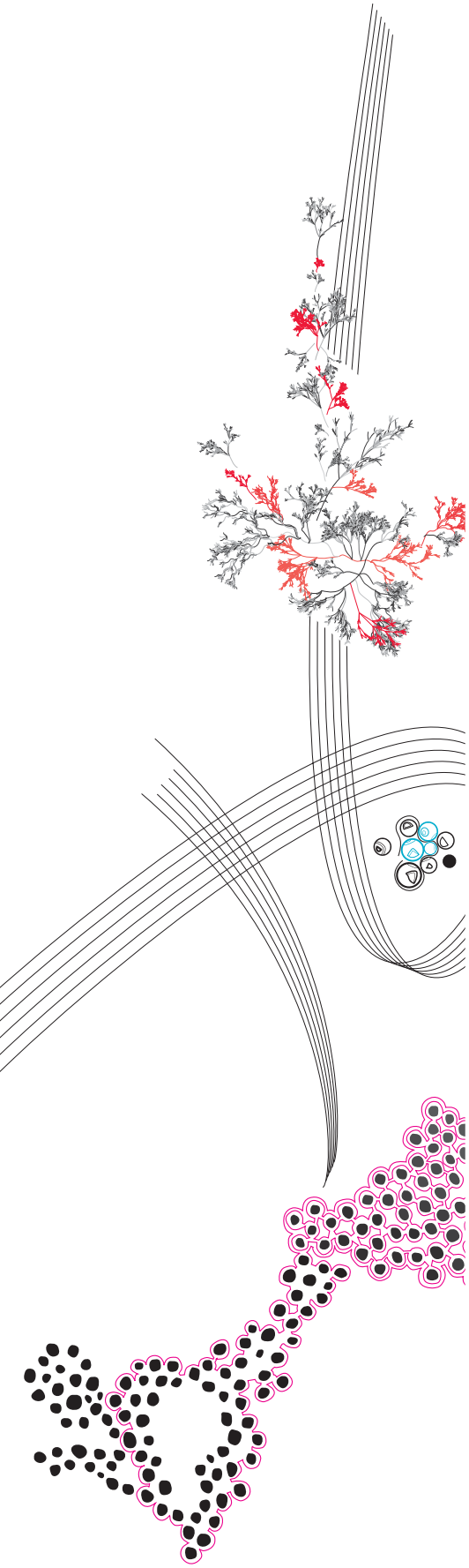
BSc Thesis Applied Mathematics



The most powerful theorem

Rutger van der Graaf

Supervisor: Clara Stegehuis

January, 2025

Department of Applied Mathematics
Faculty of Electrical Engineering,
Mathematics and Computer Science

UNIVERSITY OF TWENTE.

The most powerful theorem

R. van der Graaf*

January, 2025

Abstract

We are given a large dataset including all theorems, axioma's, definitions and more that exist within mathematics. This dataset is represented as a directed graph. In this graph, the theorems, axioma's and definitions are the vertices. There is a directed edge from vertex a to vertex b if vertex a is used to proof the existence of vertex b. This graph will be called the LPG.

We are interested in finding the strongest vertex in this graph. We first construct a measure to score each vertex which consists of the Pagerank, Betweenness centrality and Degree score. Additionally, we define the Pareto Front to identify undominated vertices.

Every vertex in the LPG has a certain category. We reduce the LPG to a graph that only contains these categories. This way, we first find out that the category 'Function' performs best in the designed statistic. We now look specifically in the sub graph that only contains vertices of the category 'Function'. We find that there are 4 vertices are performing equally good in the sub graph, of which the vertex 'Function.Injective.addMonoid' performs slightly better in the designed statistic than the other 3 vertices.

*Email: r.vandergraaf@student.utwente.nl

1 Introduction

As long as mathematics exists, formal proofs have been the corner stone of the existence of theorems. Only a correct proof can turn a proposition into a theorem. Whilst this is normally done by hand and without assistance, the technological era that we live in gives room for more assistance. An example of technological assistance is LEAN, an online programme that assists in proving mathematical theorems[1]. One can also refer to it as the LEAN Proving Assistant. For this programme to function, a large data set is created including all theorems, mathematical symbols, definitions and axioma. Every theorem in mathematics depends on other theorems via its proof. With this in mind, this dataset can be interpreted as a directed graph. In this directed graph, the theorems, symbols and every thing else that is found in the dataset is a vertex. There is a directed edge between vertices if the beginning vertex is necessary for the existence of the end vertex. The resulting graph will be referred to as the Lean Proof Graph(LPG).

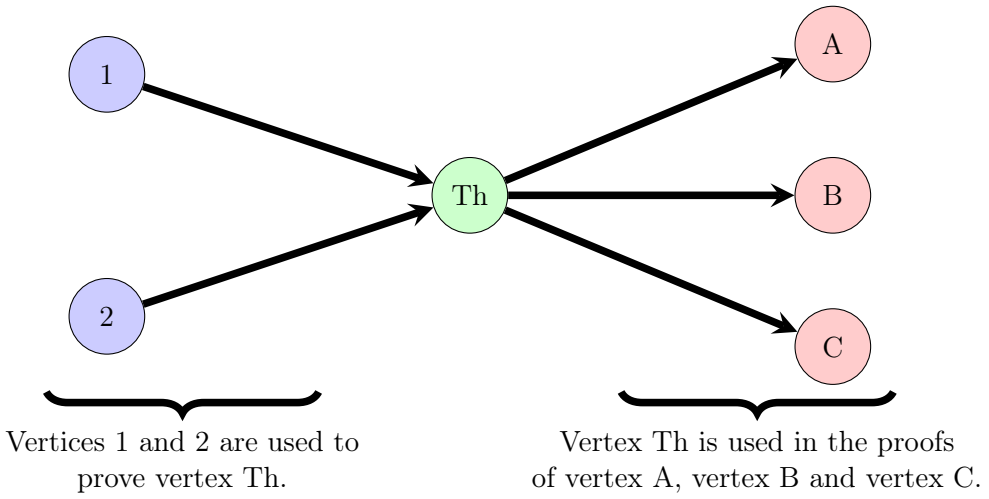


FIGURE 1: An example of the general build of the LPG

From this graph, it is interesting to find out which vertices can be said to be most important, or even which single vertex is most important. By analyzing the most important vertices in this graph, we can gain insights into the complexity and potential bottlenecks within the proof generation process. This analysis can help in optimizing the performance of proof generators but also provides a deeper understanding of the underlying mathematical principles.

The organization of LEAN already has tried to find out which theorem is most important. In their effort, they found that the equality sign is most important in the LPG. Here they only looked at one graph statistic, namely the degree. Common sense implies that the equality sign is not an interesting result, although justifiable when only looking at the degree of every vertex. Therefore, to determine what vertex is most important, it is first essential to define what properties of a vertex are of importance in this specific graph. Once we have defined what is important in the LPG, we can construct a measure, based on existing graph statistics, that will rank vertices on their importance.

All vertices in the LPG are placed in a category. Therefore, we will first reduce the LPG to a graph that only contains the categories and find out what the most important

category is. After this, we can zoom in on the most important category and find out what vertex is the most important in this category. Since this vertex is the most important vertex of the most important category, we can say that this vertex is the most important over the whole LPG.

These four steps can be written out as a research question and three sub research questions.

Research question
What is the most important vertex in the 'LEAN Proof Graph'?
Sub research questions

1. *What does it mean for a vertex to be important in the 'LEAN Proof Graph'?*
2. *What measure is most effective at finding the most important vertex in the 'LEAN Proof Graph'?*
3. *What is the most important category in the reduced 'LEAN Proof Graph'?*

2 Existing graph statistics

There exist several graph statistics that indicate how important a certain vertex is in a graph. We will provide a small summary of the graph statistics to get an idea how the statistic works and what the statistic actually measures. For all graph statistics, we will be looking at a directed graph $G = (V, E)$ where V is the set of vertices and E is the set of directed edges. Also, we will say that $|V| = n$ and $|E| = m$.

2.1 Betweenness centrality

Betweenness centrality is a graph statistic that exists both for directed and undirected graphs[7]. Betweenness centrality gives a value of vertex importance for a vertex j. We get this value by looking at the amount of shortest paths from vertex i to vertex k that include vertex j: $g_{i,k}(v_j)$. Relative to all the shortest paths from vertex i to vertex k: $g_{i,k}$. This gives

$$b_{i,k}(v_j) = \frac{g_{i,k}(v_j)}{g_{i,k}}. \tag{1}$$

We need to do this for every combination of vertices to get a good understanding of how important vertex j is. We do this by first defining how important vertex j is to vertex i by summing over all the vertices k. This gives

$$d^*_{i,j} = \sum_{i=k}^{n} b_{i,k}(v_j), i \neq j \neq k. \tag{2}$$

We can interpret this as a matrix with entries $d^*_{i,j}$ on the spot (i, j) where we can see how dependent vertex i is on vertex j in reaching all the other vertices. From this we can finally compute the Betweenness centrality of vertex j by summing over all vertices i, which gives

$$C_B(v_j) = \sum_{i=1}^{n} d^*_{i,j}. \tag{3}$$

The value of $C_B(v_j)$ becomes large once a graph becomes large. Therefore, we normalize (3) with the maximal Betweenness centrality that is possible for a graph with the same properties as the graph that is analyzed. To compute this we need the number of vertices with incoming arcs, N_I, the number of vertices of with outgoing arcs, N_O, and the number of vertices with incoming and outgoing arcs, N_S. With these three values, the maximal Betweenness centrality C_B^* is defined as follows.

$$C_B^* = (N_I - 1)(N_O - 1) - (N_S - 1). \tag{4}$$

This formula comes from the fact that a star-like graph has the highest Betweenness centrality, which is proved by R. White and P. Borgatti[7]. Therefore, if you can create a star-like graph with the same N_I, N_O and N_S as the graph that is analysed, you can calculate the maximal Betweenness centrality. Next, we get the relative Betweenness centrality $\hat{C_B}$ of a vertex j.

$$\hat{C_B}(v_j) = \frac{C_B(v_j)}{C_B^*} = \frac{C_B(v_j)}{(N_I - 1)(N_O - 1) - (N_S - 1)}. \tag{5}$$

For a large graph like the LPG, it might be very time costly to compute the Betweenness centrality. This mostly comes from counting all the possible shortest paths between every pair of 2 vertices. There are exactly $n \cdot n - n$ pairs. This becomes extremely large for large n, which is the case for the LPG with around a million vertices.

In the LPG, if a vertex has a high Betweenness centrality it means that this vertex is used in relatively many shortest paths from one theorem to another theorem. A path from a theorem to another theorem can be interpreted as the beginning theorem indirectly also being needed in the final theorem. Being in this shortest path, it implies that this theorems needs the beginning theorem for its proofs and indirectly helps proving the final theorem. Therefore, we can say that this theorem is important. Since we look at every combination of theorems, a theorem with high Betweenness can be considered important in the LPG. Therefore, vertices with high Betweenness centrality are strong in the LPG.

2.2 Katz Centrality

Katz centrality is a measure that looks not only at the immediate connections but also at connections that are further away from the vertex that is analysed[4]. This means that a certain vertex will get a higher score is the vertices around it also have a high score.

Katz centrality is calculated as follows. First you need to calculate the Katz Matrix KC. This is done as follows:

$$KC = (I - \alpha A)^{-1} - I. \tag{6}$$

In equation (6), A represents the adjacency matrix of the given graph. The damping value α should be chosen to be greater than zero and smaller than the maximal eigenvalue of A, λ_{max}. In simple sense, the damping value determines how much further into the graph we are looking when calculating the centrality of a vertex. The identity matrix is given by I. To get the Katz centrality of a vertex v_i, you simply sum over the row i of

the Katz Matrix KC as follows.

$$KC(v_i) = \sum_{j=1}^{N} KC_{i,j}. \tag{7}$$

In the LPG, it might be interesting to investigate vertices with high Katz centrality. If a vertex has high Katz centrality, it means that the theorem itself can be considered important because the theorems that are close to it in the graph are also considered important. Katz centrality stands out as a measure since it looks not only at the immediate connections but also at vertices that are further away.

2.3 Vertex degree

The vertex degree might be one of the easier methods. In a directed graph, the degree $d(i)$ of a vertex v_i consists of an in-degree, $d_{in}(i)$, and an out-degree, $d_{out}(i)$. This gives that de degree of an vertex v_i is $d(i) = (d_{in}(i), d_{out}(i))$.

For our graph, if a vertex has a high in-degree, it means that a lot of theorems were needed to complete the proof of this theorem. This may imply that this theorem was hard to prove.

If a vertex has a high out-degree, it means that this theorem is used to prove a lot of different other theorems. This might imply that this theorem is important, but we still need to be careful. Since the LPG has some "theorems" that are used in every day practice and almost every proof, these theorems have high out-degree, whilst not being interesting as the strongest theorem.

From the degree, we can construct part of the final measure used to score vertices in the LPG. Before we will construct this part of the final measure, we need to investigate how the in- and out-degrees are distributed over all the vertices. Later, we will set up a valid measure.

2.4 Eigenvector centrality

Eigenvector centrality works almost the same as Katz centrality. The only difference is the fact that eigenvector centrality looks only at the first neighbour of the vertex instead of looking further than one vertex[4]. It is calculated as follows:

$$\lambda_{max} x = xA. \tag{8}$$

Here, just like the Katz centrality, A is the adjacency matrix and λ_{max} is the maximal eigenvalue of A. When solving this equation for x, x_i is the eigenvector centrality for vertex i.

In literature, it can be found that the eigenvector centrality does not give a proper indication of centrality if the graph is acyclic[4]. The LPG is an acyclic graph by definition. Therefore, the eigenvector centrality is not that useful.

2.5 Pagerank

Pagerank is a measure created to rank the importance of a webpage[6]. It can be seen as a score based on how often a certain vertex is passed through during a random walk in

the graph in combination with the chance of randomly jumping in the graph. In some sense it is another variant of eigenvector and Katz centrality. The Pagerank vector P satisfies[2]:

$$P = \gamma A^T P + \frac{1-\gamma}{n}\mathbf{1}. \tag{9}$$

Here A is once again the adjacency matrix. The equation is split up in two parts. The first part accounts for the random walk that is being done on the graph. The second part accounts for the random jump to another vertex in the graph. We set γ as the damping value. This value gives the distribution between the random walk and the random jump. That is, with probability γ, the random walk chooses a neighbor and with probability $1-\gamma$ we jump to a random vertex. Generally, this value is set to 0.85. Solving equation (9) gives a 1 by n vector P. The Pagerank of a vertex i is equal to the value in position i in P.

Due to the possibility of a random jump, vertices that are normally not that central in the graph still get the chance to get a score in the Pagerank. In the end, if a certain vertex is more central, they will still get the higher score since they will appear more often in the random walks. Pagerank gives a well distributed score to each vertex based on how important the vertex is. The Pagerank does favor the in-degree of vertices over the out-degree of vertices.

3 Basic dataset Analysis

To get a better understanding of how the LPG behaves and what the LPG looks like, we perform some general analysis on the data. This includes getting an idea of how big the dataset is, looking into the distribution of in- and out-degrees of all the vertices and looking into the different categories the data is split up in. The LPG has 1.146.768 vertices and 14.889.516 directed edges. Since the LPG is a large graph, we will not give a visual representation of the LPG.

3.1 In- and out-degree

We can visualize the distribution of in- and out-degrees of all vertices. For the in- and out-degree, we will look at the degree of a vertex against the probability that a certain vertex has higher or equal degree. That is, on the x-axis all possible in- and out-degrees and on the y-axis $P(x \geq d_{in/out})$, see Figure 2 and 3.

Figure 2 and 3 give us some insight on how the graph and its vertices behave. Note that in the LPG, an in-degree of x means that this theorem needs x theorems to be proven and an out-degree of x means that this theorem is used in x different proofs. In Figure 2 we can see that the minimal in-degree of a vertex is 0 and the maximal in-degree of a vertex is 570. This means that there exists a theorems that uses 570 different theorems in its proof. Interestingly, about 18% of all theorems in the LPG have an in-degree of 0. This means that 18% of the dataset is accepted as true without requiring a proof. When looking at the theorems that fall in this 18%, you can see that this generally consists of axioma and objects that are needed for completeness of the LEAN program, like the vertex called 'obj_'.

