Dynamic Programming in Dutch Secondary Education

Educational Research Project - 10 EC

Mathematics

Author:

Jarco Slager (s1850881)

Supervisors:

Dr. Ir. M. Timmer

Dr. Ir. J.A.H Alkemade

May 27, 2021

UNIVERSITY OF TWENTE.

Abstract

This research project focused on how Computational Thinking can be taught to students in upper secondary education (*Dutch: bovenbouw vwo*). For this purpose, a lesson series has been designed around the concept of Dynamic Programming. Motivation and comprehension played a central role in the design of the lesson series. The lesson series has been analysed by a panel of experts. Their feedback was used to design a second iteration. Two students in secondary education have participated in the lesson series and have been interviewed. Their findings are used to give further recommendations to improve the lesson series. The final product of this research project is a lesson series that can be used by teachers in the Netherlands and abroad to teach students basic understanding of Dynamic Programming and to teach them several Computational Thinking skills.

Keywords: Computational Thinking, Dynamic Programming, mathematics, education

Preface

Teaching mathematics is more than just my study or my work. It is a hobby. Even though, when I was in secondary education, it was not something I particularly liked. My main interests were languages, such as French and Spanish. However, I had inspiring mathematics teachers who motivated me to be curious about mathematics and, eventually, encouraged me to study mathematics. Studying mathematics taught me theory about many subjects, including Dynamic Programming.

Dynamic Programming appealed greatly to me in my courses, both in bachelor degree courses and master degree courses. I am delighted to have used this strong, mathematical tool to design a lesson series that I hope can help students to develop their Computational Thinking skills and their interest in mathematics.

In general, the educational research project is focused on Dutch education only. Even though this design is mainly focused on and tested with Dutch students, an English version is also available. I was requested to write the report in English, as a third party was interested to use lesson material for a European project.

First of all, I would like to thank Mark Timmer for the great supervision. Many Wednesday evenings over the past couple of months you have helped to improve the lesson design and my English. Additionally, you motivated me even more to ensure that I have designed a lesson series that I am proud of. Furthermore, I would like to thank the panel of experts for providing feedback and the participating students for giving me the possibility to test the lesson series. Finally, I would like to thank Jelle for his support and his good care.

Contents

1 Introduction

Educational changes and developments occur every day (Activiteiten Inspectie van het Onderwijs, 2019). These developments are initiated to imporve the quality of education. The subject of mathematics is not an exception to this rule. Also for mathematics modifications and innovations arise, for example the proposals from Curriculum.nu (2019). One of these proposals is that more attention is needed for what they call "mathematical thinking- and working procedures". These procedures are, for example, algorithmic thinking, problem solving and the use of technology. In this research, an interest is taken in a possible subject that is suitable for developing these procedures. This leads to the term Computational Thinking (CT).

CT is a concept that becomes increasingly important in secondary education (Timmer and Tolboom, 2019). One of the founders of CT, Wing (2014), describes it as "The thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer -human or machine- can effectively carry out." An important concept in this regard is computational complexity. The development of CT offers a great perspective to improve the skills of students regarding the mathematical processes of modelling, reasoning and problem solving (Calao et al., 2015). Even though Calao et al. (2015) conducted their research focusing on students of primary education, this may also apply to students in secondary education.

To teach the computational thinking-skill, different views are discussed in literature (Kong, 2019). It could be taught "cross-curricular", i.e. teaching computational thinking in different subjects. It could also be part of computer science, since some believe that computational thinking should be taught via programming (Kong, 2016). Even though this lesson series does not focus on programming specifically, some of its aspects, such as computational complexity, will be explored.

To teach CT around a mathematical subject, Dynamic Programming (DP) could be used. DP requires many aspects of CT and thus offers perspective as the central theme to teach several features of CT. The fundamental idea of DP is that complex problems that can be solved by breaking them down into smaller problems. These smaller problems can (usually) be solved in a straightforward way. Afterwards, the solutions to these smaller problems can be combined to solve the larger, more complex problem. There are many examples in which DP offers a powerful solution tool, which will be mentioned later in this chapter.

The question that arises is how DP should be taught in secondary education. Anderle et al. (2019) give an approach on how to teach DP and recursion (these two have a strong correlation) before students go to college/university. Furthermore, they propose examples that are suitable to solve with DP. However, these examples have been designed for the PRASK, a competition in Slovakia for the promotion of algorithmic and programming skills. This is focused on talented youths from the age of 11-15. In principle, this should be suitable for students in upper secondary school (aged 15-18) (*Dutch: bovenbouw havo/vwo*). The problems that Anderle et al. (2019) give are mainly focused on games, e.g. chess and the NIM-game (for an explanation of NIM, see for example (Berlekamp et al., 2018)).

There are many other subjects in which DP is a powerful method. Bockenhauer et al. (2019) describe a way to educate students regarding DP without any knowledge in computer science/programming. Bockenhauer et al. (2019) introduce the concepts of DP in a way that is particularly suitable for biology students; they focus on comparing two DNA sequences by using DP. For two small sequences, this problem can be solved by hand, which is demonstrated in the article. However, they plead for an approach in which students also solve the problem by using MATLAB or PYTHON in order to give them

more insight in how to solve larger and more complex problems. The authors also mention that there were experiments with students from secondary education; however, there is no further information about this matter.

There are many other problems in which DP can offer the solution. Winston (2004) gives and explains many well-known problems like the resource-allocation problem (knapsack), inventory problems and network problems. The latter might be suitable to connect mathematics to the real world experience of students. Furthermore, Winston (2004) provides small examples that can easily solved by hand and mentions how the knapsack-problem can be solved by using Spreadsheets. This way, students are not required to possess any knowledge concerning MATLAB or PYTHON to tackle larger problems. Thus, using Spreadsheets can offer an adequate alternative to other programming languages.

Because of the developments in education and the possibility of CT earning a place in the curriculum, the focus of this research lies on the development of a lesson series that teaches the students basic concepts of CT. The topic used for this purpose is DP. Thus, the students also should acquire basic knowledge of DP. In addition, the students should be motivated to learn something about these topics. Therefore, this research has two main objectives:

(I) The lesson series should motivate students to learn about DP and CT.

(II) The lesson series should improve the mathematical knowledge/skills of students with respect to DP and CT.

The outline of this report is as follows. In chapter 2, the theoretical framework is given. The research goals (also stated above) are elaborated in chapter 3. In chapter 4 the methodology is given for the design and the testing of the lesson series. Then, in chapters 5 and 6 the results and conclusions are given. The appendices A - F contain feedback obtained from a panel of experts, a previous iteration, the answers to the exercises from students, a transcription of the interviews with students, the complete lesson series and a teacher's guide. The lesson series and teacher's guide are available in Dutch and English.

2 Theoretical Framework

This chapter describes the mathematical background of DP, as well as the didactical aspects of teaching DP and CT skills.

2.1 Mathematical Background

This section gives the definition of DP that will be used throughout this research. Afterwards, the relation with Markov Decision Processes is discussed. Finally, this theory is illustrated by a specific example for which DP is useful: finding an optimal (shortest) route in a network.

The definition of DP
The definition of DP that is used throughout this research is that of Winston (2004):

Dynamic programming is a technique that can be used to solve many optimization problems. In most applications, dynamic programming obtains solutions by working backward from the end of a problem towards the beginning, thus breaking up a large, unwieldy problem into a series of smaller, more tractable problems.

In chapter 1, a few examples of applications of DP have already been given. There are many different methods to solve DP problems, and also different types of problems. Two types are deterministic dynamic programs (DDPs) and probabilistic dynamic programs (PDPs). To solve PDPs, basic knownledge about probability theory is required, since the expectation of a random variable plays a central role. In this research, this is not prerequired knowledge for the students, and therefore this research restricts itself to DDPs. However, the concepts that are introduced in the remainder of this section can also be applied to PDPs.

DP and Markov Decision Process
A DP problem can be formulated as a subproblem of a Markov Decision Process (MDP) (Puterman, 1994). For the definition of an MDP and a DP problem, the terminology of Puterman (1994) is used.

An MDP is defined as the set $\{T, S, A, p_t(\cdot|s,a), r_t(s,a), r_N(s)\}$, with:

- Time periods/stages $T = \{1, 2, \ldots, N\}$, with $N \in \mathbb{N} \setminus \{0\}$. These are the moments in which an action a is chosen. Note that in this research only finite horizons are considered, i.e. there is only a finite number of stages $N < \infty$.
- The state space $S = \{S_t\}$, where S_t consists of the states that the system can attain in stage $t \in T$. The set $S' = \bigcup_{t=1}^{N} S_t$ denotes the set of all states.
- The action space $A = \{A_s | s \in S'\}$, a set consisting of the sets A_s with $s \in S'$ (the action space for state s). An action $a \in A_s$ can be chosen when the system is in state s, independent of t (i.e. it does not change over time).
- The transition probability $p_t(\cdot|s,a)$, where $p_{t'}(j|s,a)$ denotes the probability of transitioning to state $j \in S'$ given that at time $t = t'$ the system is in state s and action $a \in A_s$ is chosen.
- The immediate reward $r_t(s,a)$, with $r_N(s)$ the salvage value. The salvage value $r_N(s)$ is the reward obtained if the final state is state s, thus at $t = N$.

For a DDP it holds that the next stage can be determined with certainty based on the current state of the system s and the chosen action a. For a formal definition of what is meant by this property of the DP, consider the transfer function $\pi_t(s,a) : S' \times A_s \to S'$. This function indicates what the state

of the system is in stage $t+1$ in the case that in time period t the system is in state s and action $a \in A_s$ is chosen. For a DDP it is then true that the transition probability is given by

$$p_t(j|s,a) = \begin{cases} 1 & \text{if } \pi_t(s,a) = j \\ 0 & \text{if } \pi_t(s,a) \neq j. \end{cases}$$

Since the transition probability is independent of t, this probability is simply denoted as $p(\cdot|s,a)$.

For choosing the actions, only Markovian choices are considered in this research. Let $\mathscr{L}(A_s)$ be the set of all possible probability distributions over the elements of A_s. Let $q_s(\cdot) \in \mathscr{L}(A_s)$ be a probability distribution and let $A_s = \{a_1, a_2, \ldots, a_k\}$.

The interpretation of $q_s(a)$ is that the action a is chosen with probability $q_s(a)$ when the system is in state s. For example, choosing a "mixture" of actions could have the following distribution:

$$q_s(a) = \begin{cases} \frac{1}{4} & \text{if } a = a' \\ \frac{3}{4} & \text{if } a = a'' \\ 0 & \text{otherwise.} \end{cases}$$

As only deterministic choices are focused on in this research, it holds that $q_s(a') = 1$ for action a' and that $q_s(a) = 0$ for $a \neq a'$. The "mixture" of actions is not allowed.

Solving the routing problem

The theory will now be illustrated by means of the routing problem of Figure 1. The problem consists of finding the shortest route from node 1 to node 8. The arrows between the nodes indicate the possibility of moving from one state to the other, in the direction of the arrow. The travel costs (for example travel time or distance) for the route are specified next to the arrows. Note that working with costs is analogous to working with rewards. The shortest route consists of the collection of arrow going from node 1 to node 8 such that the total (added) costs are minimal.

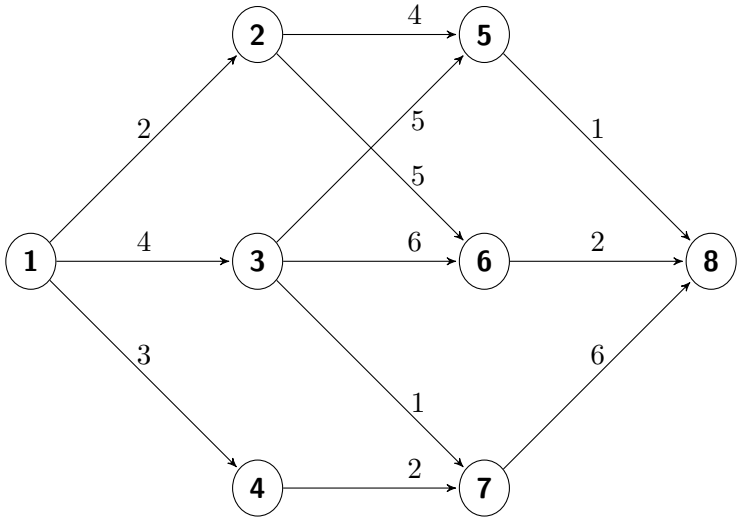


Figure 1: *An example of a routing problem.*

In terms of an MDP, the problem can be described in the following way:

- $T = \{1, 2, 3, 4\}$, since an action has to be chosen in three different stages to arrive at the fourth stage. An action has to be chosen in node 1, in nodes 2/3/4 and in nodes 5/6/7.
- $S = \{S_1, S_2, S_3, S_4\}$, with $S_1 = \{1\}$, $S_2 = \{2, 3, 4\}$, $S_3 = \{5, 6, 7\}$ and $S_4 = \{8\}$. Note that in this case, the sets S_t are disjoint, i.e. $S_{t'} \cap S_{t''} = \emptyset$ for $t' \neq t''$. For general problems (also routing problems), these sets S_t need not be disjoint.
- The action space depends on the state of the system s. Note that $S' = \{1, 2, \ldots, 8\}$. Let a_{ij} be the action of going from node i to node j. Then:
  1. $A_1 = \{a_{12}, a_{13}, a_{14}\}$.
  2. $A_2 = \{a_{25}, a_{26}\}$.
  3. $A_3 = \{a_{35}, a_{36}, a_{37}\}$.
  4. $A_4 = \{a_{47}\}$.
  5. $A_5 = \{a_{58}\}$.
  6. $A_6 = \{a_{68}\}$.
  7. $A_7 = \{a_{78}\}$.
  8. $A_8 = \emptyset$.
- Since this problem is a DDP, $p(j|s, a) = 1$ for $a = a_{s,j}$, else $p(j|s, a) = 0$.
- The immediate rewards $r_t(s, a)$ are the costs next to the arrow in Figure 1. The salvage value is $r_4(8) = 0$.

To tackle this problem, one could apply the "brute-force" technique. That would require the computation of the total costs for all six routes and then choosing the route with the smallest costs. As a matter of fact, this is an adequate approach to solve this particular problem. However, this is not always the case, as is illustrated in the literature (see for example Winston (2004)). In many cases, it becomes clear that DP is preferable over the "brute-force" technique. In this example, a DP algorithm will be used, based on the algorithm by Puterman (1994), who calls it the backwards induction algorithm. The algorithm for this problem is given by Algorithm 1:

Algorithm 1: An algorithm for the routing problem.

Initialization:

$u_N^*(s) = r_N(s) \quad \forall s \in S_N$;

$t = N - 1$;

while $t > 1$ **do**

$u_t^*(s) = \min_{a \in A_s} \left\{ r_t(s, a) + \sum_{j \in S} p(j|s, a) u_{t+1}^*(j) \right\}$ for each $s \in S_t$;

Set $A_{s,t}^* = \arg\min_{a \in A_s} \left\{ r_t(s, a) + \sum_{j \in S} p(j|s, a) u_{t+1}^*(j) \right\}$ for each $s \in S_t$;

$t = t - 1$;

end

In Algorithm 1, $u_t^*(s)$ denotes the optimal value for the shortest route when the system is in state s at stage t. As the name backwards induction (Puterman, 1994) indicates, the problem is solved by

starting in the last stage (at node 8). There is no reward for finishing in node 8, thus $r_N(8) = 0$. In each iteration, the fastest route from the current node to the last node is stored. This way, when that node is reached via a node from a different stage, the fastest route to reach the last node is known. This ensures that there is no faster route to reach that last node. For example, if it is known that going from node 3 to node 8 is the fastest via node 5, then for the computation of going from node 1 to node 8, the routes that go from node 3 to node 6 or 7 do not need to be considered, since those routes are guaranteed to have a higher cost than going via node 5. Due to the deterministic setting, Algorithm 1 can be simplified. The simplified version is given in Algorithm 2, which is used to solve the routing problem. Since the transition probabilities are not written down explicitly in Algorithm 2, this is incorporated in the optimal value: $u_t^*(j|s,a)$ denotes the optimal value for the shortest route when the system is in state j at stage t, when the system was state s and action a was chosen in stage $t-1$.

Algorithm 2: An algorithm for the deterministic routing problem.

Initialization:
$u_N^* = r_N(s) \quad \forall s \in S_N$;
$t = N - 1$;
while $t > 1$ **do**
  $u_t^*(s) = \min_{a \in A_s} \left\{ r_t(s,a) + u_{t+1}^*(j|s,a) \right\}$ for each $s \in S_t$;
  Set $A_{s,t}^* = \arg\min_{a \in A_s} \left\{ r_t(s,a) + u_{t+1}^*(j|s,a) \right\}$ for each $s \in S_t$;
  $t = t - 1$;
end

Below, the solutions for the routing problem of Figure 1 is given.

Solution

Initialisation:
$t = 4$ en $u_4^*(8) = 0$.
Iteration:
$t = 3$
$u_3^*(5) = \min\{1\} = 1$, hence $A_{5,3}^* = a_{58}$.
$u_3^*(6) = \min\{2\} = 2$, hence $A_{6,3}^* = a_{68}$.
$u_3^*(7) = \min\{6\} = 6$, hence $A_{7,3}^* = a_{78}$.

$t = 2$
$u_2^*(2) = \min\{4 + u_3^*(5), 5 + u_3^*(6)\} = \min\{5, 7\} = 5$, hence $A_{2,2}^* = a_{25}$.
$u_2^*(3) = \min\{5 + u_3^*(5), 6 + u_3^*(6), 1 + u_3^*(7)\} = \min\{6, 8, 7\} = 6$, hence $A_{3,2}^* = a_{35}$.
$u_2^*(4) = \min\{2 + u_3^*(7)\} = \min\{8\} = 8$, hence $A_{4,2}^* = a_{47}$.

$t = 1$
$u_1^*(1) = \min\{2 + u_2^*(2), 4 + u_2^*(3), 3 + u_2^*(4)\} = \min\{7, 10, 11\} = 7$, hence $A_{1,1}^* = a_{12}$.

The solutions that have been given will now be discussed. Since $u_t^*(s)$ denotes the optimal value from stage t with the system in state s, the shortest route to go from node 1 to node 8 has value $u_1^*(1) = 7$. This optimal value is attained by choosing action a_{12} at $t = 1$, thus at $t = 2$ the system is in node 2. At node 2, the optimal choice is to choose action a_{25}. Then at $t = 3$ the system is at node 5. From

here, the optimal (and only) choice is to take action a_{58}. In conclusion, the best route is (1) → (2) → (5) → (8) with a total cost of 7.

Even though DP can substantially reduce the computational complexity of optimisation problems, there are also problems that are too large to solve with DP. A large problem can suffer from the so-called 'three curses of dimensionality', which are the following:

- The state space is too large.
- The solution space is too large.
- The action space is too large.

It essentially boils down to the fact that problems can be too vast to solve using the standard approach of DP. To solve these problems, the technique of Approximate Dynamic Programming (ADP) can be used. Information about the details of ADP are given by Powell (2007). ADP is out of the scope for this research, as the standard DP approach will suffice.

2.2 Didactical aspects of teaching DP

Sections 2.2.1 and 2.2.2 discuss the theoretical background regarding motivation and comprehension, respectively. Furthermore, in each section the design requirements for the lesson series are introduced, which are based on the theory. The final list of design requirements can be found in chapter 3.

Another goal of this section is to demonstrate why DP is a suitable theme for this lesson series. This is discussed in both subsections.

2.2.1 Motivation

When teaching new theory or a new concept, students' motivation should be taken into account. For students to learn new theory, they should be motivated to carry out the learning tasks as these tasks will help them to grasp this theory.

To motivate students to carry out learning tasks, research shows that three factors play an important role (Marzano, 2018):

1) Students should experience the exercise as meaningful.
2) Students should be confident that they are capable of carrying out the learning task.
3) Students should have a clear idea about what is expected of them.

Marzano (2018) gives several ideas to integrate these factors into practice. The ideas that are suitable for this research are described in the remainder of this section.

Factor 1: Students should experience the exercise as meaningful
In general, it is sufficient for a teacher to explain to the students where they can apply the new theory. For example, this can be done by explaining where this knowledge is useful in daily life, both within and outside of school. Regarding DP, there are many examples that show the relationship between the theory and daily life, as mentioned in chapter 1, which could demonstrate the connection between the subject of mathematics and the "outside world". To enthuse students about mathematics and science, it is important that they get in touch with examples they could encounter in daily life (Simpkins et al., 2006). Parents and teachers can play a central role by showing students examples of mathematics

they encounter every day. To stimulate this, the developed lesson series can be used. For example, it is possible to explain a concrete example to the students in detail such as the routing problem stated in section 2.1. This leads to the first motivational requirement of the lesson design, referred to as (M1): it should contain applications of the theory in real-world problems.

Another way to give insight to students about what they can do with that knowledge is by letting them explain to each other what they think is the use of a learning task (again, this could be done using the example of the routing problem from section 2.1). Both the students and the teacher could solve this problem and afterwards, the students could conclude themselves what the point is of the mathematical theory they just learned. The usefulness of DP in this context is that it provides an efficient method to find the fastest route between a starting node A and a destination node B. This is something that every student could agree with.

Furthermore, it is important that the teacher knows what he or she is talking about and that the learning tasks that are developed are suitable for both the learning outcomes and the interest of students. For example, a context about the comparison of two DNA-strings as mentioned by Bockenhauer et al. (2019) is interesting for students that follow a biology course (which is not mandatory in upper secondary education in the Netherlands). On the other hand, for the students that do not follow a biology course, this example is not suitable. The routing problem or finding an optimal strategy for a game (see (Anderle et al., 2019)) would be more suitable for a broader audience.

Factor 2: Students should be confident that they are capable of carrying out the learning task

Being confident that a task can be successfully executed can lead to students putting effort into solving that task. Furthermore, it also has an effect on how long students will keep on working on an exercise despite obstacles they encounter (Bandura, 1977). Lack of success and the perception that failure is due to inability may undermine students' motivation to learn (Dweck, 1986).

According to Marzano (2018) it is important for a teacher to be a model to their students. Therefore, it is essential that sufficient examples are discussed and explained in detail, using the correct strategy. This can also be achieved by providing the students with worked out examples. This way, students acquire the skill to solve such a problem themselves. This leads to requirement (M2): the lesson series should contain examples and explanations on how to solve exercises.

Furthermore, it is important that a teacher can offer help and guidance and (perhaps) more importantly, students should have the feeling that they can ask for advice. To achieve this, (difficult) exercises could have an additional remark that the teacher can be consulted if necessary. However, creating an environment in which students can ask questions is considered to be a task of the teacher instead of the lesson series. Therefore, additional remarks that questions can be asked is not taken into account for the design of the lesson series.

Lastly, the teacher can make sure that the students realise they have the skills to carry out the learning task. This can be done by discussing which skills are necessary to do the exercise, but also by showing that the new learning task is linked to the exercises that the students have done in a prior stage. This stimulates students to stay motivated (Ebbens, 2015). Here, it is essential that students' prior knowledge is activated (denoted as requirement (M3)). This can be accomplished by, for instance, using the examples that the students previously dealt with as a basis for the new learning task. Another method to accomplish this is by connecting the new learning task to the mathematical theory they studied over the past few years.

Factor 3: Students should have a clear idea about what is expected of them
Firstly, the goal of the learning task should be clear to a student and secondly, the students should know what is asked of them. Therefore, when designing a lesson series it is important that another party goes over the exercises, so one can see how the task is perceived. For students, it is of great importance that they know what they have to do and how they have to do that (Teitler, 2017). Therefore, during the lesson series, the exercises should contain clear instructions and indicate in each exercise what a students should do and how the student should do that. This is referred to as requirement (M4).

In case of a summative assessment of the exercises, it is important students should know what they will be graded on. This can be done by for example a rubric or by an example of a solution from the teacher. For all exercises, clear instruction regarding the final product of the exercise are crucial.

Not only should the exercises give clear instructions, they should also challenge the students in different ways. Proposing the theory in a variety of ways is important for students to learn in an active way (Ebbens, 2015). Therefore, the last motivational requirement (M5) is that the design should contain different types of learning tasks.

2.2.2 Comprehension

This section focuses on how to teach DP and how to teach CT skills. Even though DP is used as a tool to teach CT skills, students will learn about both topics. Thus, both DP and CT are discussed in this section. The method to design questions for the lesson series is discussed at the end of this section.

Dynamic Programming
To be certain that the students understand the mathematical theory of DP, information is gathered on how DP should be taught. Anderle et al. (2019) argue that DP is important in the theory of algorithms, but that in general DP is seen as something that is difficult to understand and that is counter intuitive, even by university/college students. Therefore, they have opted not to explain DP in full detail as a technique to design algorithms. Alternatively, they decided to shed light on certain aspects of DP to give students some intuition regarding this concept. They believe that when students are older and learn more about the technique of DP, they can connect these different aspects more easily and therefore get a better view of what DP is and how it works. As stated in chapter 1, Anderle et al. (2019) confront students with different games like NIM and chess. In these games, students should discover a solution strategy given a specific situation or context.

The didactical aspects of Anderle et al. (2019) are now discussed. They have chosen to explain DP in different "pieces", and not explain the whole concept at once. Their goal is to make sure that designing an algorithm with the use of DP is not seen as a "black box". Linking to the didactics of mathematics, Anderle et al. (2019) split the theory in smaller, separate pieces to explain the concepts to the students. The separation of theory affects the students' cognitive development of this theory (Piaget, 1971). According to Van Dormolen (1974), breaking up a concept in small chunks leads to short-term success, however, for long-term memory this leads to incoherent cognitive schema's. Van Dormolen (1974) therefore argues that it is important to teach new theory using one central theme.

On the other hand, one can argue in favor of the approach of Anderle et al. (2019). A particular disadvantage of learning using the schema's of Piaget (1971) is time pressure (Drijvers et al., 2019). As the goal of our research is to give students an introduction to and intuition for DP, and not to

teach students formal mathematical theory in secondary school, this approach is can be used for our purpose. Moreover, Anderle et al. (2019) point out that their approach has been successful.

Since working from one central theme is beneficial for the students' understanding of the theory, this will be done in the lesson series as well. The theme that will be used to teach students CT skills (see the part on CT below) is DP. Therefore, the design will need an introduction to this theme, which is the first requirement regarding comprehension ((C1)).

From the central theme of DP, students should learn both CT skills and ideas from DP. To succeed in that regards, several applications from DP should be considered and discussed. This leads to requirement (C2): the lesson series should contain examples of applications of DP.

Bockenhauer et al. (2019) argue that one should not use a formal (mathematical) introduction to DP. Their starting point is that students do not have any prior knowledge on recursion and they believe that the students learn this gradually. Their idea is inspired on the work of Aho (1974), however, this book has been written in a formal mathematical way and the level is above the level of students of Dutch secondary schools. Therefore, Bockenhauer et al. (2019) have a different approach. Besides letting students work with problems on paper, they argue that using MATLAB or PYTHON has a positive effect on the students' understanding of DP. Unfortunately, this is not applicable in Dutch secondary education, as these programming languages are (currently) not part of the curriculum (Examenblad, 2020b,a). As some form of programming can be beneficial for the comprehension of DP, requirement (C3) is that computer software should be used to solve a DP problem. An alternative for programming with MATLAB or PYTHON is Spreadsheets, as used by Winston (2004). Spreadsheets is a better option, since it is more accessible to students in secondary education.

Demonstrating the inefficiency of the brute-force method in class is a widely used reason to show students a different approach is required to tackle certain problems. For the introduction of DP, many real-world problems can be given, since this method has been developed to solve realistic problems and therefore a large number of application exists. This enables students to connect the concepts of DP to the real world, which complies with realistic mathematical education (Freudenthal, 2002). Real-world problems are taken into account in requirement (M1), but using these problems students are given the opportunity to learn another aspect of DP: the advantages it has compared to the brute-force approach. As the computational advantage of DP is an important aspect and one of the reasons why it has been developed, students should learn the advantages of DP (requirement (C4)).

Computational Thinking
CT is a broad topic. As there are time constraints to the lesson series (the lesson series will consist of three separate lessons), only a few aspects can be taken into account. STEM (Science, Technology, Engineering and Mathematics) researchers developed a taxonomy that divides CT STEM-practices in four different categories with several examples (Weintrop et al., 2016):

- *Data practices*: data collection, creation, manipulation, analysis, and visualisation.
- *Modeling & Simulation Practices*: using computational models to understand a concept or to find and test solutions, or assessing, designing, and constructing computational models.
- *Computational Problem Solving Practices*: preparing problems for computational solutions, programming, choosing effective computational tools, assessing different approaches/solutions to a problem, troubleshooting/debugging.
- *System Thinking Practices*: investigating a complex system as a whole, understanding the rela-

tionships within a system, defining systems, and managing complexity.

In addition, Weintrop et al. (2016) give an explicit list of CT skills:

1. Ability to deal with open-ended problems.
2. Persistence in working through challenging problems.
3. Confidence in dealing with complexity.
4. Representing ideas in computationally meaningful ways.
5. Breaking down large problems into smaller problems.
6. Creating abstraction for aspects of problem at hand.
7. Reframing problem into a recognizable problem.
8. Assessing strengths/weaknesses of a representation of data/representational system.
9. Generating algorithmic solutions.
10. Recognizing and addressing ambiguity in algorithms.

Thus, several aspects of CT exist that can be focused on. Since students are not expected to have knowledge about programming, the programming aspects cannot be taken into account. Fortunately, certain CT topics do not require this prior knowledge.

The category *System Thinking Practices* could be suitable for students that do not know programming languages, since students can be asked about the complexity of a system (i.e. the number of computations) without any knowledge about how a computer tackles the problem. In chapter 7 of (Kong, 2019), students have to explore a model by changing its parameters to learn more about the model. That same approach could be applicable to this research. For example, students can be asked to explain what happens when the dimensions of the problem change and how that affects the computational time. It is indeed possible to teach student CT skills without the sole use of computers. Thus, students should learn several CT skills in the lesson series (e.g. computational complexity), which is requirement (C5). The list of skills that Weintrop et al. (2016) give is the basis of this requirement.

Designing Exercises

In addition to this, different types of exercises should be considered. Several methods and techniques can be used to achieve an appropriate design, for example the 6E-model (Windels, 2012). This model is meant for teachers to prepare classes. The main components are a strong sense of guidance but also a strong sense of exploration for the students. An important aspect of this model is that not everything is explained to the students, but that they have to come up with their own ideas or concepts using specific examples. The teacher (or an exercise) asks solely questions, such that the student has the opportunity to "explore" the theory instead of receiving the information from the teacher. This guidance to teach students is also known as Guided Reinvention (Freudenthal, 2002). Furthermore, the 6E-model is also in line with the theory that activating prior knowledge is beneficial for learning (Wetzels et al., 2011). Concluding, the lesson series can be designed using elements from the 6E-model. The design needs a balance between two types of exercises: exercises that guide students (C6) and exercises that give room for the students to explore the theory (C7).

3 Research Goal

In this chapter, the research objectives are discussed. In Section 2.2, both motivational and comprehensional aspects of the lesson series were considered. This gives guidelines and examples of how a lesson series could be designed with DP as its central theme. Moreover, the focus lies on (Dutch) secondary education. Students require a certain level to understand the mathematical concepts of DP. Therefore, the target audience is students in (Dutch) pre-university education. (*For Dutch readers, the target audience is students in bovenbouw vwo*).

In conclusion, the two research goals for the design of the lesson series are:

(I) The lesson series should motivate students to learn about DP and CT.

(II) The lesson series should improve the mathematical knowledge/skills of students with respect to DP and CT.

In section 2.2, the design requirements have been proposed and explained. Below, the list of all the design requirements is given as an overview. To achieve the research goals, the design requirements are used to assess used as the basis for the lesson series.

Motivation
The design for the lesson series should include...:

(M1) applications of the theory in real-world problems.

(M2) examples and explanations on how to solve the exercises.

(M3) references to prior knowledge.

(M4) clear instructions on what students have to do in a learning task.

(M5) different types of learning tasks (i.e. reading text, doing exercises, watching videos, playing educational games, etc.).

Comprehension
The design for the lesson series should include...:

(C1) an introduction to DP.

(C2) examples of DP.

(C3) an exercise that can be solved using computer software.

(C4) advantages of DP.

(C5) computational thinking skills.

(C6) exercises that guide students.

(C7) exercises that gives room for students to explore the theory.