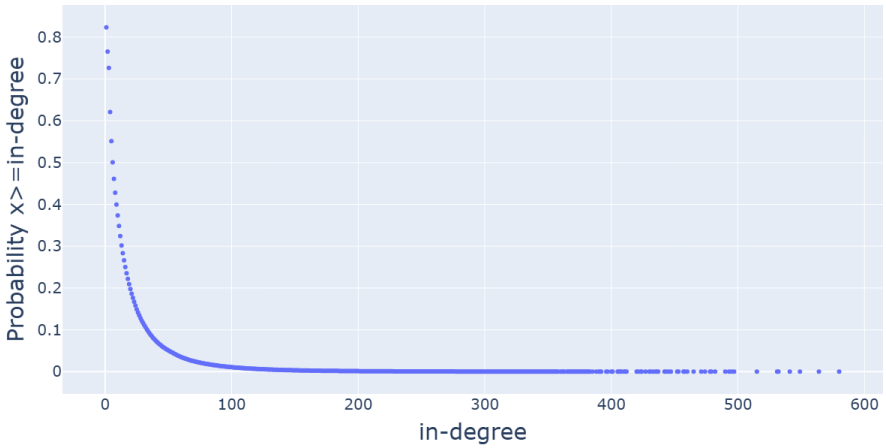


FIGURE 2: In-degrees against probability of the LPG

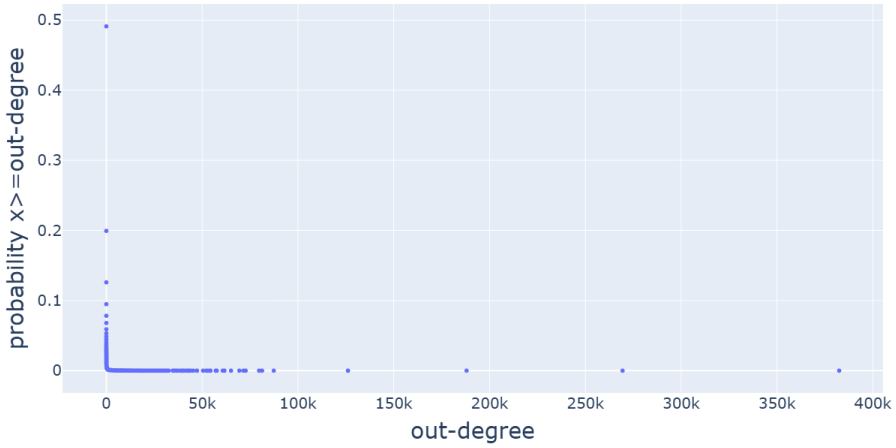


FIGURE 3: Out-degrees against probability of the LPG

The out-degrees of the vertices have larger variance. The minimal out-degree of a vertex is 0 and the maximal out-degree of a vertex is 380.000. When looking closely at the data, this distribution does kind of make sense. About 50% of all vertices have an out-degree of 0. This means that half of all the theorems in the LPG do not contribute in any of the proofs. These are probably theorems that are specific for a certain problem and therefore are not needed in the proof of any other theorems. Moreover, 95% of all theorems have an out-degree of 8 or less. On the other hand, the remaining 5% of vertices have a degree between 8 and 380.000. This means that a small percentage of the theorems are needed in many general proofs within mathematics. This could for example be the Triangle inequality, a theorem that is used in many proofs across different fields of mathematics. There are some outliers in this. For example, when looking at how many vertices have an out-degree of 50.000 or higher, we only get a list of 22 vertices. Giving this list a closer look, we notice quickly that it is not very odd that there are vertices with such a high out-degree. For example, the equality sign '=' has a out-degree of around 270.000. This simply means that the equality sign is used in about a quarter of all the theorems.

This of course makes sense, but is not an interesting result as 'most powerful theorem' whilst looking at the out-degree. There are a lot of 'theorems' in the LPG that are in literal sense important to have in mathematics but ultimately don't give an interesting result. This is something to consider whilst constructing the measure for the LPG.

To get some final insight in the distribution of the degree of all vertices, we have made a scatterplot of the in- and out-degree combination of every vertex in Figure 4. The x- and y-axis are on a logarithmic scale. Since a logarithmic scale does not allow an entry of zero, an in- or out-degree of zero is replaced by 0.1 such that it is shown.

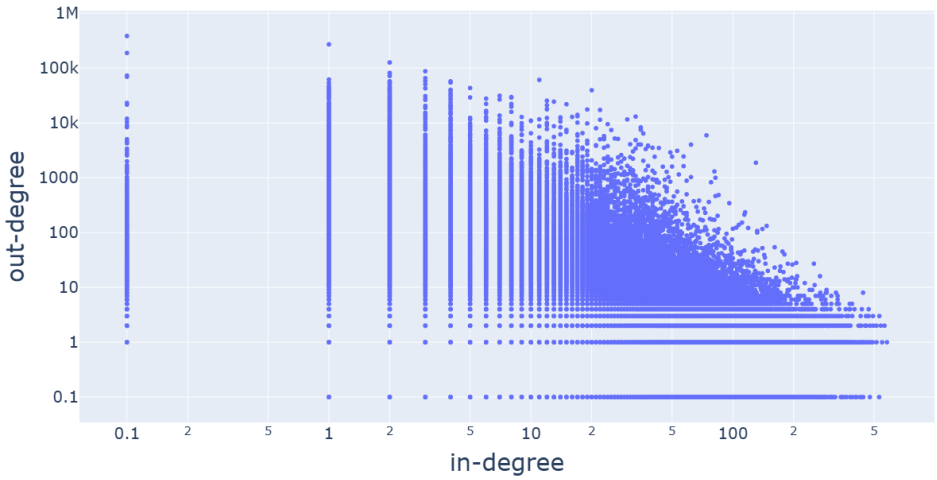


FIGURE 4: Scatterlogplot of (d_{in}, d_{out}) of all vertices of the LPG

In Figure 4 we can see wether there is a correlation between high in-degree and low out-degree and vice versa. Of course, you do have to keep in mind that the x-axis goes from 0 too 600 and the y-axis goes from 0 to 400.000. When calculating the correlation between the in- and out-degree, we find that there actually is no clear correlation with a score of -0.0059.

3.2 Categories

All the vertices in the LPG are divided into several categories. This can be seen by the names given for the vertices. Take for example the vertex called *'CategoryTheory.Sum.inl_.proof_2'*. In the name of this vertex, we can see that the name is separated with a dot. We will define the string of text before the first dot as the main category. This way, we can split up all vertices in their own respective category. For the example vertex, the category would be *'CategoryTheory'*. When doing this, we find out that there are 27.813 different categories in the LPG. On average, this would mean that there are around 40 vertices per category. When looking more closely at the different categories, we find out that this is not the case. In Figure 5 you can find a similar figure that we also used for the distribution of the in- and out-degrees of the graph. On the x-axis you can find the size of a category and on the y-axis you can find the probability of a category size being larger or equal than x.

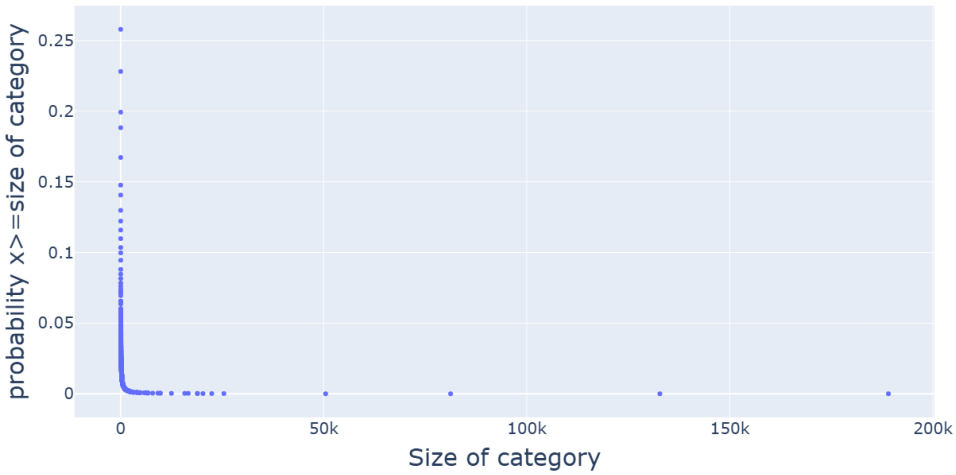


FIGURE 5: Category size against probability

From here we can see that around 25% of the different categories have a category size of 2 or more. This means that 75% of all categories have a size of 1. These are singular categories, that is categories that only consist of one theorem. This makes up for around 20.500 of the 27.000 different categories. When looking more closely at these singular categories, we see that once again most of these 'theorems' are actually objects that are needed for completeness of the LEAN program.

Range category size	Number of categories
1-10	24588
11-100	2491
101-1.000	629
1.001-10.000	93
10.001 or more	12

TABLE 1: How many categories are in the range of category sizes

In Table 1 we have divided the sizes of categories in bins and see how many categories have a size that falls into the respective bin.

4 Methods

Now that we have more insights of the properties of the graph, we will create the methods of finding the strongest vertex in the LPG. Included in the methods is the way of ranking vertices and reducing the graph to a size that manageable for computation time. We will reduce the graph to a graph that only contains the categories. Doing this, we can find the strongest category of the LPG, after which we can find the strongest vertex within the strongest category.

4.1 Measure

To find the strongest vertex in our directed graph, we will have to set up a function that scores each vertex. From the existing graph statistics that we have investigated, we will use the Pagerank, Betweenness centrality and vertex degree. We will calculate the total score in pycharm using the available package networkx to work with the graph. Betweenness centrality and Pagerank are fairly straightforward using networkx.

4.1.1 Betweenness centrality

There is a function in networkx that can calculate the Betweenness centrality for a given weighted and unweighted directed graph. The one sidenote here is the fact that the calculation of the Betweenness centrality has a quadratic complexity. This implies that the computation time increases drastically when increasing the amount of vertices. Therefore, we can only use Betweenness centrality if our graph is 'small' enough such that we can still do the calculation in reasonable time. Small enough in this context means that the graph should have at most around 5.000 vertices to do the calculation within around 15 minutes. After the calculation is done, we normalize all scores with the highest score that is given. This way, all scores for Betweenness centrality are between zero and one. The normalized Betweenness centrality of vertex i will be called $BC(i)$.

4.1.2 Pagerank

Just like Betweenness centrality, there is also a build in function that calculates the Pagerank for a given weighted and unweighted directed graph. The time complexity of Pagerank is not quadratic or worse and therefore the computation time should not be a problem for the graphs that we will use during the research. There is one adjustment that we will have to make in the calculation of the Pagerank. Recall that Pagerank favors the in-degree of a vertex over the out-degree of a vertex. In our graph, the in-degree means how many theorems were needed to prove the theorem and the out-degree means for how many different proofs this theorem is used. In the context of the LPG, a theorem can be considered to be stronger based on how many different theorems it helps prove. This means that we would need to favor the out-degree instead of the in-degree of a vertex in the calculation of the Pagerank. This is easily arranged by creating a 'placeholder graph' that is exactly inversed of the graph of which we want to calculate the Pagerank. With inversed it is meant to flip all the directions of the edges such that all the in-degrees become out-degrees and vice versa. Now we can calculate the Pagerank of the inversed graph such that we get a score that favors the out-degree of the original graph. Once again, we normalize all scores that are given with the maximal score that is given such that all scores will be between zero and one. The normalized Pagerank of a vertex i will be called $Pr(i)$.

4.1.3 Degree score

Lastly, we will include a measure that scores the vertices based on there in- and out-degree. We want to construct a measure that scores the vertices based on how close the in- and out-degree are to each other. There are some vertices that have very low in-degree and very high out-degree. In context, that would mean that this theorem is very easy to prove or even requires no proof after which it helps in many proofs. From the dataset analysis, we know that this would be the case for example for axioma or overly

simple element of mathematics like the equality sign. This would not be interesting as most important vertex. The other extreme would also not be interesting, namely very high in-degree and low out-degree. This would mean a vertex would be very hard and complex to prove or at least needs a lot of elements to prove after which it does not contribute in any other proofs, which would also not qualify as a strong or important theorem. If a theorem needs a reasonable amount of theorems to be proven and this theorem would then also help in a lot of other proofs, it qualifies to be considered strong. With this in mind we have constructed the following measure.

$$d_{score}(i) = 1 - \frac{|d_{in}(i) - d_{out}(i)|}{d_{in}(i) + d_{out}(i)} \tag{10}$$

Let us first look at the fraction in equation (10). In the numerator, we find the absolute value of the difference of the in- and out-degree. If the in- and out-degree are close to each other, the numerator will become smaller and the whole fraction will become smaller. Given that we subtract the fraction of one, it would imply that the closer the degrees are to each other, the higher the score will be. Note that when the in- and out-degree are equal, the degree score will be equal to one. The denominator of the fraction is the two degrees added to each other. This is done to take the size of these values into account. Take for example the degree combination (10,11) and (100,101). The difference of degrees in both these cases are 1. Relatively, the combination (100,101) is closer to each other than (10,11). When calculating the score, we find that the degree score of (10,11) is 0,952 and the degree score of (100,101 is) 0.995. Another property of this measure are that when one of the degrees is zero, the degree score will also be zero. Once again, if the in- and out-degree are equal, the degree score will be 1. That would mean that the combination (2,2) would get the same maximal score as (100,100), whilst (100,100) is clearly more interesting. This flaw will be fixed since the complete measure gives a combination of the three presented measures and in Pagerank and Betweenness centrality, (100,100) would get higher score than (2,2). If the in- and out-degree are both equal to zero, we define the d_{score} to be also equal to zero.

4.1.4 Complete measure

We will construct the complete measure with a linear combination of the three statistics that are presented above. This complete measure will look as follows.

$$TotalScore(i) = \beta_1 \cdot d_{score}(i) + \beta_2 \cdot Pr(i) + \beta_3 \cdot BC(i) \tag{11}$$

Where $\beta_1, \beta_2, \beta_3$ are values between zero and one and the sum off these is equal to 1. Since the degree score, Pagerank and Betweenness centrality all lie between zero and one and the $TotalScore$ is a linear combination of these three, we know that the $TotalScore$ is also a value between zero and one. The vertex that gets a higher $TotalScore$ than another vertex can considered to be more important.

The three variables $\beta_1, \beta_2, \beta_3$ can be adjusted. To get an evenly divided $TotalScore$, we can set all these values to $\frac{1}{3}$, but we can also adjust these variables if we for example find that the Betweenness centrality is less important then the Pagerank of a vertex.

4.1.5 Pareto Front

There is also another way of finding vertices with a good score. This way is based on the Pareto Front[5]. The Pareto Front is based on the dominance of a combination A

over a combination B. If you can find a combination that is not dominated by any other combination, we can say that this combination is in the Pareto Front. A combination A is said to be dominated by a combination B if B is as least as good as A in every objective.

We want to compare all the vertices with each other based on the three graph statistics that a vertex is given, which are the degree score, the Pagerank and the Betweenness centrality. For this we have defined three objectives in which the vertices can be compared to each other. The three objectives are

$$\begin{aligned} obj_1(i) &= d_{score}(i) + Pr(i), \\ obj_2(i) &= d_{score}(i) + BC(i), \\ obj_3(i) &= Pr(i) + BC(i). \end{aligned}$$

A vertex i is dominated if there exist a vertex i^* in the graph such that:

$$obj_k(i) < obj_k(i^*), \quad \forall\ k = 1, 2, 3 \tag{12}$$

All vertices that are not dominated can be considered to be the vertices with the best combination of scores. After doing this, we found that there are some vertices that were undominated but still have one or even two of the three scores quite small. Therefor, from the list of undominated vertices, we remove these outliers. A vertex i in the undominated vertices is an outlier if at least one of the three scores, $d_{score}(i)$, $Pr(i)$ or $BC(i)$ is 0.1 or less.

This way, we can find the best vertices before we determine how we set up the linear combination of the three statistics.

4.2 Creating Category Graph

Since we have knowledge of how the different categories of the graph behave, we can construct the reduced graph that only contains the different categories in the LPG.

The structure of the category graph(CG) is defined as follows. From the LPG, we take all the vertices of one category and turn it into a single vertex. All the edges that would go from a vertex to a vertex in the same category are disregarded. All the edges that would go from a vertex of one category to a vertex of a different category remain in the graph. Since the LPG will be heavily reduced, it is very possible that there will be multiple edges from one category to another category. Instead of keeping multiple edges, we will turn this into one edge with a weight equal to the amount of edges. This way, we can reduce that LPG to the CG. More formally, let $C_1, \ldots, C_k$ be the vertex categories. Then the CG has vertices $1, \ldots, k$. The weight of a egde i, j in the CG is $|\{\{u, v\} \in E_{LPG} : u \in C_i, v \in C_k\}|$

When originally creating the CG, we only used this rule. This gave an CG of around 27.000 vertices. From this graph, we removed the categories with a size of 25 or lower, resulting in a graph with around 1.100 vertices. After doing a calculation of the three different statistics, we found that the scores of Pagerank were highly scewed. This category with the highest score was ‘Eq’. Due to this high score, we found that ‘Eq’ is not a singular category, but a category with size at least 25 vertices. We first assumed that this is a singular category, since the vertex name ‘Eq’ has no separating dot in

its name. In fact, the size of the category is 67. For example, there is also a vertex with the name 'Eq.substr'. This category got such a relatively high Pagerank since the weighted out-degree was very high. This is due to the individual vertex 'Eq' that is also included in the category 'Eq'. Due to this individual outlier, the whole category became an outlier. Therefore, we found that we need to make a distinction between two types of outliers.

The first type of outliers are the vertices that have a high degree in the original LPG. The second type of outliers are the categories with a small enough size such that we will not have to consider them. The reason why we have to consider these both is the fact that the one may not imply the other.

Take for example the vertex 'Eq' again. This vertex has an out-degree of around 270.000. We thought that this vertex was a singular category since the vertex name has no separating dot and would therefor be disregarded when only looking at the second type of outliers. We discovered that the category 'Eq' has a size of 67, which means that this category would probably not be labeled as an outlier of the second type. If the vertex 'Eq' would be entered in the category 'Eq', this category would suddenly have a very high weighted out-degree which would strongly influence the Pagerank of this category and all the other categories as well. Therefore, it is important to first exclude the outliers of type 1 and afterwards the outliers of type 2 in creating the CG.

Due to this reduction of the graph, we will be able to calculate the the score of each vertex in reasonable time.

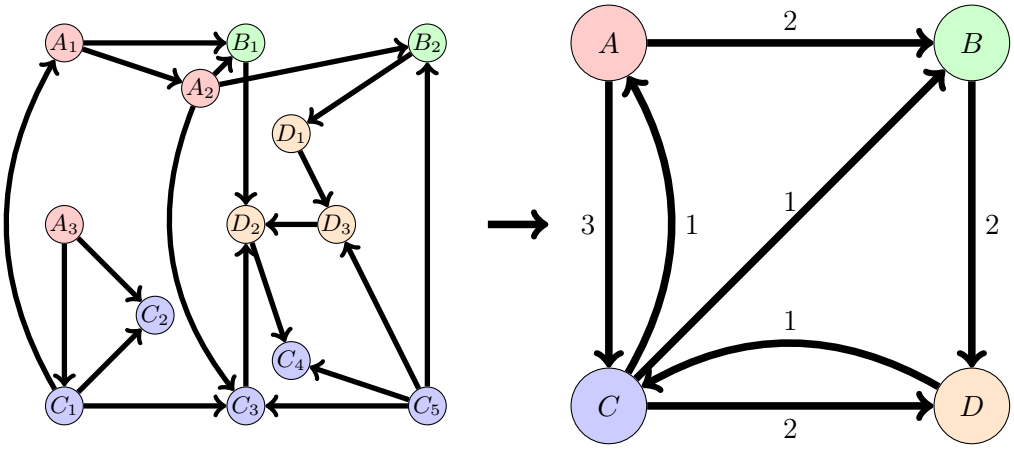


FIGURE 6: Reduction of the LPG to the Category Graph shown in an example graph.

5 Results Category Graph

Now that we have a properly defined graph and a properly defined measure in the $TotalScore(i)$ and the Pareto Front, we can start calculating the results.

For these results, all vertices with an out-degree of 50.000 or higher in the original LPG are considered type 1 outliers. Doing this, we get a list of 24 vertices that have been disregarded in the creation of the CG. Next, the CG is created. Now that we have the complete CG, we look at all the different sizes of the categories and remove all type

2 outliers. That is, all categories with a category size of x or below. To establish if the results are consistent, or are heavily dependent on the inclusion of small categories, different values for x will be considered. More specifically, we will look at x ranging from 5 to 100, where we will take steps of 5. This gives us a total of 20 cases that are considered.

There are also four categories that are significantly larger than the rest of the categories. These are the categories that have a category size of 50.000 or higher. These are the categories 'Lean', 'Mathlib','CategoryTheory' and '_private'. We have done the calculation once including these categories and once excluding these categories. This gives 20 cases when including the large categories and 20 cases when excluding the categories, giving a total of 40 cases that are considered. The complete results for each individual case can be found in the Appendix 10. Below we will give the most important findings.

In general, we find that the vertices that are undominated defined by the Pareto front are also vertices that are almost always in the top 5 vertices when the vertices are ranked based on there $TotalScore$. You can find these results in the Appendix 10.1. This gives good confidence that the Pareto front and the $TotalScore$ are both good ways of finding the strongest vertex in the CG. The only exception here is the vertex 'CategoryTheory'. This category is in de Pareto Front when the category size removed below is 55 (Figure 25) until 100 (Figure 34). This vertex is only in the top 10 vertices based on the $TotalScore$ when the category size removed below is 55 (Figure 25) until 65 (Figure 27).

To give some more validation of the different measures that are used in the calculation, we can take a look at the correlation matrix which includes the different statistics we have for each vertex. If there is a low correlation between various statistics, we validate that every statistic calculates a different property of a vertex. These different properties are explained in section 4.1. The statistics that are included in the correlation matrix are the d_{score}, Pagerank, Betweenness centrality, weighted in- and out-degree and the category size. This gives Figure 7.

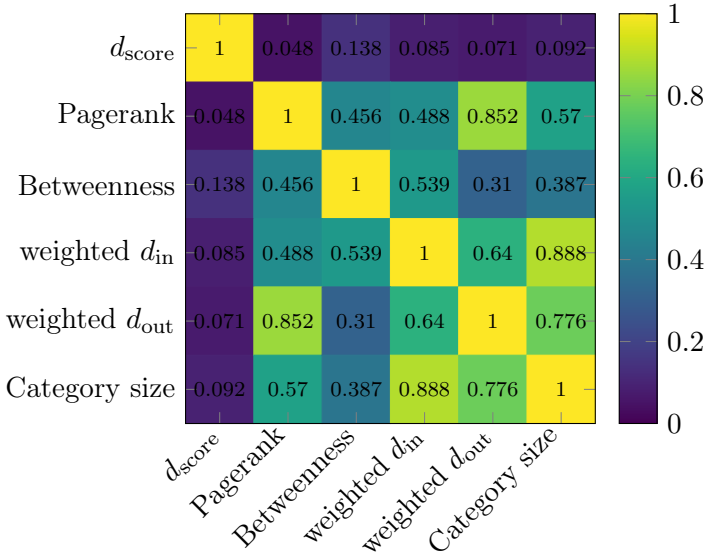


FIGURE 7: Correlation matrix of different graph statistics based on the CG

Note that this is a symmetric matrix. Also, in the diagonal we see that there is a correlation of 1. This is the case since in the diagonal, the statistics are compared with themselves, which logically results in a full correlation. We are mostly interested in the correlation between the d_{score}, Pagerank and Betweenness, since these are the three main statistics that are used. We find that there is practically no correlation with the d_{score}, which means that this statistic calculates a different importance of a vertex in comparison with the Pagerank and Betweenness. Between the Pagerank and the Betweenness, we find a correlation of 0.456, which indicates a correlation between vertices that receive a high Pagerank and a high Betweenness. This is not that surprising, since both statistics are generally used to calculate the importance of a vertex. Still it is good to see that there is no complete correlation, which shows that the two measures do rank the vertices in a different way.

Furthermore, we see a high correlation between the Pagerank and the weighted out-degree, being 0.852. This does not come as a surprise since the Pagerank was manipulated to favor the out-degree of a vertex, as seen in section 4.1.2. Here we see the validation of the manipulation.

5.1 Results including large categories

First we will look at the results that include the large categories. To get some insight in the results, we have made a table that includes the vertices that have appeared in the Pareto Front, how often they have appeared in the Pareto Front, their average $TotalScore$ and the size of the category. The $TotalScore$ is calculated with $\beta_1 = \beta_2 = \beta_3 = \frac{1}{3}$. Note that a category appear a maximal of 20 times in the Pareto Front.

Category	In Pareto Front	Average $Totalscore$	Category size
Lean	20	0.6650	189.049
Function	20	0.4792	3.952
Mathlib	18	0.5390	81.236
Array	2	0.3954	25.418
CategoryTheory	10	0.3556	132.763

TABLE 2: Results of categories that appeared in the Pareto Front at least once when large categories are included. $\beta_1 = \beta_2 = \beta_3 = \frac{1}{3}$

When looking at the individual results which can be found in the Appendix 10.1, we see that 'Lean', 'Mathlib' and 'Function' occupy the top 3 spots 85% of the time when looking at the $TotalScore$, with an exeption when the category size removed below is 5 (Figure 15), 95 (Figure 33) or 100 (Figure 34). 'Array' can always be found in the top 10, mostly varying between the 5^{th} and the 8^{th} place. Even though 'CategoryTheory' appears more often in the Pareto Front then 'Array', the average $TotalScore$ of 'Array' is better then that of 'CategoryTheory'.

To get even more insight on how the scores of the different categories behave, we can make a rankplot. In a rankplot, we can see how different categories rank relative to each other regarding different statistics. For the rankplot that we will make, we will look at the statistics d_{score}, Pagerank, Betweenness centrality and the $TotalScore$. The categories that are included in the rankplot are categories that appear in the top 5 at

least once in the statistic Pagerank, Betweenness centrality and $TotalScore$. The d_{score} is not included within the selection of the categories, since there are many vertices with d_{score} equal to 1 and therefor there is no clear top 5. Furthermore, these vertices generally have low degree, like (5,5), and will therefor also not score high in the other categories. We will only make the rankplot for the case where the category size removed below is 50 to provide some general insight.

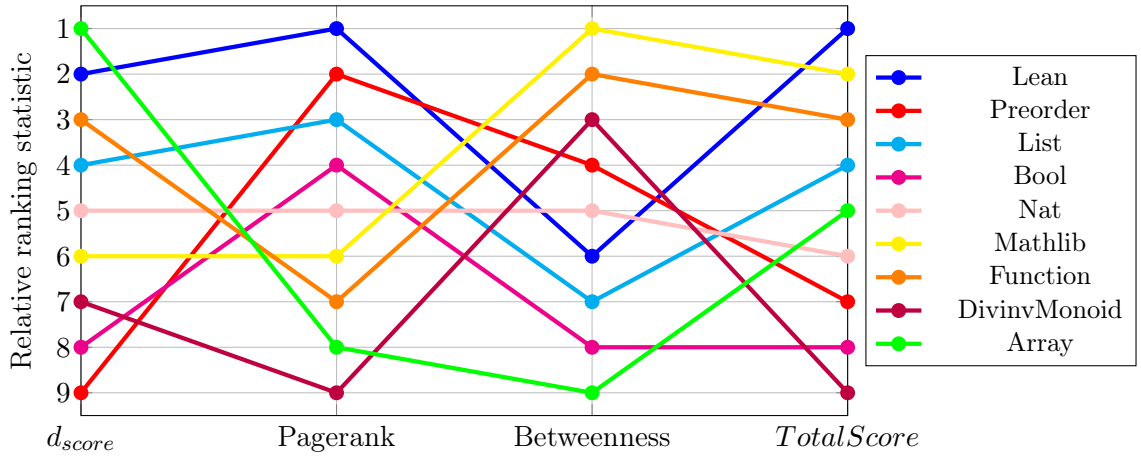


FIGURE 8: Rank plot of the vertices that are in the top 5 of at least one statistic when including large categories. Category size removed below is 50 and $\beta_1 = \beta_2 = \beta_3 = \frac{1}{3}$

As defined above, we see that every category is at least once in the top 5. Now we see that some categories that have not been mentioned before are also included. For example the category 'Bool' relatively has the 4^{th} highest Pagerank, but in the d_{score} and Betweenness centrality it scores very poorly, which gives it a final relative 8^{th} place in the $TotalScore$. Categories that we already saw before, like 'Lean', 'Function' and 'Mathlib' are seen to be scoring relatively good in more than one statistic, which in turn helps in scoring good in the $TotalScore$.

5.2 Results excluding large categories

Now we will look at the result when excluding the large categories. First, we will look at a table that is similair to Table 2 that gives the vertices that have appeared in the Pareto Front, how often they have appeared in the Pareto Front, there average $TotalScore$ and the size of the category. The $TotalScore$ is calculated with $\beta_1 = \beta_2 = \beta_3 = \frac{1}{3}$. Note that a category appear a maximal of 20 times in the Pareto Front.

Category	In Pareto Front	Average $Totalscore$	Category size
List	20	0.6198	20.269
Function	20	0.7017	3.952
Nat	13	0.5725	9.679
Filter	3	0.3970	5.796

TABLE 3: Results of categories that appeared in the Pareto Front at least once when large categories are excluded. $\beta_1 = \beta_2 = \beta_3 = \frac{1}{3}$

Something that we notice is that the category 'Function' shows up again. When competing with the big categories, the category already showed how important it is. When the larger categories are not in the picture anymore, this category does an even better job. Together with the category 'List', the categories show up in every Pareto Front. These two categories are consistently in the top 3 of the $TotalScore$ which can be seen in the Appendix 10.2. The category 'Function' beats 'List' in the cases where the category size removed below is 5 (Figure 35) until 80 (Figure 50). The category 'List' manages to get in front of of 'Function' when the category sizes removed below get to 85 (Figure 51) until 100 (Figure 54). The Category 'Nat' is in 85% of the cases present in the top 3. The category 'Filter' can only be found in the top 10 when the category sizes removed below are between 45 (Figure 43) and 60 (Figure 46).

We will also take a look at the rankplot when we exclude the large categories. The rankplot will have the same structure as Figure 8, only including categories that appear at least once in the top 5 of the Pagerank, Betweenness centrality or $TotalScore$. The category size removed below is 50.