The next chapter describes the method to test these requirements. For the comprehension requirements, learning outcomes have been set up for each lesson. These learning outcomes also form the basis for the design of the lesson series. They are given on the next page.

Lesson 1
Students need an introduction to DP. Therefore, they should be able to apply it in a simple context. As Anderle et al. (2019) state that playing games is a good approach to teach students DP, this is the first learning outcome lesson 1: students should be able to come up with a winning strategy for a game using DP. This is then connected to requirement (C1).

Furthermore, they should be able to explain the basic idea of dynamic programming, which also follows from requirement (C1). Hence, being able to explain what dynamic programming entails is the second learning outcome.

Finally, as the students should know applications in the real world (requirement (M1)) and know examples of DP (requirement (C2)), the third learning outcome is that the students should be able to give examples of applications of DP.

Lesson 2
To teach students more about DP, they should be able to apply the theory. Therefore, students should be able to solve small instances of problems. The problem that is focused on in this research is the routing problem. So, as an introduction to DP (requirement (C1)), the first learning outcome is that they can compute the optimal route in a small routing problem. The quantity "small" will be specified in the in section 5.1.1.

Computational complexity is one of the CT skills. As DP can reduce the computational complexity of optimisation problems, a learning outcome is set up to take these two requirements into account ((C4) and (C5)): the students should be able to explain why DP decreases the number of computations.

Finally, the students should solve an exercise with computer software (requirement (C3)). Note that the students do not have prerequisite knowledge about computer programming. Thus, they have to learn some notation regarding minima that is common in computer languages (and mathematics). This will help them when they have to solve the exercise with computer software. Thus, the third learning outcome of this lesson is that they have to recognise and execute computations using the notation $\min_x\{\dots\}$ and $\arg\min_x\{\dots\}$. The reason that they learn this specific notation is because they will have to determine minima in Spreadsheets.

Lesson 3
Solving an exercise with computer software will result in finding the minimum in the routing problem (requirement (C3)). For this, students will need to learn how to find the minimum in a column with several values in Excel. That is the first learning outcome of this lesson.

Computational complexity is one of the aspects of CT that is chosen to be developed, since DP lends itself for this aspect (requirement (C5)). Therefore, the last two learning outcomes of this lesson (series) are that the students should be able to estimate the number of computations that need to be executed in the routing problem and that they should be able to explain how the number of computations changes when the size of the routing problem changes.

In the overview of the lesson series in section 5.1.1, these learning outcomes are listed as well. In the next chapter, there will be references to the learning outcomes, as they create a way to verify whether the design requirements are present in the design.

4 Methodology

This chapter describes the approach that was used to achieve the research goals. Furthermore, the chapter explains how the data was collected. Finally, the quality of this research is assessed by its reliability and validity.

The Methodology is divided into two separate parts. Firstly, the methodology is given on how the design was developed, which includes a walk-through by a panel of experts, resulting in a second iteration of the lesson series. Secondly, the methodology is given regarding the testing of the lesson series with students.

4.1 Methodology A - The design

This section describes the methodology regarding the design of the lesson series, which is based on the theoretical framework given in chapter 2.

Procedure
The design of the lesson series is based on the design requirements. Each of these requirements is contained in exercises, explanations or both.

Each requirement could be integrated into the lesson series in a different way. For example, regarding requirement (M1), mentioning applications in the explanation is a straightforward approach. In addition, this requirement was also suitable for being covered by an exercise as simple applications of DP exist (for example, strategic games (Anderle et al., 2019)).

Preferably, each requirement is met in several ways, such that it is completely interwoven into the lesson series. However, some of the requirements were more suitable for exercises than for explanations/theory (for instance (C6) and (C7)). In those cases, several exercises were designed that contain these requirements. An in-depth substantiation for the design can be found in Section 5.1.2.

Afterwards, the design was analysed to see which requirements have and have not been met. Each requirement was analysed separately to see if it was present or not. This has been done by a panel of experts that were given the list of requirements and that gave feedback on whether the design agrees with the design requirements by doing a walk-through of the design. Furthermore, they made general comments/remarks that were analysed to further improve the lesson series. This provided a primary check to verify if the research goals (I) and (II) were achieved. To obtain a complete verification of achievement of the research goals (I) and (II), the lesson series was tested with students and semi-structured interviews were conducted with those students (see section 4.2).

Respondents
For designing the lesson series, the expert panel has been approached. This panel consists of four experts: two teachers of the target audience (with both more than 15 years of experience in teaching), one student in mathematics education and one student in mathematics (focused on Operations Research).

Instruments
For the design, no specific instruments have been used. Solely literature is consulted in the design of the lesson series.

Analysis
As stated in the procedure, the design has been analysed to see which of the requirements have been met and which have not. Using the information of the expert panel, a distinction was made which requirements were present sufficiently and which requirements needed more consideration. This way, ideas for explanations/exercises that could be concluded were established to ensure that all requirements were met.

4.2 Methodology B - Testing with students

To test whether or not this lesson series is effective to achieve the students' learning outcomes, the lesson series was tested with two students. The procedure, respondents, instruments and analysis for this part of the methodology will be described below.

Procedure
Data has been collected to find out to which degree this lesson series is both motivational and effective to achieve the students' learning outcomes. To obtain this data, students have been asked to do the exercises of this lesson series. After doing the exercises in pairs, students had to hand in their work. The researcher was present during the execution of the lesson series to observe the time it took the students and to answer questions (which was not necessary). All three lessons were done in one session for logistic reasons.

After collecting the students' work, their answers were assessed to see if the students managed to complete the exercises and if they managed to achieve the learning outcomes. More information about how the graded work has been analysed can be found in the paragraph regarding analysis.

Additionally, one-on-one semi-structured interviews were conducted by the researcher after the students completed the lesson series, to find out their views on the material and to qualitatively assess if the students achieved the learning outcomes. The type of questions that were asked can be found in the paragraph regarding instruments. Important aspects include the design of the lesson series, the level of motivation and the level of difficulty. In conclusion, these questions are used to analyse the motivational and comprehensional aspects of the lesson series.

Since this procedure is executed with students of Dutch secondary education, there are also ethical aspects to this research. The students that participated will remain anonymous in this report. The ethical committee of the University of Twente has been approached and has given their approval for executing the research in this way.

Respondents
The respondents were students selected from a secondary school that has close connections with the University of Twente. The participants take both mathematics for scientific studies (*Dutch: wiskunde B*) and advanced mathematics (*Dutch: wiskunde D*). The age of students in Dutch pre-university education is 16-18 years.

In total, two students have participated in this study. It was not practically feasible to use a larger group of students for the experiment - especially during the current corona pandemic - since the students have a strict schedule to adhere to.

Instruments
The answers that the students have produced are used to give recommendations about further improvements on the lesson series. Furthermore, semi-structured interviews have been conducted. The interviews have been designed with a focus on motivation and comprehension, since those are the

aspects that help to answer whether or not the research goals have been achieved. The interviews are semi-structured, thus if a student gave an interesting answer to a question, the researcher asked more about that subject. This has given ideas for improvements for the lesson series, which is the main objective of testing the lesson series with students.

Motivation
The questions regarding motivation are based on the design requirements (Section 2.2). After each question it is indicated to which requirement it corresponds.

- What did you think of the examples/applications? (M1)
- Were there examples/explanations on how to solve the exercises? (M2)
- Did you feel you had enough prior knowledge to complete the lesson series? (M3)
- Did you know what you had to do in the exercises/did you know what was expected of you? (M4)
- Did you think the type of activities/exercises was diverse? (M5)

Comprehension
The questions regarding comprehension are partly based on the design requirements (section 2.2).Regarding comprehension, the students' results are also available. Therefore, if a student did not do well during the exercises, more questions could be asked to see if the students did or did not achieve the learning outcomes of the lessons. If a student did well, the standard questions provided enough information to assess whether he/she understood the theory. Similar to the questions for motivation, there is an indication to which requirement it corresponds (if applicable).

- Can you explain what dynamic programming is? (C1)
- Can you give examples of applications of dynamic programming? (C2)
- Did you use computer software to solve a problems? And did this help you to understand the theory? (C3)
- Can you explain why dynamic programming is faster at finding the shortest route versus comparing all the different routes? (C4)
- Given a (small) routing network, do you think you could find the shortest route using dynamic programming? (C5)
- Were there questions that led you step by step to the correct answer? (C6)
- Where there questions that required more effort than others to come up with the answer? (C7)
- Do you have any comments on the lesson series?

For requirement (C5), there are many different aspects that can be tested. To ensure the interviews were be conducted within a reasonable time frame, the conclusions on this requirement are primarily based on the students' work. Students were allowed to give any feedback in the last question of the interview in which they could mention anything that was not discussed before.

Regarding reliability of the research, the focus did not solely lie on the work that the students handed in, but also on the semi-structured interviews. With these interviews, an extra verification step is

introduced to investigate if the students did or did not achieve the learning outcomes. The analysis will be discussed in the next paragraph.

Analysis
For each of the learning outcomes, the students' answers have been analysed qualitatively. Furthermore, the students' elaboration in the semi-structured interview have also been taken into account. Combining these two types of data, a verification is made if students have achieved the learning outcomes. This indicates whether or not the lesson series is suitable to teach students various skills of computational thinking. The answers that the students gave regarding comprehension provided a complete idea of the achievements of the students. Achieving the learning outcomes indicates that the exercise is probably effective. On the other hand, if students did not achieve these learning outcomes, improvements should be proposed such that the lesson series becomes more effective.

Furthermore, in the semi-structured interview the students were asked about their opinion on the lesson series. Questions regarding all aspects of motivation and comprehension in the lesson series. This way, an analysis is made to assess whether or not the lesson series contains the design requirements. The semi-structured interviews were transcribed and coded per interview question.

Regarding reliability, note that the group of students is extremely small. It is possible that these students are very enthusiastic, or exactly the opposite. This can give a wrong view of the quality of the lesson series.

Another downside of a small group is that it might not yield enough results. To obtain as much information as possible from the students, the aspect of comprehension was tested in two ways: by analysing the work and by analysing the semi-structured interviews.

It is important to understand which part of the research answers which questions. First of all, the lesson series was conducted. Then, the students handed in their work which which assessed and they participated in a semi-structured interview. The data from the exercises and the interview help to answer the question whether or not the students achieved the **learning outcomes** and if they were **motivated** to do the exercises. The results from this step are used to verify if the design meets the **requirements**. Based on this verification, a conclusion can be drawn regarding the accomplishment of the **research goals**. A schematic overview of this process can be found in Figure 2.

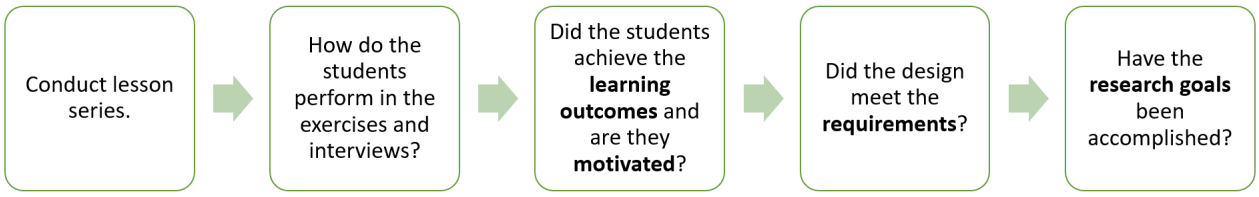


Figure 2: *A schematic overview of the testing process with students.*

5 Results

In this chapter, the results will be discussed. Firstly, an overview of the lesson design is given. Secondly, the lesson series itself is given and explained. Thirdly, the feedback from the panel of experts is stated. Then the results from testing the lesson series with students is considered, including the semi-structured interviews. Finally, the chapter concludes with an overview of the results.

5.1 Lesson Design

In this section, the overview of the lesson series is stated. In section 5.1.2 an explanation for how and why the lesson series has been designed this way is given. It also includes which aspects of CT are present in which exercises. This substantiation is done for the English lesson series for consistency in language. The Dutch and English versions, without explanations, can be found in appendix E. The teacher's guide and solutions to the exercises can be found in appendix F (both available in Dutch and English).

5.1.1 Overview

In this overview, a short introduction is given on the contents of each lesson. The lesson series consists of three consecutive lessons. Learning outcomes have been set in line with so-called SMART learning outcomes (Donders and Ruijs, 2013). In short, this means that the learning outcomes ought to be specific, measurable, acceptable, realistic and time-bound. They have already been described in chapter 3. They are stated again to obtain a complete overview of the lesson series and the goals of each lesson.

Lesson 1: Introduction to Dynamic Programming
In this lesson, the students play the match-game (NIM-game). Then, they find a winning strategy for the starting player. In the last exercise of this introduction, students try to get an idea of what DP entails. The teacher finalises this lesson by giving some examples of the application of DP.

Learning outcomes: At the end of the lesson, students are able to ...

- come up with a winning strategy for a game using dynamic programming.
- explain what dynamic programming entails.
- give examples of applications of dynamic programming.

Lesson 2: The routing problem
Students start with the routing problem. They are given a small network of stations in which they will apply dynamic programming to find the shortest route. New notation for minimisation is introduced, which will be used in lesson 3. For a function $f(x)$, this regards the minimum $\min_x\{f(x)\}$, but also the argument $\text{argmin}_x\{f(x)\}$.

Learning outcomes: At the end of the lesson, students are able to ...

- compute the optimal route in a small routing problem with a unique solution using dynamic programming. *Small: maximum 4 phases and 10 nodes.*
- give an explanation why the number of computations to find the optimal solution in a routing network decreases when solving it with dynamic programming instead of minimizing over all possible routes.

- recognise and execute computations using the notation $\min_x\{\dots\}$ and $\arg\min_x\{\dots\}$.

Lesson 3: The routing problem with Excel
In this last lesson, students start with a larger routing problem than in the former lesson. They will have to find the optimal solution using Excel. The lesson series contains a brief explanation how Excel can be used for the final exercise. They will first evaluate the problem on paper to understand the context. Then, they will move to Excel to find the optimal costs. Finding the optimal route is out of the scope due to the limitations in prior knowledge regarding Excel.

Learning outcomes: At the end of this lesson, students are able to ...

- use the function MIN in Excel to find the minimum of the sum of two columns (of equal length).
- give an estimate of the number of computations that have to be executed to find the optimal route in a routing problem.
- explain how the number of computations that have to be executed changes when the size of the routing problem changes.

The Excel file is explained in appendix E.3 (in English). The author may be contacted for the original file.

5.1.2 Substantiation of the design

In this section, the design of the lessons and exercises is explained in detail. Furthermore, as the goal of this lesson series is to teach CT, an overview is given of which CT skills can be found back in the exercises. The skills correspond to the skills on page 16. See Table 1 for the exercises per lesson and the CT skills per exercise.

Table 1: *The skills of computational thinking indicated for each exercise.*

Skill	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
Exercises lesson 1										
Ex. 1										
Ex. 2					X				X	
Ex. 3				X	X					
Exercises lesson 2										
Ex. 1					X					
Ex. 2						X				
Ex. 3						X				
Exercises lesson 3										
Ex. 1	X	X	X		X					
Ex. 2								X		
Ex. 3										
Final Assignment		X			X		X			

Note that not all exercises are focused on CT skills. Furthermore, some skills are more present than others and the last skill has not been included in any exercise. The concept of DP is suitable for all these skills, however the fifth skill is the main focus. This is due to skill five covering the main idea of DP: breaking down large problems into smaller subproblems.

In the following paragraphs, the exercises and explanations are discussed per lesson, in which a substantiation is given for the design. The substantiation will be given below each exercise. To distinguish the exercise from the substantiation, the explanation is written in blue.

Lesson 1: Introduction to Dynamic Programming
Exercise 1 - (M1)(M2)(M4)(M5)
You are going to play a strategy game, the match-game. This game is played in pairs, with player A and player B. The rules of the game (read them carefully before you start playing):

1. Start with 30 matches on the table.
2. Choose the person to start, who will be player A. Player A can choose to pick up either 1, 2 or 3 matches from the table.
3. Now it is player B's turn. Player B can also pick up 1, 2 or 3 matches from the table. You repeat this alternately. You always have to pick up at least 1 match from the table.
4. The player that has to pick up the last match from the table loses. For example: it is player A's turn and there are 3 matches on the table. If player A picks up 2 matches from the table, then there is 1 left on the table. Player B must pick up this one and loses. Enjoy playing the game!

This exercise has been created mainly to engage students in the new theory. Berlekamp et al. (2018) believe that games are interesting to students. Therefore, the goal of this first exercise is to engage students in the lesson series. This is related to the first learning outcome of this lesson.

Exercise 2 - (M4)(C6)
It is possible to construct a strategy to ensure that player A (the starting player) wins the match-game. Try to come up with such a strategy to guarantee that player A wins. Write down the steps that player A has to follow to ensure victory.

This exercise focuses on CT skills 5 and 9. To find a winning strategy, one should start with a small subproblem (skill 5) and then generalise this to larger problems. With this knowledge, students come up with a strategy/plan that makes sure that player A wins the game. In other words, the students have to come up with an algorithmic solution (skill 9).

Furthermore, it is very explicit what is expected of students. They played the game in exercise one and know what the game is. It is clear that they have to give a product, namely a step by step instruction on how to win for player A (the starting player). This motivates students to carry out the learning task (Marzano, 2018).

Exercise two is also related to the first learning outcome of this lesson.

Exercise 3 - (M1)(M2)(C1)(C2)(C4)(C6)
We just discussed how such a strategy can be found. The key is to first analyse the situation when a single match is on the table. Finding an optimal strategy/approach this way is an example of **dynamic programming**. For dynamic programming, problems are solved by breaking them up in smaller chunks, or by starting "at the end" for example. When the solution is obtained for a smaller problem, this is then used to solve the larger problem.

Question
How has dynamic programming been used in exercise 2?

Similar to exercise 2, the students are confronted with skill number 5. In this exercise, students are required to explain what DP actually is by looking back at exercise 2. The teacher's guide (appendix F) also mentions that exercise two should be discussed with the teacher before students move on to exercise two to make sure that they understand the concept.

Thus, reading the definition of DP and then reflecting on how this has been used in the former exercise aids students with developing skill 5, since they have to know how this problem has been broken down into smaller problems.

After exercise 3, there is some information for students regarding the applications of DP. This is related to the third learning outcome of this lesson. The teacher's guide specifies that the teacher should talk about this with students. Explaining that this subject has many applications and is used widely can make sure that the students experience this exercise or lesson series as meaningful. Hence, this is also connected to the motivational aspect of the lesson series.

Other applications are comparing DNA-sequences and finding the shortest route. The latter example will be used to learn more about dynamic programming. This will be done in the coming two lessons.

To give the students some more context, the teacher can explain something more in detail about this text (as explained in the teachers guide). Furthermore, it gives the students information about the coming lesson, so they know what will be discussed.

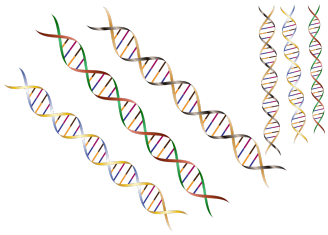


Figure 3: *Comparing DNA-sequences.*

Figure 4: *Finding the shortest route.*

Lesson 2: The routing problem
In the previous lesson, you have seen several applications of dynamic programming. One of these applications was the routing problem. An example of a route can be seen in Figure 5.

Suppose that this routing problem is a network of trains. The circles indicate **stations**. The starting station is station s_1. The arrows between the different stations show to which other stations you can travel. Additionally, the numbers next to the arrows indicate how long it takes to travel between these stations, which we call the *costs*.

For example: from station s_3 we can go to station s_5 with costs 5. Also, we could travel to station s_6 with costs 6 and to station s_7 with costs 1. The lower the costs, the shorter the travel time.

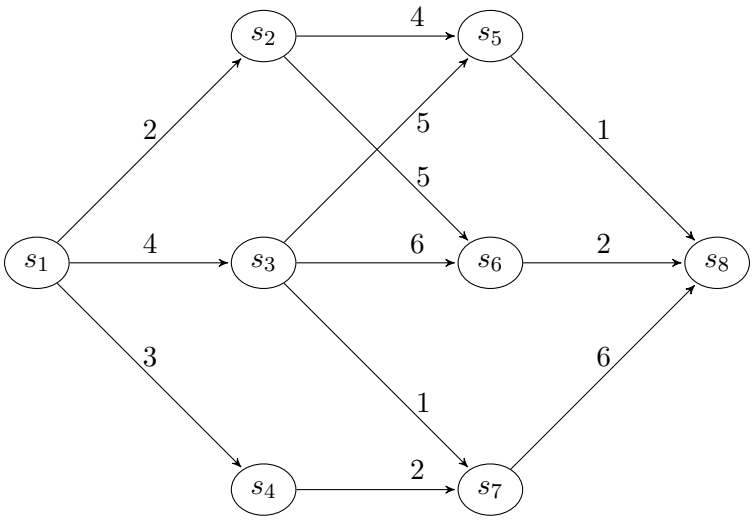


Figure 5: *An example of a routing problem.*

Exercise 1 -(M1)(M2)(M4)(C2)(C4)(C6)(C7)
The goal is to go from station s_1 to station s_8, with the lowest possible costs (meaning we want to travel as quickly as possible). Before doing this, we will first investigate why dynamic programming is useful here.

(a) When we apply dynamic programming, we can look for smaller sub-problems. For example, travelling from station s_3 to station s_8. Show with a calculation that the fastest route from station s_3 to station s_8 is by travelling via station s_5. How high are the costs?

(b) Why is it not necessary to compute the costs of the routes $(s_1) \to (s_3) \to (s_6) \to (s_8)$ and $(s_1) \to (s_3) \to (s_7) \to (s_8)$ to find out what the fastest route is from station s_1 to s_8?

(c) Compute the costs of the fastest route from station s_1 to station s_8 via station s_3. You can use question (a).

This exercise develops CT skill 5. It states that first a smaller problem is solved and then the larger problem. This is done in question (a). In question (b) students are asked to explain why computing the other routes is redundant. Thus, students are told what to do, but also are allowed to explore the theory themselves in a guided setting, adopting the theory of Windels (2012) and Freudenthal (2002).

Furthermore, since the students should compute the optimal route and are asked to explain why come computations are redundant, this exercise contributes to both the first and the second learning outcome of this lesson series.

We will continue doing this type of computations to find the fastest route. We want to come up with a well-structured approach. We do this as follows. In the figure, we already see some form of a structure with "groups", namely s_2, s_3 and s_4 are together and s_5, s_6 and s_7 as well. Therefore, we divide the stations in four stages:

- *Stage 1:* (s_1).
- *Stage 2:* (s_2), (s_3), (s_4).
- *Stage 3:* (s_5), (s_6), (s_7).
- *Stage 4:* (s_8).

We are going to find the solution by using dynamic programming. This means we start in *Stage 4*, thus we "start" at station s_8. The steps we go through in each stage are the following:

1. Check for each station what the fastest way is to go to station s_8 from the current station. The costs of the chosen route are the minimal costs.
2. Remember what the best choice was for the next station, so we remember to which station we have to travel next.

We will use the following table:

Stage	**Station**	**Possible routes to**	**Minimal costs**	**Next station?**
4	(s_8)	None.	0	None.
3	(s_5)	(s_8): Costs 1	1	(s_8)
	(s_6)	(s_8): Costs 2	2	(s_8)
	(s_7)	(s_8): Costs 6	6	(s_8)
2	(s_2)	(s_5): Costs $4+1=5$, (s_6): Costs $5+2=7$,	5	(s_5)
	(s_3)	(s_5): Costs $5+1=6$, (s_6): Costs $6+2=8$, (s_7): Costs $1+6=7$,	6	(s_5)
	(s_4)	(s_7): Costs $2+6=8$	8	(s_7)
1	(s_1)	(s_2) (s_3) (s_4)		

In the table, the following has been done. In *Stage 4* we do not choose a next station. The minimal costs to travel from station s_8 to station s_8 is equal to 0: we're already there.

Next, we look at all the station in *Stage 3*. For example, from station s_5 we can only travel to station s_8 with costs 1. Thus, the minimal costs to go from station s_5 to station s_8 are equal to 1. We can only go to station s_8, so this will also be the next station. This approach has also been done for the other station in *Stage 3*.

We repeat this process for *Stage 2* and *Stage 1*. We give an outline for the computation for station s_3. From station s_3 we can go to stations s_5, s_6 or station s_7. The costs to go from station s_3 to station s_8 via station s_5 are equal to the costs to go from station s_3 to station

s_5 (=5), plus the minimal costs to go from station s_5 to station s_8 (=1), what we can find in *Stage 3*. This results in a total costs of 6. We will also do this for the other stations and find that the minimal costs are equal to 6. So, from station s_3 we choose s_5 to be our next station, because this results in the lowest costs.

The explanation above gives information how DP is applied to the routing problem. In addition, an example is given on how to compute the minimal costs, which students will do in exercise two. This explanation is therefore related to requirement (M2).

Exercise 2 - (M1)(M2)(M4)(C2)
Our goal is to find the optimal route from station s_1. In *Stage 1* we start in station s_1. From here, we can go to station s_2, s_3 or s_4. Using the minimal costs from the stations in *Stage 2* we can find the optimal route.

(a) Compute the costs to go from station s_1 to station s_8 via all three stations in *Stage 2*. Write these values down in the table.

(b) If you did this correctly, one of these values is minimal. What does this value mean?

(c) Complete the table. What is the optimal route? Colour this route in Figure 5.

In this exercise, skill 6 is developed. Instead of doing the computations on their own answer sheets, the students are asked to solve the problem by filling in the given table. By translating the problem into a table/algorithm, the students come into contact with abstractions of the problem at hand.

This exercise also deals with finding an optimal route in a small routing problem and thus focuses on the first learning outcome of this lesson.

We now completed the table. Because we want to have the costs as low as possible, we kept choosing the route that gave us the minimal costs. In other words, we have computed the minimum over and over again. You already know how to compute the minimum of a function. This is what you will revise in the next exercise, and you will learn some new notation for this.

Exercise 3 - (M2)(M3)(M4)(C5)(C6)
Computing a minimum for a function f is something you have done quite often.

(a) Given the function $f(x) = \frac{1}{3}x^2 - 2x + 5$, for $x \geq 0$, with the following graph:

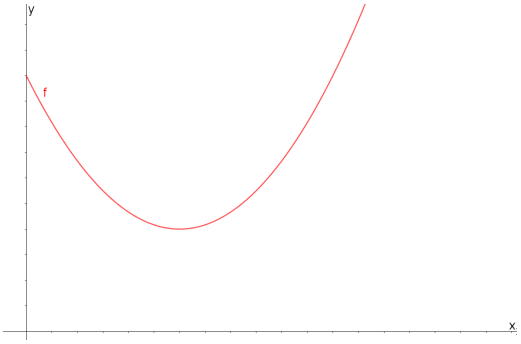


Figure 6: *The graph of f.*

The minimum is the minimal function value of $f(x)$. The x-value that is used as an input is called the argument. Show, using the derivative, that the argument $x = 3$ yields a minimum of 2.

This can be connected to the routing problem. In each stage of the routing problem, we choose a route for which the costs are minimal and in question (a) we "choose" an x-value for which $f(x)$ is a minimum. These are examples of finding a minimum, where a certain "choice" (the route or x-value) results in a minimum. We introduce two important concepts:

- We vary the x-value (the argument) to find the minimum of $f(x)$, which we write as $\arg\min_x\{f(x)\}$. The subscript x means that it is x that we vary. In our example this x-value equalled 3, thus we have $\arg\min_x\{f(x)\} = 3$.
- When you found this argument, you can compute the minimum. That minimum we write in a similar way: $\min_x\{f(x)\}$. The subscript x again means that we can vary the x-value. In the example of question (a) the minimum equalled 2 (by inserting $\arg\min_x\{f(x)\} = 3$), so here we have $\min_x\{f(x)\} = f(3) = 2$.

This notation is widely used in mathematics and is seen a lot in programming languages, in which larger networks of the routing problem can be solved. Go back to the table on page 95. We are going to practise with the notation we just learnt.

Take a look at station s_2 in *Stage 2* in the table. We already computed that the minimal costs are equal to 5, because if we were to choose between the costs of 5 and 7, we would choose 5 (we vary the station to which we can travel). The station we have chosen was station s_5 and not station s_6 (so the argument for these costs is station s_5). Therefore, we write $\arg\min_{\text{station}}\{costs(station)\} = s_5$. The minimal costs are then the costs we obtain when we travel to this station, which are equal to 5. When we use the notation in this example, we write it as $\min_{\text{station}}\{costs(station)\} = 5$ (so we take the minimum over the stations, with a minimal value of 5).

(b) Use this notation to find the minimum and the argument for the minimum for the other stations in *Stage 2*, so for stations s_3 and s_4.

(c) If you were to describe $f(x)$ and x in terms of costs and next station, what is $f(x)$ and what is x?

Similarly to exercise 2, this exercise also trains skill 6. The students are already familiar with finding minima and maxima by using the derivative of a function, as this is required pre-knowledge for this lesson series. It becomes more abstract as the notation for min and argmin are introduced to the students (which will be necessary for lesson 3), and thus skill 6 is used here.

By activating prior knowledge in question (a), we try to improve the setting for learning new concepts and give students the opportunity to add the new notation to their existing cognitive schema (Wetzels et al., 2011; Piaget, 1971). As the notation $\min_x\{\dots\}$ and $\arg\min_x\{\dots\}$ is introduced in this exercise, it corresponds to the last learning outcome of this lesson.

Lesson 3: The routing problem with Excel
In Figure 7 we see a new routing problem. Again, there are stations, called s_1 to s_27 and the arrows indicate possible routes. The numbers are not given next to the arrows, since then it will become confusing, because from each station in each stage there is a route to each station of the next stage.

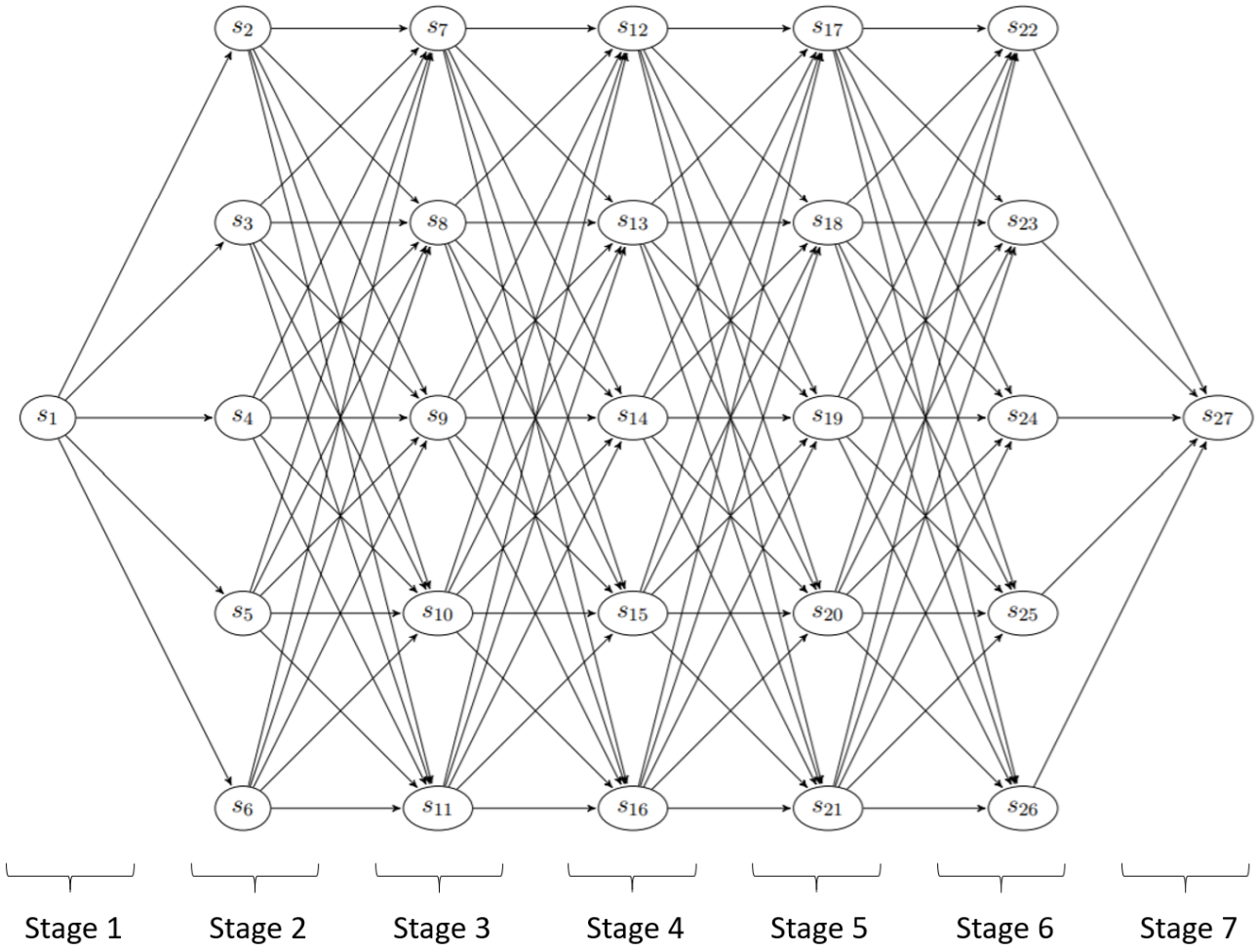


Figure 7: *An example of a (larger) routing problem.*

Exercise 1 -(M1)(M3)(M4)(C2)(C5)(C7)
In this exercise, we are going to look at the routing problem in Figure 7. If we want to know what the fastest route is from station s_1 to station s_2, then we have to compute the costs of the routes. However, before we do that, we would like to know something about the **number** of routes and with that, the number of computations.