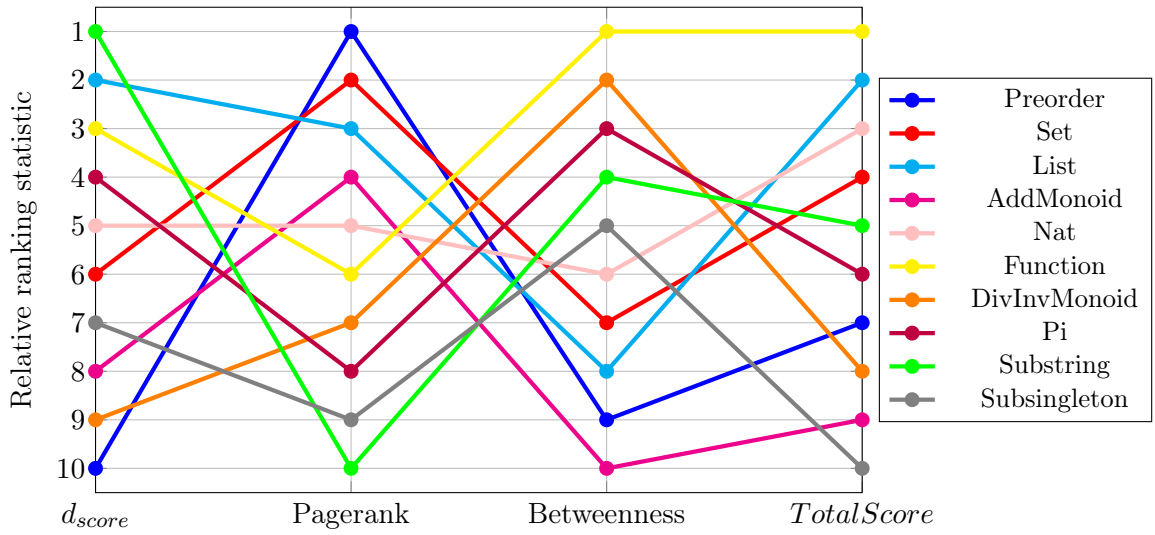


FIGURE 9: Rank plot of the vertices that are in the top 5 of at least one statistic when excluding large categories. Category size removed below is 50 and $\beta_1 = \beta_2 = \beta_3 = \frac{1}{3}$

Here we can see the behaviour of the various categories and there relative scores. As defined, every category appears in a top 5 at least once. We see that the category 'Subsingleton' is barely included in this rankplot with just a relative 5^{th} spot in the Betweenness centrality. Categories that we have looked at before in Table 3, like 'Function', 'Nat' and 'List' all do relatively well in every statistic, which gives more validation to these categories being stronger than other categories.

5.3 General result

The category 'Function' shows up in the Pareto Front for all 40 cases. Furthermore, this category finds itself in the top 3 based on the $TotalScore$ in 38 of the cases, where in the other two cases 'Function' be found on the 4^{th} spot (Figure 33 and Figure 34). In the 20 cases including the big categories, the average $TotalScore$ of 'Function' is 72% of the maximal average $TotalScore$, which is 'Lean'. In the 20 cases excluding the big

categories, 'Function' itself has the highest average $TotalScore$. Also, from the rankplots we find that 'Function' does relatively well in all different statistics, instead of having one strong statistic that carries the $TotalScore$. From these insights we can conclude that 'Function' is the strongest category in our statistic.

6 Analysis category 'Function'

Now that it is established that 'Function' is the strongest category in our statistic, we now will look at the graph that only contains the category 'Function'. We will call this graph the Function Graph(FG). The FG will have vertices that are in the category 'Function'. The edges that are included in the FG are only edges where the starting- and endpoint of the edge is a vertex of the category 'Function'. More formally, the FG will have the vertex set $V_{FG} = \{v \in V_{LPG} : v \in Function\}$ and edge set $E_{FG} = \{\{u, v\} : u \in V_{FG}, v \in V_{FG}\}$. We know from the category size, which can be found in Table 3, that this graph has 3.952 vertices. Besides that, their are 7.486 edges in the FG. To get some insight on how the FG behaves and is constructed, we will do a similar analysis as seen in Section 3.

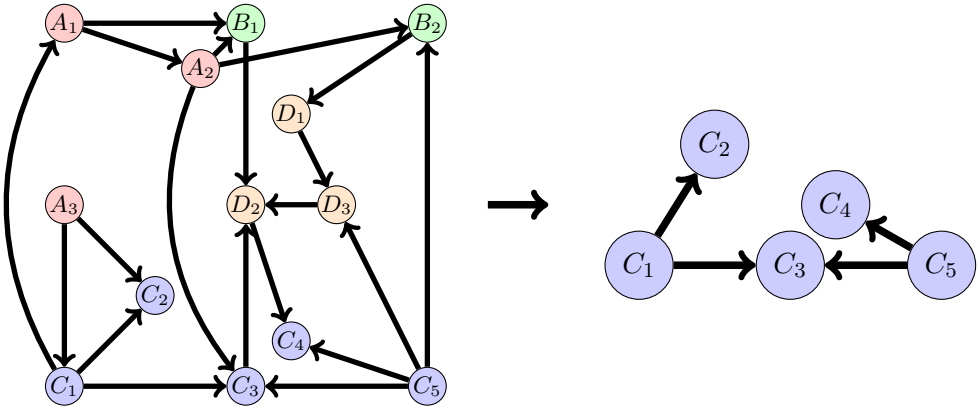


FIGURE 10: Reduction of the LPG to the Function Graph shown in an example graph.

To get some insight in the distribution of the in- and out-degrees of the vertices, we once again will look at the degree of a vertex against the probability that a certain vertex has higher or equal degree. That is on the x-axis all possible in- and out-degrees and on the y-axis $P(x \geq d_{in/out})$.

In Figure 11 and Figure 12 we see that there is less variance in the in-degree than in the out-degree, just as we saw in the LPG. It is again important to understand what the in- and out-degree mean in this sub graph. If a vertex has an in-degree of x it means that this vertex needs x vertices within the category 'Function' in its proof. Note that this is slightly different than the LPG, since the vertex might need more theorems to be proven, but these are from a different category that is not included in the sub graph. The same reasoning holds for the out-degree. An out-degree of x means that a certain vertex is used in x proofs within 'Function'. It might be that the vertex is used in even more proofs, but that might be of vertices that are not included in this sub graph.

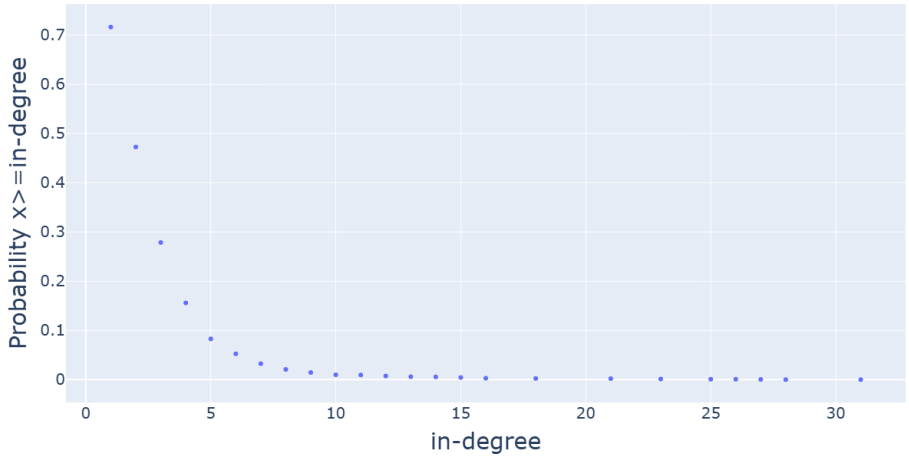


FIGURE 11: In-degrees against probability for the Function Graph

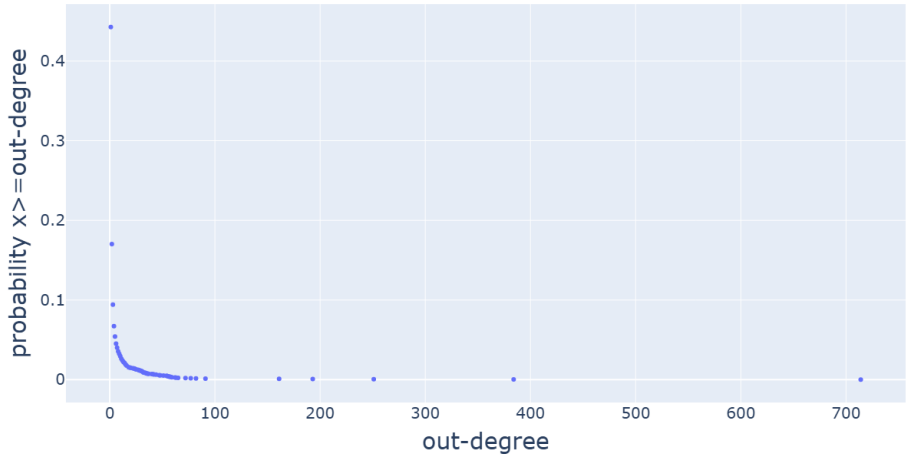


FIGURE 12: Out-degrees against probability for the Function Graph

From Figure 11 we find that the in-degree lies between 0 to 30. About 29% of the vertices have an in-degree of 0. From figure 12 we find that the out-degree of the vertices is ranging from 0 to 713. Moreover, around 55% of all vertices have an out-degree of 0. This means that more than half of the vertices does not contribute in a proof within the category 'Function'. About 9% of the vertices have an out-degree between 3 and 31. From this figure we can see vertices that have a significant higher out-degree than the rest of the vertices. These are the vertices with out-degree of 100 or more, which are 5 vertices in total.

Next we will see how the combination of in- and out-degrees are distributed in the FG.

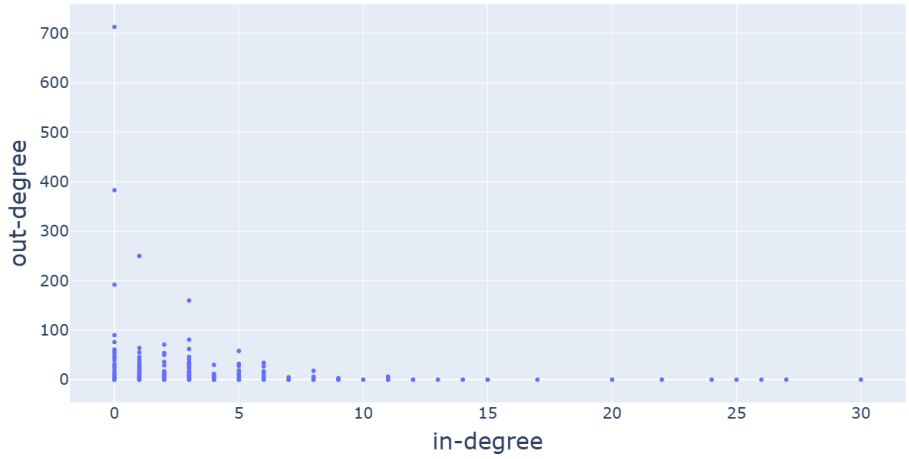


FIGURE 13: In- and out-degree of all vertices in the Function Graph

Here we find that the vertices of very high out-degree also have a low in-degree, being the combinations (0,713),(0,383),(1,250),(0,192) and (3,160). Since these high out-degrees will give a skewed result in the measure, we will consider them outliers.

Furthermore, interestingly there are vertices that have an in- and out-degree of 0. These are isolated vertices in the FG. This is due to the definition of the FG. In the LPG, these vertices are not isolated. Since we only include edges that have the starting- and endpoint within the category 'Function', edges that have only the starting- or endpoint in the category 'Function' are not included. In total there are around 450 vertices that are isolated.

7 Results category 'Function'

Now that we have some insight in the category 'Function', we can calculate the different statistics and Pareto Front.

As concluded in section 6, we will consider the vertices with an out-degree of 100 or more outliers. Furthermore, the isolated vertices will remain in the graph. These vertices will only have a slight influence on the Pagerank due to the random jump property of the Pagerank, but this influence will be minimal.

In the results for the Category Graph, we were able to look at different cases to obtain a wide spread of data after which a clear conclusion could be drawn. These cases were distinguished by removing categories with a certain category size. We will not be able to make a similar distinction for the Function Graph. The only variables that we will be able to adjust in the results is the linear combination of the graph statistics that are calculated. That is, we will be able to calculate a different $TotalScore$ of a vertex by adjusting the variables β_1, β_2 and β_3. Note that, when only adjusting these variables, the values for the d_{score}, Pagerank and Betweenness centrality will remain the same. This means that the Pareto Front will also be the same for all the different cases by its definition found in section 4.1.5.

With this in mind, we can look at different cases adjusting the variables β_1, β_2 and β_3

and thus the $TotalScore$, whilst only looking at vertices that appear in the Pareto Front.

There are 4 vertices that appear in the Pareto Front. These vertices can be found in Table 4 together with their respective d_{score}, Pagerank and Betweenness centrality.

Vertex	d_{score}	Pagerank	Betweenness
Function.Injective.addMonoid	0.1290	0.2728	0.5057
Function.Injective.monoid	0.2222	0.2872	0.4168
Function.instEmbeddingLikeEmbedding	0.6667	0.3193	0.1216
Function.Embedding.trans.proof_1	1.0000	0.2169	0.1275

TABLE 4: Vertices of the category 'Function' that appear in the Pareto Front

As said, the values presented in Table 4 will not change when β_1, β_2 and β_3 change. Furthermore, for convenience, from now on we will refer to the vertices only by the name after the last dot. That is, we will have the vertices 'addMonoid', 'monoid', 'instEmbeddingLikeEmbedding' and 'proof_1'.

As seen in Section 6, we find that all vertices have a relative low degree when comparing Figure 13 to Figure 4. Furthermore, we find that if vertices have an in-degree equal to the out-degree, these degrees are generally 10 or lower. From the definition of the degree score found in section 4.1.3, this gives a degree score of 1. In the Category Graph, the variance in degree is much higher than in the Function Graph. Therefor, in the CG the Pagerank and Betweenness centrality would balance the $TotalScore$ out appropriately with a distribution of $\beta_1 = \beta_2 = \beta_3 = \frac{1}{3}$. In the FG, the balancing out will be done less heavily. Therefor, the d_{score} will be counted less heavily in the different $TotalScore$'s that will be calculated.

The distribution of the variables β_1, β_2 and β_3 in the different cases is as follows.

Case	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
β_1	0.1	0.1	0.1	0.1	0.1	0.1	0.2	0.2	0.2	0.2	0.2	0.3	0.3	0.3	0.3
β_2	0.2	0.3	0.4	0.5	0.6	0.7	0.2	0.3	0.4	0.5	0.6	0.2	0.3	0.4	0.5
β_3	0.7	0.6	0.5	0.4	0.3	0.2	0.6	0.5	0.4	0.3	0.2	0.5	0.4	0.3	0.2

TABLE 5: Distribution $(\beta_1, \beta_2, \beta_3)$ for the $TotalScore$

In Table 5 we see that β_1 ranges from 0.1 to 0.3, whilst β_2 and β_3 range from 0.2 tot 0.7. This ensures that the d_{score} still has a place to show its worth in the cases 12 to 15, but also makes sure that at least one of the Pagerank or Betweenness centrality has a stronger contribution in the $TotalScore$.

To get a clear overview of how the 4 vertices in Table 4 perform against each other in the different 15 cases, we will look at a rank plot. In this rankplot, we find on the y-axis the relative rank of the 4 compared vertices and on the x-axis the case that is considered.

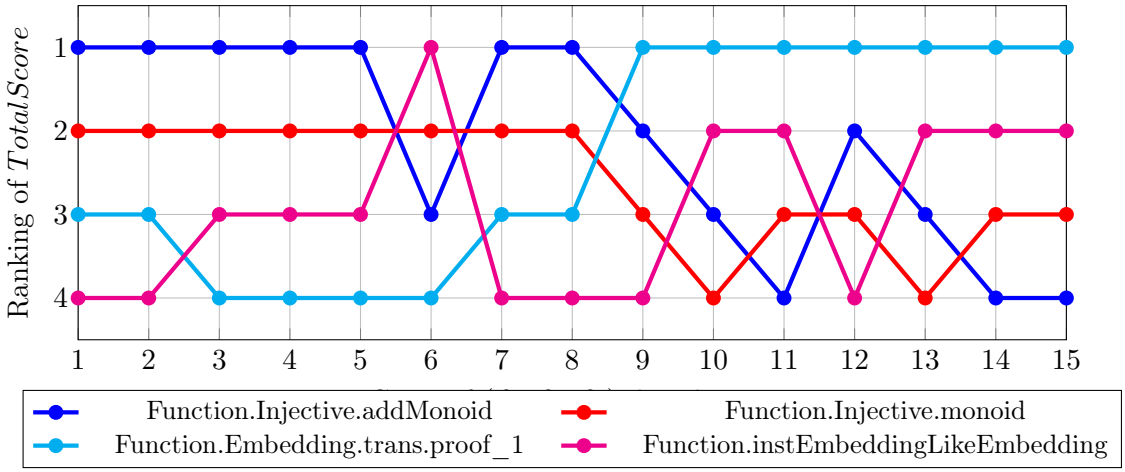


FIGURE 14: Rank plot of the vertices in Table 4 using the cases in Table 5

In Figure 14 we see that there is clear shift in the ranking when looking at the different cases. It is not surprising that the vertex 'proof_1' scores well in the cases 12 to 15, as these are the cases where the d_{score} has a higher contribution. This vertex also ranks first in the cases 9, 10 and 11. In these cases, the d_{score} contributes 0.2 and the Pagerank contributes more or equal than the Betweenness.