(a) How many possible routes are there to travel from station s_1 to station s_{27}?

(b) Suppose an extra stage is added between stage 6 and 7 (so again 5 extra stations). How many **extra** routes do we obtain? And what if another stage of 5 stations is added?

(c) We assume we have the situation as in Figure 7. Suppose an extra stage of 5 stations is

added between stage 5 and 6. What is the influence on the number of computations if:

i. We compute the costs of **all** routes and then look for the minimum?

ii. We compute the fastest route with Dynamic Programming?

Give a conclusion about the efficiency of methods i. and ii. if we were to add 100 extra stages for example. Use the terms "linear relation" and "exponential relation".

In this exercise, skills 1, 2, 3 and 5 are developed. This is a very large problem for the students and can be challenging. Furthermore, this exercise gives the students some basic ideas on what computational complexity is, but without formally introducing it. The problem is broken into sub-problems to make it more approachable, for instance by only looking at what happens when a certain phase is added.

Question (a) is an easier exercise, as the students should know how to compute the number of routes using combinations. In question (b) and (c) students have to find out the computational advantages of using DP. By using routing network, realistic mathematical education is ensured (Freudenthal, 2002).

Since the number of computations and efficiency is discussed in this exercise, it deals with the second and the third learning outcome of the lesson.

Now we will make it more concrete. For each station, the costs are known to travel to the other stations in the next stage. The costs from station s_1 to station s_2 are equal to 7. Also the costs from station s_1 to the other stations are known. They are as follows:

- $s_1 \to s_2$: costs 7.
- $s_1 \to s_3$: costs 8.
- $s_1 \to s_4$: costs 9.
- $s_1 \to s_5$: costs 5.
- $s_1 \to s_6$: costs 8.

These values are also in the first column of Table 2. The values indicate the costs of going to the stations in the next stage, in the order from top to bottom (as depicted in Figure 7). We will first have a look at the other data in the table.

Table 2: *The distance between the different station of the next stage.*

Stage 1		**Stage 2**		**Stage 3**		**Stage 4**		**Stage 5**		**Stage 6**		**Stage 7**
s_1	7	s_2	6	s_7	5	s_{12}	5	s_{17}	10	s_{22}	6	s_{27}
	8		9		6		8		10			
	9		10		10		10		6			
	5		6		10		5		8			
	8		8		7		10		10			
		s_3	9	s_8	5	s_{13}	8	s_{18}	8	s_{23}	8	
			9		5		8		7			
			10		9		5		9			
			7		9		7		8			
			10		6		8		6			
		s_4	7	s_9	6	s_{14}	8	s_{19}	9	s_{24}	8	
			6		7		7		9			
			7		9		7		8			
			9		5		5		9			
			6		9		5		8			
		s_5	10	s_{10}	6	s_{15}	9	s_{20}	9	s_{25}	5	
			8		6		10		7			
			10		9		10		9			
			6		9		9		8			
			6		7		9		9			
		s_6	6	s_{11}	5	s_{16}	6	s_{21}	6	s_{26}	7	
			9		7		5		9			
			10		6		6		10			
			9		5		5		9			
			7		9		8		8			

The explanation and the table above are explained in detail. The aim of this explanation is to make students understand the table, so they are able to complete the next learning task. Therefore, this is related to requirement (M4).

Exercise 2 - (M3)(M4)(C6)(C7)
Take another look at Table 2. The rest of the table is constructed in the same way as the column in *Stage 1*.

(a) What does the value of 6 mean next to station s_2? And what does the 9 mean next to station s_2?

(b) Take a look at the column of *Stage 6*. Why is there only one value per station?

(c) Give an explanation why there is no value given at station s_{27}. If you were to give a value to station s_{27}, what value would that be (in other words, how long does it take to travel from station s_{27} to s_{27})?

Skill 8 is developed in this exercise. Students are asked to look at the data that is represented in a table, similarly to the last lesson. In question (b) and (c), students are asked why the data is represented in a different way than in the rest of the columns. Students will encounter that

each form of representing data has strengths and weaknesses, as the table is compact but one should understand the problem to understand how the data is represented in the table. This is important to understand, as students will need this table in both exercise three as in the final assignment of this lesson.

As this question is a start for the coming exercises in the lesson series, it will help the students to achieve the learning outcomes of the next two exercises, which will be described below.

To solve this, we could write down everything on paper. Even though we tackle this problem with dynamic programming, it will be a lot of work to do these computations by hand. Fortunately, we can use computer programs to help us. We are going to solve this problem with the use of Excel.

Exercise 3 - (M3)(M4)(C6)(C7)
Open the file RoutingproblemLesson3.xl on your computer. In this file, a similar table is given to Table 2. The distance from station s_{27} to s_{27} is set to 0. In this file, the stations are called 1, 2, 3, ... instead of $s_1, s_2, s_3, \ldots$.

Beneath the table, there are 6 coloured blocks/tables with the following data:

- Stations: below this are the stations of the stage above.
- k_station: these are the minimal costs to travel to the final station (27), from that station.
- Best route: this says which route should be chosen next.

Using dynamic programming, we start at the end of the problem. That end is simple, since at station s_{27} we're already there. Then we work backwards, in which we do the computations per (coloured) block. The first two blocks are coloured red: they have not been filled out yet. The other four have been completed already. It is your job to find the values for the red blocks. If you succeed, beneath the text "The optimal route is then given by:" a sequence of stations will appear that ensures that the costs are minimal. Answer the following question:

Question: Take a look at station s_{25} in the last block. What do the 5 and 27 mean? And if you look at station s_{21}, what do the 12 and the 22 mean?

Exercise three is a short exercise and is a brief introduction for the student to the Excel file. The students receive some information on what is meant in the Excel file, which is closely related to the table they have seen before and therefore, they can relate this to what they already know.

As it is an introduction to the final assignment, this exercise will help students to achieve the learning outcomes of the final assignment, which will be described below.

Explanation Excel (needed for the final assignment)
We focus on stage 2, which we would like to complete. The best routes are still unknown, so they are set to 0 (we don't know them yet). We use the minimum from stage 3 to find out which station in stage 2 gives us the lowest value to the final station, which requires the exact same approach we used in the previous lesson. In Excel we can do this with the function MIN. You will do this yourselves as well, but first read the instructions below. You will need to compute

the minimum of the sum of two columns (which will come back in the final assignment). How to do this is explained below:

1. Suppose you have the numbers in the cells $A1$ to $A4$ and in the cells $C6$ to $C9$, see Figure 8:

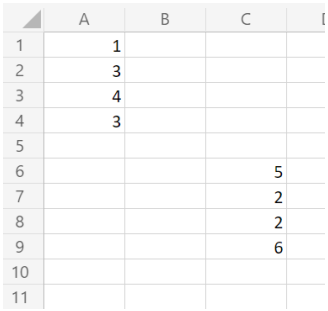


Figure 8: *The example to use the function* MIN.

2. Select an empty cell in which you want to have the minimal value. In our example we take $E1$. We select the cell $E1$ and type =min($A1$:$A4$+$C6$:$C9$). Excel now adds the values of the first row of both columns ($A1 + C6$), adds the values of the second row of both columns ($A2 + C7$), and so on. Then, Excel computes the minimum, which is 5 (the sum of the numbers in the second row, 3+2). See the figure below.

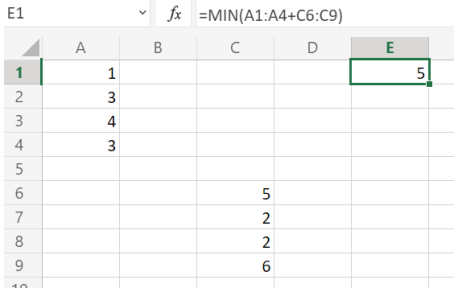


Figure 9: *The answer for using the function* MIN.

As students are not required to have any knowledge about the functions in Excel, the explanation above tries to give a clear view on how to compute the minimum of the sum of two columns (needed for the final assignment). The MIN function in Excel is not restricted to the sum of two columns; however for the Exercise this is the only information the students will need. Other aspects of Excel are therefore not considered.

Final assignment (Eindopdracht)(M1)(M2)(M3)(M4)(M5)(C3)(C5)(C6)(C7)
In the final assignment, you are going to compute the costs of the fastest route to go from station s_1 to station s_{27}. Use the file from the former exercise (RoutingproblemLesson3.xl).

(a) First compute the optimal costs from station s_2 to station s_{27}. Use the approach of the previous lesson. You will need to use the explanation on how to compute a minimum in Excel. The best route from station s_2 appears as well, which is station s_7. *Hint: you have to compute k_station for station* s_2.

(b) Do the same for the other stations in stage 2. Give the lowest costs for each station to travel to station s_{27}. When you did this correctly, the best routes appear for each station in stage 2. *Hint: if a 0 remains below Best route, then you have not found the optimal value yet.*

(c) Finally, what are the minimal costs to travel from station s_1 to station s_{27}? Give the optimal route by filling in the sequence below:

$(s_1) \rightarrow \ldots \rightarrow \ldots \rightarrow \ldots \rightarrow \ldots \rightarrow \ldots \rightarrow (s_{27})$

In the final assignment, skills 2, 5 and 7 are developed. As this is a difficult exercise in which all the learnt theory should be used, students should show persistence in working through this problem. Furthermore, using DP the students have to break up the problem into easier problems, which they have to use in Excel. This is what they have done before using the table in the second lesson.

The students are both guided with the hints and the explanation above, but also have to explore Excel and the knowledge they have about the routing problem to learn how to find the optimal solution. This improves the way students learn the theory (Wetzels et al., 2011). Furthermore, in this exercise, students are allowed to use Excel and get into contact with a simplified version of computer programming, which is useful for learning how DP works (Bockenhauer et al., 2019).

In combination with the theory above the exercise, the final assignment focuses on the first learning outcome of the lesson series as the students have to use the function MIN in Excel to find the optimal route.

This is the end of the lesson series.

5.2 Feedback Walk-through

In this section, the most important points of the feedback are discussed. Furthermore, the reasons why or why not the feedback has been implemented to improve the quality of the lesson series. The feedback from the expert panel can be found in Appendix A. The textual remarks that have been resolved can be found in Appendix A.5. Note that this feedback has been used to obtain the lesson series that has been given in the former section. This feedback has been given on the first draft of the lesson series, which can be found in appendix B

Design requirements
In general, few comments were given on the design requirements. This indicates that, overall, the design meets the requirements. However, one of the experts mentioned that requirement (C7) has room for improvement. The expert's proposed solution is to add a homework exercise. In this exercise, the students should be given an example in which DP can be applied. The students can then study this example in depth, such that they indeed explore the theory.

However, there is little time for students to complete a homework exercise. As the requirement (C7) is already present in other exercises in the lesson series, the choice is made to not add this extra exercise. If this lesson series were to be extended, adding the proposed exercise is a suitable option to enlarge the presence of requirement (C7) in the lesson series.

General remarks
The remarks will be discussed per lesson, since this format is also used by the expert panel. Afterwards, remarks that do not specifically pertain to a lesson will be covered.

Lesson 1
Remarks for improvement have been given regarding exercise three. The question about advantages and disadvantages of DP was criticised. Some members of the panel expressed concerns about the difficulty of the exercise, as they were uncertain whether students would be able to complete the exercise.

A possible solution to this problem was proposed by a different expert. Perhaps students could be given examples and argue why or why not DP would be useful. However, since students know very little about DP, this will also be challenging. Therefore, the exercise has been changed such that students only have to think about how DP was applied in exercise two. The role of the teacher will be to shed some light on other problems, e.g. DNA-sequences and routing problems. The teacher should mention how DP could be applied in those cases, without going into the details. This way, students will still see some examples of DP (in line with requirements (M1) and (C2)).

For the last exercise of this lesson, one of the experts mentioned that it is unclear how a strategy was found. However, the exercise mentions that this has just been discussed. This is a good point, but the expert panel has only seen the exercises. The teacher's guide (including keys and guidelines) has not been shared with them. In this guide, the teacher is advised to discuss the strategy that has been found by the students in exercise two. Hence, by providing the teacher's guide, the raised issue has been resolved.

Lesson 2
According to one of the experts, a specific link should be made to make it clear that the lower the travel time is, the lower the costs are. To ensure that this is clear to students, this specific link has been added to the text.

Another remark has been made about being specific in the text. One of the experts believed that it should specifically stated that solving a problem with DP implies starting at the end of the problem. Even though it was stated that the students have to start in *Stage 4*, it was not clear enough for the expert. If it is not clear to an expert, it is likely that it is also not clear for students. Therefore, it is also specified that the students start in *Stage 4*, thus at the end of the problem (station s_8).

For the first question, a remark is made to change the structure of the exercise: students could solve the problem step by step and then find out that it might be useful to insert stages. This is a different approach to obtain relational understanding of the topic, since currently the focus lies on *why* the same steps can be repeated and thus why DP is suitable. If more time is available in a lesson series, this would be a suitable approach to teach the students why applying stages is useful for solving the problem with DP. As the exercise as it is already focuses on relational understanding, it has not been changed.

Another suggestion to improve the first exercise is to include answers in the exercises. This is not necessary, since the supervising teacher can either guide the students or give the answer. It is important that the students think about the right approach rather than finding the correct

answer, as is in line with relational understanding (Skemp, 2006). Therefore, the answers are not given to the students, but they are included in the teacher's guide.

A different aspect that can be incorporated in lesson two is that the costs and the route can be related to the notation of $f(x)$ and x respectively. This can be done in the last exercise by extending it. Giving a complete description of $f(x)$ and x is not mandatory, a simple and intuitive explanation is sufficient. Since the new notation that is introduced in exercise three will be used in the last lesson of this series, exercise three will remain at the end of lesson two, which goes against the suggestion by one of the experts.

A final remark on exercise three is that no specification is given on how students should solve the problem in question (a). The problem can be solved by using the derivative or by using $x_{\min} = -\frac{b}{2a}$. Both options are suitable, however, using the derivative reflects the idea that students have to "search" for the argument, instead of using a simple formula for this. Because of this, it will be specified that students have to use the derivative. Additionally, this follows design requirements (M2) and (M4), as specifying the approach makes the instructions clear.

Lesson 3
Regarding the first exercise, different numbers of stations per stage could be given (some stages with more stations, some with less). This is an interesting suggestion, because it requires students to delve deeper into the computational complexity of the problem. Since this is the first time that students encounter this and the students should be able to connect this easily to the theory of the last lesson (in which also the number of stations per stage were equal), this has not been changed in the second iteration.

For the third exercise, a suggestion was given to change the names of the station s_1 to A, s_2 to B, and so on. Using indices is standard in mathematics and as it is important that students can work with this notation, it has not been changed.

Other
Remarks that were not specifically directed towards a lesson or exercise are discussed below.

An expert stated that graph theory could be included in this lesson series. This is already done implicitly, since the routing network is a directed graph. However, exact definitions and questions relating to graph theory are not incorporated. As there is not enough time in the lesson series to give more attention to graph theory, this has not been added.

Regarding the time schedule of the lessons, one expert mentioned that each lesson can be conducted within fifty minutes. Furthermore, this expert stated that the last lesson is the longest of the three. This is virtually the same as was stated in the teacher's guide in appendix F. Therefore, the exercises will not be extended/shortened significantly, except for the minor changes mentioned above.

Finally, one of the teachers does not understand why the students have to find a minimum of a quadratic function first and afterwards a minimum using Excel. This has been done to activate students' prior knowledge and to give an introduction about new notations and definitions. Even though this is not clear at first sight, both the teacher's guide and the substantiation of the design help to clarify these reasons to teachers.

5.3 Testing

In this section, the students' results are discussed, entailing both the answers to the exercises and the answers from the interviews. As the students worked together on the exercises, only one answer sheet is available.

5.3.1 Execution

The test was done with two students, as specified in chapter 4. All three lessons were done on the same day due to logistical circumstances. The researcher supervised the process.

The students did not ask questions on how to solve the exercises. They read the theory separately, then discussed the question. When they agreed on the correct answer, one student wrote those down (see Appendix C).

Below, the time it took the students to read the theory and complete the exercises is given:

- Lesson 1: 17 minutes.
- Lesson 2: 34 minutes.
- Lesson 3: 34 minutes.
- Total completion time: 85 minutes.

Note that the setup had been done before the lessons started; the matches for exercise one of lesson one had already been counted and the Excel file for lesson three was opened on a computer.

5.3.2 Answers to exercises

In this section, the students' answers to the exercises (see Appendix C) are analysed.

Lesson 1 - Exercise 2
The goal of this exercise was to find a winning strategy for the starting player A in the match-game. The students gave a correct strategy, including the use of the variable x to indicate the amount of matches taken by player B. However, a specification that steps 2 and 3 should be repeated until the situation in step 4 is obtained was missing. Despite this being an incomplete algorithm because this recursive step not was specified explicitly, the students did have the correct idea for the winning strategy.

Lesson 1 - Exercise 3
The students correctly stated that they started at the end of the problem, which is indeed how DP is used. They explain that they use the solution of the end of the problem to find the winning strategy. This explanation is in the right direction. However, key words such as "solving backwards" or "step by step" would have been indications that students understand what DP is and how it has been used in the previous exercise.

Lesson 2 - Exercise 1
The students computed the correct values for the routes in question (a). Furthermore, they managed to give a reason why certain routes can be ignored to find the optimal route in question (b). Then they use this information in exercise (c) successfully, computing the optimal route from s_1 to s_8 via s_3.

Lesson 2 - Exercise 2
In all three subquestions, the students managed to give the correct answers. The table is filled in correctly, they explained that the lowest value indicated the shortest route and coloured the correct route in the figure.

Lesson 2 - Exercise 3
In the first question, the students computed the correct derivative and used that to show that there is indeed a minimum at $x = 2$ with a value of 3. Since the graph of the function was given, this is enough to say that it is a minimum. In question (b), the students correctly gave the notation for the minimum. However, it was also required to use the notation for the argument, which they did not do. Then in question (c), they correctly linked the costs of the route to the function value f and the choice of the station to the argument x.

Lesson 3 - Exercise 1
In question (a), they found 5^{21} routes, without an explanation. This value is higher than the correct answer. In the same way, question (b) shows higher values than expected. Furthermore, the computation of the number of extra routes is incorrect. In question (c), they correctly said that for method i the number of options increases exponentially, whereas in method ii it is a linear increase. The argumentation for method ii is not clear since the reason for the linear increase is not indicated. In the explanation for method i the students claim that the minimum will be higher, which is incorrect.

Lesson 3 - Exercise 2
All three subquestions have been answered correctly. The students found the correct interpretations for the values in the table and also managed to explain that the stations from the last stage only had one corresponding value in the column next to the stage. Finally, they correctly stated that the value of the last station should be zero, since there are no costs to travel to that station.

Lesson 3 - Exercise 3
This exercise has been answered correctly, as they managed to explain the meaning of the values in the Excel file.

Lesson 3 - Final assignment
In the final assignment, the students give the correct answers. They managed to use the correct formulas in Excel and to find the optimal route, which is given in their Excel file and on their solution paper.

5.3.3 Interviews

In this section, the interviews regarding the design requirements are discussed (for the design requirements, see chapter 3). Any additional remarks will be discussed at the end of this section. Student 1 and student 2 are referred to as S1 and S2, respectively. To ensure anonymity, the students are referred to as "they". The transcription of the interviews (in Dutch) can be found in Appendix D.

Motivation

Requirement (M1)

The students mentioned that the examples made the lesson series clear and lively. The applications, such as DNA-technology and routes, were interesting (S1). Furthermore, playing the game was also thought to be useful to improve their understanding and to obtain a complete picture (S2).

Requirement (M2)

Regarding the examples and explanations, S1 said that the explanations were very clear in the lesson series. S2 also thought that the theory was explained in a clear way. In addition, S2 seems to imply that the preceding text and theory was general, but that it could be applied to the exercises that followed.

Requirement (M3)

S1 stated that they had enough prior knowledge to complete the lesson series, even though it is a different branch of mathematics that is not assessed frequently. S2 also stated that they had enough prior knowledge and that it was accessible to start with the lesson series, even for students without advanced mathematics class (*Dutch: wiskunde D*). However, to complete the exercise with the derivative, one needs to know how to compute that.

Requirement (M4)

Both students thought that they had clear instructions on what to do in learning tasks. S2 added that sometimes, they had to guess in which direction they had to think. An example in which the student encountered this, was the first exercise of lesson three. This student explained that this is due to a lack of knowledge regarding tree diagrams.

Requirement (M5)

S1 felt that the type of activities and exercises was varied, since they start with the match-game. This showed the student how to apply mathematics to something in practice, in various situations. S2 believed there was variation in the exercises, but also that there were similarities between different exercises in which only the formulation was different.

Comprehension

Requirement (C1)

Both students managed to explain that for DP, one starts with a (small) part or at the end and from that solves the entire problem back to the beginning. S1 called it a complex system, rather than an approach or method.

Requirement (C2)

S1 referenced several applications of DP, for example DNA-technology (instead of indicating it is about comparing DNA strings). Also the match-game was mentioned by this student. S2

mentioned as applications to go to Amsterdam and to find out which route is the shortest. Also, the student remembered something about DNA.

Requirement (C3)
Both students mentioned that the software Excel has been used to solve an exercise. S1 said it helped to understand their way of thinking, since it confirmed if they were on the right track. S2, on the other hand, said that the theory preceding the final assignment was more clarifying than Excel, partially due to the lack of prior knowledge in Excel. S2 mentioned that as soon as one understands Excel, it can be useful to understand the theory.

Requirement (C4)
For this requirement, S1 correctly mentioned that using DP for the routing problem, one starts at the end and therefore, the longer routes can be ignored. This yields less options, it is faster to find the optimal solution compared to the "normal" method, since then also the longer routes are taken into account. S2 had to think a bit more about this question. However, the correct answer was given that in the first step, longer routes can be cut off from the problem.

Requirement (C5)
Both students stated that they think they will manage to solve the routing problem when given a small routing problem similar to the one from lesson two.

Requirement (C6)
S1 stated that in lesson two and three, exercises were given in which they were lead to the answer step by step by using subquestions. S2 also stated that these questions were present.

Requirement (C7)
For the last requirement, the students give different answers. S1 thought that the exercises were not extremely difficult, solely that a few were more time consuming than others (for example exercises with Excel). However, in the start of the lesson series they had to think longer about the exercises, since it is a completely new system that they work with. S2 immediately referred to the final assignment, which was not explained step by step. Furthermore, this student thought that the first question in lesson two was challenging. After reading it twice, it became clear what the student had to do.

Final remarks
Both students stated that the lesson series was put together nicely. S2 also remarks that they now understand what DP entails and what the advantages are. S2 did mention that it was convenient to have a teacher there to help the students, since it would be more difficult to go through it alone with this new topic. However, it should be possible according to this student, since everything was described well.

5.4 Overview of the results

A short overview of the results can be found in the table below. The table indicates for each requirement if it is present, according to the experts, the students' answers and the student interviews. The entry '+' indicates it is present, '-' indicates it is not and 'o' indicates that it is partially present. Finally, a missing entry indicates that the results do not provide an immediate score. After analysing and interpreting the results, this will be changed (see chapter 6).

Table 3: *Scores for each requirement.*

Requirement	**Experts**	**Student answers**	**Student interviews**
M1		+	+
M2			+
M3	+		+
M4			o
M5	+		o
C1		+	o
C2			+
C3	+	+	+
C4			
C5		+	
C6			+
C7	-		+

6 Conclusion and discussion

Below, the results are analysed. Then, section 6.2 describes the conclusions of the results. Afterwards, section 6.3 contains the discussion on the limitations and implications of this research.

6.1 Analysis of the results

The results regarding time, learning outcomes and design requirements will be analysed below. Similar to section 5.4, a table is given at the end of this section to provide an overview of this analysis.

Time
The time for the lessons was set on approximately fifty minutes per lesson. The students completed the lessons within the given time frame (they finished all lessons early). Since only two students were present and the preparation was already done for them (e.g. Excel was set up on a computer), time was saved. For a class of thirty students, the lessons are expected to take more time and therefore, lesson two and three are of adequate length. Since the students only took seventeen minutes for the first lesson instead of fifty, lesson one could be extended. An example to extend the first lesson is given in the paragraphs on the design requirements.

Learning outcomes
The students managed to find a winning strategy for the match-game using DP, as can be seen from their answers. Furthermore, in the interviews the students were able to explain the basic idea of DP and give examples of its applications. Thus, the learning outcomes of lesson one have been achieved.

The students were able to find the optimal route in a small routing problem using DP. In the interview, they indicated that they could solve a routing problem similar to the one of lesson two. Additionally, in exercise one of lesson two, the students explained why some routes do not have to be considered. They elaborated on this in the interviews. Hence, the first two learning outcomes of lesson two have been achieved. For the last learning outcome, the students were able to use the notation $\min_x\{\ldots\}$ correctly, but they did not use the notation $\text{argmin}_x\{\ldots\}$. Hence, it has not been verified if the students achieved the last learning outcome of this lesson. However, as they could link f to the value of the costs and x to the chosen routes in exercise three of lesson two and did not ask questions about the theory when the lessons were conducted, it seems they do understand the notation for the minimum and for the argument.

In the final assignment, the students correctly used the MIN function in Excel to find the minimum of the sum of two columns. The students did not manage to give a clear estimation of the number of computations that should be done and did also not explain how that number changes when the size of the routing problem changes. Even though they understood the basic concepts why DP is much faster, they were not able to give valid arguments for making a link with the concepts of linear and exponential growth. Therefore, only the first learning outcome has been achieved in this lesson.

Design requirements
The design requirements will be discussed. The scores in table 3 are extended by interpreting the results, which allows more entries to be examined. This extended table is given in table 4 (see page 47). The argumentation for the extension is given below.

Requirement (M1)
The experts do not mention the applications specifically. However, the students show in their solutions that they worked with several applications of DP (e.g. in the match-game and in the routing problem). Furthermore, in the interviews, the students were able to mention several applications from the lesson series. Hence, this requirement is present sufficiently.

Requirement (M2)
The feedback from students' answers does not give information regarding this requirement. In the interviews, however, the students mentioned that the theory was explained in a clear way and that they knew how to apply that in the exercises afterwards. The experts also mentioned that the explanation was very clear. Thus, this requirement is achieved.

Requirement (M3)
The experts mentioned that the exercises in lesson one are suitable to activate prior knowledge. The students also indicated that they had enough prior knowledge to complete the exercises. On the contrary, one of the students did mention they had trouble with the first exercise of lesson three, in which they had to compute the number of ways to go from s_1 to s_{27}. This is also observed in the students' answers, since they did not find the correct answer to this exercise.

Even though the experts were positive about this requirement, it still has to be improved based on the students' findings. This can be done by, for example, including an extra exercise in which students have to reduce the routing problem to a road diagram for finding the number of routes. Finding the number of possibilities in a road diagram is taught in lower secondary education (of the target group). Since there is room for an extra exercise in lesson one, an exercise on road diagrams can be implemented here. Then in lesson three, using Guided Reinvention (Freudenthal, 2002), the students can reduce the routing problem to a road diagram for the computation of the number of possible routes.

A student mentioned that knowing how to compute a derivative is necessary, which is indeed a prerequisite for this lesson series.

Requirement (M4)
The students mentioned that they understood what to do in the exercises for the most part. In exercise three, one of the students stated that they had to guess in which direction they had to think. According to this student, this was caused by a lack in knowledge regarding tree diagrams (even though tree diagrams is not the correct terminology). This relates to requirement (M3), since prior knowledge on this topic is useful. A different possibility to ensure students are able to complete the exercise correctly, is to state the correct answer for example. That way, students can verify if their solution is correct or try to derive how to obtain the correct answer.

In general, the tasks were explained in a clear way. However, there is room for improvement as students experienced "guessing" the correct direction. Thus, this requirement has both positive and negative remarks.

Requirement (M5)
The experts agree that there is a variety in learning tasks (e.g. playing a game, using Excel). This is also observed in the exercises of the students. In the interview, one student agrees fully that there is a variety of learning tasks, whereas the other student agrees partially. The latter student stated that some of the exercises seemed a bit similar. The fact that there are similar exercises (for example explaining an entry in a table or in Excel in lesson three) does not imply that the exercises are unvaried. Therefore, this requirement has been achieved.

Requirement (C1)
The students were able to give ideas how DP was applied in games and in the routing network (from both the answers to the exercises and the interviews). Even though a student called it a complex system, the students seemed to understand the basic concepts of DP. Thus, the students have been introduced to DP.

Requirement (C2)
The students worked with several examples in the exercises and were able to give applications where DP was applied. Since they were able to work with different applications of DP and were able to give applications, they have been exposed to several examples of DP in this lesson series.

Requirement (C3)
The experts, the answers to the exercises and the interviews with the students all show that the students indeed worked with computer software to solve a problem. The students also indicated that it helped them to understand the theory better. Thus, this requirement is achieved.

Requirement (C4)
The students were able to explain advantages of DP in the application they worked on the most (the routing problem). In the exercises as well as in the interviews, the students noted that long sub-routes are discarded, since they will not be optimal. Hence the students managed to give advantages of DP.

Requirement (C5)
The students did all the exercises, in which several CT skills are used (see table 1). Furthermore, the students mentioned they are able to work with similar problems in which they have to apply DP to a routing network. Based on the fact that they completed the exercises which were designed for CT skills, the requirement is achieved.

Requirement (C6)
From the students' answers, it can be observed that they indeed do exercises step by step. Furthermore, in the interviews the students agreed that some of the exercises led them to the final answer in steps. Hence, guiding exercises are present in the lesson series.

Requirement (C7)
According to the experts, students should be given more room to explore in the lesson series. One of the students explained that it was not difficult, but that some exercises simply required more work. On the other hand, this student indicated that it was more difficult in the beginning since they had to work with a new concept. The other student explained that the final assignment was an example of an exercise in which they were not guided step by step. This implies that there are exercises for students to explore the theory, but that the amount of these

exercises could be increased.

The table below is set up in the same way as table 3. Table 4 has an extra column in which the conclusion is given regarding the presence of this design requirement. Two of the entries have been changed after the interpretation of the results. This concerns requirements (M3) and (C1).

For (M3), the entry for the student interviews has been change from '+' to '-'. This has been done since the students indicated that they had enough prior knowledge to complete the lesson series and that it is accessible. However, one student mentioned they did not know a lot about tree diagrams (even though the student meant road diagrams). That statement actually implies that they did not have enough prior knowledge to complete the lesson series (which is also observed in their solution of the corresponding exercise).

For (C1), the entry for the student interviews change from 'o' to '+'. One of the students used wrong terminology to explain what dynamic programming is, which resulted in the score of 'o' as the other student did give a correct explanation. However, the students showed understanding of basic DP concepts (saying that we can use smaller parts of the problem to understand the larger problem). Hence, this entry has been adjusted to a positive score.

Table 4: *Scores for each requirement, based on the conclusion.*

Requirement	Experts	Student answers	Student interviews	Conclusion
M1		+	+	+
M2	+		+	+
M3	+	-	-	-
M4		o	o	o
M5	+	+	o	+
C1		+	+	+
C2		+	+	+
C3	+	+	+	+
C4		+	+	+
C5		+	+	+
C6		+	+	+
C7	-		+	o

6.2 Conclusion

In this section, the conclusions on the research goals will given. The research goals were the following:

(I) The lesson series should motivate students to learn about DP and CT.

(II) The lesson series should improve the mathematical knowledge/skills of students with respect to DP and CT.

The conclusions from table 4 are used. Regarding motivation, requirements (M3) and (M4) should be improved. Activating prior knowledge stimulates students to be motivated according to Ebbens (2015) (requirement (M3)). Clear instructions (M5) are also important for students

to be motivated (Teitler, 2017). Ideas to improve these requirements have already been given in the text above. Taking into account that the students and the experts gave a positive review and that the overall score for motivation in table 4 are positive, the first research goal is close to being achieved. Minor changes to improve (M3) and (M4) will result in the accomplishment of the motivational research goal. This is also based on the positive reactions from the students and the expert panel.

Regarding comprehension, more attention should be given to exercises in which students have room to explore (requirement (C7)), which is based on the feedback from one of the experts. Exploring the theory enables students to come up with their own ideas and concepts to understand mathematics (Windels, 2012). A possible way to improve this has been proposed by the same expert that gave this feedback (see section 5.2). The remaining design requirements are present according to table 4. Students showed basic understanding of DP and its applications. Together with the fact that several CT skills have been practised, the second research goal has been accomplished.

6.3 Discussion

In the discussion, the limitations and implications of this research are discussed. Afterwards, ideas for further research are given.

Limitations
The limitations of this research are now discussed, regarding the procedure, the respondents and the instruments.

The first design of the lesson series has been presented to a panel of experts. These experts gave feedback to improve the lesson series, which resulted in a new iteration. A similar approach could have been done with the feedback from the students. Due to time constraints, it was not possible to create a new iteration of the design. Taking the students' feedback into account for a new iteration would improve the lesson series. Two points that could solve the students' feedback:

- The students had trouble with finding the number of possible routes in the routing problem of lesson three. An exercise could be added that shows the students how to compute this, as from the interview and the answers it could be concluded that the students did not know how to tackle this problem.
- The students had to guess in which direction to think, so requirement (M4) could be improved by giving the answers in the exercise. This could give the students more insight in how to tackle the problem.

Another limitation is that the three lessons have been taught in one afternoon. This way, the lesson series is treated as a whole rather than three separate lessons. The concepts that the students learn are needed for the next lessons. The concepts/skills the students learned in lesson one are still fresh in their mind when starting lesson two or three. This might not be the case when the lessons are conducted on different days. A new test could be done, in which there is more time between the lessons, to see how well the students remembered the concepts from preceding lessons.

Due to the current pandemic, it has been difficult to find a group of students to test the lesson series. Two students are not enough to give a complete view of the quality of the lesson series. Additionally, the participating students take advanced mathematics (*Dutch: wiskunde D*). A suitable group for a new test would be a normal class size (± thirty students) that only take mathematics for scientific studies (*Dutch: wiskunde B*), since the lesson series should also be suitable for these students.

Implications

As CT becomes more important in secondary education (Timmer and Tolboom, 2019), the development of lesson material for this subject is important. Several CT skills (with the taxonomy of Weintrop et al. (2016)) are developed throughout this lesson series. By means of DP, these CT skills are practised. The development of these skills can improve the mathematical processes of reasoning and problem solving (Calao et al., 2015).

The guidelines for the design are motivation and comprehension. The motivational factors that are considered are meaningfulness, confidence and clarity (Marzano, 2018). Comprehension is based on different views. Specifically for DP the ideas of Anderle et al. (2019) are used. Besides doing theoretical problems, this lesson series also gives students the opportunity to solve problems using Excel, for which virtually no prerequisite knowledge is needed.

This lesson series could be adopted by a teacher who wants to provide extracurricular material for high-achieving students. It is possible to use the lessons as three separate lessons, but it could also be taught in one single afternoon. Hence, the lesson series can be used with flexibility. The solution manual with some guidelines on how to teach the lesson series makes it accessible to teachers and easy to prepare. By providing the lesson series in two languages, it is suitable for both Dutch as well as international education.

Further research

Further research is divided into two parts: ideas to improve the quality of this lesson series and ideas to develop more material regarding CT.

To improve the quality of this lesson series, the students' feedback could be used to create a new iteration. One of the possibilities for this is to create a new exercise for lesson one that follows requirement (C7): an exercise that gives room for students to explore the theory. Then, the lesson series can be tested again on a larger scale; a group of thirty students that take mathematics for scientific studies. The same approach can be used to analyse their results (their answers and interviews), leading to an improved version of the lesson series.

Furthermore, the lesson series can be tested as three completely separate lessons to obtain more information on how well students are able to use the concepts of the preceding lessons when they are not done on the same day.

One of the experts mentioned an interesting topic that is closely related to the routing problem: graph theory. Graphs can be used to present networks, but it is not restricted to the routing problem. This topic might offer interesting problems suitable for students as they are practical, e.g. the matching problem or network flows.

References

Activiteiten Inspectie van het Onderwijs (2019). Jaarwerkplan 2010.

Aho, A. (1974). *The design and analysis of computer algorithms*. Addison-Wesley Pub. Co, Reading, Mass.

Anderle, M., Forisek, M., et al. (2019). Teaching recursion and dynamic programming before college. *Bulletin of EATCS*, 3(129).

Bandura, A. (1977). Self-efficacy: toward a unifying theory of behavioral change. *Psychological review*, 84(2):191.

Berlekamp, E. R., Conway, J. H., and Guy, R. K. (2018). *Winning Ways for Your Mathematical Plays, Volume 3*, volume 3. CRC Press.

Bockenhauer, H.-J., Kohn, T., Komm, D., Serafini, G., et al. (2019). An elementary approach towards teaching dynamic programming. *Bulletin of EATCS*, 2(128).

Calao, L. A., Moreno-León, J., Correa, H. E., and Robles, G. (2015). Developing mathematical thinking with scratch. In *Design for teaching and learning in a networked world*, pages 17–27. Springer.

Curriculum.nu (2019). Toelichting Rekenen & Wiskunde. *Samen bouwen aan het primair en voortgezet onderwijs van morgen*.

Donders, W. and Ruijs, L. (2013). *Praktische gespreksvoering*. Boom Lemma uitgevers.

Drijvers, P., van Streun, A., and Zwaneveld, B. (2019). *Handboek wiskundedidactiek*. Epsilon Uitgaven.

Dweck, C. S. (1986). Motivational processes affecting learning. *American psychologist*, 41(10):1040.

Ebbens, S. (2015). *Effectief leren: basisboek*. Noordhoff Uitgevers, Groningen.

Examenblad (2020a). Examenprogramma informatica havo/vwo. URL https://www.examenblad.nl/examen/informatica-vwo/2021?topparent=vg41h1h4i9qe [accessed 24-12-2020].

Examenblad (2020b). Examenprogramma wiskunde B vwo. URL https://www.examenblad.nl/examen/wiskunde-b-vwo-2/2021 [accessed 24-12-2020].

Freudenthal, H. (2002). *Revisiting Mathematics Education - China Lectures*. Kluwer academic publishers.

Kong, S.-C. (2016). A framework of curriculum design for computational thinking development in k-12 education. *Journal of Computers in Education*, 3(4):377–394.

Kong, S.-C. (2019). *Computational thinking education*. Springer, Singapore.

Marzano, R. (2018). *Leren in vijf dimensies: moderne didactiek voor het voortgezet onderwijs*. Koninklijke Van Gorcum, Assen.

Piaget, J. (1971). *The Epigenetic System and the Development of Cognitive Functions*. Edingburgh University Press.

Powell, W. B. (2007). *Approximate Dynamic Programming: Solving the curses of dimensionality*, volume 703. John Wiley & Sons.

Puterman, M. (1994). *Markov decision processes: discrete stochastic dynamic programming*. Wiley-Interscience, Hoboken, N.J. Great Britain.

Simpkins, S., Davis-Kean, P., and Eccles, J. (2006). Math and science motivation: A longitudinal examination of the links between choices and beliefs. *Developmental Psychology*, 42(1):70–83. cited By 375.

Skemp, R. R. (2006). Relational understanding and instrumental understanding. *Mathematics Teaching in the Middle School*, 12(2):88–95.

Teitler, P. (2017). *Lessen in orde: handboek voor de onderwijspraktijk*. Uitgeverij Coutinho, Bussum.

Timmer, M. and Tolboom, J. (2019). Computational thinking–een vooruitblik op het wiskundeonderwijs van de toekomst. *Nieuw archief voor wiskunde*, 5(1):42–45.

Van Dormolen, J. (1974). *Didactiek van de wiskunde*. Utrecht: Oosthoek.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., and Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, 25(1):127–147.

Wetzels, S. A., Kester, L., and Van Merrienboer, J. J. (2011). Adapting prior knowledge activation: Mobilisation, perspective taking, and learners' prior knowledge. *Computers in Human Behavior*, 27(1):16–21.

Windels, B. (2012). Het 6E-model. *Nieuwe Wiskrant*, 32(2):20–26.

Wing, J. M. (2014). Computational thinking benefits society. *40th Anniversary Blog of Social Issues in Computing*, 2014.

Winston, W. (2004). *Operations Research: Applications and Algorithms*. Thomson/Brooks/-Cole, Belmont, CA.

A Feedback Walk-through (Dutch)

The feedback that the expert panel has given via the walk-through is given below. Since the experts are Dutch and the lesson series was only available in Dutch at the time of the walk-through, their feedback has been given in Dutch. They gave feedback on the design requirements and provided general remarks about the lesson series.

A.1 Student in mathematics

Feedback OvO – Jarco Slager (Dynamisch Programmeren)

Ontwerpaspecten/Eisen:
Welke onderdelen zitten er nog niet voldoende in?

Het zit goed in elkaar, veel variërende opdrachten. Er is duidelijk gekeken naar de ontwerpeisen, ze komen goed terug in de verschillende opdrachten. De ontwerpeis die het minst verwerkt is, is C7, maar dit is ook lastig verwerken in een lessenserie lijkt me. Je zou nog als huiswerk kunnen geven na les 1 dat ze één van de genoemde voorbeelden mogen kiezen en nadenken over hoe dynamisch programmeren kan worden toegepast. Dan zijn ze 'vrijgelaten om zelf de theorie te verkennen'.

Algemene feedback:
Waar kan ik dingen verbeteren/aanpassen? Dat mogen ook algemene opmerkingen zijn.

Ik heb 2 tekstuele suggesties:

1. *Les 2, vraag 3: In het tweede puntje bovenaan pagina 35 staat: Ook hier betekent het subscript x weer dat we de x-waarde hebben aangepast. In plaats van 'hebben aangepast' zou ik hier 'kunnen variëren' neerzetten.*
2. *Les 3, eindopdracht, vraag a: In de zin 'De beste route vanaf station s2 verschijnt nu ook, namelijk 7' zou ik neerzetten 'namelijk via station 7'.*

Verder zit het echt heel goed in elkaar naar mijn mening. Goede opbouw, duidelijke uitleg. De moeilijkheid vind ik erg lastig inschatten.

Het lijken me goede lessen voor een lesduur van 50 minuten. Zelf denk ik aan de volgende tijdsverdeling:

Les 1:

- *Vraag 1: 15 minuten,*
- *Vraag 2: 20 minuten inclusief bespreken,*
- *Vraag 3: 10 minuten inclusief bespreken*

Les 2:

- *Alle opdrachten ongeveer 15 minuten inclusief bespreken*

Les 3:

- *Vraag 1: 10 minuten inclusief bespreken*
- *Vraag 2: 10 minuten inclusief bespreken*
- *Vraag 3: 10 minuten inclusief bespreken*
- *Eindopdracht: 15 minuten inclusief bespreken*

Van alle lessen, denk ik dat les 3 de meeste tijd kan gaan kosten.

A.2 Student in education

Feedback OvO – Jarco Slager (Dynamisch Programmeren)

Ontwerpaspecten/Eisen:
Welke onderdelen zitten er nog niet voldoende in?

Bij opdracht 3 les 1 mis ik een soort van afsluiter waarbij een voorbeeld wordt gegeven waarbij de leerlingen bijvoorbeeld beargumenteren waarom dynamisch modelleren wel goed past of niet goed past. Dus na het noemen van voor en nadelen dit ook kunnen meenemen in afweging of je bij een probleem dynamisch zou willen programmeren ja of nee.

Les 2 opdracht 1 is het stappenplan opstellen niet ideaal. Hier zou je denk ik beter eerst 1. 2. 3. Stapsgewijs kunnen zeggen 1. we kijken naar het laatste station en de routes die hierop aansluiten. 2. We zoeken wat hiervan de optimale route is. 3. We doen van onze net gevonden stations een stapje terug en kijken wat hiervan de kosten per route zijn. 4... Dan terug komen op het feit dat we hier onze stappen steeds herhalen dus de fasen aanbrengen. Bij opdracht 3 zou ik meer nadruk leggen of neem f(x) zijn de kosten voor de reis en x is de route die we kiezen dus eerst wat duidelijker koppelen x= $x_1 \to x_3 \to x_6 \to x_8$ bijvoorbeeld. Hier kom je iets later mee. Zou ik dus iets naar voren schuiven.

Les 3 opdracht 1 zijn bij elke fase evenveel stations. Is het niet interessanter dit per fase een ander aantal te laten zijn? Opgave c is erg mooi 😊

Opdracht 3 les 3 zou ik de stations A, B,etc tm ZZ nummeren om verwarring te voorkomen. Zeker in de fase van nieuwe cognitieve schema's maken is het gevaarlijk als er een fout in de denkstap zit.

Algemene feedback:
Waar kan ik dingen verbeteren/aanpassen? Dat mogen ook algemene opmerkingen zijn.

A.3 Teacher 1

Feedback OvO – Jarco Slager (Dynamisch Programmeren)

Ontwerpaspecten/Eisen:
Welke onderdelen zitten er nog niet voldoende in?

Heel erg leuk!

Ik houd me aanbevolen voor de uitwerkingen..haha

Algemene feedback:
Waar kan ik dingen verbeteren/aanpassen? Dat mogen ook algemene opmerkingen zijn.

Ik heb het laatste gedeelte met excel niet gedaan. Ik vind de opdracht erg leuk.

Eventueel zou je nog iets met grafentheorie kunnen toevoegen, want dat zit hier ook in.

Les 1:

Opdracht 1,2 leuke manier om voorkennis te activeren.

Opdracht 3: weten ze al voldoende van dynamisch programmeren om voordeel en nadeel te noemen?

Les 2: Ik zou het figuur onder de tekst plaatsen.

"Bij elke pijl staat aangegeven hoe lang het duurt om tussen de stations te reizen, wat we de kosten noemen"

Dit is beetje vreemd toch? Je bedoelt Hoe korter het duur hoe lager de kosten?

In de tabel staat mogelijke route, het maakt het misschien iets duidelijker voor leerlingen als er staat

"mogelijk route naar"

Opdracht 3

Maakt neem ik aan niet uit hoe leerlingen dit doen?

Les 3: Ik dacht WOW die tabel...hoe dan! Maar erg leuk en heel gestructureerd!

A.4 Teacher 2

Opdr. 2 → stappenplan → zodat A wint

Opdracht 3 → onduidelijk wat je bedoelt met de 1e zin; ... of maar 5.

→ Zonet is besproken hoe zo'n strategie gevonden kan worden.
3: Is de vraag voordeel/nadeel niet te vroeg?

Les 2 Routeprobleem: Intro → wat we hier kosten noemen

→ Misschien ook antw. erbij vermelden.
→ zodat je kunt controleren of je op de goede weg zit

blz 33 → bij de 2 stappen vermelden dat je bij het eindstation begint. Hier dus fase 4

opdr. 3 → Het berekenen van een min bij een gegeven functie f.

Algemeen → voor wie bestemd?
→ minimum bepalen → $x = \frac{-b}{2a}$ of afgeleide?

Les 3 opdr 1 i we de lengte van alle routes berekenen en het min zoeken

Ik begrijp niet helemaal lineair verb en exp. verb (blz) 37

→ Ik snap niet helemaal hoe het onderdeel minimum bepalen bij een 2e gr functie, terwijl je daarna een min. in Excel gaat bepalen.

A.5 Textual remarks

In the feedback from panel of experts, textual remarks were given. In addition to giving the textual changes in Dutch, these changes will also be given in English. Remarks on the lay-out are also given here.

- Lesson 2, introduction: "wat we vaak de kosten noemen" wordt "wat we hier de kosten noemen". (*English: "what we often call the costs" becomes "what we call the costs here".*)
- Lesson 2, introduction: the figure of the routing problem has been placed below the explanation.
- Lesson 2, table: "Mogelijke routes" wordt "Mogelijke routes naar" (*English: "Possible routes" becomes "Possible routes to".*)
- Lesson 2, exercise 3: "hebben veranderd" wordt "kunnen variëren". (*English: "have changed" becomes "can vary".*)
- Lesson 3, exercise 1: "we de lengte alle routes berekenen" wordt "we de lengte van alle routes berekenen". (*English: "we compute the length all routes" becomes "we compute the length of all routes".*)
- Lesson 3, final assignment: "namelijk 7" wordt "namelijk via station s_7". (*English: "i.e. 7" becomes "i.e. via station s_7".*)

B Iteration 1 (Dutch)

Les 1 - Introductie Dynamisch Programmeren

Opdracht 1
Jullie gaan een strategisch spel spelen, namelijk het lucifer-spel. Dit spel spelen jullie in tweetallen, speler A en B. De spelregels (lees ze allemaal eerst helemaal door):

1. Begin met 30 lucifers op tafel.
2. Kies de persoon die gaat beginnen, speler A. Speler A mag kiezen om 1, 2 of 3 lucifers van tafel te halen.
3. Nu is het de beurt van speler B. Ook speler B mag nu 1, 2 of 3 lucifers van tafel halen. Dit herhalen jullie om en om. Je moet altijd minimaal 1 lucifer van tafel halen.
4. De speler die de laatste lucifer van tafel moet halen, verliest. Bijvoorbeeld: het is de beurt van speler A en er liggen 3 lucifers. Als speler A nu 2 lucifers van tafel haalt, dan ligt er nog 1. Speler B moet deze nu wel van tafel halen en verliest. Veel plezier!

Opdracht 2
Het is mogelijk om een strategie te bedenken waardoor speler A (de speler die begint) met zekerheid wint met het lucifer-spel. Probeer een strategie te bedenken waarmee speler A wint. Schrijf hiervoor een stappenplan op, zodanig dat speler A wint.

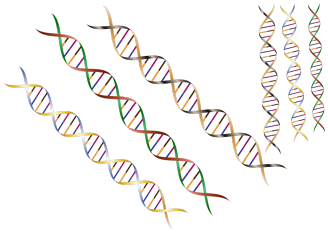
Opdracht 3
Zonet is besproken hoe zo'n strategie gevonden kan worden, namelijk door eerst te kijken wat er aan de hand is met maar 1 lucifer, of maar 5. Het vinden van een optimale strategie/aanpak op deze manier is een voorbeeld van **dynamisch programmeren**. Bij dynamisch programmeren proberen we problemen op te lossen door het probleem in kleinere stukken te hakken, of bijvoorbeeld door "aan het eind" te beginnen. Als we de oplossing hebben gevonden van een klein probleem, dan gebruiken we die voor het oplossen van het grotere probleem.

(a) Hoe hebben we dynamisch programmeren gebruikt in Opdracht 2?

(b) Noem een voordeel van dynamisch programmeren.

(c) Noem een nadeel van dynamisch programmeren.

Dynamisch programmeren kent veel toepassingen. Voorbeelden hiervan zijn het vergelijken van DNA-ketens of het vinden van de snelste route. Dit laatste voorbeeld gaan we gebruiken om meer te weten te komen over dynamisch programmeren, wat in de volgende lessen aan bod komt.

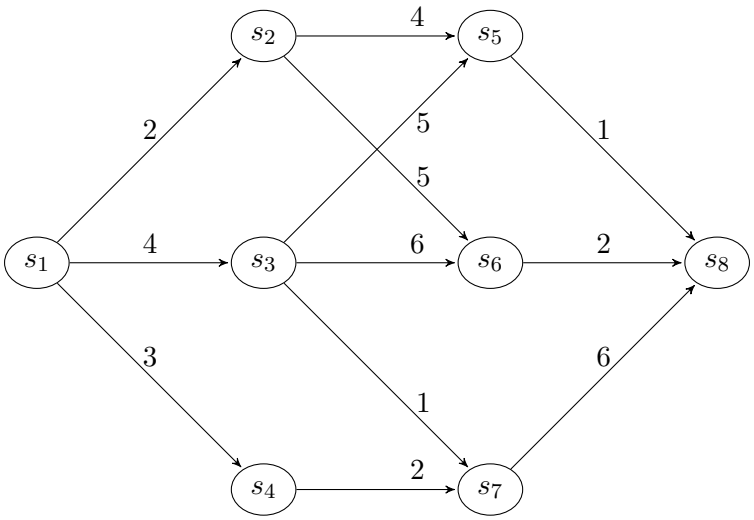


Figuur 10: *Vergelijken van DNA-ketens.*

Figuur 11: *Vinden van de snelste route.*

Les 2 - Het routeprobleem

In de vorige les zijn er voorbeelden aan bod gekomen waarbij dynamisch programmeren gebruikt kan worden. Eén van deze voorbeelden was het routeprobleem. Een voorbeeld van een route is te zien in Figuur 12.



Figuur 12: *Een voorbeeld van een routeprobleem.*

Stel dat dit routeprobleem een treinnetwerk voorstelt. De rondjes stellen de **stations** voor. Het beginstation is station s_1. De pijlen tussen de verschillende stations geven aan waar je naartoe mag reizen vanaf een station. Bij elke pijl staat aangegeven hoe lang het duurt om tussen deze stations te reizen, wat we vaak de **kosten** noemen.

Bijvoorbeeld: vanaf station s_3 kunnen we naar station s_5 reizen met kosten 5. Ook kan er worden gereisd naar station s_6 met kosten 6 en naar station s_7 met kosten 1.

Opdracht 1

Het doel is om vanaf station s_1 naar station s_8 te reizen, met de kosten zo laag mogelijk (wat betekent dat we zo snel mogelijk willen reizen). Voordat we dit doen, bekijken we eerst waarom dynamisch programmeren hier handig kan zijn.

(a) Voor dynamisch programmeren kijken we naar kleinere problemen. Bijvoorbeeld, als we van station s_3 naar station s_8 willen reizen. Laat met een berekening zien dat de snelste route van station s_3 naar station s_8 via station s_5 is. Hoe hoog zijn de kosten?

(b) Waarom hoef je niet de kosten van de routes $(s_1) \to (s_3) \to (s_6) \to (s_8)$ en $(s_1) \to (s_3) \to (s_7) \to (s_8)$ te berekenen om erachter te komen wat de korste route is van station s_1 naar s_8?

(c) Bereken de kosten van de snelste route van station s_1 naar station s_8, via station s_3. Gebruik hiervoor vraag (a).

Dit soort berekeningen gaan we steeds uitvoeren om erachter te komen wat de snelste route is. Hiervoor willen we een gestructureerde aanpak bedenken. Dat doen we als volgt. In het

plaatje zien we al een structuur met "groepjes", namelijk s_2, s_3 en s_4 staan bij elkaar en s_5, s_6 en s_7. Daarom verdelen we de routes in vier verschillende fases:

- *Fase 1:* (s_1).
- *Fase 2:* (s_2), (s_3), (s_4).
- *Fase 3:* (s_5), (s_6), (s_7).
- *Fase 4:* (s_8).

We gaan de oplossing zoeken met behulp van dynamisch programmeren. Dat betekent dat we in *Fase 4* beginnen. De stappen die we elk fase doorlopen zijn de volgende:

1. Bekijk voor elk station hoe we het snelst bij station s_8 komen vanaf het huidige station. Dit noemen we de minimale kosten.
2. Onthoud wat vanaf het huidige station de snelste keuze was, dus we onthouden naar welk station we daarna moeten reizen.

We maken gebruik van de volgende tabel:

Fase	**Station**	**Mogelijke routes**	**Minimale kosten**	**Volgende station?**
4	(s_8)	Geen.	0	Geen.
3	(s_5)	(s_8): Kosten 1	1	(s_8)
	(s_6)	(s_8): Kosten 2	2	(s_8)
	(s_7)	(s_8): Kosten 6	6	(s_8)
2	(s_2)	(s_5): Kosten $4 + 1 = 5$, (s_6): Kosten $5 + 2 = 7$,	5	(s_5)
	(s_3)	(s_5): Kosten $5 + 1 = 6$, (s_6): Kosten $6 + 2 = 8$, (s_7): Kosten $1 + 6 = 7$,	6	(s_5)
	(s_4)	(s_7): Kosten $2 + 6 = 8$	8	(s_7)
1	(s_1)	(s_2) (s_3) (s_4)		

In de tabel is het volgende gedaan. In *Fase 4* kiezen we geen volgende route. De minimale kosten om vanuit station s_8 naar station s_8 te gaan is 0: we zijn er namelijk al.

Vervolgens kijken we naar alle stations in *Fase 3*. Bijvoorbeeld, vanaf station s_5 kunnen we alleen naar station s_8 gaan met kosten 1. Dus, de minimale kosten om van station s_5 naar station s_8 te gaan zijn gelijk aan 1. We kunnen alleen naar station s_8, dus dat is het volgende station. Zo is dit ook gedaan voor de andere stations in *Fase 3*.

We herhalen dit voor *Fase 2* en *Fase 1*. We schetsen de berekening voor station s_3. Vanuit station s_3 kunnen we naar station s_5, s_6 of s_7. De kosten om van station s_3 naar station s_8 te gaan via station s_5 zijn gelijk aan de kosten om van station s_3 naar station s_5 te gaan (=5), plus de minimale kosten om van station s_5 naar station s_8 te gaan (= 1), wat we uit *Fase 3* halen. Dat komt neer op kosten van 6. Dit doen we ook voor de andere stations en vinden dat de minimale kosten gelijk zijn aan 6. Dus, vanuit station s_3 kiezen we als volgende station s_5, want deze zorgt voor de laagste kosten.

Opdracht 2

Het doel is nu om te kijken wat de optimale route is vanuit station s_1. In *Fase 1* beginnen we in station s_1. Hier kunnen we naar station s_2, s_3 of s_4. Met behulp van de minimale kosten vanuit de stations uit *Fase 2* kunnen we de optimale route vinden.

(a) Bereken de kosten om van station s_1 naar station s_8 te gaan via alle drie stations in *Fase 2*. Vul deze waarden in in de tabel.

(b) Als het goed is, is één van deze waardes minimaal. Wat betekent deze waarde?

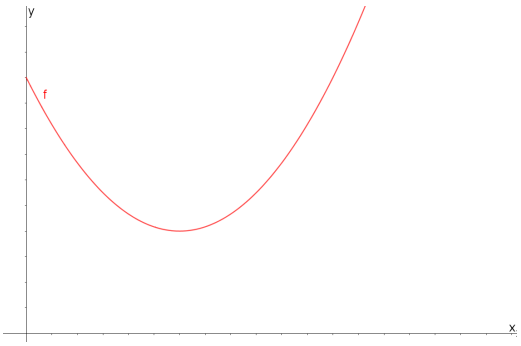
(c) Maak de tabel helemaal af. Wat is de optimale route? Kleur deze route in Figuur 12.

We hebben nu de tabel afgemaakt. Omdat we de kosten zo laag mogelijk willen houden, hebben we steeds de route gekozen die de minimale kosten geeft. In feite hebben we dus steeds het minimum berekend. Het berekenen van een minimum is iets wat jullie wel kunnen gegeven een functie. Dit gaan we in de volgende opdracht herhalen en leren hier nieuwe notatie voor.

Opdracht 3

Het berekenen van een minimum gegeven een functie f hebben jullie al vaak gedaan.

(a) Gegeven de functie $f(x) = \frac{1}{3}x^2 - 2x + 5$, voor $x \geq 0$, met de volgende grafiek:



Figuur 13: *De grafiek van f.*

Het minimum is de minimale functiewaarde van $f(x)$. De x-waarde die we invullen noemen we ook wel het argument. Laat op exacte wijze zien dat het minimum optreedt bij het argument $x = 3$ en dat het minimum gelijk is aan 2.

Dit kunnen we weer koppelen aan het routeprobleem. In elke fase van het routeprobleem kiezen we een route waarvoor de kosten minimaal zijn en vraag (a) "kiezen" we een x waarvoor $f(x)$

minimaal is. Dit zijn voorbeelden van het vinden van een minimum, waarbij een bepaalde "keuze" (de route of x-waarde) zorgt voor dat minimum. We introduceren hiervoor twee belangrijke notaties aan de hand van wat we net hebben gedaan:

- We variëren de x-waarde (het argument) om zo de minimale waarde te vinden van $f(x)$, wat we schrijven als $\arg\min_x\{f(x)\}$. Het subscript x betekent dat het de x is die we variëren. In ons voorbeeld was deze x-waarde gelijk aan 3, dus hier geldt $\arg\min_x\{f(x)\} = 3$.
- Als je dit argument hebt gevonden, dan kun je het minimum berekenen. Dat minimum noteren we dan op een vergelijkbare manier: $\min_x\{f(x)\}$. Ook hier betekent het subscript x weer dat we de x-waarde hebben aangepast. In ons voorbeeld van vraag (a) was het minimum gelijk aan 2 (door $\arg\min_x\{f(x)\} = 3$ in te vullen), dus hier geldt $\min_x\{f(x)\} = f(3) = 2$.

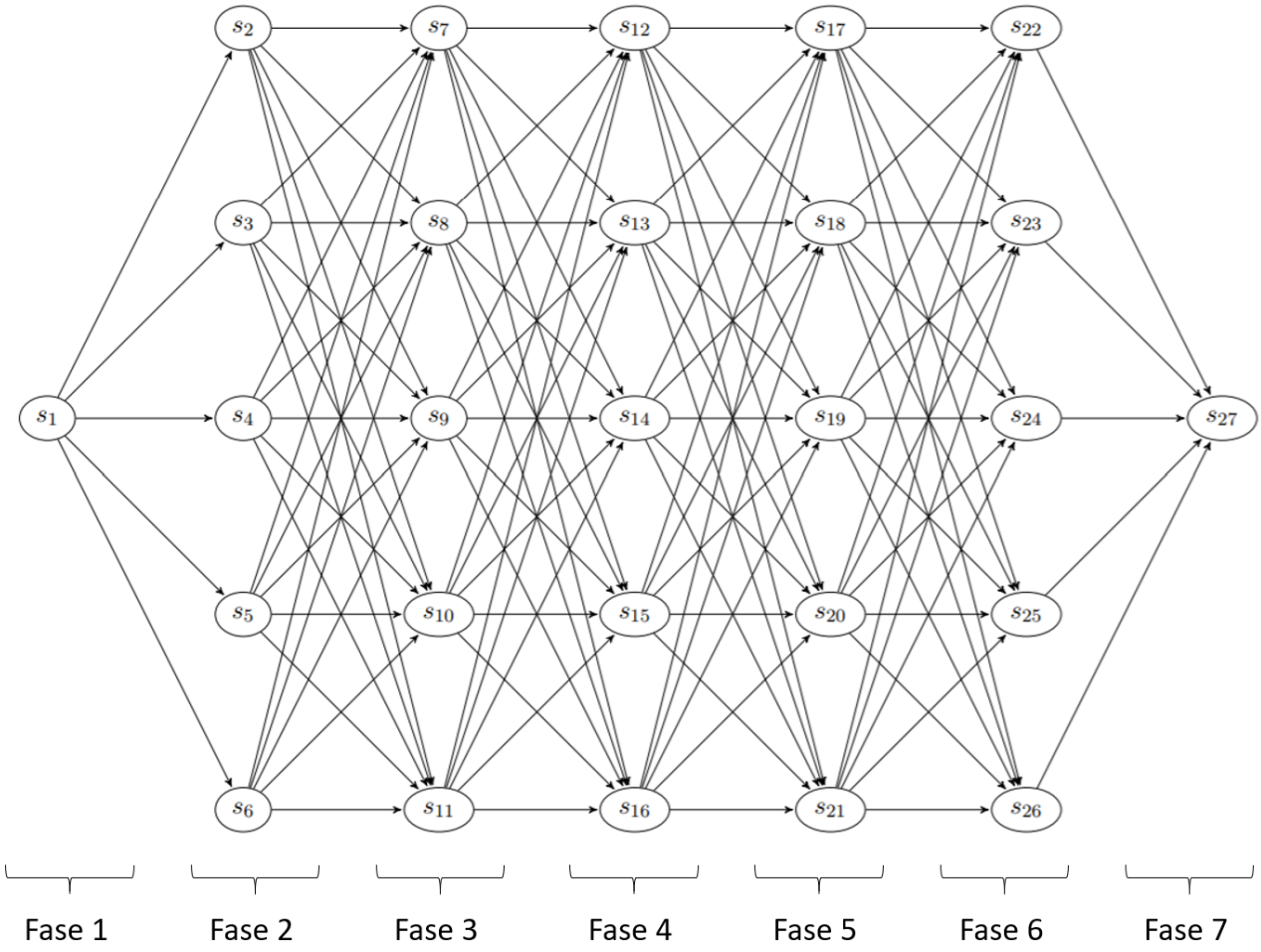
Deze notatie wordt veel gebruikt bij wiskunde en komt bijvoorbeeld terug bij programmeren, waarbij bijvoorbeeld het routeprobleem voor grote netwerken kan worden opgelost. Houd nu de tabel van pagina 60 erbij. We gaan oefenen met de notatie die we net hebben geleerd.