There is no vertex that ranks strictly worse than another vertex. This partly due to the fact that the compared vertices are in the Pareto Front and therefor are undominated in at least one objective by definition. Still, when comparing the vertices with each other, we see that one is scoring better than the other. When comparing the ranks of the vertex 'addMonoid' to the other three vertices 'monoid', 'instEmbeddingLikeEmbedding' and 'proof_1', we find that 'addMonoid' outranks all vertices with a score of 11-4, 9-6 and 8-7 respectively. Next, we can compare 'monoid' to the vertices 'instEmbeddingLikeEmbedding' and 'proof_1'. In these two cases, 'monoid' outranks both vertices with a score of 9-6 and 8-7 respectively. Lastly, when comparing 'instEmbeddingLikeEmbedding' to 'proof_1', we find that 'proof_1' wins with a score of 11-4.

We can also take a look at the average rank of each vertex based on these 15 cases. This average rank can been seen in the following table.

Vertex	addMonoid	monoid	instEmbeddingLikeEmbedding	proof_1
Average rank	2.13	2.60	2.93	2.33

TABLE 6: Average rank over the cases in Table 5

A higher average rank means that a vertex did well on average over all the 15 cases defined in Table 5. We saw that 'addMonoid' outranked every vertex and here we see that this vertex also has the highest average rank. Only 'monoid' and 'proof_1' have swapped places when compared to the outranking. 'monoid' outranks 'proof_1' with a score of 8-7, but 'proof_1' has a clear higher average rank.

From this analysis, we find that the vertex 'addMonoid' is performing the best, looking at the determinded 15 cases. Nevertheless, all four vertices that are in the Pareto Front are

performing well against each other and therefor we can argue that all these four vertices are the best, with a slight edge towards the vertex 'Function.Injective.addMonoid'.

8 Conclusion

In this paper, we first compared different graph statistics to get an insight on what properties of a vertex a specific statistic ranks. We find that for the LPG, the Pagerank and the Betweenness centrality are two statistics that rank important properties. Furthermore, we design a statistic based on the in- and out-degree of a vertex, giving a vertex a higher score when the in- and out-degree are relatively close to each other.

From these three statistics, we construct two measures to find the strongest vertex. The two measures are the $TotalScore$ and the Pareto Front. The $TotalScore$ ranks a vertex via a linear combination of the three statistics. The Pareto Front finds undominated vertices via objective functions based on the three statistics.

Using the measures that we have defined, we showed that there are 5 categories when large categories are included to be strong, and 4 categories when large categories are excluded to be considered strong. Since there is one vertex overlap in these two sets, we obtain eight categories to be considered strong out of approximately 27.000 categories in the LPG. These eight categories are 'Lean', 'Function', 'Mathlib', 'Array', 'CategoryTheory', 'List', 'Nat' and 'Filter'. From these eight, we found that the category 'Function' is the strongest considering all cases based on the statistic.

Within the category 'Function', we once again looked at what vertices are the strongest. Of the 3.952 vertices that are within this category, we considered four vertices to be the strongest. These four vertices are 'Function.Injective.addMonoid', 'Function.Injective.monoid', 'Function.instEmbeddingLikeEmbedding' and 'Function.Embedding.trans.proof_1'. The scores of these four vertices are all very close over 15 cases that were considered. The best score is given to the vertex 'Function.Injective.addMonoid'. Nevertheless, since the scores are close there is some room for interpretation and therefore all four vertices are considered the strongest.

When looking at the meaning of all vertices on the website of LEAN[3], we find that the vertex 'Function.Injective.addMonoid' has something to do with the additivity property of a function. That is, $f(x+y) = f(x) + f(y)$. For more information, you can go to the website of LEAN[3] and typing in the vertex name in the search bar.

9 Discussion

There are many options for further research regarding the analysis of the LPG. First of all, there are several options too look at whilst still using the same measure that has been presented. In the research, we applied the measure only on reduced versions of the LPG, namely the Category Graph and the Function Graph. This was mainly done to reduce the computational time significantly. With the use of a supercomputer and a more optimized code, one would be able to perform the measure on the complete LPG in one go. Next, instead of only looking at the category 'Function', you can also take a look at how the vertices perform in the other categories that were in consideration for the strongest category to gain more general insight on how these categories perform.

Another adjustment that can be made in my research lies in the definition of the Function Graph. The only edges that were included in this graph were edges which had their beginning- and endpoint incident with a vertex of the category 'Function'. As a result, vertices that had high out-degree to vertices in a different category had an out-degree of 0 in the Function Graph. In further research, It would be interesting to see what happens with the results when these edges are also included in the graph, whilst only applying the measure to the vertices of the category 'Function'.

The data that was provided by LEAN included everything that is used to make the proving assistant work. This means that there are not only theorems in the dataset, but for example also definitions. On the website of LEAN[3], we see that every vertex has a 'type'. This type can for example be 'def', 'theorem', 'abbrev' or many more. This meant that finding the strongest 'theorem' was harder, since a lot of different types of vertices also performed well due to being central in the graph. In further research it would be interesting to only include vertices from the graph that are of the type 'theorem'. In this new graph, we would than certainly find the strongest theorem instead of the strongest vertex.

A whole different approach that can be done is looking at longest directed paths in the LPG. The endpoint of this longest path can be considered to contain all the information of the vertices that came before in the path.

What remains to be done is continuing obtaining interesting data from the LPG. Since the LPG is very large, there are likely many more different statistics that can be obtained from the LPG. This can for example be done by one of the approaches which we refer to above, or even a completely different approach.

References

[1] https://lean-lang.org/theorem_proving_in_lean4/introduction.html. Accessed: 2024-12-17.

[2] https://towardsdatascience.com/pagerank-algorithm-fully-explained-dc794184b4af. Accessed: 2024-11-10.

[3] https://leanprover-community.github.io/mathlib4_docs/. Accessed: 2025-01-09.

[4] Seyed Mojtaba Hosseini Bamakan, Ildar Nurgaliev, and Qiang Qu. Opinion leader detection: A methodological review. *Expert Systems with Applications*, 115:208–209, 2019.

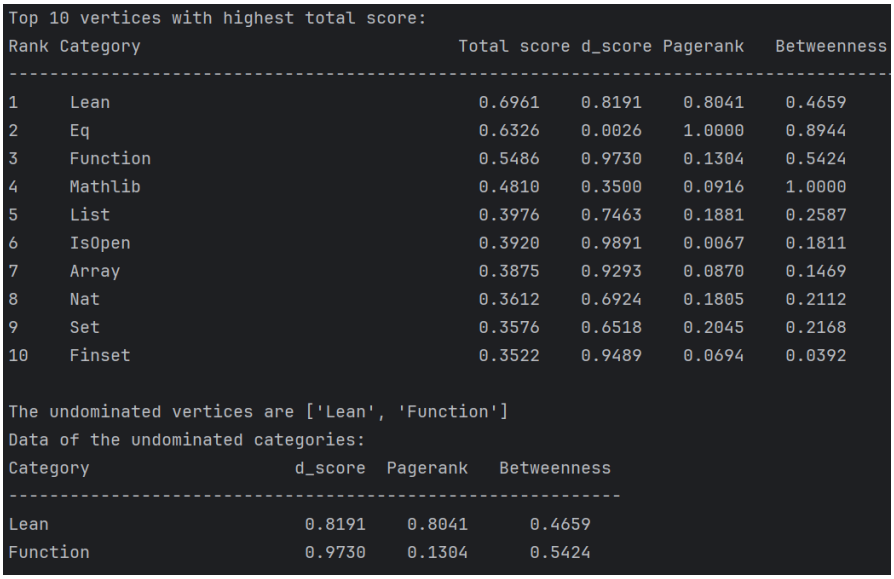
[5] Naru Okumura, Keiki Takadama, and Hiroyuki Sato. Pareto front estimation model optimization for aggregative solution set representation. In *2024 IEEE Congress on Evolutionary Computation (CEC)*, page 2. IEEE, 2024.

[6] Lawrence Page. The pagerank citation ranking: Bringing order to the web. Technical report, Technical Report, 1999.

[7] Douglas R. White and Stephen P. Borgatti. Betweenness centrality measures for directed graphs. *Social Networks 16*, pages 335–346, 1994.

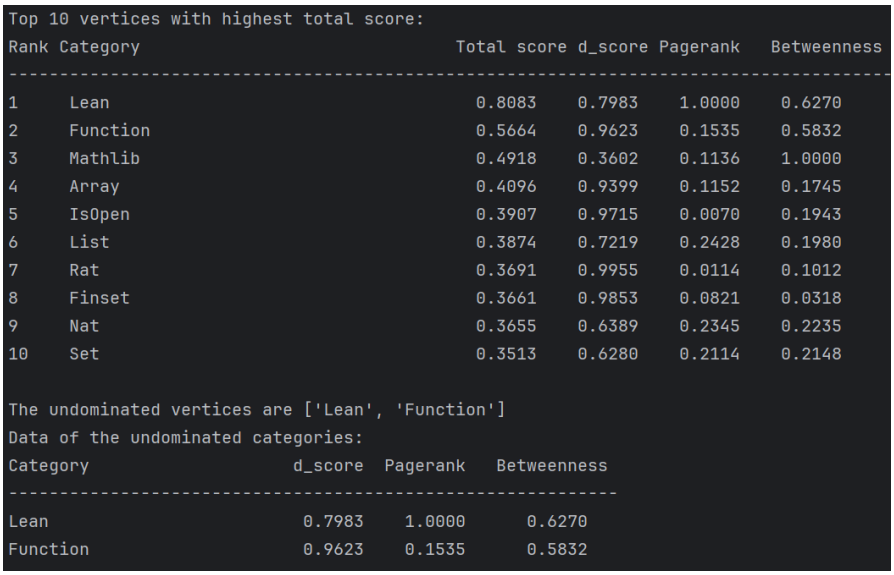
10 Appendix

10.1 Results Category Graph including large categories



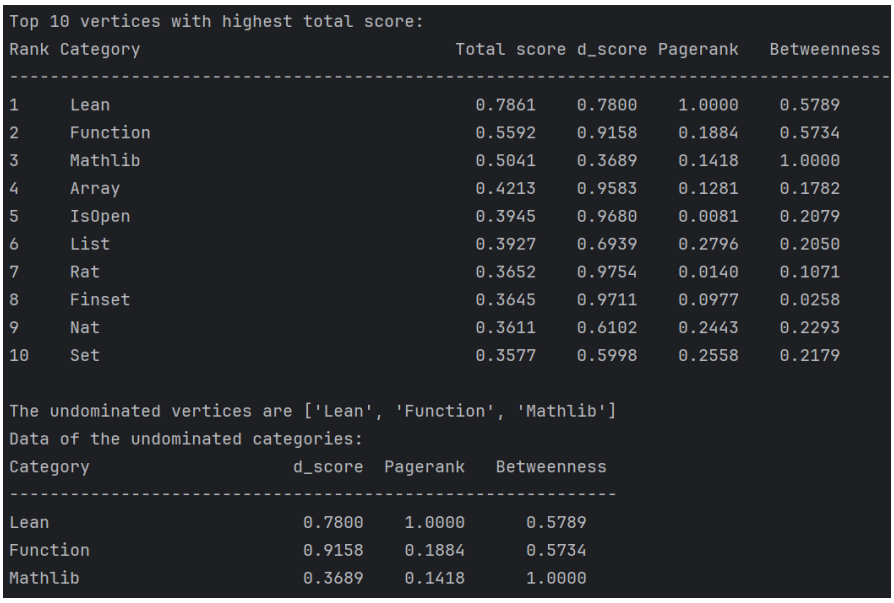
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Lean                                    0.6961    0.8191    0.8041    0.4659
2    Eq                                      0.6326    0.0026    1.0000    0.8944
3    Function                                0.5486    0.9730    0.1304    0.5424
4    Mathlib                                 0.4810    0.3500    0.0916    1.0000
5    List                                    0.3976    0.7463    0.1881    0.2587
6    IsOpen                                  0.3920    0.9891    0.0067    0.1811
7    Array                                   0.3875    0.9293    0.0870    0.1469
8    Nat                                     0.3612    0.6924    0.1805    0.2112
9    Set                                     0.3576    0.6518    0.2045    0.2168
10   Finset                                  0.3522    0.9489    0.0694    0.0392

The undominated vertices are ['Lean', 'Function']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-----------------------------------------------------------
Lean                       0.8191    0.8041      0.4659
Function                   0.9730    0.1304      0.5424
```

FIGURE 15: Result Category Graph when large categories are included. Category size below 5 removed



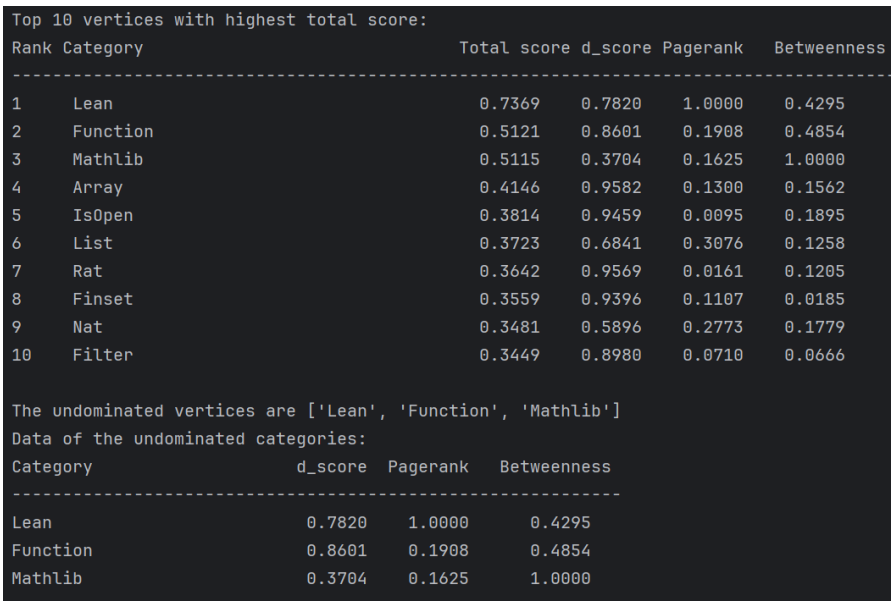
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Lean                                    0.8083    0.7983    1.0000    0.6270
2    Function                                0.5664    0.9623    0.1535    0.5832
3    Mathlib                                 0.4918    0.3602    0.1136    1.0000
4    Array                                   0.4096    0.9399    0.1152    0.1745
5    IsOpen                                  0.3907    0.9715    0.0070    0.1943
6    List                                    0.3874    0.7219    0.2428    0.1980
7    Rat                                     0.3691    0.9955    0.0114    0.1012
8    Finset                                  0.3661    0.9853    0.0821    0.0318
9    Nat                                     0.3655    0.6389    0.2345    0.2235
10   Set                                     0.3513    0.6280    0.2114    0.2148

The undominated vertices are ['Lean', 'Function']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-----------------------------------------------------------
Lean                       0.7983    1.0000      0.6270
Function                   0.9623    0.1535      0.5832
```

FIGURE 16: Result Category Graph when large categories are included. Category size below 10 removed



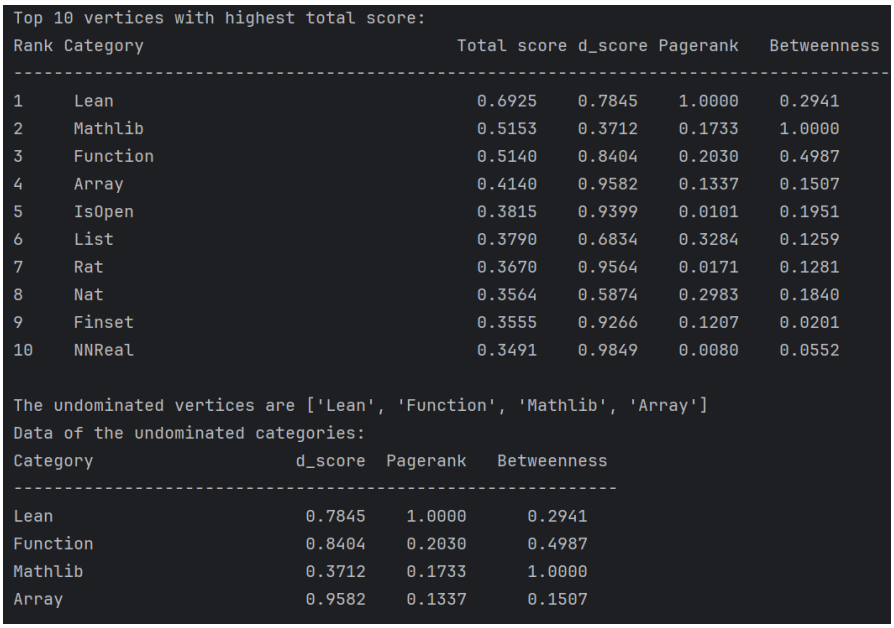
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Lean                                   0.7861    0.7800    1.0000    0.5789
2     Function                               0.5592    0.9158    0.1884    0.5734
3     Mathlib                                0.5041    0.3689    0.1418    1.0000
4     Array                                  0.4213    0.9583    0.1281    0.1782
5     IsOpen                                 0.3945    0.9680    0.0081    0.2079
6     List                                   0.3927    0.6939    0.2796    0.2050
7     Rat                                    0.3652    0.9754    0.0140    0.1071
8     Finset                                 0.3645    0.9711    0.0977    0.0258
9     Nat                                    0.3611    0.6102    0.2443    0.2293
10    Set                                    0.3577    0.5998    0.2558    0.2179

The undominated vertices are ['Lean', 'Function', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
Lean                       0.7800    1.0000      0.5789
Function                   0.9158    0.1884      0.5734
Mathlib                    0.3689    0.1418      1.0000
```

FIGURE 17: Result Category Graph when large categories are included. Category size below 15 removed



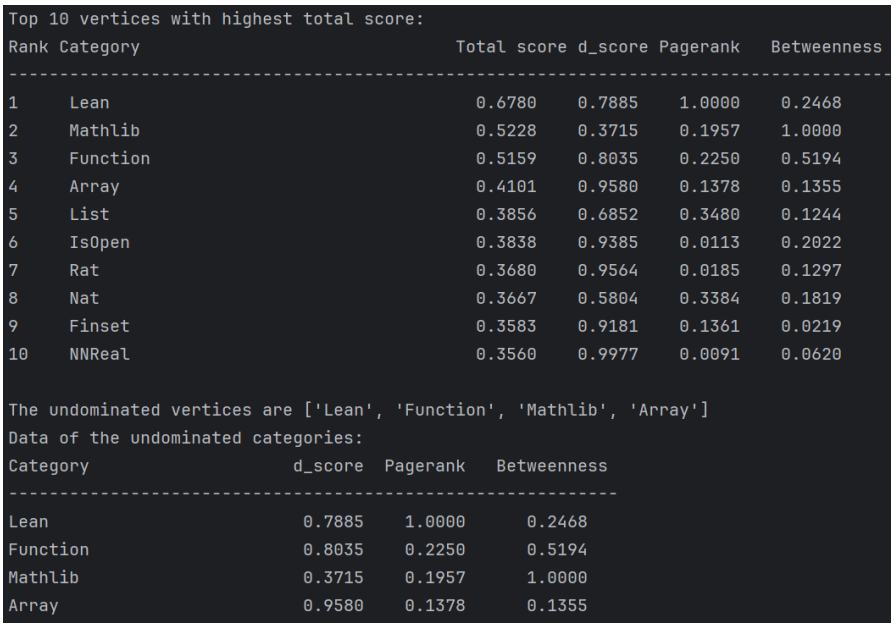
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Lean                                   0.7369    0.7820    1.0000    0.4295
2     Function                               0.5121    0.8601    0.1908    0.4854
3     Mathlib                                0.5115    0.3704    0.1625    1.0000
4     Array                                  0.4146    0.9582    0.1300    0.1562
5     IsOpen                                 0.3814    0.9459    0.0095    0.1895
6     List                                   0.3723    0.6841    0.3076    0.1258
7     Rat                                    0.3642    0.9569    0.0161    0.1205
8     Finset                                 0.3559    0.9396    0.1107    0.0185
9     Nat                                    0.3481    0.5896    0.2773    0.1779
10    Filter                                 0.3449    0.8980    0.0710    0.0666

The undominated vertices are ['Lean', 'Function', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
Lean                       0.7820    1.0000      0.4295
Function                   0.8601    0.1908      0.4854
Mathlib                    0.3704    0.1625      1.0000
```

FIGURE 18: Result Category Graph when large categories are included. Category size below 20 removed



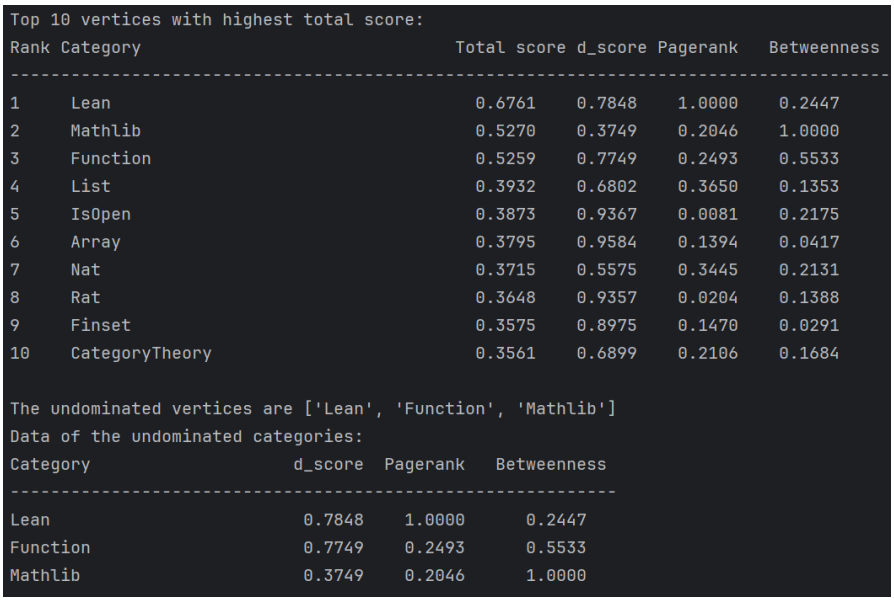
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Lean                                   0.6925    0.7845    1.0000    0.2941
2     Mathlib                                0.5153    0.3712    0.1733    1.0000
3     Function                               0.5140    0.8404    0.2030    0.4987
4     Array                                  0.4140    0.9582    0.1337    0.1507
5     IsOpen                                 0.3815    0.9399    0.0101    0.1951
6     List                                   0.3790    0.6834    0.3284    0.1259
7     Rat                                    0.3670    0.9564    0.0171    0.1281
8     Nat                                    0.3564    0.5874    0.2983    0.1840
9     Finset                                 0.3555    0.9266    0.1207    0.0201
10    NNReal                                 0.3491    0.9849    0.0080    0.0552

The undominated vertices are ['Lean', 'Function', 'Mathlib', 'Array']
Data of the undominated categories:
Category                   d_score  Pagerank   Betweenness
-----------------------------------------------------------
Lean                        0.7845    1.0000      0.2941
Function                    0.8404    0.2030      0.4987
Mathlib                     0.3712    0.1733      1.0000
Array                       0.9582    0.1337      0.1507
```

FIGURE 19: Result Category Graph when large categories are included. Category size below 25 removed



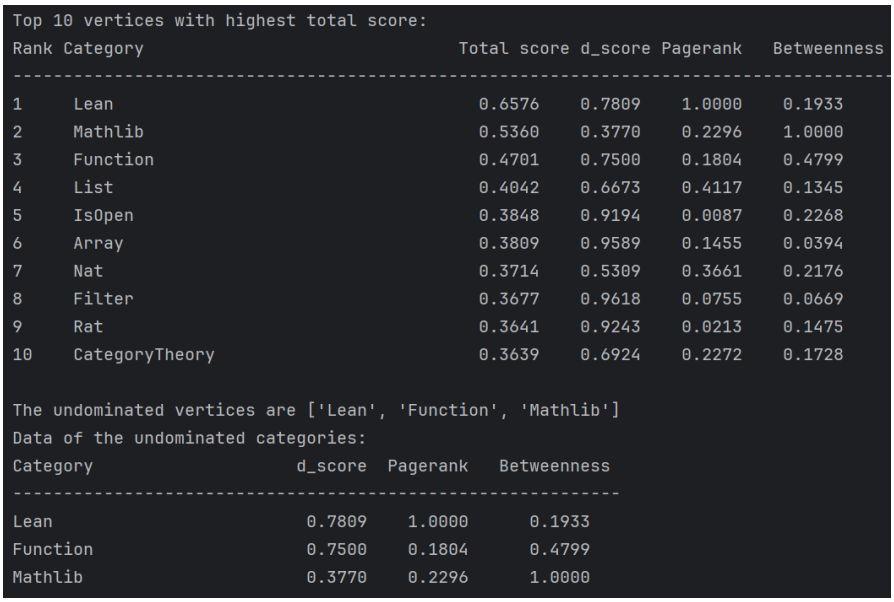
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Lean                                   0.6780    0.7885    1.0000    0.2468
2     Mathlib                                0.5228    0.3715    0.1957    1.0000
3     Function                               0.5159    0.8035    0.2250    0.5194
4     Array                                  0.4101    0.9580    0.1378    0.1355
5     List                                   0.3856    0.6852    0.3480    0.1244
6     IsOpen                                 0.3838    0.9385    0.0113    0.2022
7     Rat                                    0.3680    0.9564    0.0185    0.1297
8     Nat                                    0.3667    0.5804    0.3384    0.1819
9     Finset                                 0.3583    0.9181    0.1361    0.0219
10    NNReal                                 0.3560    0.9977    0.0091    0.0620

The undominated vertices are ['Lean', 'Function', 'Mathlib', 'Array']
Data of the undominated categories:
Category                   d_score  Pagerank   Betweenness
-----------------------------------------------------------
Lean                        0.7885    1.0000      0.2468
Function                    0.8035    0.2250      0.5194
Mathlib                     0.3715    0.1957      1.0000
Array                       0.9580    0.1378      0.1355
```

FIGURE 20: Result Category Graph when large categories are included. Category size below 30 removed



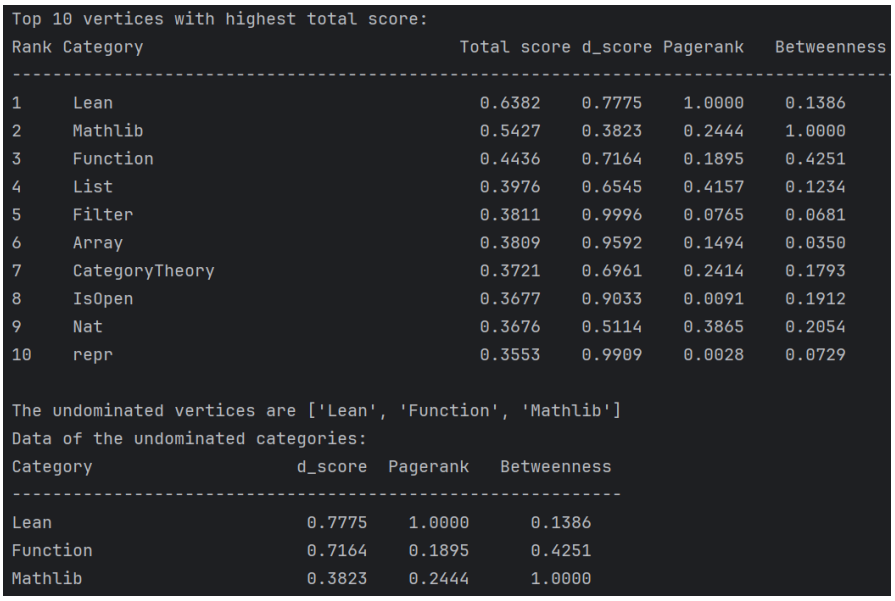
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Lean                                    0.6761    0.7848    1.0000    0.2447
2    Mathlib                                 0.5270    0.3749    0.2046    1.0000
3    Function                                0.5259    0.7749    0.2493    0.5533
4    List                                    0.3932    0.6802    0.3650    0.1353
5    IsOpen                                  0.3873    0.9367    0.0081    0.2175
6    Array                                   0.3795    0.9584    0.1394    0.0417
7    Nat                                     0.3715    0.5575    0.3445    0.2131
8    Rat                                     0.3648    0.9357    0.0204    0.1388
9    Finset                                  0.3575    0.8975    0.1470    0.0291
10   CategoryTheory                          0.3561    0.6899    0.2106    0.1684

The undominated vertices are ['Lean', 'Function', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
Lean                       0.7848    1.0000      0.2447
Function                   0.7749    0.2493      0.5533
Mathlib                    0.3749    0.2046      1.0000
```

FIGURE 21: Result Category Graph when large categories are included. Category size below 35 removed



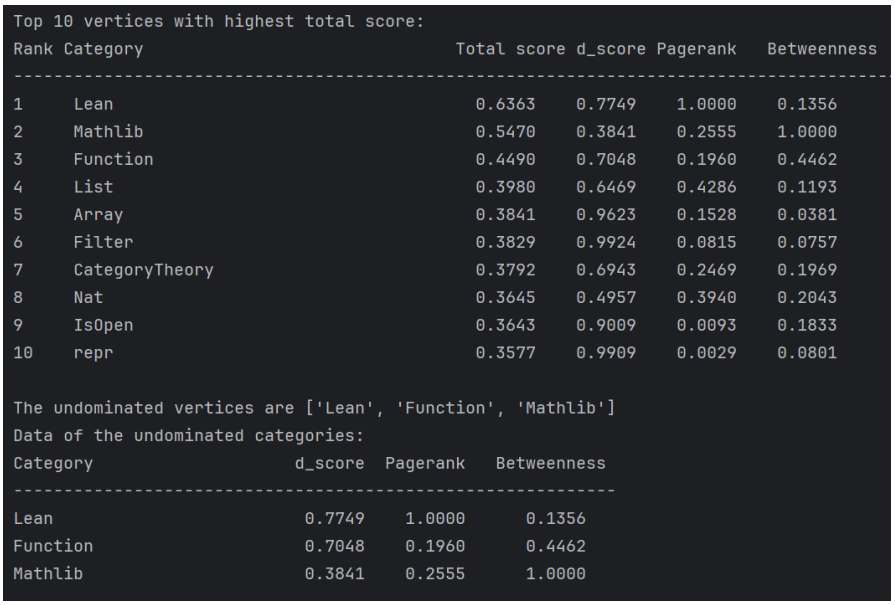
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Lean                                    0.6576    0.7809    1.0000    0.1933
2    Mathlib                                 0.5360    0.3770    0.2296    1.0000
3    Function                                0.4701    0.7500    0.1804    0.4799
4    List                                    0.4042    0.6673    0.4117    0.1345
5    IsOpen                                  0.3848    0.9194    0.0087    0.2268
6    Array                                   0.3809    0.9589    0.1455    0.0394
7    Nat                                     0.3714    0.5309    0.3661    0.2176
8    Filter                                  0.3677    0.9618    0.0755    0.0669
9    Rat                                     0.3641    0.9243    0.0213    0.1475
10   CategoryTheory                          0.3639    0.6924    0.2272    0.1728

The undominated vertices are ['Lean', 'Function', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
Lean                       0.7809    1.0000      0.1933
Function                   0.7500    0.1804      0.4799
Mathlib                    0.3770    0.2296      1.0000
```