Kijk eens naar station s_2 in *Fase 2* in de tabel. We hebben al berekend dat de minimale kosten gelijk zijn aan 5, want als we moeten kiezen tussen kosten 5 of 7, dat we dan 5 kiezen (we variëren de stations waar we naartoe kunnen reizen). Het station dat we hebben gekozen was station s_5 en niet station s_6 (dus het argument voor deze kosten is station s_5). Daarom schrijven we $\arg\min_{\text{stations}}\{kosten(stations)\} = s_5$. De minimale kosten zijn dan de kosten die we krijgen als we naar dit station reizen, die gelijk zijn aan 5. Als we de notatie gebruiken in dit voorbeeld, dan schrijven we dat als $\min_{\text{stations}}\{kosten(stations)\} = 5$ (we nemen dus het minimum over de stations, waarbij de minimale waarde dus 5 is).

(b) Gebruik deze notatie om ook het minimum en het argument voor het minimum te geven voor de andere stations in *Fase 2*, dus voor stations s_3 en s_4.

Les 3 - Het routeprobleem met Excel

In Figuur 14 zien we een nieuw routeprobleem. We zien hier weer stations, genaamd s_1 t/m s_{27} en de pijlen geven de mogelijke routes aan. Er staan hier geen getallen bij de pijlen, omdat het dan onoverzichtelijk wordt, omdat vanuit elk station uit elke fase een route is naar elk station uit de volgende fase.



Figuur 14: *Een voorbeeld van een routeprobleem.*

Opdracht 1

In deze opdracht gaan we kijken naar het routeprobleem uit Figuur 14. Als we willen weten wat de snelste route is van station s_1 naar station s_{27}, dan moeten we de kosten van de routes weten. Echter, eerst willen we iets meer weten over het **aantal** routes en daarmee het aantal berekeningen.

(a) Hoeveel mogelijke routes zijn er om van station s_1 naar station s_{27} te reizen?

(b) Stel dat er nog een extra fase bijkomt tussen fase 6 en 7 (dus weer 5 extra stations). Hoeveel **extra** routes krijgen we dan? En als er daarna nog een fase van 5 stations bijkomt?

(c) We nemen aan dat we gewoon de situatie hebben zoals in Figuur 14. Stel dat er tussen fase 5 en 6 een extra fase bijkomt van 5 stations. Wat voor invloed heeft dat op het aantal berekeningen die we uit moeten voeren als:

i. We de lengte **alle** routes berekenen en het minimum zoeken?

ii. We de minimale route zoeken met Dynamisch Programmeren?

Geef hiermee een conclusie over de efficiëntie van manier i. en van ii. als we bijvoorbeeld 100 fases zouden toevoegen. Gebruik in je conclusie de termen "lineair verband" en "exponentieel verband".

We gaan het nu concreter maken. Voor elk station zijn de kosten bekend om naar de andere stations te reizen van de volgende fase. De kosten van station s_1 naar station s_2 gelijk zijn aan 7. Ook de kosten van station s_1 naar de andere stations zijn bekend. Die zijn als volgt:

- $s_1 \rightarrow s_2$: kosten 7.
- $s_1 \rightarrow s_3$: kosten 8.
- $s_1 \rightarrow s_4$: kosten 9.
- $s_1 \rightarrow s_5$: kosten 5.
- $s_1 \rightarrow s_6$: kosten 8.

Deze waardes staan ook in de eerste kolom van Tabel 5. De waardes staan dus voor de kosten naar de stations in de volgende fase, in de volgorde van boven naar beneden (zoals in Figuur 14). We gaan eerst eens kijken naar de andere gegevens in de tabel.

Tabel 5: *De afstand tussen de verschillende stations van de volgende fase.*

Fase 1		**Fase 2**		**Fase 3**		**Fase 4**		**Fase 5**		**Fase 6**		**Fase 7**
s_1	7	s_2	6	s_7	5	s_{12}	5	s_{17}	10	s_{22}	6	s_{27}
	8		9		6		8		10			
	9		10		10		10		6			
	5		6		10		5		8			
	8		8		7		10		10			
		s_3	9	s_8	5	s_{13}	8	s_{18}	8	s_{23}	8	
			9		5		8		7			
			10		9		5		9			
			7		9		7		8			
			10		6		8		6			
		s_4	7	s_9	6	s_{14}	8	s_{19}	9	s_{24}	8	
			6		7		7		9			
			7		9		7		8			
			9		5		5		9			
			6		9		5		8			
		s_5	10	s_{10}	6	s_{15}	9	s_{20}	9	s_{25}	5	
			8		6		10		7			
			10		9		10		9			
			6		9		9		8			
			6		7		9		9			
		s_6	6	s_{11}	5	s_{16}	6	s_{21}	6	s_{26}	7	
			9		7		5		9			
			10		6		6		10			
			9		5		5		9			
			7		9		8		8			

Opdracht 2
Kijk nog eens naar Tabel 5. De rest van de tabel is net zo opgebouwd als de kolom van **Fase 1**.

(a) Wat betekent de waarde 6 die bij station s_2 staat? En wat betekent de 9 die bij station s_2 staat?

(b) Kijk eens naar de kolom van **Fase 6**. Waarom is er per station maar 1 waarde?

(c) Geef een verklaring waarom er bij station s_{27} geen waarde is gegeven. Als je station s_{27} een waarde moest geven, welke waarde zou dat dan zijn (ofwel, hoe lang duurt het om vanaf s_{27} naar s_{27} te reizen)?

Om dit op te lossen, zouden we alles met de hand uit kunnen schrijven. Ook al doen we dit met behulp van dynamisch programmeren, het blijft veel werk om dat allemaal te berekenen. Gelukkig kunnen hiervoor computerprogramma's gebruiken om ons te helpen. Wij gaan dit probleem op lossen met behulp van Excel.

Opdracht 3
Open het bestand RouteprobleemLes3.xl op je computer. In dit bestand staat een vergelijkbare tabel als Tabel 5. Toegevoegd is dat de afstand van station s_{27} naar zichzelf gelijk is aan 0. In dit bestand heten de stations station 1, 2, 3,... in plaats van $s_1,\ s_2,\ s_3, \ldots$.

Onderaan de tabel staan 6 gekleurde blokken/tabellen met allemaal de volgende gegevens:

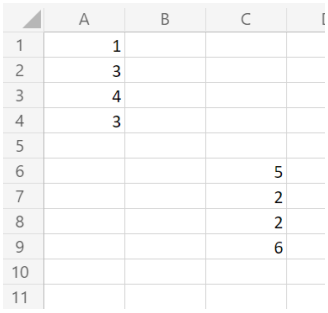
- Stations: hieronder staan de stations in de fase erboven.
- k_station: dit zijn de minimale kosten om te reizen naar het eindstation (27), vanaf dat station.
- Beste route: dit geeft aan welke route als volgt moet worden gekozen.

Met dynamisch programmeren beginnen we aan het eind van het probleem. Dat eind is weer simpel, want bij station s_{27} zijn we er al. Dan gaan we terugwerken, waarbij we de berekeningen uitvoeren in de verschillende blokken. De eerste twee blokken zijn roodgekleurd: deze zijn nog niet ingevuld. De vier andere zijn wel ingevuld. Aan jullie de taak om de rode blokken ook in te vullen. Dan verschijnt onder het kopje "De optimale route is dan:" een reeks met stations die ervoor zorgen dat de kosten minimaal zijn. Beantwoord de volgende vraag:

Vraag: Kijk eens naar station s_{25} in het laatste blok. Wat betekenen de 5 en 27? En als we kijken naar station s_{21}, wat betekenen de 12 en 22?

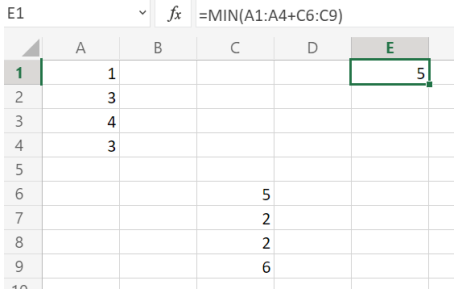
Uitleg Excel (voor de eindopdracht)
We focussen nu op fase 2, die we graag in willen vullen. De beste routes zijn nog onbekend, dus die staan nu op 0 (die weten we namelijk niet). We gebruiken het minimum uit fase 3 om voor elk station uit fase 2 de laagste kosten te vinden om naar het eindstation te reizen, wat precies dezelfde aanpak vereist als we in de vorige les hebben gedaan. In Excel kan dat met de functie MIN. Dit gaan jullie straks zelf doen, maar lees eerst de instructies hieronder. Jullie hebben nodig dat je het minimum kunt berekenen van de som van twee kolommen (dit komt dus terug in de eindopdracht). We laten nu zien hoe je dat moet doen:

1. Stel je hebt een kolom met getallen in de vakken $A1$ tot en met $A4$ en in de vakken $C6$ tot en met $C9$, zie het plaatje van Figuur 15:



Figuur 15: *Het voorbeeld voor het gebruiken van* MIN.

2. Selecteer een leeg vakje waar je de minimale waarde wilt hebben. In ons voorbeeld nemen we $E1$. We selecteren het vakje $E1$ en vullen in $=\min(A1{:}A4{+}C6{:}C9)$. Excel telt nu de waardes van de eerste rij van beide kolommen bij elkaar op $(A1 + C6)$, de getallen van de tweede rij van beide kolommen $(A2 + C7)$, enz. Dan berekent Excel het minimum, wat 5 is (dat is hier dus de som van de getallen van de tweede rij, namelijk 3 + 2). Zie de figuur hieronder.



Figuur 16: *Het antwoord voor het gebruiken van de functie* MIN.

Eindopdracht

In deze eindopdracht gaan jullie berekenen hoe lang de kortste route is om van station s_1 naar station s_{27} te reizen. Gebruik hiervoor het bestand van de vorige opgave (Routeprobleem-Les3.xl).

(a) Bereken hiertoe eerst de optimale kosten van station s_2 naar station s_{27}. Gebruik de aanpak van de vorige les. Je zult de uitleg over het berekenen van een minimum in Excel nodig hebben. De beste route vanaf station s_2 verschijnt nu ook, namelijk 7. *Let op: je moet dus het vakje k_station invullen voor station* s_2.

(b) Doe hetzelfde voor de andere station in fase 2. Geef van elk station de laagste kosten om naar station s_{27} te reizen. Als het goed is, verschijnen de beste routes vanaf elk station in fase 2. *Let op: als er 0 blijft staan bij Beste route, dan klopt je optimale waarde nog niet.*

(c) Tot slot, wat zijn de minimale kosten om van station s_1 naar station s_{27} te reizen? Geef de optimale route door het schema hieronder in te vullen:

$(s_1) \rightarrow \ldots \rightarrow \ldots \rightarrow \ldots \rightarrow \ldots \rightarrow \ldots \rightarrow (s_{27})$

Dit is het einde van de lessenserie.

C Solutions of the students (Dutch)

The students worked together on this lesson series and worked out the solutions on one sheet. Their solutions are given below, consisting of two pages. Afterwards, the solutions to exercise 2a) and 2c) of lesson two are given. At the end of this appendix, three figures depict how the students solved the final assignment.

Les 1

(2) 1. Speler A pakt 1 lucifer.
2. Speler B pakt x lucifers
3. Speler A pakt dan $4-x$ lucifers.
~~4. Zo door tot er 5 over zijn~~
4. Er blijft nog 1 over waardoor A wint.

(3) We zijn op het eind begonnen. Aan de hand hiervan zijn we verder gaan kijken hoe we het op konden lossen.

Les 2

(1) a $S_3 \rightarrow S_7 \rightarrow S_8$: $1+6=7$
$S_3 \rightarrow S_6 \rightarrow S_8$: $6+2=8$
$S_3 \rightarrow S_5 \rightarrow S_8$: $5+1=6$

b $S_3 \rightarrow S_6 \rightarrow S_8$ en $S_3 \rightarrow S_7 \rightarrow S_8$ zijn sowieso al hoger dan $S_3 \rightarrow S_5 \rightarrow S_8$

c $S_1 \rightarrow S_3$ is één mogelijkheid dus
$S_1 \rightarrow S_3 \rightarrow S_5 \rightarrow S_8$: $4+5+1=10$

(2) b De kosten voor die route het laagst zijn. Je bent er via die route dus het snelst

(3) a $f(x) = \frac{1}{3}x^2 - 2x + 5$
$f'(x) = \frac{2}{3}x - 2$
$\frac{2}{3}x - 2 = 0$
$\frac{2}{3}x = 2$
$x = 3$
~~$f(\frac{1}{3}) = \frac{1}{3}\cdot(\frac{1}{3})^2 - 2\cdot\frac{1}{3} + 5 = \frac{1}{3}\cdot\frac{1}{9} - \frac{2}{3} + 5 =$~~
~~$\frac{1}{27} - \frac{2}{3} + 5 =$~~

Eindopdracht : $S_1 \rightarrow S_5 \rightarrow S_{11} \rightarrow S_{12} \rightarrow S_{17} \rightarrow S_{25} \rightarrow S_{27}$

$f(x) = \frac{1}{3} \cdot (3)^2 - 2 \cdot 3 + 5 =$
$\frac{1}{3} \cdot 9 - 6 + 5 =$
$3 - 6 + 5 = 2$

b S_3 : $\min_{\text{station}} \{\text{kosten (station)}\} = 6$
S_4 : $\min_{\text{station}} \{\text{kosten (station)}\} = 8$

c $f(x) =$ kosten (station)
$x =$ station

Les 3

(1) a 5^{21}

b $21 + 5 = 26 \rightarrow 5^{26}$
$26 + 5 = 31 \rightarrow 5^{31}$

dus in het eerste geval 5^5
" tweede geval 5^{10}

c i : exponentieel verband. Alle routes worden meegenomen dus het aantal mogelijkheden neemt exponentieel toe. Minimum ~~stijgt~~ zal hierdoor hoger liggen

ii : lineair verband. We beginnen achteraan met het oplossen. Je zoekt naar een oplossing dus gaat het lineair

(2) a eerste 6 bij S_2 gaat naar S_7
tweede 6 bij S_2 gaat naar S_{10}
de 9 bij S_2 gaat naar ~~28~~ S_8

b Hij gaat alleen maar naar 27

c 0. Je bent ~~ook~~ er al.

(3) ~~kosten~~ kosten 5 naar stations 27
6+6 ~~8+5~~ geeft 12 om naar 27 ~~te~~ te komen
↓station
vanaf 21 naar 22 naar 27

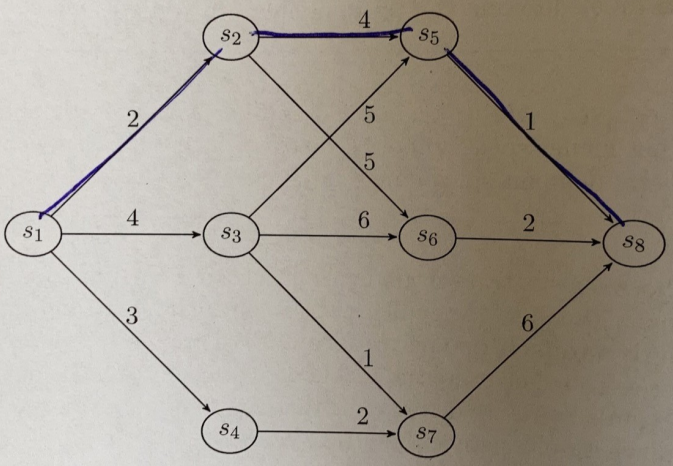
station we daarna moeten reizen.

We maken gebruik van de volgende tabel:

Fase	Station	Mogelijke routes naar	Minimale kosten	Volgende station?
4	s_8	Geen.	0	Geen.
3	s_5	s_8: Kosten 1	1	s_8
	s_6	s_8: Kosten 2	2	s_8
	s_7	s_8: Kosten 6	6	s_8
2	s_2	s_5: Kosten $4+1=5$, s_6: Kosten $5+2=7$,	5	s_5
	s_3	s_5: Kosten $5+1=6$, s_6: Kosten $6+2=8$, s_7: Kosten $1+6=7$,	6	s_5
	s_4	s_7: Kosten $2+6=8$	8	s_7
1	s_1	s_2 kosten $2+5=7$ s_3 kosten $4+6=10$ s_4 kosten $3+8=11$	7	S2

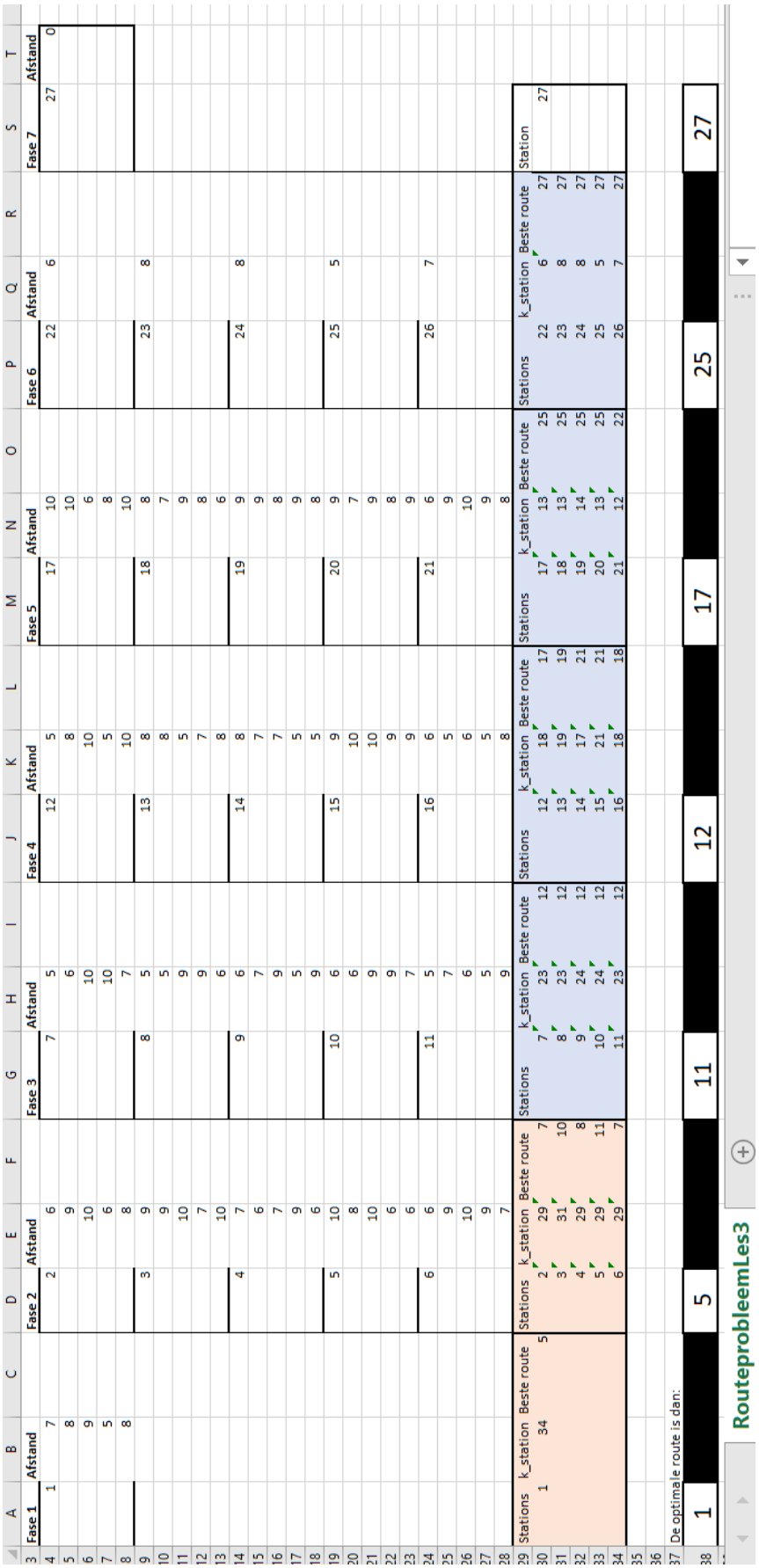
In de tabel is het volgende gedaan. In *Fase 4* kiezen we geen volgende route. De minim kosten om vanuit station s_8 naar station s_8 te gaan is 0: we zijn er namelijk al.

worden gereisd naar station s_6 met kosten 6 en naar station s_7 met kosten 1. Hoe lager de kosten, hoe korter de reistijd.



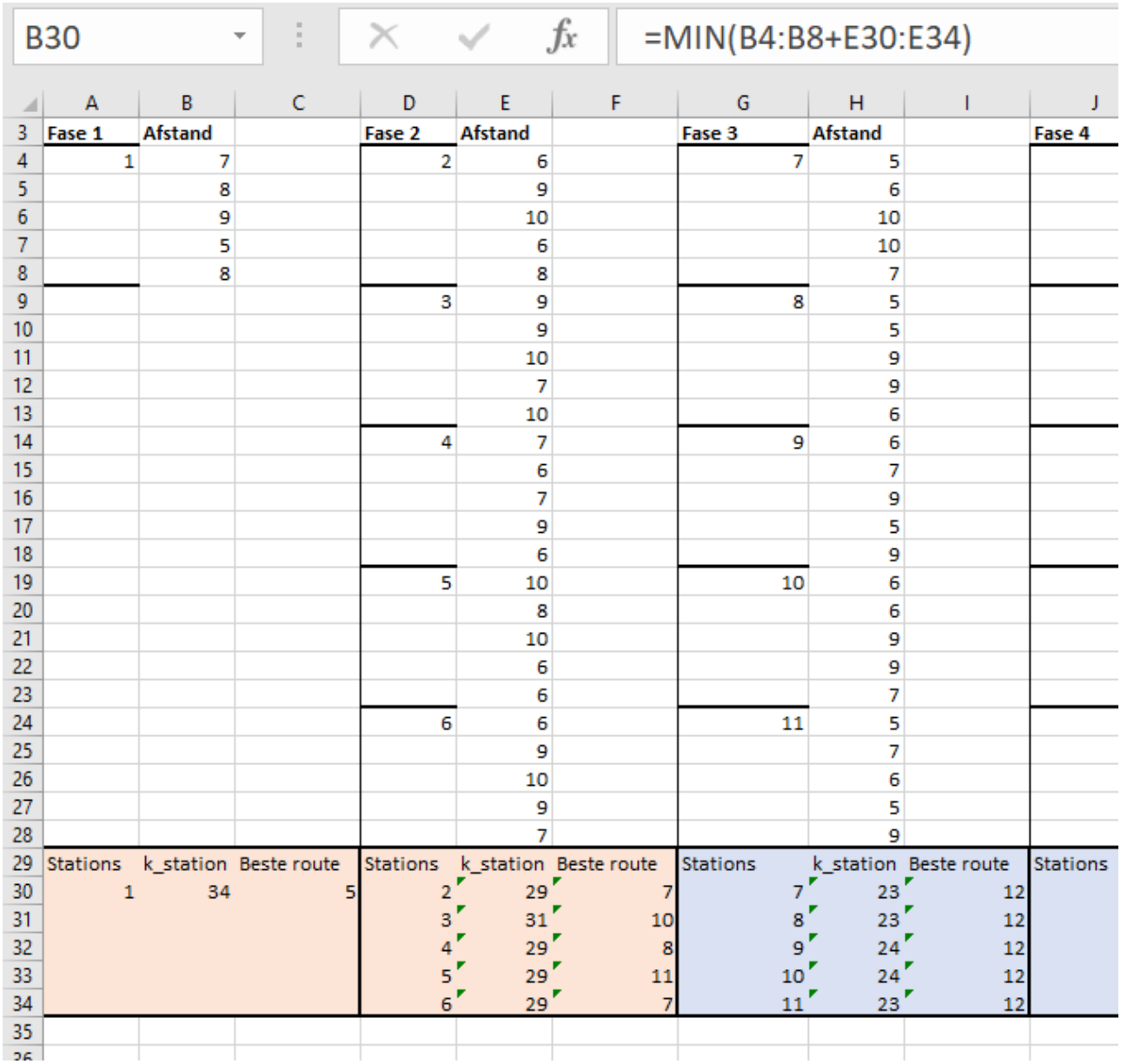
Figuur 5: *Een voorbeeld van een routeprobleem.*

Opdracht 1


RouteprobleemLes3
De optimale route is dan:
1
5
11
12
17
25
27

E30 =MIN(E4:E8+H30:H34)

	A	B	C	D	E	F	G	H	I	J
3	**Fase 1**	**Afstand**		**Fase 2**	**Afstand**		**Fase 3**	**Afstand**		**Fase 4**
4	1	7		2	6		7	5		12
5		8			9			6		
6		9			10			10		
7		5			6			10		
8		8			8			7		
9				3	9		8	5		13
10					9			5		
11					10			9		
12					7			9		
13					10			6		
14				4	7		9	6		14
15					6			7		
16					7			9		
17					9			5		
18					6			9		
19				5	10		10	6		15
20					8			6		
21					10			9		
22					6			9		
23					6			7		
24				6	6		11	5		16
25					9			7		
26					10			6		
27					9			5		
28					7			9		
29	Stations	k_station	Beste route	Stations	k_station	Beste route	Stations	k_station	Beste route	Stations
30	1	34	5	2	29	7	7	23	12	12
31				3	31	10	8	23	12	13
32				4	29	8	9	24	12	14
33				5	29	11	10	24	12	15
34				6	29	7	11	23	12	16
35										
36										



	A	B	C	D	E	F	G	H	I	J
3	**Fase 1**	**Afstand**		**Fase 2**	**Afstand**		**Fase 3**	**Afstand**		**Fase 4**
4	1	7		2	6		7	5		
5		8			9			6		
6		9			10			10		
7		5			6			10		
8		8			8			7		
9				3	9		8	5		
10					9			5		
11					10			9		
12					7			9		
13					10			6		
14				4	7		9	6		
15					6			7		
16					7			9		
17					9			5		
18					6			9		
19				5	10		10	6		
20					8			6		
21					10			9		
22					6			9		
23					6			7		
24				6	6		11	5		
25					9			7		
26					10			6		
27					9			5		
28					7			9		
29	Stations	k_station	Beste route	Stations	k_station	Beste route	Stations	k_station	Beste route	Stations
30	1	34	5	2	29	7	7	23	12	
31				3	31	10	8	23	12	
32				4	29	8	9	24	12	
33				5	29	11	10	24	12	
34				6	29	7	11	23	12	
35										

D Transcription of the interviews with students (Dutch)

In this appendix, the transcription of the interviews with the students is given. The letters S and I indicated what the student and the interviewer said, respectively. Since these interviews were conducted in Dutch, the transcription is also given in Dutch.

D.1 Student 1

I: Wat vond jij van de voorbeelden en de toepassingen die in de lessenserie naar voren kwamen?

S: De voorbeelden maakten het wel duidelijker en wat levendiger en beter te begrijpen. De toepassingen in DNA-technologie en routes interesseren mij dan ook nog weer van hoe wordt het daar dan gebruikt, vooral bij DNA.

I: Dat met DNA kwam niet precies naar voren, maar je weet wel dat het daar dus wordt gebruikt.

S: Ja.

I: Maarja, hoe dat precies werkt, dat is een heel andere lessenserie. Waren er volgens jou voorbeelden en uitleg over hoe je de opdrachten op moest lossen?

S: Het werd allemaal heel duidelijk beschreven in de lessenserie.

I: Fijn. Vind je ook dat je genoeg voorkennis had om de lessenserie uit te voeren?

S: Uhh, ja. Ik denk het wel, al is het een andere tak van wiskunde die niet veel getoetst wordt.

I: Nee, dat kan ik me goed voorstellen. Dat is inderdaad een heel andere manier. In de opdrachten, wist jij wat je moest doen en wat er van je werd verwacht?

S: Bij de meeste opdrachten overal wel. En af en toe kon ik wel gokken welke richting het zou moeten gaan. In dat opzichte kon ik wel alle opdrachten goed begrijpen en maken.

I: Ok, weet je ook nog bij welke opdracht je dacht van hmm, hier snap ik het niet zo goed of vind ik het moeilijk?

S: Volgens mij was dat met de boomdiagrammen, omdat mijn eigen kennis daar wat minder in was.

I: Ok, dankjewel. Vond je dat het type opdrachten/activiteiten gevarieerd was?

S: Eh, ja. Beginnend met zo'n luciferssspel en daarna doorgaan naar de wiskunde erachter liet voor mij echt zien hoe je dan iets uit de praktijk helemaal wiskundig toepast en dan op andere situaties gaat toepassen.

I: Ok, heel goed. Dan nu een stukje begrip. Kun jij uitleggen wat dynamisch programmeren is?

S: Dynamisch programmeren is een bepaald systeem, een complex systeem, beginnend bij een klein eindeel en vanaf daar terug proberen te redeneren hoe een groter systeem in elkaar steekt en hoeveel routes of mogelijkheden je op een bepaalde manier kan hebben.

I: Ja, ok. En kun je ook voorbeelden noemen van toepassingen van dynamisch programmeren?

S: Ja, routes bepalen in de kortste route, of DNA technologie zoals erin stond. Of anders misschien de manier om te winnen bij zo'n luciferssspel. Dat zijn wat voorbeelden uit de praktijk.

I: Ok heel goed. Heb je een computerprogramma gebruikt om een probleem op te lossen?

S: Ja, Excel!

I: Maar, heeft dit jou ook geholpen de theorie beter te begrijpen?

S: Uh, ja. Want als je dat eenmaal invult, dan maakt Excel de rest van de berekening af. Dus welke route nou het snelst was. Dat hielp mij wat om gewoon te kijken van ik had wel een vermoeden, ohja dat is inderdaad de korste route. Dus voor de complexere berekeningen is Excel echt een hulpmiddel om te kijken van, zit ik met mijn eigen gedachtegang op de goede weg en is dit ook wel echt de kortste route.

I: Ok ja, dankjewel. Stel nou dat ik jou nou zo'n klein routenetwerk zou geven zoals in les twee, denk je dat je dan met behulp van dynamisch programmeren de kortste route zou kunnen vinden?

S: Ja, ik denk het wel.

I: Kun je ook uitleggen waarom dynamisch programmeren sneller is om de korste route te vinden dan als je bijvoorbeeld alle routes met elkaar zou vergelijken?

S: Bij dynamisch programmeren begin je op het eind dus kun je alle andere langere routes al achterwege laten en krijg je dus al veel minder mogelijkheden om uit te zoeken wat nou de kortste is. Waarbij je als bij de normale manier, als je alle mogelijkheden neemt, ook alle langere routes mee moet nemen, dus ook de langste. Maar die wil je juist niet hebben.

I: Ok, heel goed. Even kijken, even over de types van opgaven. Waren er vragen die jou stap voor stap naar het antwoord toe leidden?

S: Ja. Volgens mij is dat les twee en les drie ook wel. Daar gingen ze echt stap voor stap richting het antwoord toe. Dus beginnend met een deelvraag en zo werken naar een eindconclusie.

I: Ok. En waren er ook meer vragen waarbij het wat meer moeite kostte dan anderen om het antwoord te vinden?

S: Eh ja, met Excel was iets meer werk. Maar echt moeite, dat weet ik zo niet.

I: Ok. En verder geen andere opdrachten waarbij je dacht dit vind ik moeilijk of hier moest ik wat meer zelf nadenken?