FIGURE 22: Result Category Graph when large categories are included. Category size below 40 removed



```
Top 10 vertices with highest total score:
Rank Category                          Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Lean                                0.6382    0.7775    1.0000    0.1386
2    Mathlib                             0.5427    0.3823    0.2444    1.0000
3    Function                            0.4436    0.7164    0.1895    0.4251
4    List                                0.3976    0.6545    0.4157    0.1234
5    Filter                              0.3811    0.9996    0.0765    0.0681
6    Array                               0.3809    0.9592    0.1494    0.0350
7    CategoryTheory                      0.3721    0.6961    0.2414    0.1793
8    IsOpen                              0.3677    0.9033    0.0091    0.1912
9    Nat                                 0.3676    0.5114    0.3865    0.2054
10   repr                                0.3553    0.9909    0.0028    0.0729

The undominated vertices are ['Lean', 'Function', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
Lean                       0.7775    1.0000      0.1386
Function                   0.7164    0.1895      0.4251
Mathlib                    0.3823    0.2444      1.0000
```

FIGURE 23: Result Category Graph when large categories are included. Category size below 45 removed



```
Top 10 vertices with highest total score:
Rank Category                          Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Lean                                0.6363    0.7749    1.0000    0.1356
2    Mathlib                             0.5470    0.3841    0.2555    1.0000
3    Function                            0.4490    0.7048    0.1960    0.4462
4    List                                0.3980    0.6469    0.4286    0.1193
5    Array                               0.3841    0.9623    0.1528    0.0381
6    Filter                              0.3829    0.9924    0.0815    0.0757
7    CategoryTheory                      0.3792    0.6943    0.2469    0.1969
8    Nat                                 0.3645    0.4957    0.3940    0.2043
9    IsOpen                              0.3643    0.9009    0.0093    0.1833
10   repr                                0.3577    0.9909    0.0029    0.0801

The undominated vertices are ['Lean', 'Function', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
Lean                       0.7749    1.0000      0.1356
Function                   0.7048    0.1960      0.4462
Mathlib                    0.3841    0.2555      1.0000
```

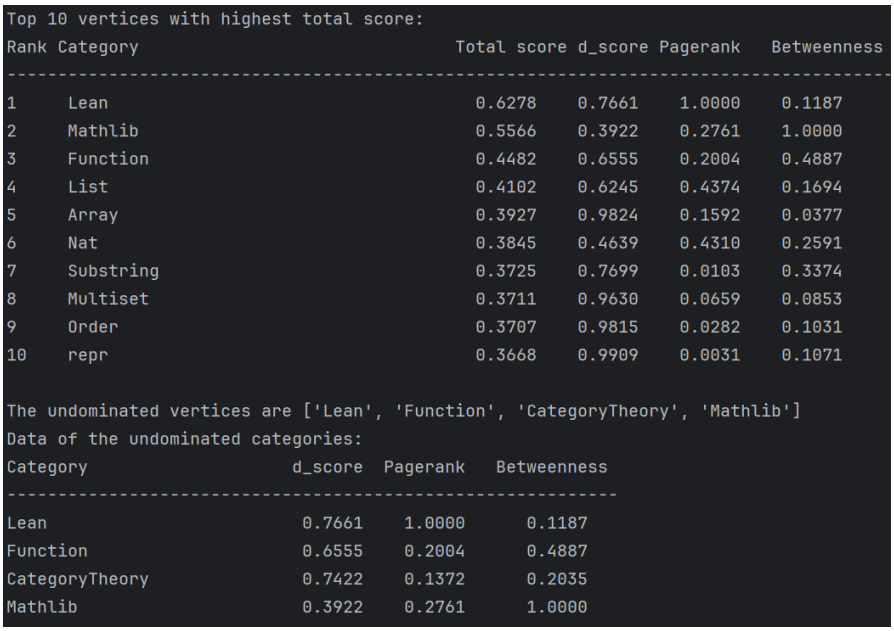
FIGURE 24: Result Category Graph when large categories are included. Category size below 50 removed

```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Lean                                    0.6303    0.7728    1.0000    0.1197
2    Mathlib                                 0.5499    0.3846    0.2638    1.0000
3    Function                                0.4606    0.6989    0.1981    0.4848
4    List                                    0.4061    0.6454    0.4274    0.1462
5    CategoryTheory                          0.3931    0.7171    0.2523    0.2103
6    Array                                   0.3857    0.9685    0.1549    0.0347
7    Filter                                  0.3851    0.9872    0.0806    0.0884
8    Nat                                     0.3738    0.4950    0.4037    0.2230
9    Substring                               0.3621    0.7750    0.0104    0.3012
10   repr                                    0.3594    0.9909    0.0030    0.0851

The undominated vertices are ['Lean', 'Function', 'CategoryTheory', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
Lean                       0.7728    1.0000      0.1197
Function                   0.6989    0.1981      0.4848
CategoryTheory             0.7171    0.2523      0.2103
Mathlib                    0.3846    0.2638      1.0000
```

FIGURE 25: Result Category Graph when large categories are included. Category size below 55 removed

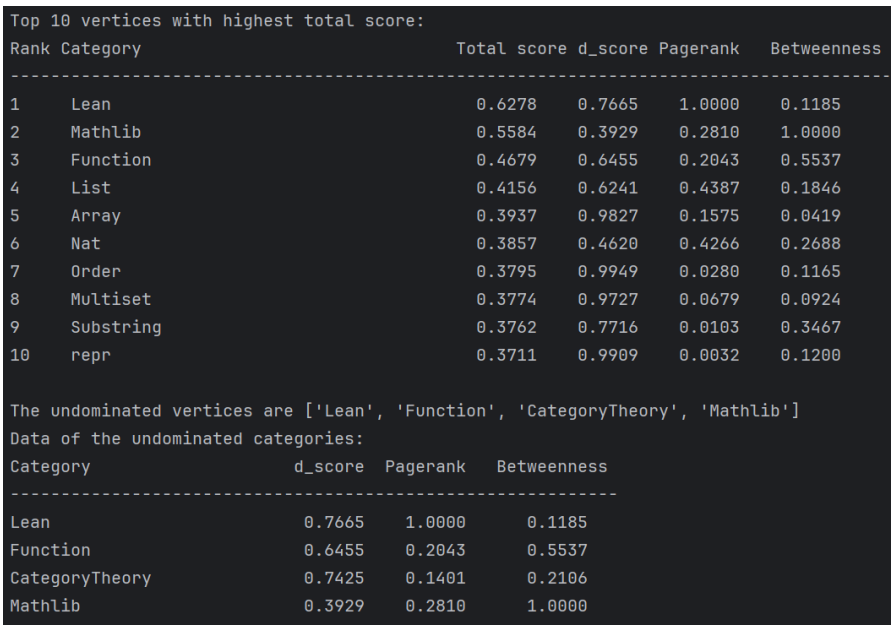
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Lean                                    0.6302    0.7706    1.0000    0.1215
2    Mathlib                                 0.5609    0.3890    0.2925    1.0000
3    Function                                0.4514    0.6800    0.1933    0.4806
4    List                                    0.4104    0.6412    0.4240    0.1666
5    Array                                   0.3916    0.9816    0.1577    0.0365
6    Nat                                     0.3855    0.4871    0.4165    0.2532
7    Filter                                  0.3837    0.9531    0.0974    0.1015
8    CategoryTheory                          0.3821    0.7270    0.1907    0.2289
9    Substring                               0.3686    0.7781    0.0103    0.3174
10   repr                                    0.3645    0.9909    0.0031    0.1002

The undominated vertices are ['Lean', 'Function', 'CategoryTheory', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
Lean                       0.7706    1.0000      0.1215
Function                   0.6800    0.1933      0.4806
CategoryTheory             0.7270    0.1907      0.2289
Mathlib                    0.3890    0.2925      1.0000
```

FIGURE 26: Result Category Graph when large categories are included. Category size below 60 removed



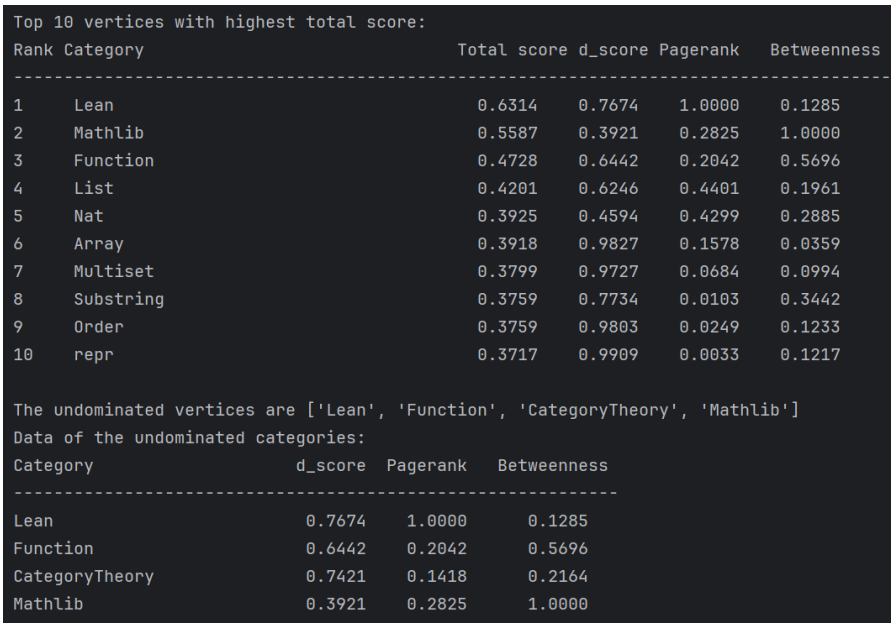
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Lean                                    0.6278    0.7661    1.0000    0.1187
2    Mathlib                                 0.5566    0.3922    0.2761    1.0000
3    Function                                0.4482    0.6555    0.2004    0.4887
4    List                                    0.4102    0.6245    0.4374    0.1694
5    Array                                   0.3927    0.9824    0.1592    0.0377
6    Nat                                     0.3845    0.4639    0.4310    0.2591
7    Substring                               0.3725    0.7699    0.0103    0.3374
8    Multiset                                0.3711    0.9630    0.0659    0.0853
9    Order                                   0.3707    0.9815    0.0282    0.1031
10   repr                                    0.3668    0.9909    0.0031    0.1071

The undominated vertices are ['Lean', 'Function', 'CategoryTheory', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-----------------------------------------------------------
Lean                       0.7661    1.0000      0.1187
Function                   0.6555    0.2004      0.4887
CategoryTheory             0.7422    0.1372      0.2035
Mathlib                    0.3922    0.2761      1.0000
```

FIGURE 27: Result Category Graph when large categories are included. Category size below 65 removed



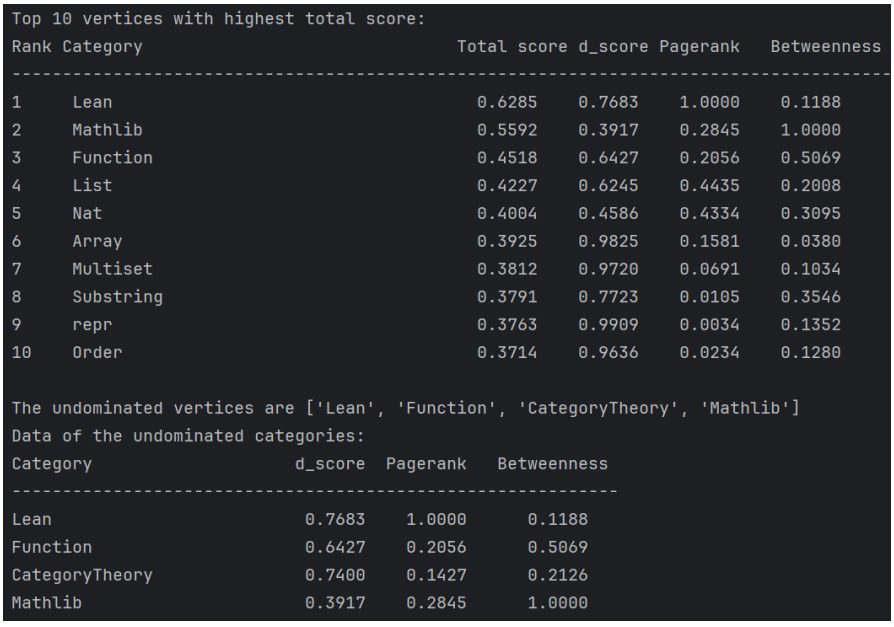
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Lean                                    0.6278    0.7665    1.0000    0.1185
2    Mathlib                                 0.5584    0.3929    0.2810    1.0000
3    Function                                0.4679    0.6455    0.2043    0.5537
4    List                                    0.4156    0.6241    0.4387    0.1846
5    Array                                   0.3937    0.9827    0.1575    0.0419
6    Nat                                     0.3857    0.4620    0.4266    0.2688
7    Order                                   0.3795    0.9949    0.0280    0.1165
8    Multiset                                0.3774    0.9727    0.0679    0.0924
9    Substring                               0.3762    0.7716    0.0103    0.3467
10   repr                                    0.3711    0.9909    0.0032    0.1200

The undominated vertices are ['Lean', 'Function', 'CategoryTheory', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-----------------------------------------------------------
Lean                       0.7665    1.0000      0.1185
Function                   0.6455    0.2043      0.5537
CategoryTheory             0.7425    0.1401      0.2106
Mathlib                    0.3929    0.2810      1.0000
```

FIGURE 28: Result Category Graph when large categories are included. Category size below 70 removed



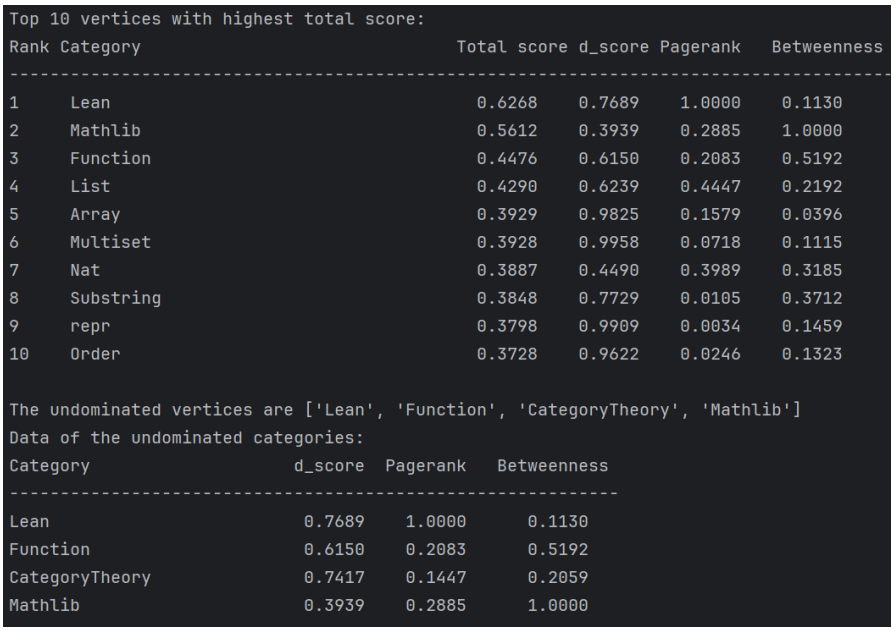
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Lean                                   0.6314    0.7674    1.0000    0.1285
2     Mathlib                                0.5587    0.3921    0.2825    1.0000
3     Function                               0.4728    0.6442    0.2042    0.5696
4     List                                   0.4201    0.6246    0.4401    0.1961
5     Nat                                    0.3925    0.4594    0.4299    0.2885
6     Array                                  0.3918    0.9827    0.1578    0.0359
7     Multiset                               0.3799    0.9727    0.0684    0.0994
8     Substring                              0.3759    0.7734    0.0103    0.3442
9     Order                                  0.3759    0.9803    0.0249    0.1233
10    repr                                   0.3717    0.9909    0.0033    0.1217

The undominated vertices are ['Lean', 'Function', 'CategoryTheory', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
Lean                       0.7674    1.0000      0.1285
Function                   0.6442    0.2042      0.5696
CategoryTheory             0.7421    0.1418      0.2164
Mathlib                    0.3921    0.2825      1.0000
```