S: Eh, ik weet zo niet meer welke opdracht het was, maar er waren wel opdrachten waar ik wat langer over na moest denken omdat het een nieuw systeem is waar je mee werkt. Dus in het begin is het even wat langzamer.

I: Ok. En dan nog even de laatste vraag: heb jij nog opmerkingen over de lessenserie?

S: Ehm niet echt. Ik vond dat het goed in elkaar zat. Het was voor mij goed te volgen.

I: Ok, dankjewel!

D.2 Student 2

I: Wat vond je van de voorbeelden en de toepassingen in de lessenserie?

S: Ja, ik vond dat eigenlijk wel heel goed, wat jij bijvoorbeeld bij die eerste vraag hebt gedaan met die lucifers. Dat je eerst één keer het spelletje speelt, dan denk je er helemaal niet eens zoveel overna. Ja, dan heb je natuurlijk wel een idee dat er een tactiek achter zit. Maar daar denk je niet zo over na. En daarna ga je er dan echt er induiken en dan vond ik wel heel leuk. En dan maak je het ook wel echt duidelijk, dat alles echt op zijn plek valt zeg maar.

I: Dat is fijn. Waren er ook voorbeelden en uitleg over hoe je de opdrachten op moest lossen?

S: Uhm, oeh daar vraag je me wat. Over hoe je de opdracht op moest lossen?

I: Ja, was dat duidelijk, waren er voorbeelden over, was er uitleg over?

S: Oh, of het duidelijk was. Ja, in de voorafgaande tekst stond natuurlijk gewoon elke keer de theorie als het ware. En dan misschien niet helemaal toegepast op de vragen die daarna helemaal komen. Maar dat, je moet die theorie gewoon toepassen, dus dan begrijp je dat automatisch, want die theorie was duidelijk hoe dat vermeld was.

I: Dat is heel fijn. Vind je ook dat jij van tevoren genoeg wist, had je genoeg voorkennis om dit te doen?

S: Ja ja ja, dat zeker wel. Ik vond eigelijk, ik denk dat je hier gewoon zo in had kunnen stappen. Je moet wel wat van wiskunde weten, en die ene vraag met de afgeleide, ja dat moet je natuurlijk wel eerst weten. Maar ik denk dat je dit ook wel vrij goed kan volgen als je geen wiskunde D achtergrond hebt.

I: Ok, dankjewel. En wist je in de opdrachten wat je moest doen en wat ervan je werd verwacht?

S: Ja, dat was wel duidelijk in de vraag.

I: Ok, heel goed. En als je kijkt naar het type activiteiten en opdrachten, vond je dat dat gevarieerd was of dat het eentonig was?

S: Ja, er zat wel wat variatie in. Maar er waren ook vragen die wel op elkaar leken. Dus ik vond het eigenijk allebei wel, wat ik zeg, zoals die afgeleide dat was dan echt een totaal andere vrraag naar mijn idee. Maar er waren ook wel, ik weet zo niet meer welke opgave dat was, maar dat opgave a bijvoorbeeld bij de ene opdracht hetzelfde was als a bij de andere opdracht, maar dan net iets anders geformuleerd.

I: Ok dankjewel. Dan gaan we nu over naar een stukje begrip, of je nog wat weet van de lessenserie.

S: Oooh, haha.

I: Kun jij uitleggen wat dynamisch programmeren is?

S: Ja, dat is toch dat je van achter naar voor werkt, dus dat je aan het eind van een probleem begint en dan moet je een oplossing te vinden door van het eind naar het begin te werken.

I: Nou dat is heel goed, zo zou ik het ook omschrijven. Kun je ook wat voorbeelden noemen van toepassingen waar we dynamisch programmeren gebruiken?

S: Eh ja. Ehm, de manier om in Amsterdam te komen bijvoorbeeld met de trein, via welke stad, welke wegen, welke spoorlijnen het kortst is. Of via de weg gewoon, naar weet ik veel, naar Enschede naar de UT, ik zeg maar wat. Of ja, jij had ook een voorbeeld, dat staat me nog bij, van DNA, had je ook een voorbeeld in het boekje gezet dacht ik. En verder, dat is wel een beetje wat ik denk.

I: Ok heel goed, dankjewel. Heb je een computerprogramma gebruikt om een probleem op te lossen?

S: Ja, Excel op het laatst toch.

I: En heeft het je ook geholpen om de theorie beter te begrijpen?

S: Ja, nou ja in het begin snapten wij natuurlijk Excel niet heel goed. Maar op een gegeven moment, als je dat duidelijk hebt, dan kan het je wel helpen. Maar ik denk dat de theorie van voorafgaand, dat was duidelijk vond ik en beter te bevatten dan Excel, maar dat komt denk ik ook omdat ik de Excel achtergrond niet helemaal heb.

I: Maar denk je, van ik heb hier wat aan gehad om het ook met een computerprogramma te doen, of denk je dat het je niet echt heeft geholpen?

S: Jawel, dat wel. Het heeft wel geholpen.

I: Heb je er dan een beter beeld van?

S: Ja, het is allemaal veel beter nou. Ik vond het serieus een goeie lessenserie, goed opgebouwd.

I: Ok. Dankjewel. En stel dat ik jou nou zo'n klein routenetwerk zou geven net als in les 2, denk je dat je dan op zou kunnen oplossen met behulp van dynamisch programmeren?

S: Ik denk dat ik wel een eind zou komen ja.

I: Haha, ok dat is heel goed.

S: Ja ik denk het wel ja.

I: En kun je ook uitleggen, als je dynamisch programmeren gebruikt om de korste route te vinden in zo'n netwerk, waarom dat nou veel sneller is dan wanneer je dat zou doen als je alle routes met elkaar vergelijkt?

S: Oeh, da's een goeie vraag. Ehm, ja ik denk omdat als je aan het eind begint dan kun je al snel zien dat een aantal routes afvallen, bijvoorbeeld in die eerste stap naar links zeg maar, als je in dat schema een beetje keek, ja ik heb dat nou een beetje in mijn hoofd hoe dat in de lessenserie stond. Als je naar links gaat, voor de eerste stap zeg maar vanaf het eind, dan is er meteen al eentje die afvalt omdat die al langer is dan de rest. Dus ik denk dat dat een reden is waarom dynamisch programmeren beter werkt dan alle routes opzoeken.

I: Ja, daar ben ik het helemaal mee eens, heel goed. Je slaat wat over, want die langere routes hoef je niet meer mee te nemen inderdaad. Heel goed. Dan over het type vragen, waren er ook vragen die jou stap voor stap naar het antwoord toe leidden?

S: Ja, die waren er ook. Er was een opdracht, waarbij elke vraag echt één stap was.

I: Ok, dat begrijp ik, dat maakt opzich niet uit. Die waren er dus wel. En waren er ook vragen waarvan je zegt, daar moest ik zelf wat meer nadenken, dat was niet zo stap voor stap, dat kostte wat meer moeite?

S: Ja die laatste opdracht, die eindopdracht, dat was niet stap voor stap, maar dat was volgens mij ook niet helemaal de bedoeling. Ja, er was nog één andere vraag die die ik de eerste keer niet helemaal begreep, maar de tweede keer dat ik 'm beter las en dat ik wel wist wat er bedoeld werd, dat ik even iets beter moest lezen.

I: Weet je nog welke dat was?

S: Nee dat weet ik zo niet meer, maar anders kan ik zo dat boekje nog wel even pakken als je wil.

I: Als je het niet weet is niet erg, maar ik dacht is het misschien die dat je het aantal routes moest vinden?

S: Nee, ik geloof dat het een vraag was dat het ermee te maken had dat je via eentje moest, via station vijf ofzo.'

I: Ohja, ik weet al welke je bedoelt. Dat was volgens mij de eerste vraag in les 2, met dat kleine routenetwerk.

S: Ja.

I: Ok, dan weet ik welke opdracht je bedoelt, heel fijn. Dan als laatst, heb jij nog opmerkingen over de lessenserie?

S: Nou, eigenlijk niet. Het doel is goed bereikt zeg maar. Dynamisch programmeren is mij dus echt wel duidelijk wat dat inhoudt en wat het voordeel ervan is. Ik vond wel, dat jij erbij zat, dat vond ik wel fijn. Als je zo'n lessenserie zelf zou moeten doolopen, dan wordt het al wel lastiger. Maar dat komt ook omdat het onderwerp helemaal nieuw is. Maar het stond wel allemaal goed beschreven in de lessenserie zeg maar, je moet je wel kunnen redden.

I: Ok, nou heel fijn om te horen!

E Lesson Design

In the following section, the lesson designs can be found in Dutch and in English.

E.1 Dutch version

Les 1 - Introductie Dynamisch Programmeren

Opdracht 1
Jullie gaan een strategisch spel spelen, namelijk het lucifer-spel. Dit spel spelen jullie in tweetallen, speler A en B. De spelregels (lees ze allemaal eerst helemaal door):

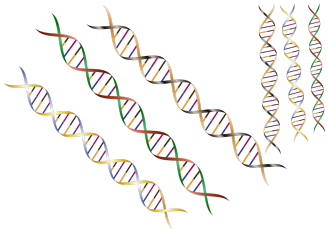
1. Begin met 30 lucifers op tafel.
2. Kies de persoon die gaat beginnen, speler A. Speler A mag kiezen om 1, 2 of 3 lucifers van tafel te halen.
3. Nu is het de beurt van speler B. Ook speler B mag nu 1, 2 of 3 lucifers van tafel halen. Dit herhalen jullie om en om. Je moet altijd minimaal 1 lucifer van tafel halen.
4. De speler die de laatste lucifer van tafel moet halen, verliest. Bijvoorbeeld: het is de beurt van speler A en er liggen 3 lucifers. Als speler A nu 2 lucifers van tafel haalt, dan ligt er nog 1. Speler B moet deze nu wel van tafel halen en verliest. Veel plezier!

Opdracht 2
Het is mogelijk om een strategie te bedenken waardoor speler A (de speler die begint) met zekerheid wint met het lucifer-spel. Probeer een strategie te bedenken waarmee speler A wint. Schrijf hiervoor een stappenplan op, zodanig dat speler A wint.

Opdracht 3
Zonet is besproken hoe zo'n strategie gevonden kan worden, namelijk door eerst te kijken wat er aan de hand is met maar 1 lucifer. Het vinden van een optimale strategie/aanpak op deze manier is een voorbeeld van **dynamisch programmeren**. Bij dynamisch programmeren proberen we problemen op te lossen door het probleem in kleinere stukken te hakken, of bijvoorbeeld door "aan het eind" te beginnen. Als we de oplossing hebben gevonden van een klein probleem, dan gebruiken we die voor het oplossen van het grotere probleem.

Vraag
Hoe is dynamisch programmeren gebruikt in opdracht 2?

Andere voorbeelden van toepassingen zijn het vergelijken van DNA-ketens en het vinden van de snelste route. Dit laatste voorbeeld gaan we gebruiken om meer te weten te komen over dynamisch programmeren, wat in de volgende lessen aan bod komt.



Figuur 17: *Vergelijken van DNA-ketens.*

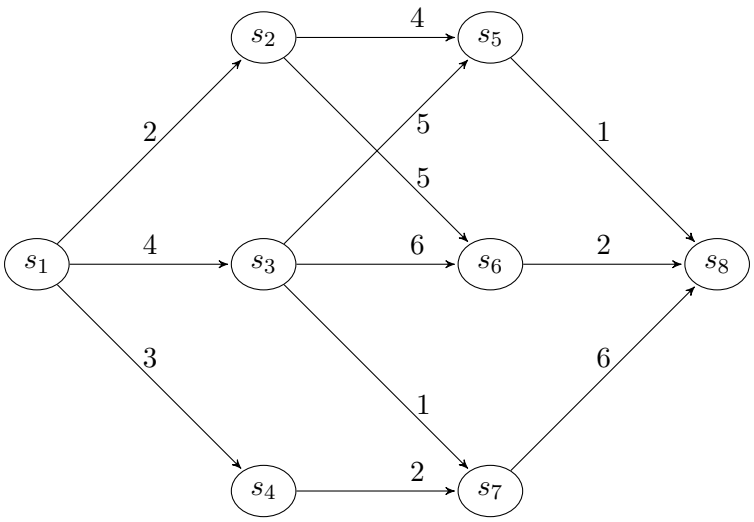
Figuur 18: *Vinden van de snelste route.*

Les 2 - Het routeprobleem

In de vorige les zijn er voorbeelden aan bod gekomen waarbij dynamisch programmeren gebruikt kan worden. Eén van deze voorbeelden was het routeprobleem. Een voorbeeld van een route is te zien in Figuur 19.

Stel dat dit routeprobleem een treinnetwerk voorstelt. De rondjes stellen de **stations** voor. Het beginstation is station s_1. De pijlen tussen de verschillende stations geven aan waar je naartoe mag reizen vanaf een station. Bij elke pijl staat aangegeven hoe lang het duurt om tussen deze stations te reizen, wat we hier de **kosten** noemen.

Bijvoorbeeld: vanaf station s_3 kunnen we naar station s_5 reizen met kosten 5. Ook kan er worden gereisd naar station s_6 met kosten 6 en naar station s_7 met kosten 1. Hoe lager de kosten, hoe korter de reistijd.



Figuur 19: *Een voorbeeld van een routeprobleem.*

Opdracht 1

Het doel is om vanaf station s_1 naar station s_8 te reizen, met de kosten zo laag mogelijk (wat betekent dat we zo snel mogelijk willen reizen). Voordat we dit doen, bekijken we eerst waarom dynamisch programmeren hier handig kan zijn.

(a) Voor dynamisch programmeren kijken we naar kleinere problemen. Bijvoorbeeld, als we van station s_3 naar station s_8 willen reizen. Laat met een berekening zien dat de snelste route van station s_3 naar station s_8 via station s_5 is. Hoe hoog zijn de kosten?

(b) Waarom hoef je niet de kosten van de routes $(s_1) \to (s_3) \to (s_6) \to (s_8)$ en $(s_1) \to (s_3) \to (s_7) \to (s_8)$ te berekenen om erachter te komen wat de korste route is van station s_1 naar s_8?

(c) Bereken de kosten van de snelste route van station s_1 naar station s_8, via station s_3. Gebruik hiervoor vraag (a).

Dit soort berekeningen gaan we steeds uitvoeren om erachter te komen wat de snelste route

is. Hiervoor willen we een gestructureerde aanpak bedenken. Dat doen we als volgt. In het plaatje zien we al een structuur met "groepjes", namelijk s_2, s_3 en s_4 staan bij elkaar en s_5, s_6 en s_7. Daarom verdelen we de stations in vier verschillende fases:

- *Fase 1:* (s_1).
- *Fase 2:* (s_2), (s_3), (s_4).
- *Fase 3:* (s_5), (s_6), (s_7).
- *Fase 4:* (s_8).

We gaan de oplossing zoeken met behulp van dynamisch programmeren. Dat betekent dat we in *Fase 4* beginnen en dus bij station s_8 "beginnen". De stappen die we elk fase doorlopen zijn de volgende:

1. Bekijk voor elk station hoe we het snelst bij station s_8 komen vanaf het huidige station. De kosten van de gekozen route zijn de minimale kosten.
2. Onthoud wat vanaf het huidige station de snelste keuze was, dus we onthouden naar welk station we daarna moeten reizen.

We maken gebruik van de volgende tabel:

Fase	**Station**	**Mogelijke routes naar**	**Minimale kosten**	**Volgende station?**
4	(s_8)	Geen.	0	Geen.
3	(s_5)	(s_8): Kosten 1	1	(s_8)
	(s_6)	(s_8): Kosten 2	2	(s_8)
	(s_7)	(s_8): Kosten 6	6	(s_8)
2	(s_2)	(s_5): Kosten $4 + 1 = 5$, (s_6): Kosten $5 + 2 = 7$,	5	(s_5)
	(s_3)	(s_5): Kosten $5 + 1 = 6$, (s_6): Kosten $6 + 2 = 8$, (s_7): Kosten $1 + 6 = 7$,	6	(s_5)
	(s_4)	(s_7): Kosten $2 + 6 = 8$	8	(s_7)
1	(s_1)	(s_2) (s_3) (s_4)		

In de tabel is het volgende gedaan. In *Fase 4* kiezen we geen volgende route. De minimale kosten om vanuit station s_8 naar station s_8 te gaan is 0: we zijn er namelijk al.

Vervolgens kijken we naar alle stations in *Fase 3*. Bijvoorbeeld, vanaf station s_5 kunnen we alleen naar station s_8 gaan met kosten 1. Dus, de minimale kosten om van station s_5 naar

station s_8 te gaan zijn gelijk aan 1. We kunnen alleen naar station s_8, dus dat is het volgende station. Zo is dit ook gedaan voor de andere stations in *Fase 3*.

We herhalen dit voor *Fase 2* en *Fase 1*. We schetsen de berekening voor station s_3. Vanuit station s_3 kunnen we naar station s_5, s_6 of s_7. De kosten om van station s_3 naar station s_8 te gaan via station s_5 zijn gelijk aan de kosten om van station s_3 naar station s_5 te gaan (=5), plus de minimale kosten om van station s_5 naar station s_8 te gaan (= 1), wat we uit *Fase 3* halen. Dat komt neer op kosten van 6. Dit doen we ook voor de andere stations en vinden dat de minimale kosten gelijk zijn aan 6. Dus, vanuit station s_3 kiezen we als volgende station s_5, want deze zorgt voor de laagste kosten.

Opdracht 2
Het doel is nu om te kijken wat de optimale route is vanuit station s_1. In *Fase 1* beginnen we in station s_1. Hier kunnen we naar station s_2, s_3 of s_4. Met behulp van de minimale kosten vanuit de stations uit *Fase 2* kunnen we de optimale route vinden.

(a) Bereken de kosten om van station s_1 naar station s_8 te gaan via alle drie stations in *Fase 2*. Vul deze waarden in in de tabel.

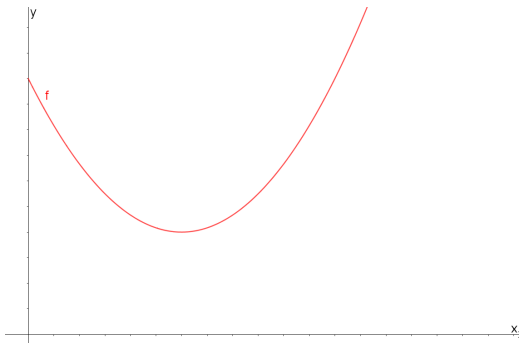
(b) Als het goed is, is één van deze waardes minimaal. Wat betekent deze waarde?

(c) Maak de tabel helemaal af. Wat is de optimale route? Kleur deze route in Figuur 19.

We hebben nu de tabel afgemaakt. Omdat we de kosten zo laag mogelijk willen houden, hebben we steeds de route gekozen die de minimale kosten geeft. In feite hebben we dus steeds het minimum berekend. Het berekenen van een minimum is iets wat jullie wel kunnen gegeven een functie. Dit gaan we in de volgende opdracht herhalen en leren hier nieuwe notatie voor.

Opdracht 3
Het berekenen van een minimum gegeven een functie f hebben jullie al vaak gedaan.

(a) Gegeven de functie $f(x) = \frac{1}{3}x^2 - 2x + 5$, voor $x \geq 0$, met de volgende grafiek:



Figuur 20: *De grafiek van f.*

Het minimum is de minimale functiewaarde van $f(x)$. De x-waarde die we invullen noemen we ook wel het argument. Laat met behulp van de afgeleide zien dat het minimum optreedt bij het argument $x = 3$ en dat het minimum gelijk is aan 2.

Dit kunnen we weer koppelen aan het routeprobleem. In elke fase van het routeprobleem kiezen we een route waarvoor de kosten minimaal zijn en vraag (a) "kiezen" we een x waarvoor $f(x)$ minimaal is. Dit zijn voorbeelden van het vinden van een minimum, waarbij een bepaalde "keuze" (de route of x-waarde) zorgt voor dat minimum. We introduceren hiervoor twee belangrijke notaties aan de hand van wat we net hebben gedaan:

- We variëren de x-waarde (het argument) om zo de minimale waarde te vinden van $f(x)$, wat we schrijven als $\arg\min_x\{f(x)\}$. Het subscript x betekent dat het de x is die we variëren. In ons voorbeeld was deze x-waarde gelijk aan 3, dus hier geldt $\arg\min_x\{f(x)\} = 3$.
- Als je dit argument hebt gevonden, dan kun je het minimum berekenen. Dat minimum noteren we dan op een vergelijkbare manier: $\min_x\{f(x)\}$. Ook hier betekent het subscript x weer dat we de x-waarde kunnen variëren. In ons voorbeeld van vraag (a) was het minimum gelijk aan 2 (door $\arg\min_x\{f(x)\} = 3$ in te vullen), dus hier geldt $\min_x\{f(x)\} = f(3) = 2$.

Deze notatie wordt veel gebruikt bij wiskunde en komt bijvoorbeeld terug bij programmeren, waarbij bijvoorbeeld het routeprobleem voor grote netwerken kan worden opgelost. Houd nu de tabel van pagina 84 erbij. We gaan oefenen met de notatie die we net hebben geleerd.

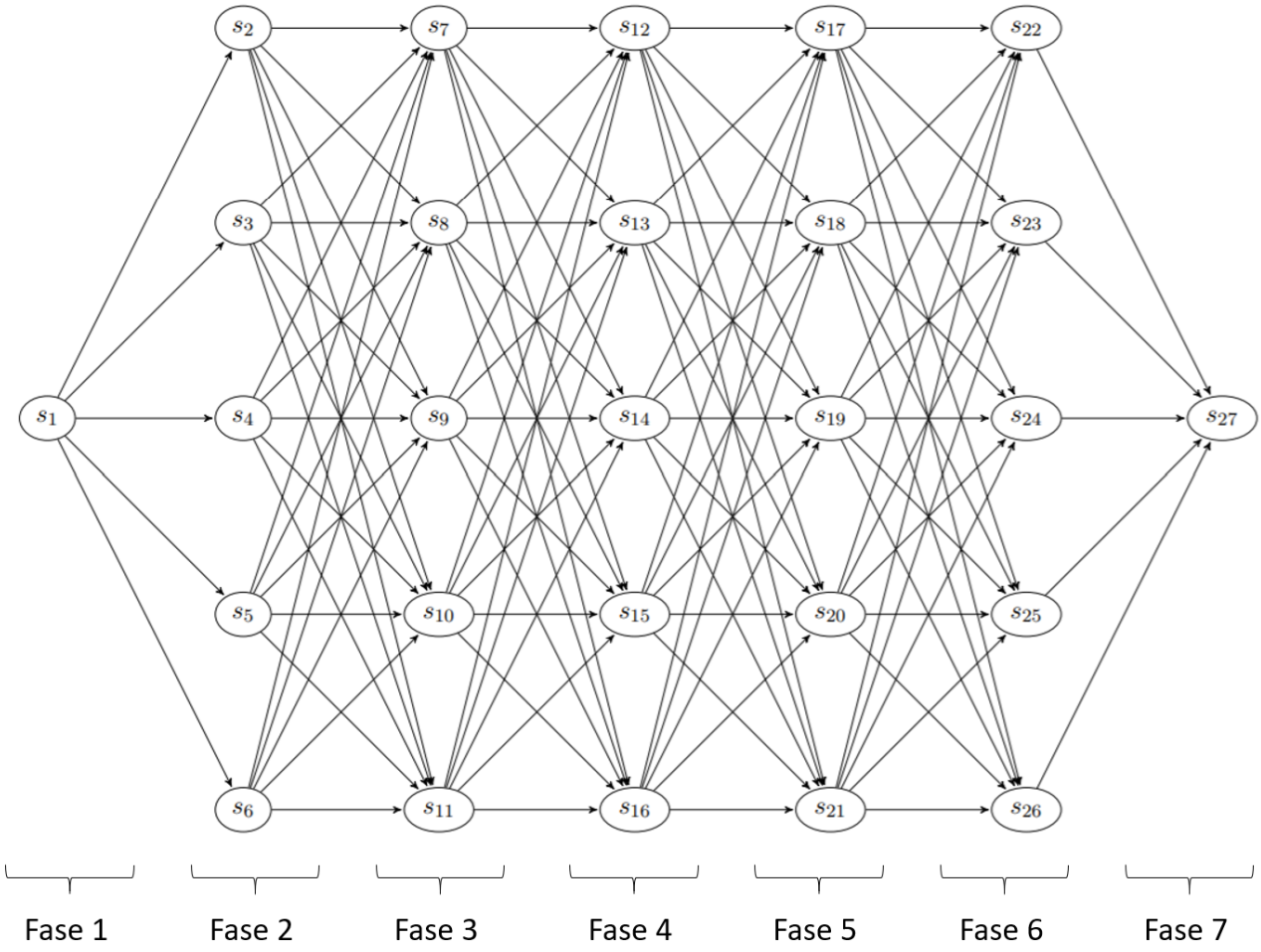
Kijk eens naar station s_2 in *Fase 2* in de tabel. We hebben al berekend dat de minimale kosten gelijk zijn aan 5, want als we moeten kiezen tussen kosten 5 of 7, dat we dan 5 kiezen (we variëren de stations waar we naartoe kunnen reizen). Het station dat we hebben gekozen was station s_5 en niet station s_6 (dus het argument voor deze kosten is station s_5). Daarom schrijven we $\arg\min_{\text{station}}\{kosten(station)\} = s_5$. De minimale kosten zijn dan de kosten die we krijgen als we naar dit station reizen, die gelijk zijn aan 5. Als we de notatie gebruiken in dit voorbeeld, dan schrijven we dat als $\min_{\text{station}}\{kosten(station)\} = 5$ (we nemen dus het minimum over de stations, waarbij de minimale waarde dus 5 is).

(b) Gebruik deze notatie om ook het minimum en het argument voor het minimum te geven voor de andere stations in *Fase 2*, dus voor stations s_3 en s_4.

(c) Als je kosten en vervolgstations in termen van $f(x)$ en x zou omschrijven, wat is dan $f(x)$ en wat is dan x?

Les 3 - Het routeprobleem met Excel

In Figuur 21 zien we een nieuw routeprobleem. We zien hier weer stations, genaamd s_1 t/m s_{27} en de pijlen geven de mogelijke routes aan. Er staan hier geen getallen bij de pijlen, omdat het dan onoverzichtelijk wordt, omdat vanuit elk station uit elke fase een route is naar elk station uit de volgende fase.



Figuur 21: *Een voorbeeld van een routeprobleem.*

Opdracht 1

In deze opdracht gaan we kijken naar het routeprobleem uit Figuur 21. Als we willen weten wat de snelste route is van station s_1 naar station s_{27}, dan moeten we de kosten van de routes weten. Echter, eerst willen we iets meer weten over het **aantal** routes en daarmee het aantal berekeningen.

(a) Hoeveel mogelijke routes zijn er om van station s_1 naar station s_{27} te reizen?

(b) Stel dat er nog een extra fase bijkomt tussen fase 6 en 7 (dus weer 5 extra stations). Hoeveel **extra** routes krijgen we dan? En als er daarna nog een fase van 5 stations bijkomt?

(c) We nemen aan dat we gewoon de situatie hebben zoals in Figuur 21. Stel dat er tussen fase 5 en 6 een extra fase bijkomt van 5 stations. Wat voor invloed heeft dat op het aantal berekeningen die we uit moeten voeren als:

i. We de lengte van **alle** routes berekenen en het minimum zoeken?

ii. We de minimale route zoeken met Dynamisch Programmeren?

Geef hiermee een conclusie over de efficiëntie van manier i. en van ii. als we bijvoorbeeld 100 fases zouden toevoegen. Gebruik in je conclusie de termen "lineair verband" en "exponentieel verband".

We gaan het nu concreter maken. Voor elk station zijn de kosten bekend om naar de andere stations te reizen van de volgende fase. De kosten van station s_1 naar station s_2 gelijk zijn aan 7. Ook de kosten van station s_1 naar de andere stations zijn bekend. Die zijn als volgt:

- $s_1 \rightarrow s_2$: kosten 7.
- $s_1 \rightarrow s_3$: kosten 8.
- $s_1 \rightarrow s_4$: kosten 9.
- $s_1 \rightarrow s_5$: kosten 5.
- $s_1 \rightarrow s_6$: kosten 8.

Deze waardes staan ook in de eerste kolom van Tabel 6. De waardes staan dus voor de kosten naar de stations in de volgende fase, in de volgorde van boven naar beneden (zoals in Figuur 21). We gaan eerst eens kijken naar de andere gegevens in de tabel.

Tabel 6: *De afstand tussen de verschillende stations van de volgende fase.*

Fase 1		**Fase 2**		**Fase 3**		**Fase 4**		**Fase 5**		**Fase 6**		**Fase 7**
s_1	7	s_2	6	s_7	5	s_{12}	5	s_{17}	10	s_{22}	6	s_{27}
	8		9		6		8		10			
	9		10		10		10		6			
	5		6		10		5		8			
	8		8		7		10		10			
		s_3	9	s_8	5	s_{13}	8	s_{18}	8	s_{23}	8	
			9		5		8		7			
			10		9		5		9			
			7		9		7		8			
			10		6		8		6			
		s_4	7	s_9	6	s_{14}	8	s_{19}	9	s_{24}	8	
			6		7		7		9			
			7		9		7		8			
			9		5		5		9			
			6		9		5		8			
		s_5	10	s_{10}	6	s_{15}	9	s_{20}	9	s_{25}	5	
			8		6		10		7			
			10		9		10		9			
			6		9		9		8			
			6		7		9		9			
		s_6	6	s_{11}	5	s_{16}	6	s_{21}	6	s_{26}	7	
			9		7		5		9			
			10		6		6		10			
			9		5		5		9			
			7		9		8		8			

Opdracht 2
Kijk nog eens naar Tabel 6. De rest van de tabel is net zo opgebouwd als de kolom van *Fase 1*.

(a) Wat betekent de waarde 6 die bij station s_2 staat? En wat betekent de 9 die bij station s_2 staat?

(b) Kijk eens naar de kolom van *Fase 6*. Waarom is er per station maar 1 waarde?

(c) Geef een verklaring waarom er bij station s_{27} geen waarde is gegeven. Als je station s_{27} een waarde moest geven, welke waarde zou dat dan zijn (ofwel, hoe lang duurt het om vanaf s_{27} naar s_{27} te reizen)?

Om dit op te lossen, zouden we alles met de hand uit kunnen schrijven. Ook al doen we dit met behulp van dynamisch programmeren, het blijft veel werk om dat allemaal te berekenen. Gelukkig kunnen hiervoor computerprogramma's gebruiken om ons te helpen. Wij gaan dit probleem op lossen met behulp van Excel.

Opdracht 3
Open het bestand RouteprobleemLes3.xl op je computer. In dit bestand staat een vergelijkbare tabel als Tabel 6. Toegevoegd is dat de afstand van station s_{27} naar zichzelf gelijk is aan 0. In dit bestand heten de stations station 1, 2, 3,... in plaats van $s_1,\ s_2,\ s_3, \ldots$.

Onderaan de tabel staan 6 gekleurde blokken/tabellen met allemaal de volgende gegevens:

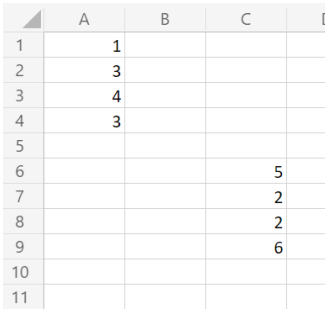
- Stations: hieronder staan de stations in de fase erboven.
- k_station: dit zijn de minimale kosten om te reizen naar het eindstation (27), vanaf dat station.
- Beste route: dit geeft aan welke route als volgt moet worden gekozen.

Met dynamisch programmeren beginnen we aan het eind van het probleem. Dat eind is weer simpel, want bij station s_{27} zijn we er al. Dan gaan we terugwerken, waarbij we de berekeningen uitvoeren in de verschillende blokken. De eerste twee blokken zijn roodgekleurd: deze zijn nog niet ingevuld. De vier andere zijn wel ingevuld. Aan jullie de taak om de rode blokken ook in te vullen. Dan verschijnt onder het kopje "De optimale route is dan:" een reeks met stations die ervoor zorgen dat de kosten minimaal zijn. Beantwoord de volgende vraag:

Vraag: Kijk eens naar station s_{25} in het laatste blok. Wat betekenen de 5 en 27? En als we kijken naar station s_{21}, wat betekenen de 12 en 22?

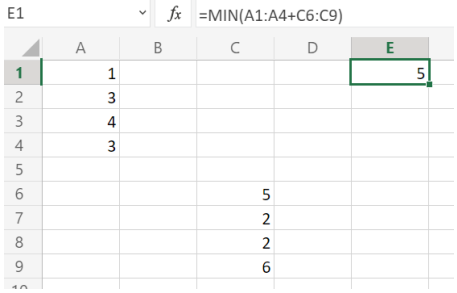
Uitleg Excel (voor de eindopdracht)
We focussen nu op fase 2, die we graag in willen vullen. De beste routes zijn nog onbekend, dus die staan nu op 0 (die weten we namelijk niet). We gebruiken het minimum uit fase 3 om voor elk station uit fase 2 de laagste kosten te vinden om naar het eindstation te reizen, wat precies dezelfde aanpak vereist als we in de vorige les hebben gedaan. In Excel kan dat met de functie MIN. Dit gaan jullie straks zelf doen, maar lees eerst de instructies hieronder. Jullie hebben nodig dat je het minimum kunt berekenen van de som van twee kolommen (dit komt dus terug in de eindopdracht). We laten nu zien hoe je dat moet doen:

1. Stel je hebt een kolom met getallen in de vakken $A1$ tot en met $A4$ en in de vakken $C6$ tot en met $C9$, zie het plaatje van Figuur 22:



Figuur 22: *Het voorbeeld voor het gebruiken van* MIN.

2. Selecteer een leeg vakje waar je de minimale waarde wilt hebben. In ons voorbeeld nemen we $E1$. We selecteren het vakje $E1$ en vullen in $=\min(A1{:}A4{+}C6{:}C9)$. Excel telt nu de waardes van de eerste rij van beide kolommen bij elkaar op $(A1 + C6)$, de getallen van de tweede rij van beide kolommen $(A2 + C7)$, enz. Dan berekent Excel het minimum, wat 5 is (dat is hier dus de som van de getallen van de tweede rij, namelijk 3 + 2). Zie de figuur hieronder.



Figuur 23: *Het antwoord voor het gebruiken van de functie* MIN.

Eindopdracht

In deze eindopdracht gaan jullie berekenen hoe lang de kortste route is om van station s_1 naar station s_{27} te reizen. Gebruik hiervoor het bestand van de vorige opgave (Routeprobleem-Les3.xl).

(a) Bereken hiertoe eerst de optimale kosten van station s_2 naar station s_{27}. Gebruik de aanpak van de vorige les. Je zult de uitleg over het berekenen van een minimum in Excel nodig hebben. De beste route vanaf station s_2 verschijnt nu ook, namelijk via station s_7. *Let op: je moet dus het vakje k_station invullen voor station* s_2.

(b) Doe hetzelfde voor de andere station in fase 2. Geef van elk station de laagste kosten om naar station s_{27} te reizen. Als het goed is, verschijnen de beste routes vanaf elk station in fase 2. *Let op: als er 0 blijft staan bij Beste route, dan klopt je optimale waarde nog niet.*

(c) Tot slot, wat zijn de minimale kosten om van station s_1 naar station s_{27} te reizen? Geef de optimale route door het schema hieronder in te vullen:

$(s_1) \rightarrow \ldots \rightarrow \ldots \rightarrow \ldots \rightarrow \ldots \rightarrow \ldots \rightarrow (s_{27})$

Dit is het einde van de lessenserie.

E.2 English version

Lesson 1 - Introduction Dynamic Programming

Exercise 1
You are going to play a strategy game, the match-game. This game is played in pairs, with player A and player B. The rules of the game (read them carefully before you start playing):

1. Start with 30 matches on the table.
2. Choose the person to start, who will be player A. Player A can choose to pick up either 1, 2 or 3 matches from the table.
3. Now it is player B's turn. Player B can also pick up 1, 2 or 3 matches from the table. You repeat this alternately. You always have to pick up at least 1 match from the table.
4. The player that has to pick up the last match from the table loses. For example: it is player A's turn and there are 3 matches on the table. If player A picks up 2 matches from the table, then there is 1 left on the table. Player B must pick up this one and loses. Enjoy playing the game!

Exercise 2
It is possible to construct a strategy to ensure that player A (the starting player) wins the match-game. Try to come up with such a strategy to guarantee that player A wins. Write down the steps that player A has to follow to ensure victory.

Exercise 3
We just discussed how such a strategy can be found. The key is to first analyse the situation when a single match is on the table. Finding an optimal strategy/approach this way is an example of **dynamic programming**. For dynamic programming, problems are solved by breaking them up in smaller chunks, or by starting "at the end" for example. When the solution is obtained for a smaller problem, this is then used to solve the larger problem.

Question
How has dynamic programming been used in exercise 2?

Other applications are comparing DNA-sequences and finding the shortest route. The latter example will be used to learn more about dynamic programming. This will be done in the coming two lessons.

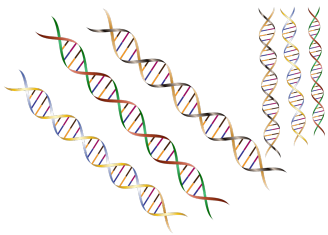


Figure 24: *Comparing DNA-sequences.*

Figure 25: *Finding the shortest route.*

Lesson 2 - The routing problem

In the previous lesson, you have seen several applications of dynamic programming. One of these applications was the routing problem. An example of a route can be seen in Figure 26.

Suppose that this routing problem is a network of trains. The circles indicate **stations**. The starting station is station s_1. The arrows between the different stations show to which other stations you can travel. Additionally, the numbers next to the arrows indicate how long it takes to travel between these stations, which we call the *costs*.

For example: from station s_3 we can go to station s_5 with costs 5. Also, we could travel to station s_6 with costs 6 and to station s_7 with costs 1. The lower the costs, the shorter the travel time.

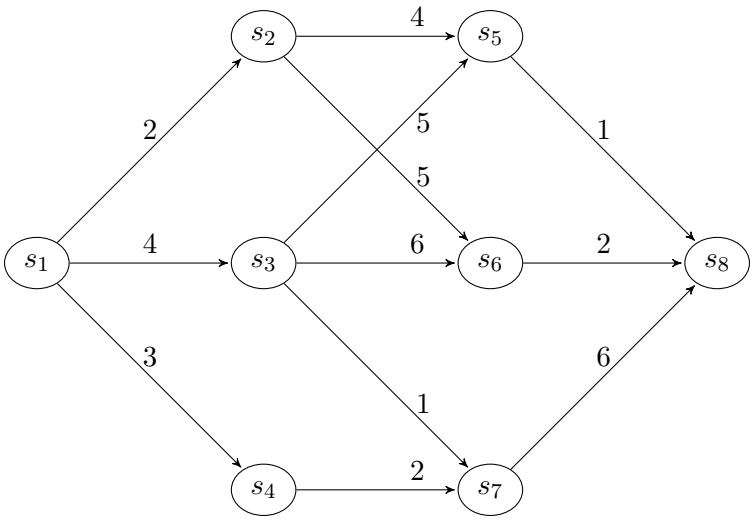


Figure 26: *An example of a routing problem.*

Exercise 1

The goal is to go from station s_1 to station s_8, with the lowest possible costs (meaning we want to travel as quickly as possible). Before doing this, we will first investigate why dynamic programming is useful here.

(a) When we apply dynamic programming, we can look for smaller sub-problems. For example, travelling from station s_3 to station s_8. Show with a calculation that the fastest route from station s_3 to station s_8 is by travelling via station s_5. How high are the costs?

(b) Why is it not necessary to compute the costs of the routes $(s_1) \to (s_3) \to (s_6) \to (s_8)$ and $(s_1) \to (s_3) \to (s_7) \to (s_8)$ to find out what the fastest route is from station s_1 to s_8?

(c) Compute the costs of the fastest route from station s_1 to station s_8 via station s_3. You can use question (a).

We will continue doing this type of computations to find the fastest route. We want to come up with a well-structured approach. We do this as follows. In the figure, we already see some form of a structure with "groups", namely s_2, s_3 and s_4 are together and s_5, s_6 and s_7 as well.

Therefore, we divide the stations in four stages:

- *Stage 1:* (s_1).
- *Stage 2:* (s_2), (s_3), (s_4).
- *Stage 3:* (s_5), (s_6), (s_7).
- *Stage 4:* (s_8).

We are going to find the solution by using dynamic programming. This means we start in *Stage 4*, thus we "start" at station s_8. The steps we go through in each stage are the following:

1. Check for each station what the fastest way is to go to station s_8 from the current station. The costs of the chosen route are the minimal costs.
2. Remember what the best choice was for the next station, so we remember to which station we have to travel next.

We will use the following table:

Stage	**Station**	**Possible routes to**	**Minimal costs**	**Next station?**
4	(s_8)	None.	0	None.
3	(s_5)	(s_8): Costs 1	1	(s_8)
	(s_6)	(s_8): Costs 2	2	(s_8)
	(s_7)	(s_8): Costs 6	6	(s_8)
2	(s_2)	(s_5): Costs $4+1=5$, (s_6): Costs $5+2=7$,	5	(s_5)
	(s_3)	(s_5): Costs $5+1=6$, (s_6): Costs $6+2=8$, (s_7): Costs $1+6=7$,	6	(s_5)
	(s_4)	(s_7): Costs $2+6=8$	8	(s_7)
1	(s_1)	(s_2) (s_3) (s_4)		

In the table, the following has been done. In *Stage 4* we do not choose a next station. The minimal costs to travel from station s_8 to station s_8 is equal to 0: we're already there.

Next, we look at all the station in *Stage 3*. For example, from station s_5 we can only travel to station s_8 with costs 1. Thus, the minimal costs to go from station s_5 to station s_8 are equal to 1. We can only go to station s_8, so this will also be the next station. This approach has also been done for the other station in *Stage 3*.

We repeat this process for *Stage 2* and *Stage 1*. We give an outline for the computation for station s_3. From station s_3 we can go to stations s_5, s_6 or station s_7. The costs to go from

station s_3 to station s_8 via station s_5 are equal to the costs to go from station s_3 to station s_5 (=5), plus the minimal costs to go from station s_5 to station s_8 (=1), what we can find in *Stage 3*. This results in a total costs of 6. We will also do this for the other stations and find that the minimal costs are equal to 6. So, from station s_3 we choose s_5 to be our next station, because this results in the lowest costs.

Exercise 2

Our goal is to find the optimal route from station s_1. In *Stage 1* we start in station s_1. From here, we can go to station s_2, s_3 or s_4. Using the minimal costs from the stations in *Stage 2* we can find the optimal route.

(a) Compute the costs to go from station s_1 to station s_8 via all three stations in *Stage 2*. Write these values down in the table.

(b) If you did this correctly, one of these values is minimal. What does this value mean?

(c) Complete the table. What is the optimal route? Colour this route in Figure 26.

We now completed the table. Because we want to have the costs as low as possible, we kept choosing the route that gave us the minimal costs. In other words, we have computed the minimum over and over again. You already know how to compute the minimum of a function. This is what you will revise in the next exercise, and you will learn some new notation for this.

Exercise 3

Computing a minimum for a function f is something you have done quite often.

(a) Given the function $f(x) = \frac{1}{3}x^2 - 2x + 5$, for $x \geq 0$, with the following graph:

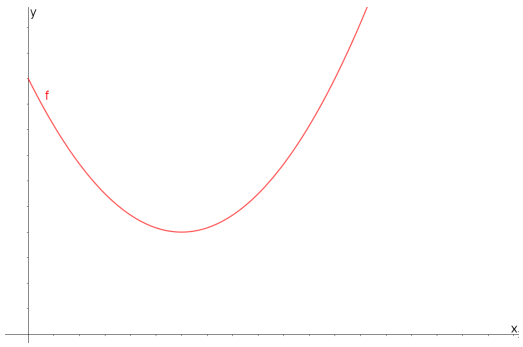


Figure 27: *The graph of f.*

The minimum is the minimal function value of $f(x)$. The x-value that is used as an input is called the argument. Show, using the derivative, that the argument $x = 3$ yields a minimum of 2.

This can be connected to the routing problem. In each stage of the routing problem, we choose a route for which the costs are minimal and in question (a) we "choose" an x-value for which $f(x)$ is a minimum. These are examples of finding a minimum, where a certain "choice" (the route or x-value) results in a minimum. We introduce two important concepts:

- We vary the x-value (the argument) to find the minimum of $f(x)$, which we write as $\arg\min_x\{f(x)\}$. The subscript x means that it is x that we vary. In our example this x-value equalled 3, thus we have $\arg\min_x\{f(x)\} = 3$.
- When you found this argument, you can compute the minimum. That minimum we write in a similar way: $\min_x\{f(x)\}$. The subscript x again means that we can vary the x-value. In the example of question (a) the minimum equalled 2 (by inserting $\arg\min_x\{f(x)\} = 3$), so here we have $\min_x\{f(x)\} = f(3) = 2$.

This notation is widely used in mathematics and is seen a lot in programming languages, in which larger networks of the routing problem can be solved. Go back to the table on page 95. We are going to practise with the notation we just learnt.

Take a look at station s_2 in *Stage 2* in the table. We already computed that the minimal costs are equal to 5, because if we were to choose between the costs of 5 and 7, we would choose 5 (we vary the station to which we can travel). The station we have chosen was station s_5 and not station s_6 (so the argument for these costs is station s_5). Therefore, we write $\arg\min_{\text{station}}\{costs(station)\} = s_5$. The minimal costs are then the costs we obtain when we travel to this station, which are equal to 5. When we use the notation in this example, we write it as $\min_{\text{station}}\{costs(station)\} = 5$ (so we take the minimum over the stations, with a minimal value of 5).

(b) Use this notation to find the minimum and the argument for the minimum for the other stations in *Stage 2*, so for stations s_3 and s_4.

(c) If you were to describe $f(x)$ and x in terms of costs and next station, what is $f(x)$ and what is x?

Lesson 3 - The routing problem with Excel

In Figure 28 we see a new routing problem. Again, there are stations, called s_1 to s_27 and the arrows indicate possible routes. The numbers are not given next to the arrows, since then it will become confusing, because from each station in each stage there is a route to each station of the next stage.

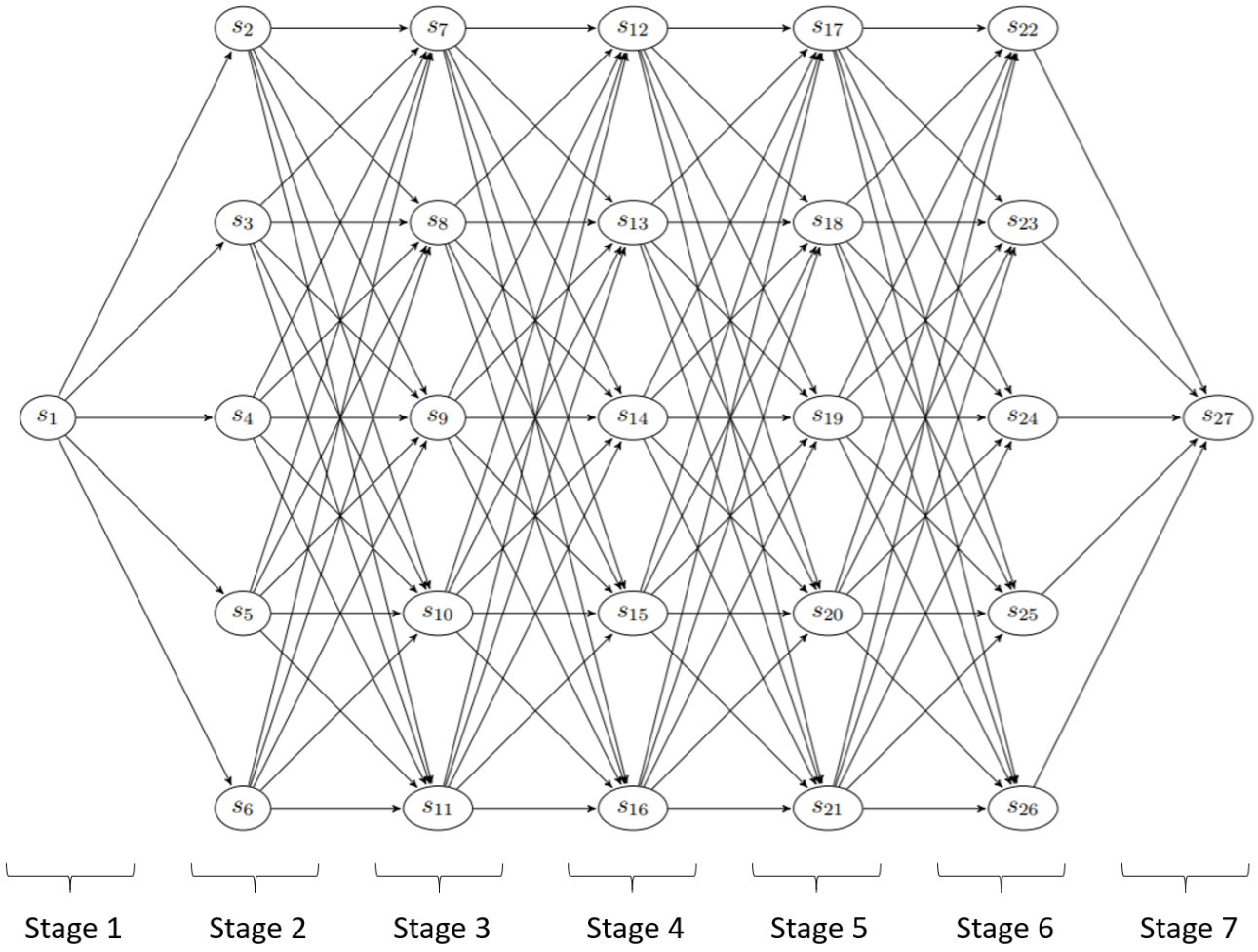


Figure 28: *An example of a (larger) routing problem.*

Exercise 1

In this exercise, we are going to look at the routing problem in Figure 28. If we want to know what the fastest route is from station s_1 to station s_2, then we have to compute the costs of the routes. However, before we do that, we would like to know something about the **number** of routes and with that, the number of computations.

(a) How many possible routes are there to travel from station s_1 to station s_{27}?

(b) Suppose an extra stage is added between stage 6 and 7 (so again 5 extra stations). How many **extra** routes do we obtain? And what if another stage of 5 stations is added?

(c) We assume we have the situation as in Figure 28. Suppose an extra stage of 5 stations is

added between stage 5 and 6. What is the influence on the number of computations if:

i. We compute the costs of **all** routes and then look for the minimum?

ii. We compute the fastest route with Dynamic Programming?

Give a conclusion about the efficiency of methods i. and ii. if we were to add 100 extra stages for example. Use the terms "linear relation" and "exponential relation".

Now we will make it more concrete. For each station, the costs are known to travel to the other stations in the next stage. The costs from station s_1 to station s_2 are equal to 7. Also the costs from station s_1 to the other stations are known. They are as follows:

- $s_1 \to s_2$: costs 7.
- $s_1 \to s_3$: costs 8.
- $s_1 \to s_4$: costs 9.
- $s_1 \to s_5$: costs 5.
- $s_1 \to s_6$: costs 8.

These values are also in the first column of Table 7. The values indicate the costs of going to the stations in the next stage, in the order from top to bottom (as depicted in Figure 28). We will first have a look at the other data in the table.

Table 7: *The distance between the different station of the next stage.*

Stage 1		**Stage 2**		**Stage 3**		**Stage 4**		**Stage 5**		**Stage 6**		**Stage 7**
s_1	7	s_2	6	s_7	5	s_{12}	5	s_{17}	10	s_{22}	6	s_{27}
	8		9		6		8		10			
	9		10		10		10		6			
	5		6		10		5		8			
	8		8		7		10		10			
		s_3	9	s_8	5	s_{13}	8	s_{18}	8	s_{23}	8	
			9		5		8		7			
			10		9		5		9			
			7		9		7		8			
			10		6		8		6			
		s_4	7	s_9	6	s_{14}	8	s_{19}	9	s_{24}	8	
			6		7		7		9			
			7		9		7		8			
			9		5		5		9			
			6		9		5		8			
		s_5	10	s_{10}	6	s_{15}	9	s_{20}	9	s_{25}	5	
			8		6		10		7			
			10		9		10		9			
			6		9		9		8			
			6		7		9		9			
		s_6	6	s_{11}	5	s_{16}	6	s_{21}	6	s_{26}	7	
			9		7		5		9			
			10		6		6		10			
			9		5		5		9			
			7		9		8		8			

Exercise 2
Take another look at Table 7. The rest of the table is constructed in the same way as the column in *Stage 1.*

(a) What does the value of 6 mean next to station s_2? And what does the 9 mean next to station s_2?

(b) Take a look at the column of *Stage 6.* Why is there only one value per station?

(c) Give an explanation why there is no value given at station s_{27}. If you were to give a value to station s_{27}, what value would that be (in other words, how long does it take to travel from station s_{27} to s_{27})?

To solve this, we could write down everything on paper. Even though we tackle this problem with dynamic programming, it will be a lot of work to do these computations by hand. Fortunately, we can use computer programs to help us. We are going to solve this problem with the use of Excel.

Exercise 3
Open the file RoutingproblemLesson3.xl on your computer. In this file, a similar table is given to Table 7. The distance from station s_{27} to s_{27} is set to 0. In this file, the stations are called 1, 2, 3, ... instead of $s_1, s_2, s_3, \ldots$.

Beneath the table, there are 6 coloured blocks/tables with the following data:

- Stations: below this are the stations of the stage above.
- k_station: these are the minimal costs to travel to the final station (27), from that station.
- Best route: this says which route should be chosen next.

Using dynamic programming, we start at the end of the problem. That end is simple, since at station s_{27} we're already there. Then we work backwards, in which we do the computations per (coloured) block. The first two blocks are coloured red: they have not been filled out yet. The other four have been completed already. It is your job to find the values for the red blocks. If you succeed, beneath the text "The optimal route is then given by:" a sequence of stations will appear that ensures that the costs are minimal. Answer the following question:

Question: Take a look at station s_{25} in the last block. What do the 5 and 27 mean? And if you look at station s_{21}, what do the 12 and the 22 mean?

Explanation Excel (needed for the final assignment)
We focus on stage 2, which we would like to complete. The best routes are still unknown, so they are set to 0 (we don't know them yet). We use the minimum from stage 3 to find out which station in stage 2 gives us the lowest value to the final station, which requires the exact same approach we used in the previous lesson. In Excel we can do this with the function MIN. You will do this yourselves as well, but first read the instructions below. You will need to compute the minimum of the sum of two columns (which will come back in the final assignment). How to do this is explained below:

1. Suppose you have the numbers in the cells $A1$ to $A4$ and in the cells $C6$ to $C9$, see Figure 29:

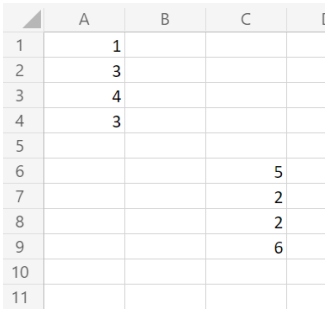


Figure 29: *The example to use the function* MIN.

2. Select an empty cell in which you want to have the minimal value. In our example we take $E1$. We select the cell $E1$ and type =min($A1$:$A4$+$C6$:$C9$). Excel now adds the values of the first row of both columns ($A1 + C6$), adds the values of the second row of both columns ($A2 + C7$), and so on. Then, Excel computes the minimum, which is 5 (the sum of the numbers in the second row, 3+2). See the figure below.

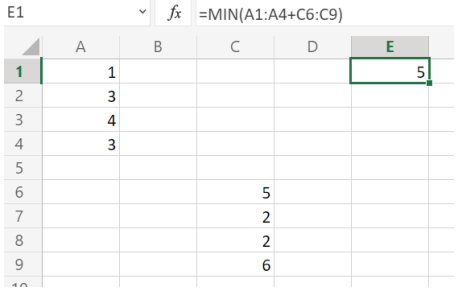


Figure 30: *The answer for using the function* MIN.

Final assignment

In the final assignment, you are going to compute the costs of the fastest route to go from station s_1 to station s_{27}. Use the file from the former exercise (RoutingproblemLesson3.xl).

(a) First compute the optimal costs from station s_2 to station s_{27}. Use the approach of the previous lesson. You will need to use the explanation on how to compute a minimum in Excel. The best route from station s_2 appears as well, which is station s_7. *Hint: you have to compute k_station for station s_2.*

(b) Do the same for the other stations in stage 2. Give the lowest costs for each station to travel to station s_{27}. When you did this correctly, the best routes appear for each station in stage 2. *Hint: if a 0 remains below Best route, then you have not found the optimal value yet.*

(c) Finally, what are the minimal costs to travel from station s_1 to station s_{27}? Give the optimal route by filling in the sequence below:

$$(s_1) \to \ldots \to \ldots \to \ldots \to \ldots \to \ldots \to (s_{27})$$

This is the end of the lesson series.

E.3 Explanation Excel file

The Excel file for the students looks as follows:

Stage 1	Distance	Stage 2	Distance	Stage 3	Distance	Stage 4	Distance	Stage 5	Distance	Stage 6	Distance	Stage 7	Distance
1	7	2	6	7	5	12	5	17	10	22	6	27	0
	8		9		6		8		10				
	9		10		10		10		6				
	5		6		10		5		8				
	8		8		7		10		10				
		3	9	8	5	13	8	18	8	23	8		
			9		5		8		7				
			10		9		5		9				
			7		9		7		8				
			10		6		8		6				
		4	7	9	6	14	8	19	9	24	8		
			6		7		7		9				
			7		9		7		8				
			9		5		5		9				
			6		9		5		8				
		5	10	10	6	15	9	20	9	25	5		
			8		6		10		7				
			10		9		10		9				
			6		9		9		8				
			6		7		9		9				
		6	6	11	5	16	6	21	6	26	7		
			9		7		5		9				
			10		6		6		10				
			9		5		5		9				
			7		9		8		8				

Stations	k_station	Best route
1		0

Stations	k_station	Best route
2		0
3		0
4		0
5		0
6		0

Stations	k_station	Best route
7	23	12
8	23	12
9	24	12
10	24	12
11	23	12

Stations	k_station	Best route
12	18	17
13	19	19
14	17	21
15	21	21
16	18	18

Stations	k_station	Best route
17	13	25
18	13	25
19	14	25
20	13	25
21	12	22

Stations	k_station	Best route
22	6	27
23	8	27
24	8	27
25	5	27
26	7	27

Station
27

The optimal route is then given by:

1		#N/B		#N/B		#N/B		#N/B		#N/B		#N/B	

The blue entries are already correct, the red ones have to be computed. This is done in the final assignment (lesson 3). Since finding the argument is too difficult for the students, when they find the correct answer, Excel automatically outputs the best route from that station. The optimal value can be found by using the MIN function. For station 7, we compute k_station = MIN(H4:H8+K30:K34). This means that the distance from station 7 is computed to all the next stations, from which we know the shortest route.

When the students find all the correct answers, the optimal route is given in the bar below.

F Teacher's Guide & Solutions to exercises

In this chapter, the solutions to the exercises along with some guidance on how these lessons can be conducted are given. Moreover, the learning outcomes are mentioned prior to the discussion of each lesson.

Both Dutch and English versions are available in sections F.1 and F.2 respectively.

F.1 Teacher's Guide - Dutch

F.1.1 Les 1: Introductie Dynamisch Programmeren

Deze les is ter introductie van dynamisch programmeren. De leerlingen spelen eerst het lucifer-spel (een variant van "The NIM-game"). Vervolgens bedenken de leerlingen zelf een strategie om te winnen met het lucifer-spel. Tot slot krijgen ze een opdracht over de definitie van dynamisch programmeren. Dit laatste is een korte vooruitblik op de volgende les; we spreken hier namelijk over enkele toepassingen. We geven eerst de leerdoelen van de les en daarna bespreken we per opdracht wat de antwoorden zijn (indien mogelijk) en geven een aantal tips/houvast voor de docent die de les geeft.

Leerdoelen les 1
Aan het einde van de les kunnen de leerlingen...

- een winnende strategie bedenken voor een spel met behulp van dynamisch programmeren.
- uitleggen wat dynamisch programmeren inhoudt.
- voorbeelden noemen van toepassingen waar dynamisch programmeren wordt gebruikt.

Opdracht 1: 2 + 10 = 12 minuten
In deze opdracht spelen de leerlingen het lucifer-spel. De instructies staan in de opdracht beschreven, maar het kan handig zijn om het idee met de leerlingen in ongeveer 2 minuten te bespreken. Dit moet in **duo's** worden gespeeld, dus let op dat er per duo 30 lucifers beschikbaar moeten zijn. Het is mogelijk om af te wijken van het aantal lucifers, maar met minder lucifers wordt het wellicht te gemakkelijk voor de leerlingen. De leerlingen kunnen dit herhaald spelen, voor ongeveer 10 minuten. Dan geeft de docent het signaal dat de tijd om is en dat de leerlingen aan de tweede opdracht mogen werken.

Opdracht 2: 10 + 5 = 15 minuten
In deze opdracht gaan de leerlingen een strategie bedenken om te winnen met het lucifer-spel. Hiervoor schrijven ze een stappenplan in dezelfde duo's.

Met dynamisch programmeren beginnen we achteraan. De strategie kun je als volgt uitvogelen. Speler A wint met zekerheid als speler B nog maar 1 lucifer overhoudt. Dus, als speler A moet je zorgen dat speler B, ongeacht zijn of haar keuzes, altijd 1 overhoudt. Dat lukt speler A als er in de beurt van speler A nog 2, 3 of 4 lucifers liggen (dan haal je respectievelijk 1, 2 of 3 lucifers weg), maar met 5 lucifers op tafel lukt dit niet. Dus, in de beurt van speler B moeten 5 lucifers op tafel liggen, zodat in de beurt van speler A er nog 2, 3 of 4 liggen (wat altijd gebeurt, wat speler B ook kiest). Hier kan speler A voor zorgen als er nog 6, 7 of 8 lucifers op tafel liggen, maar niet met 9. Dus, speler A moet zorgen dat er in de beurt van speler B 9 lucifers op tafel liggen, om dezelfde reden als eerder. Dit patroon herhaalt zich, dus we halen hieruit dat we

er altijd voor moeten zorgen dat speler B 1, 5, 9, 13, 17, 21, 25, of 29 lucifers moet hebben in zijn of haar beurt. Als we beginnen met 30 lucifers, dan kan speler A hier altijd voor zorgen en dus met zekerheid winnen. Het is leuk om op te merken dat deze winst niet verzekerd is als je begint met 29 lucifers. Dan kan speler B de strategie toepassen.

Dat stappenplan kan er als volgt uitzien (aannemend dat we met 30 lucifers spelen):

1) We beginnen met 30 lucifers, haal er 1 weg. Nu begint speler B met 29 lucifers.
2) Wacht af wat speler B doet. Er liggen dan nog 26, 27 of 28 lucifers na de keuze van speler B. Haal 1, 2 of 3 weg zodat speler B zijn of haar beurt begint met 25 lucifers.
3) Herhaal zodat speler B zijn of haar beurt begint met 4 minder dan de vorige beurt van speler B, dus speler B begint altijd met de 21, 17, 13, 9, 5 of 1 lucifers op tafel.
4) Omdat speler B met 1 lucifer eindigt, moet hij of zij deze weghalen en verliest het spel. Hiermee wint speler A.

Het is belangrijk dat de leerlingen in hun stappenplan noemen dat er altijd 1, 5, 9, ..., 29 lucifers moeten liggen aan het begin van de beurt van speler B. De leerlingen krijgen 10 minuten om het stappenplan te maken. Hierna bespreekt de docent gedurende 5 minuten wat de mogelijke strategie is en vraagt aan leerlingen wat hun strategieën zijn. Benoem dat het belangrijk is om achteraan te beginnen, dus wat er gebeurt met 1 lucifer en hoe de situatie verandert als het meer lucifers worden. Schrijf dit stappenplan uit. Nu mogen de leerlingen aan de slag met opdracht 3.

Opdracht 3: 5 + 10 = 15 minuten
In deze opdracht komt de term dynamisch programmeren aan bod. De leerlingen gaan eerst zelf aan de slag met het lezen van de opdrachten en het maken van de vraag. Het antwoord op de vraag is: In opdracht 2 hebben we dynamisch programmeren gebruikt door “achteraan” te beginnen. We weten hoe we het kleine probleem op kunnen lossen (namelijk met 1 lucifer). Dit gebruiken we om de andere problemen op te lossen, voor meerdere lucifers.

Het is belangrijk om dit onderdeel te bespreken met de leerlingen, voor ongeveer 5 minuten. Tot slot kan de docent een vooruitblik doen op de volgende les, waarbij we naar het routeprobleem gaan kijken. Daarnaast zijn er nog veel meer toepassingen voor dynamisch programmeren die kunnen worden aangestipt. Voor leerlingen met biologie is het interessant dat je DNA-ketens met elkaar kunt vergelijken met behulp van dynamisch programmeren. Illustraties voor het routeprobleem en van DNA-ketens staan ook bij de opdracht.

Ook matrixvermenigvuldiging en het knapzak-probleem zijn voorbeelden waar dynamisch programmeren van toepassing is. De docent kan zelf invulling geven aan welke toepassingen hij of zij bespreekt (nu is matrixvermenigvuldiging geen onderdeel van het huidige curriculum, maar wellicht voor de docent interessant om te weten). Er wordt wel aangeraden om in ieder geval het routeprobleem te benoemen.

F.1.2 Les 2: Het routeprobleem

In deze les komt het routeprobleem naar voren. De leerlingen krijgen een voorbeeld van wat het routeprobleem is en gaan kijken waarom dynamisch programmeren hier van toepassing is. Computationele voordelen komen hier ook naar voren. De leerlingen gaan met behulp van een

tabel dynamisch programmeren gebruiken om de oplossing te vinden. Tot slot komen de termen min en argmin naar voren, wat nodig is voor de volgende les. In deze les loopt de docent vooral rond voor vragen. De leerlingen moeten de les zelf doorwerken.

Leerdoelen les 2
Aan het einde van de les kunnen de leerlingen...

- voor kleine routeproblemen de optimale route berekenen met behulp van dynamisch programmeren.
- een verklaring geven waarom er over het algemeen minder berekeningen hoeven worden uitgevoerd als het routeprobleem wordt opgelost met dynamisch programmeren in plaats van het berekenen van alle mogelijkheden.
- De notatie $\min_x\{\dots\}$ en $\arg\min_x\{\dots\}$ herkennen en hiermee berekeningen uitvoeren.

Opdracht 1: 10 minuten
In deze opdracht gaan de leerlingen kijken naar een voorbeeld van het routeprobleem en worden ze hier bekend mee. Ze krijgen een paar definities, zoals wat stations zijn, wat routes zijn en wat kosten zijn. De antwoorden op de vragen zijn als volgt:

(a) Als we via station s_5 gaan zijn de kosten $5 + 1 = 6$. Via station s_6 en s_7 zijn de kosten respectievelijk $6 + 2 = 8$ en $1 + 6 = 7$. Dus via station s_5 is inderdaad het snelst met kosten 6.

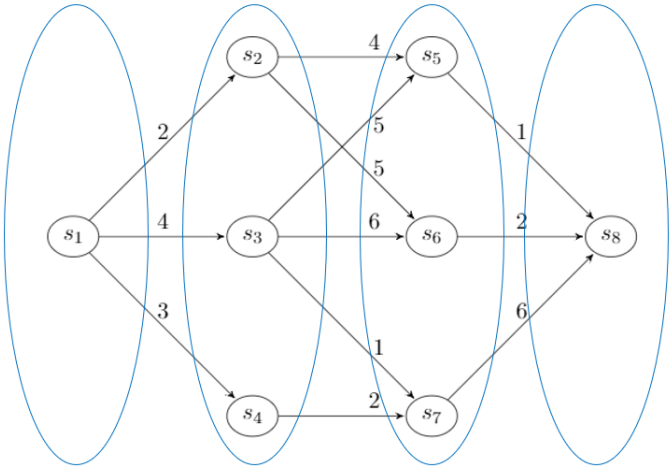
(b) Als ik zo snel mogelijk van station s_1 naar s_8 wil gaan via station s_3, dan weet ik al dat ik via station s_5 moet gaan, aangezien ik weet dat de kosten vanaf station s_3 via s_6 en s_7 hoger zijn.

(c) Vanaf station s_3 zijn mijn kosten 6. Het reizen van s_1 naar s_3 heeft kosten 4 (de enige optie), dus de kosten van de snelste route van s_1 naar s_8 via s_3 zijn $6 + 4 = 10$.

Na opdracht 1 worden de stations verdeeld in fases. De docent kan dit op het bord verduidelijken door bijvoorbeeld een plaatje te tekenen als in figuur 31:

Bij elke fase kan nog worden aangegeven om welke fase het gaat. De notatie van stations wordt op verschillende manieren gebruikt. Voor het eerste station schrijven we soms s_1 en soms $\textcircled{s_1}$. Dit is gedaan om het overzichtelijk te houden in bijvoorbeeld tabellen of lange stukken tekst; in principe zijn de notaties uitwisselbaar. Vervolgens wordt in de tekst uitgelegd dat we in fase 4 beginnen, omdat we met dynamisch programmeren werken. Er is een tabel gegeven die per fase is uitgewerkt. In fase 1 ontbreekt de berekening, het is dus aan de leerlingen om die in te vullen. Dat wordt gedaan in opdracht 2. Het lezen van de tekst en de tabel vergt ongeveer 10 minuten.

Opdracht 2: 20 minuten
In opdracht 2 berekenen we de optimale route met de optimale kosten, aan de hand van de tabel die in de tekst staat. Ook is deze opdracht een opstapje naar wat het minimum is en wat het argument van het minimum is. Dit is gekoppeld aan het vinden van een minimum met behulp van differentiëren, wat de leerlingen dus moeten beheersen. De antwoorden zijn als volgt:



Figuur 31: *Het routenetwerk met de fases.*

(a) De optimale waardes vanaf station s_2, s_3 en s_4 zijn te vinden in de tabel. De afstand van station s_1 naar de andere stations is te halen uit de grafische weergave van het routeprobleem aan het begin van les 2. We krijgen

- $\textcircled{s_2}$: Kosten $2 + 5 = 7$;
- $\textcircled{s_3}$: Kosten $3 + 6 = 9$;
- $\textcircled{s_4}$: Kosten $3 + 8 = 11$.

(b) De kosten bij station s_2 zijn minimaal, namelijk 7. Deze waarde betekent dat de minimale kosten om van station s_1 naar station s_8 te reizen gelijk is aan 7.

(c) De waardes uit vraag (a) kunnen worden ingevuld, net als de minimale kosten uit vraag (b). De minimale waarde hoort bij station s_2, dus dat is het volgende station.

Voor de docent is het belangrijk om aan te stippen wat het minimum is en wat het argument is van het minimum.

Opdracht 3: 10 minuten

In deze opdracht gaan de leerlingen werken met het minimum en met het argument van het minimum, maar dan niet voor de functies die ze gewend zijn. Eerst moeten de leerlingen ophalen wat een minimum was, wat in vraag (a) gebeurt. Vervolgens gaan de leerlingen de nieuwe notatie toepassen op het routeprobleem. Het is verstandig om de leerlingen eerst zelf hiermee te laten puzzelen en daarna de opdracht samen te doen. De leerlingen moeten goed weten dat ze het minimum berekenen en dan moeten weten welke actie voor dit minimum zorgt, ofwel het argument. Het antwoord op de vragen:

(a) Dit kan dus met behulp van de afgeleide, $f'(x) = \frac{2}{3}x - 2$. Dit gelijkstellen aan 0 geeft $\frac{2}{3}x - 2 = 0$, dus $x = 3$. Dus het minimum treedt op bij $x = 3$ en het minimum is dus

$f(3) = 2$.

(b) Voor station s_3 krijgen we $\min_{\text{station}}\{kosten(station)\} = 6$. De actie die hiervoor zorgt is s_5, dus $\text{argmin}_{\text{station}}\{kosten(station)\} = s_5$. Voor station s_4 kan hetzelfde worden gedaan, alleen zijn er niet echt keuzes. We kunnen het wel zo opschrijven: $\min_{\text{station}}\{kosten(station)\} = 8$. Dit is via station s_7, dus $\text{argmin}_{\text{station}}\{kosten(station)\} = s_7$.

(c) De kosten zijn $f(x)$ en de vervolgstations zijn x.

F.1.3 Les 3: Het routeprobleem met Excel

In deze laatste sessie gaan we kijken naar een groter routeprobleem dat we op gaan lossen met behulp van Excel. In de opdrachten wordt eerst gekeken naar de computationele aspecten van het probleem, zonder waardes te geven. Daarna krijgt het probleem een concretere aard door kosten te koppelen aan de verschillende routes. In de opdrachten die volgen moeten de leerlingen zowel met een tabel werken als met Excel, waarbij ook terug wordt gekoppeld aan de functie min aan het eind van de vorige les. Deze les is weer zelfstandig, maar het is belangrijk dat de leerlingen gebruik kunnen maken van een laptop en een docent die hier mee overweg kan (vooral de functies in Excel).

Leerdoelen les 3
Aan het einde van de les kunnen de leerlingen...

- De functie "min" kunnen gebruiken voor het vinden van een minimum.
- Een schatting maken van het aantal berekeningen dat moet worden uitgevoerd om de optimale route te kiezen in een routenetwerk.
- Uitleggen hoe het aantal berekeningen dat moet worden uitgevoerd verandert als de grootte van het routeprobleem wordt veranderd.

Opdracht 1: 10 minuten
Deze opdracht gaat over computationele aspecten van dynamisch programmeren in het routeprobleem. Een vrij uitgebreid netwerk van routes is gegeven, dat sterk lijkt op die van de voorgaande les. Echter, hier zijn nog geen waardes gekoppeld aan de verschillende routes en is het aantal stations uitgebreid van 8 naar 27. De antwoorden zijn als volgt:

(a) We zien 7 fases. Vanuit station s_1 zijn 5 mogelijkheden om naar fase 2 te gaan. Elk station in fase twee heeft ook weer 5 mogelijkheden om naar een station in fase 3 te gaan, enzovoort. Alleen in fase 6 is er geen vrijheid meer in het kiezen van een route om naar fase 7 te gaan, aangezien elk station maar 1 mogelijkheid heeft om naar fase 7 te gaan. Dit levert dat $5^5 = 3125$ mogelijke routes om van station s_1 naar station s_{27} te reizen.

(b) Met dezelfde redenering als in vraag (a) krijgen we dan $5^6 = 15.625$ verschillende routes, aangezien er 5 nieuwe stations elk weer voor 5 nieuwe routes zorgen. Dus dat levert $5^6 - 5^5 = 12.500$ extra routes. Stel dat er nog een fase tussenkomt, dan wordt het $5^7 - 5^6 = 62.500$ extra routes.

(c) Voor type i. neemt het aantal berekeningen exponentieel toe: voor elke fase van 5 stations moeten we ons vorige aantal met 5 vermenigvuldigen. Voor type ii. is het lineair: voor elke fase komen er ongeveer 5 berekeningen bij. Let op: dit antwoord is intuïtief, het gaat erom

dat de leerlingen inzien hoe het aantal berekeningen groeit. Het aantal berekeningen dat moet worden gedaan ligt iets hoger bij dynamisch programmeren, aangezien je meerdere ongelijkheden oplost. Die details zijn op dit moment te ingewikkeld voor de leerlingen en worden daarom achterwege gelaten.

Voor het maken van vraag (c) van deze opgave kan het handig zijn om de leerlingen te steunen. Ze zijn nog niet erg bekend met computationele complexiteit en het kan hier dus belangrijk zijn om de leerlingen te sturen door de opdracht voor te doen en te kijken naar wat er gebeurt met het aantal berekeningen als de stations veranderen.

Nu worden er waardes toegekend aan de verschillende routes. In de tekst voor de leerlingen staat aangegeven wat de waardes allemaal betekenen, maar het kan zijn dat leerlingen hier moeite mee hebben. Het lezen en begrijpen van informatie is ook een belangrijk onderdeel van computationeel denken en wordt hier dus op de proef gesteld. Om het beter te begrijpen maken de leerlingen opdracht 2.

Opdracht 2: 5 minuten
In deze opdracht gaan we proberen om de leerlingen meer inzicht te bieden in wat de waardes in de tabel betekenen. Zodra de leerlingen zien wat de tabel precies inhoudt, zal deze opdracht niet lang duren.

1. De eerste 6 bij station s_2 is het **eerste** gegeven getal. Dat betekent dat dit de kosten zijn van het **eerste** station in de volgende fase. Dus dit zijn de kosten om van station s_2 naar station s_7 te reizen. De tweede 6 bij station s_2 is het **vierde** gegeven getal, wat de kosten zijn van s_2 naar s_{10}. De 9 zijn dan de kosten om van station s_2 naar station s_8 te reizen.
2. Er staat maar 1 waarde omdat je vanaf al die stations alleen naar station s_{27} kunt reizen. Dat is dus die gegeven waarde.
3. Station s_{27} is het eindstation, dus daar staat geen waarde. Als je wel een waarde toe zou moeten kennen zou dat bijvoorbeeld 0 kunnen zijn, aangezien de kosten 0 zijn om naar zichzelf te reizen. Dit is van belang voor het rekenen met Excel.

Nu gaan de leerlingen in het Excelbestand kijken hoe we de optimale route moeten berekenen. Hiervoor gaan ze eerst wat meer inzicht krijgen in wat er in het bestand staat. Dat wordt gedaan in opdracht 3.

Opdracht 3: 5 minuten
In deze opdracht kijken we naar de het Excelbestand. Dit is bijna een kopie van de tabel die in de opdrachten staan. Er hoort één vraag bij deze opdracht. Het antwoord hierop is als volgt:

De 5 na de 25 betekent dat het de kosten 5 zijn om zo snel mogelijk naar station 27 te reizen. De 27 betekent dat dat het volgende station is. Bij station 21 betekent de 12 dat de minimale kosten om bij station 27 aan te komen gelijk zijn aan 12. De 22 betekent dat dat het volgende station is om die minimale kosten te halen.

De rest van de tekst is de uitleg over het gebruik van het minimum in Excel. Voor de docent is het van belang dat hij of zij goed weet hoe dit werkt en dit oefent als hij of zij hier geen ervaring mee heeft. Dit is namelijk nodig voor de eindopdracht. Nu is het de bedoeling dat de leerlingen de laptop gaan gebruiken en het routeprobleem op gaan lossen. De leerlingen gaan alleen de optimale **kosten** berekenen, niet de optimale route. Voor het berekenen van de

optimale route is meer kennis nodig van Excel en van de functie argmin. We beperken ons dus tot het vinden van het minimum. Het Excelbestand is zo geschreven dat, wanneer de leerlingen de juiste kosten vinden, de optimale route automatisch wordt gegeven. De leerlingen maken als het ware de code af met behulp van de instructies. Dit wordt dan het einde van de lessenserie.

Eindopdracht: 25 minuten
De leerlingen moeten dus het juiste bestand openen en de rood/roze blokken invullen. Het is de bedoeling dat de k_station invullen. Bijgeleverd is het bestand RouteprobleemLes3Antwoord, waarin alle getallen zijn ingevuld en de correcte route is gegeven. Hieronder volgt de uitleg waarom deze berekeningen correct zijn:

(a) We hoeven met dynamisch programmeren alleen te kijken naar de stations in de volgende fase. Om de beste route te vinden, zoek je naar welke stations je kunt gaan vanaf station s_2 met de bijbehorende kosten (te vinden in $E4 : E8$). Deze waardes moet je respectievelijk optellen bij de k_station van de volgende fasen (dat zijn $H30 : H34$), waarvan je de minimumwaarde zoekt. Dit is te vinden door bij station s_2 in te vullen dat k_station $=\min(E4 : E8 + H30 : H34)$, wat als antwoord 29 geeft en beste route 7. *Dit werkt omdat de kosten zo gestructureerd zijn dat je de juiste waardes bij elkaar optelt.*

(b) Dit werkt op precies dezelfde manier in vraag (a). Nu pak je van het volgende station de kosten van de routes plus de k_station van de stations van de volgende fase (nog steeds fase 3). Voor station 3 vullen we dan bij k_kosten in $=\min(E9 : E13 + H30 : H34)$. De antwoorden zijn als volgt:

- Station 3: k_station = 31, Beste route = 10.
- Station 4: k_station = 29, Beste route = 8.
- Station 5: k_station = 29, Beste route = 11.
- Station 6: k_station = 29, Beste route = 7.

(c) Tot slot doen we hetzelfde voor station s_1. We nemen nog steeds de kosten om naar de stations in de volgende fase te reizen (dat zijn de afstanden in kolom B). We vullen dus voor k_station in $=\min(B4 : B8 + E30 : E34)$. Dit geeft 34 als antwoord en beste route 5. Dus, de minimale kosten om van station s_1 naar station s_{27} te reizen zijn gelijk aan 34. De optimale route is dan te vinden in het Excelbestand. Die is als volgt:
$(s_1) \to (s_5) \to (s_{11}) \to (s_{12}) \to (s_{17}) \to (s_{25}) \to (s_{27})$.

Dit is het einde van de lessenserie. De leerlingen kunnen de opdrachten inleveren bij de docent die ze nakijkt.

F.2 Teacher's Guide - English

F.2.1 Lesson 1: Introduction Dynamic Programming

This lesson is an introduction to dynamic programming. Students first play the match-game (a variation on "The NIM-game"). Afterwards, students come up with a strategy to win the match-game. Finally, they will do an exercise about the definition of dynamic programming. The latter is a short preview of the next lesson; we will talk about several applications. First,

we state the learning outcomes of the lesson and afterwards we discuss per exercise what the answers are (if applicable) and give a few ideas for the educator that teaches the lesson series.

Learning outcomes lesson 1
At the end of the lesson, students are able to ...

- come up with a winning strategy for a game using dynamic programming.
- explain what dynamic programming entails.
- give examples of applications of dynamic programming.

Exercise 1: 2 + 10 = 12 minutes
In this exercise, the students play the match-game. The instructions are described in the exercise, but it might be useful to give the students an idea of the game in an explanation of approximately 2 minutes. This is played in **pairs**, so make sure that there are 30 matches available per pair of students. It is possible to deviate from the number of matches, but with less matches it might be too easy for students. The students can repeat this game for approximately 10 minutes. Then, the teacher signals that the time is up and that the students can start working on the second exercise.

Exercise 2: 10 + 5 = 15 minutes
In this exercise, the students are going to come up with a strategy to win with the match-game. They write down the steps in the same pairs.

For dynamic programming, we start at the end of the problem. A strategy can be found in the following way. Player A wins with certainty if player B is left with 1 match. So, player A has to ensure that player B is left with 1 match, no matter their choice. Player A can do that when there are 2, 3 or 4 matches on the table (then player A can remove 1, 2 or 3 matches respectively), but with 5 matches this is impossible. So, in player B's turn there should be 5 matches on the table, such that player A has 2, 3 or 4 when their turn starts (which happens, whatever the choice of player B is). Player A can ensure this situation to happen when there are 6, 7 or 8 matches on the table, but not with 9. So, player A has to play such that player B has 9 matches on the table in their turn, following the same reasoning as before. This pattern keeps repeating itself, so we find that player B needs 1, 5, 9, 13, 17, 21, 25, or 29 matches in their turn. Since we start with 30 matches, player A can make sure of this and thus win with certainty. Note that when starting with 29 matches, you do not win with certainty. Then player B could apply this strategy.

The steps could look as follows (assuming we play with 30 matches):

1) We start with 30 matches, pick up 1. Then player B starts with 29 matches.
2) Wait for player B's action. There will be 26, 27 or 28 matches left after their choice. Pick up 1, 2, or 3 such that player B has to start their turn with 25 matches.
3) Repeat this to have player B starting with 4 less than the turn before, so player B always starts with 21, 17, 13, 9, 5 or 1 on the table.
4) Because player B ends with 1 match on the table, they have to pick it up and they lose the game. Thus, player A wins.

It is important that students show in their steps that there always should be 1, 5, 9, ..., 29 matches on the table at the start of player B's turn. Students get 10 minutes to come up with the steps. Then, the teacher discuss during 5 minutes what the possible strategy is and asks the students about their approaches. Show the students that it is important to start at the end, so to given them an idea of what happens with 1 match and how the situation changes if the number of matches increase. Write down the steps. How, the students can start with exercise 3.

Exercise 3: 5 + 10 = 15 minutes
In this exercise, the concept of dynamic programming is the central theme. The students first start with reading the exercise and answering the question. The answer to the question is: in exercise 2, we applied dynamic programming by starting at the "end". We know how to solve the small problem (with 1 match). We use this to solve other problems, for multiple matches.

It is important to discuss this part with students for approximately 5 minutes. At the end, teachers can do a preview on the next lesson, in which we will have a look at the routing problem. Furthermore, there are many more applications of dynamic programming that can be mentioned. For students that take biology courses, it is interesting to know that you can compare DNA-sequences using dynamic programming. Illustrations for the routing problem and DNA-sequences are given in the exercises.

Additionally, matrix multiplication and the knapsack-problem are examples where dynamic programming can be applied. The teacher can decide which applications they want to discuss (matrix multiplication is not part of the current Dutch curriculum, but perhaps it is interesting for the teacher to know this). It is highly recommended to at least discuss the routing problem.

F.2.2 Lesson 2: The routing problem

In this lesson, the routing problem is the central theme. Students obtain an example of what the routing problem is and start looking why dynamic programming is applicable here. Computational advantages are also mentioned here. The students will use a table to use dynamic programming for finding the solution. Finally, the notation for the minimum and the argument of the minimum are mentioned, which the students will be needing for the next lesson. In this lesson, the teacher is mainly present to answer questions. The students should be able to work on the exercises themselves.

Learning outcomes lesson 2
At the end of the lesson, students are able to ...

- compute the optimal route in a small routing problem with a unique solution using dynamic programming. *Small: maximum 4 phases and 10 nodes.*
- give an explanation why the number of computations to find the optimal solution in a routing network decreases when solving it with dynamic programming instead of minimising over all possible routes.
- recognise and perform computations using the notation $\min_x\{\dots\}$ and $\arg\min_x\{\dots\}$.

Exercise 1: 10 minutes
In this exercise, the students are going to consider an example of the routing problem and will get familiar with it. They will learn some basic definitions, such as what a station is, what routes are and what costs are. The answers to the exercises are as follows:

(a) If we travel via station s_5 the costs are $5 + 1 = 6$. Via station s_6 and s_7 the costs are $6 + 2 = 8$ and $1 + 6 = 7$, respectively. So it is indeed the fastest route to go via station s_5, with costs 6.

(b) If I want to go as quickly as possible from station s_1 to s_8 via station s_3, then I already know I have to travel via station s_5, since I know that the costs of travelling from station s_3 via s_6 and s_7 are higher.

(c) From station s_3, the costs are 6. Travelling from station s_1 to s_3 hast costs 4 (the only option), so the costs of the fastest route from station s_1 to s_8 via s_3 are $6 + 4 = 10$.

After exercise 1, the stations are divided in stages. The teacher can clarify this by making a sketch of this situation on the whiteboard, like in Figure 32:

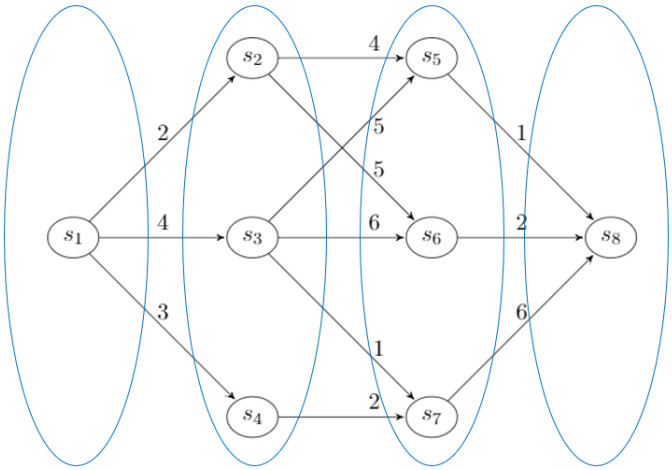


Figure 32: *The routing network with the stages.*

At each stage, one could write down which stage we are talking about. The notation of stations is used in different ways. For the first station, we sometimes write s_1 and sometimes $\textcircled{s_1}$. This is done to keep it ensure that the clear overview of the tables and long texts is not lost; in principle the notation is interchangeable. Afterwards, the text explains we start in stage 4, since we are working with dynamic programming. In the table the computations are done per stage. In stage 1 the computation is missing and it is up to the students to complete it. This is done in exercise 2. Reading the text and the table will take the students approximately 10 minutes.

Exercise 2: 20 minutes
In exercise 2, we compute the optimal route with the optimal costs, with the help of the given table. Also, this exercise is the first in the direction of what the minimum is and what the argument of the minimum is. this is linked to finding a minimum with the help of a derivative, a technique that the students should already master. The answers are as follows:

(a) The optimal values from station s_2, s_3 and s_4 can be found in the table. The distance from station s_1 to the other station can be found in the graphical representation of the routing problem at the start of lesson 2. We obtain

- (s_2): Costs $2 + 5 = 7$;
- (s_3): Costs $3 + 6 = 9$;
- (s_4): Costs $3 + 8 = 11$.

(b) The costs to arrive at station s_2 are minimal, equalling 7. This value means that the minimal costs to travel from station s_1 to s_8 is equal to 7.

(c) The values from question (a) can be put into the table, similar to the minimal costs from question (b). the minimal value belongs to station s_2, thus that will be the next station.

For a teacher, it is important to mention what the minimum is and what the argument of the minimum is.

Exercise 3: 10 minutes
In this exercise, student will work with the minimum and the argument of the minimum, but not for functions that they are used to. First, the students have to be reminded what a minimum was, which happens in question (a). Afterwards, the students will apply the newly learnt notation to the routing problem. It might be useful to let the students work on this themselves for a bit. Afterwards, it can be done together with the students. The students have to understand they are computing the minimum and understand which action causes this minimum, in other words the argument. The answers to the questions:

(a) This can be done with the derivative, $f'(x) = \frac{2}{3}x - 2$. Setting the derivative equal to 0 yields $\frac{2}{3}x - 2 = 0$, so $x = 3$. Hence the minimum occurs at $x = 3$ and the minimum is then $f(3) = 2$.

(b) For station s_3 we obtain $\min_{\text{station}}\{costs(station)\} = 6$. The action that causes this is s_5, so $\text{argmin}_{\text{station}}\{costs(station)\} = s_5$. For station s_4 the same can be done, but there is not a lot of freedom of choice. We can write it that way, however: $\min_{\text{station}}\{costs(station)\} = 8$. This is via station s_7, so $\text{argmin}_{\text{station}}\{costs(station)\} = s_7$.

(c) The costs are $f(x)$ and the next stations are x.

F.2.3 Lesson 3: The routing problem with Excel

In this last lesson, we are going to take a look at a larger routing problem that we will solve with Excel. First, the computational aspects are considered without relating them to costs yet. Afterwards, the problem will become more concrete by linking costs to the different routes. In the exercises that follow, the students have to work with both a table and Excel, in which also

is referred back to the function MIN. It is again possible for the students to work through the exercises themselves, but it is important that students have access to a laptop and a teacher who knows how to work with this (especially the functions in Excel).

Learning outcomes lesson 3
At the end of this lesson, students are able to ...

- use the function MIN in Excel to find the minimum of the sum of two columns (of equal length).
- give an estimation of the number of computations that have to be executed to find the optimal route in a routing problem.
- explain how the number of computations that have to be executed changes when the size of the routing problem changes.

Exercise 1: 10 minutes
This exercise deals with the computational aspects of dynamic programming in the routing problem. A fairly complex network is given, that has a strong resemblance with problem of the previous lesson. However, values have not been given to the arcs and the number of stations is extended from 8 to 27. The answers to the questions are the following:

(a) We see 7 stages. From station s_1 there are 5 possibilities to go to stage 2. Each station in stage 2 also has 5 possibilities to go to a next station in stage 3, and so on. Only in stage 6 there is no freedom anymore in choosing a route to go to stage 7, since each station has just 1 option to go to stage 7. This yields $5^5 = 3125$ possible routes to go from station s_1 to station s_{27}.

(b) With the same reasoning as in question (a), we obtain $5^6 = 15.625$ different routes, since the 5 new stations each cause 5 new routes. This yields $5^6 - 5^5 = 12.500$ extra routes. Suppose another stage is added, then there are $5^7 - 5^6 = 62.500$ extra routes.

(c) For type i. the number of computations increases exponetially: for each stage of 5 stations we have to multiply the previous number with 5. For type ii. it is linear: for each stage there are 5 extra computations. Note: this is an intuitive answer. The goal is that students see how the number of computations grow. The number of computations that have to be done is somewhat higher at dynamic programming, since there are multiple inequalities that are solved. The details are too complicated for students and are therefore not discussed.

For doing question (c) of this exercise, it might be useful to support the students a bit more. They are not used to computational complexity and it might be important to aid students. This can be done by doing the exercise together and letting them think how the number of computations is affected by the changing number of stations.

Now, values are given to the different routes. The text states what all the values mean, but it might be possible that the students struggle with it. Reading and understanding information is also part of computational thinking and is therefore also practised here. To understand the theory better, the students work on exercise 2.

Exercise 2: 5 minutes
In this exercise, we are going to try to offer the students more insights in what the values in the table mean. When the students know what they mean, this exercise will be finished rather quickly.

1. The first 6 at station s_2 is the **first** number. That means that these indicate the costs to go to the **first** station in the next stage. So, these are the costs to go from station s_2 to station s_7. The second 6 at station s_2 is the **fourth** number, which are the costs to go from station s_2 to station s_{10}. The 9 indicates the costs to go from station s_2 to station s_8.
2. There is only 1 value because from those stations, you can only travel to station s_{27}, so that is the given value.
3. Station $s27$ is the final station, so there is no value given there. If you were to give it a value, that could be 0, since costs are 0 to travel to the station you are in.

Now, the students are going to investigate using the Excel file how to find the optimal route. The first get more insight in what is in the file. This is done in exercise 3.

Exercise 3: 5 minutes
In this exercise, we go to the Excel file. This is virtually a copy of the table that is given in the exercises. There is just one question in this exercise. The answer is the following:

The 5 after the 25 indicates that the costs are 5 to go to station 27. The 27 signifies the next station. At station 21, the 12 indicates that this is the lowest value to get to station 27. The 22 means that this is the next station to obtain these costs.

The rest of the text explains the use of the minimum function in Excel. For the teacher it is important that they know how this works and that they practise this when they do not have experience with this. This is important for the final assignment. Now, the students will use the laptop and solve the routing problem. The students will only focus on the optimal **costs**, not the optimal route. For the computation of the optimal route, the students need more knowledge on Excel and of the function argmin. We therefore only consider finding the minimum. The Excel file has been written in such a way that when students find the correct value, the optimal route will be given automatically. The students finish the code by following certain instructions. This is then the end of the lesson series.

Final assignment: 25 minutes
The students have to open the correct file and complete the red/pink blocks. The goal is that they find k_station. For the teacher, the file Routeprobleem-Les3Antwoord contains the correct solutions with the correct route. Below, there is an explanation why these computations are correct:

(a) For dynamic programming, we only have to look at the stations in the next stage. To find the optimal route, you find out to which stations you can go from station s_2 with the corresponding costs (found in $E4 : E8$). These values should be added to the k_station of the next stage (which are $H30 : H34$), of which we want to find the minimum. This can be found by filling in at s_2 that k_station =min($E4 : E8 + H30 : H34$), which results in the answer of 29 and best route 7. *This works, because the structure of the file is set up*

in such a way that the correct values are added to each other.

(b) This works in the exact same way as in question (a). For the next station, you add the costs to the values of k_station of the next stage (still stage 3). For station 3 we say k_station equals =min($E9 : E13 + H30 : H34$). The answers are as follows:

- Station 3: k_station = 31, Best route = 10.
- Station 4: k_station = 29, Best route = 8.
- Station 5: k_station = 29, Best route = 11.
- Station 6: k_station = 29, Best route = 7.

(c) Finally, we do the same for station s_1. We still take the costs to go to the next stage (which are the distances in column B). We set k_station =min($B4 : B8 + E30 : E34$). This gives as an answer 34 and the best route 5. So, the minimal costs to travel from station s_1 to station s_{27} equals 34. The optimal route can be found in the Excel file. This is the following:
$(s_1) \rightarrow (s_5) \rightarrow (s_{11}) \rightarrow (s_{12}) \rightarrow (s_{17}) \rightarrow (s_{25}) \rightarrow (s_{27})$.

This is the end of the lesson series. The students can hand in the exercises and the teacher can grade them.