FIGURE 29: Result Category Graph when large categories are included. Category size below 70 removed



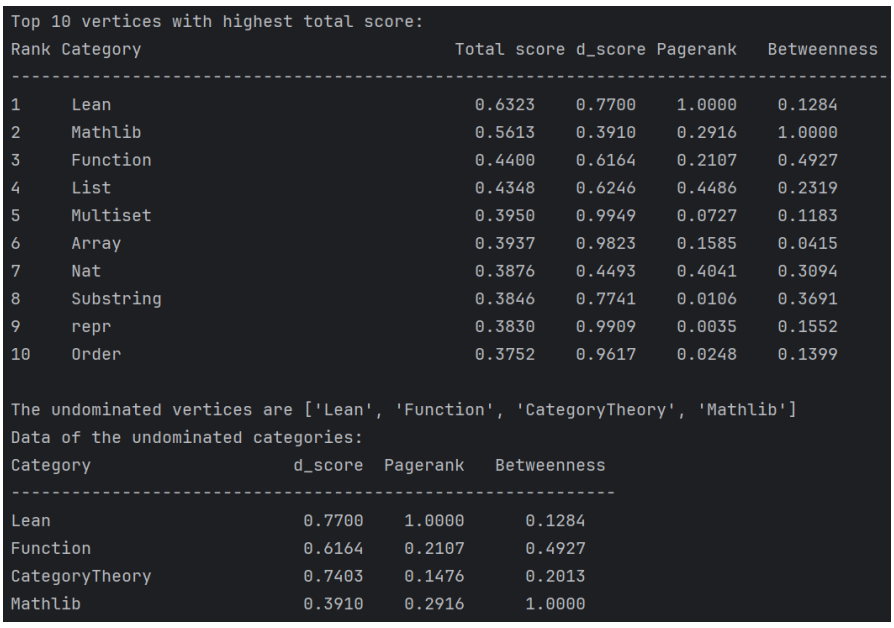
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Lean                                   0.6285    0.7683    1.0000    0.1188
2     Mathlib                                0.5592    0.3917    0.2845    1.0000
3     Function                               0.4518    0.6427    0.2056    0.5069
4     List                                   0.4227    0.6245    0.4435    0.2008
5     Nat                                    0.4004    0.4586    0.4334    0.3095
6     Array                                  0.3925    0.9825    0.1581    0.0380
7     Multiset                               0.3812    0.9720    0.0691    0.1034
8     Substring                              0.3791    0.7723    0.0105    0.3546
9     repr                                   0.3763    0.9909    0.0034    0.1352
10    Order                                  0.3714    0.9636    0.0234    0.1280

The undominated vertices are ['Lean', 'Function', 'CategoryTheory', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
Lean                       0.7683    1.0000      0.1188
Function                   0.6427    0.2056      0.5069
CategoryTheory             0.7400    0.1427      0.2126
Mathlib                    0.3917    0.2845      1.0000
```

FIGURE 30: Result Category Graph when large categories are included. Category size below 80 removed



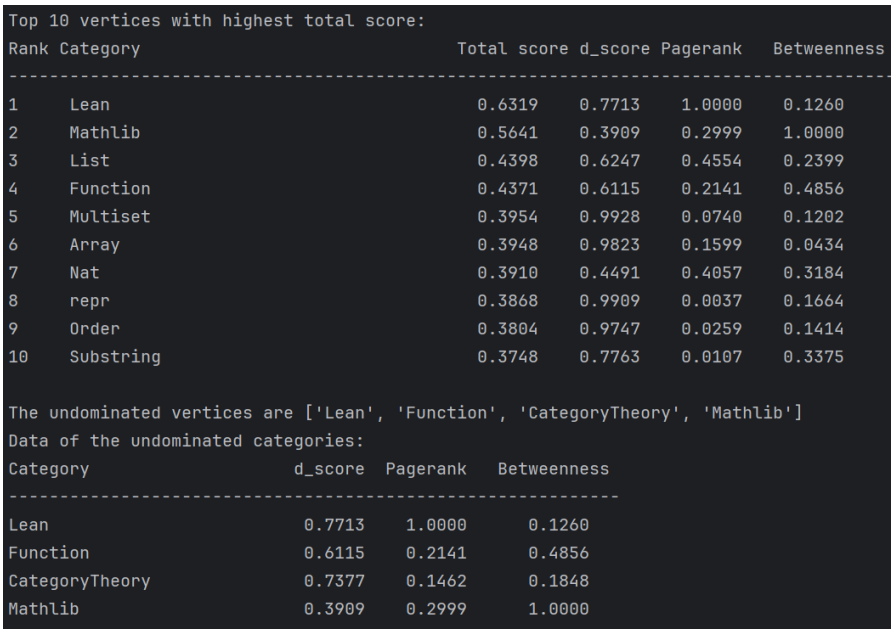
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Lean                                   0.6268    0.7689    1.0000    0.1130
2     Mathlib                                0.5612    0.3939    0.2885    1.0000
3     Function                               0.4476    0.6150    0.2083    0.5192
4     List                                   0.4290    0.6239    0.4447    0.2192
5     Array                                  0.3929    0.9825    0.1579    0.0396
6     Multiset                               0.3928    0.9958    0.0718    0.1115
7     Nat                                    0.3887    0.4490    0.3989    0.3185
8     Substring                              0.3848    0.7729    0.0105    0.3712
9     repr                                   0.3798    0.9909    0.0034    0.1459
10    Order                                  0.3728    0.9622    0.0246    0.1323

The undominated vertices are ['Lean', 'Function', 'CategoryTheory', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-----------------------------------------------------------
Lean                       0.7689    1.0000      0.1130
Function                   0.6150    0.2083      0.5192
CategoryTheory             0.7417    0.1447      0.2059
Mathlib                    0.3939    0.2885      1.0000
```

FIGURE 31: Result Category Graph when large categories are included. Category size below 85 removed



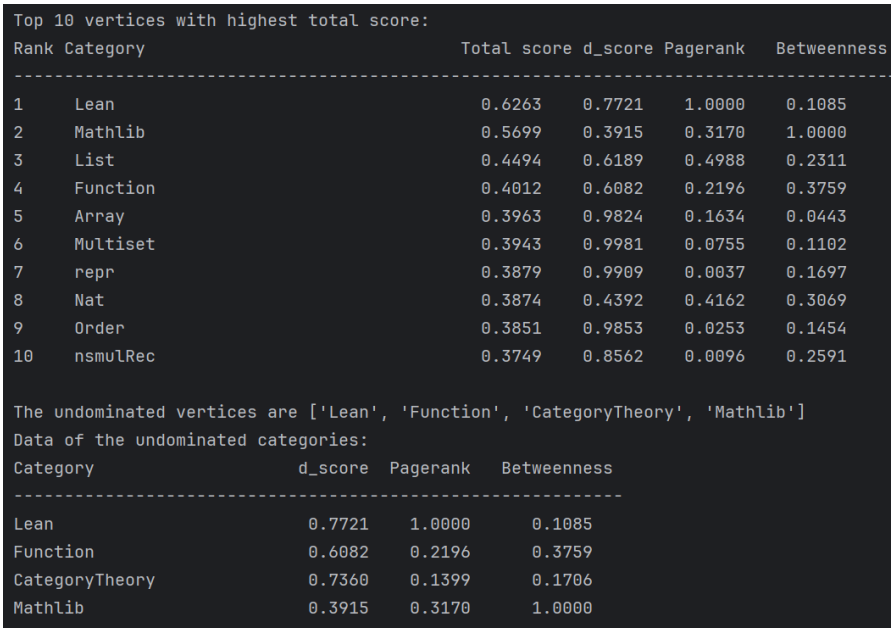
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Lean                                   0.6323    0.7700    1.0000    0.1284
2     Mathlib                                0.5613    0.3910    0.2916    1.0000
3     Function                               0.4400    0.6164    0.2107    0.4927
4     List                                   0.4348    0.6246    0.4486    0.2319
5     Multiset                               0.3950    0.9949    0.0727    0.1183
6     Array                                  0.3937    0.9823    0.1585    0.0415
7     Nat                                    0.3876    0.4493    0.4041    0.3094
8     Substring                              0.3846    0.7741    0.0106    0.3691
9     repr                                   0.3830    0.9909    0.0035    0.1552
10    Order                                  0.3752    0.9617    0.0248    0.1399

The undominated vertices are ['Lean', 'Function', 'CategoryTheory', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-----------------------------------------------------------
Lean                       0.7700    1.0000      0.1284
Function                   0.6164    0.2107      0.4927
CategoryTheory             0.7403    0.1476      0.2013
Mathlib                    0.3910    0.2916      1.0000
```

FIGURE 32: Result Category Graph when large categories are included. Category size below 90 removed



```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Lean                                   0.6319    0.7713    1.0000    0.1260
2     Mathlib                                0.5641    0.3909    0.2999    1.0000
3     List                                   0.4398    0.6247    0.4554    0.2399
4     Function                               0.4371    0.6115    0.2141    0.4856
5     Multiset                               0.3954    0.9928    0.0740    0.1202
6     Array                                  0.3948    0.9823    0.1599    0.0434
7     Nat                                    0.3910    0.4491    0.4057    0.3184
8     repr                                   0.3868    0.9909    0.0037    0.1664
9     Order                                  0.3804    0.9747    0.0259    0.1414
10    Substring                              0.3748    0.7763    0.0107    0.3375

The undominated vertices are ['Lean', 'Function', 'CategoryTheory', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
Lean                       0.7713    1.0000      0.1260
Function                   0.6115    0.2141      0.4856
CategoryTheory             0.7377    0.1462      0.1848
Mathlib                    0.3909    0.2999      1.0000
```

FIGURE 33: Result Category Graph when large categories are included. Category size below 95 removed



```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Lean                                   0.6263    0.7721    1.0000    0.1085
2     Mathlib                                0.5699    0.3915    0.3170    1.0000
3     List                                   0.4494    0.6189    0.4988    0.2311
4     Function                               0.4012    0.6082    0.2196    0.3759
5     Array                                  0.3963    0.9824    0.1634    0.0443
6     Multiset                               0.3943    0.9981    0.0755    0.1102
7     repr                                   0.3879    0.9909    0.0037    0.1697
8     Nat                                    0.3874    0.4392    0.4162    0.3069
9     Order                                  0.3851    0.9853    0.0253    0.1454
10    nsmulRec                               0.3749    0.8562    0.0096    0.2591

The undominated vertices are ['Lean', 'Function', 'CategoryTheory', 'Mathlib']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
Lean                       0.7721    1.0000      0.1085
Function                   0.6082    0.2196      0.3759
CategoryTheory             0.7360    0.1399      0.1706
Mathlib                    0.3915    0.3170      1.0000
```

FIGURE 34: Result Category Graph when large categories are included. Category size below 100 removed

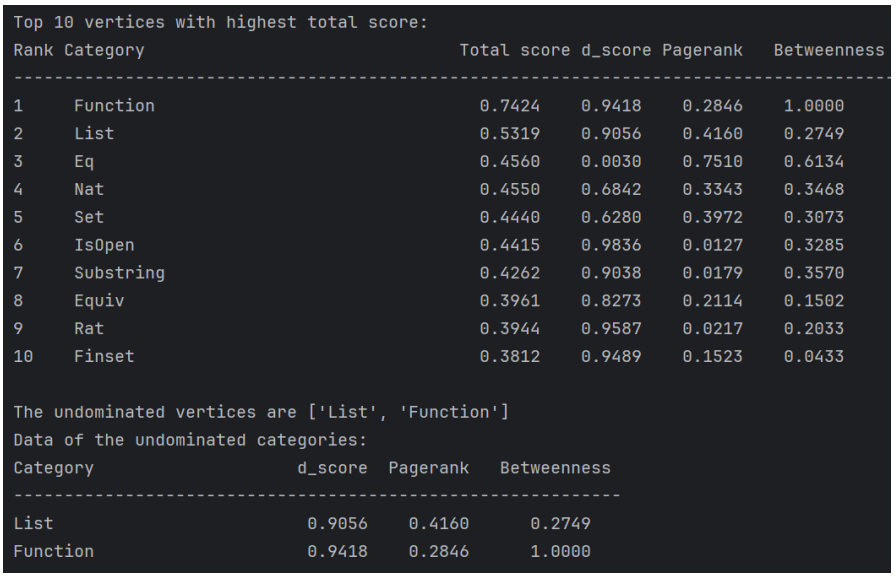
10.2 Results Category Graph excluding large categories

```
Top 10 vertices with highest total score:
Rank Category                                Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Eq                                        0.6681    0.0033    1.0000    1.0000
2    Function                                  0.6074    0.9442    0.1365    0.7411
3    List                                      0.4918    0.9694    0.1977    0.3089
4    IsOpen                                    0.4085    0.9963    0.0070    0.2227
5    Nat                                       0.4072    0.7842    0.1699    0.2680
6    Set                                       0.3915    0.6845    0.2134    0.2768
7    Pi                                        0.3823    0.8615    0.0273    0.2585
8    Equiv                                     0.3709    0.8903    0.1125    0.1106
9    IsUnit                                    0.3646    0.9974    0.0030    0.0942
10   Algebra                                   0.3576    0.9859    0.0192    0.0687

The undominated vertices are ['List', 'Function']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-----------------------------------------------------------
List                       0.9694    0.1977      0.3089
Function                   0.9442    0.1365      0.7411
```

FIGURE 35: Result Category Graph when large categories are excluded. Category size below 5 removed

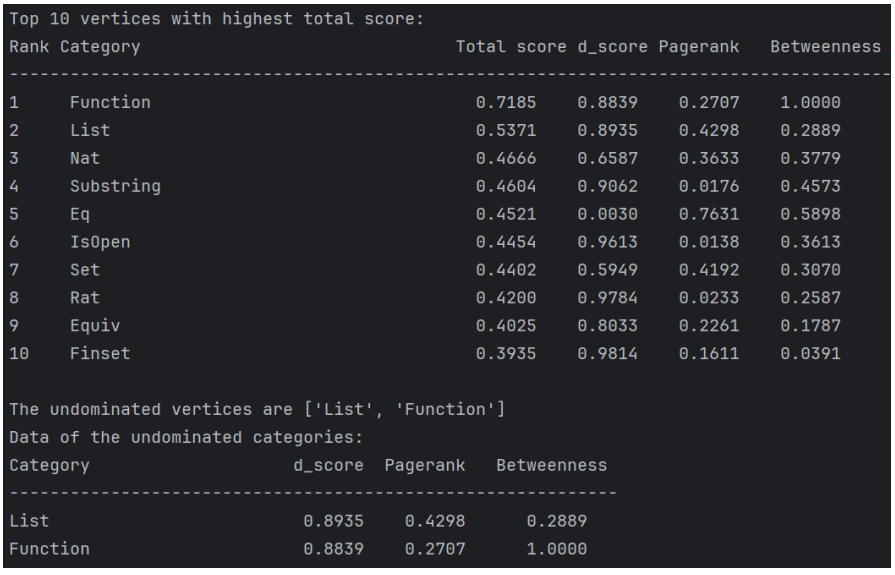
```
Top 10 vertices with highest total score:
Rank Category                                Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Function                                  0.7460    0.9896    0.2476    1.0000
2    List                                      0.5303    0.9392    0.3993    0.2531
3    Eq                                        0.5286    0.0032    0.8688    0.7133
4    Nat                                       0.4658    0.7195    0.3466    0.3317
5    IsOpen                                    0.4342    0.9864    0.0116    0.3048
6    Set                                       0.4333    0.6588    0.3472    0.2943
7    Substring                                 0.4208    0.9215    0.0192    0.3221
8    Equiv                                     0.3892    0.8562    0.1772    0.1349
9    Pi                                        0.3862    0.8129    0.0350    0.3110
10   Rat                                       0.3798    0.9373    0.0187    0.1840

The undominated vertices are ['List', 'Function']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-----------------------------------------------------------
List                       0.9392    0.3993      0.2531
Function                   0.9896    0.2476      1.0000
```

FIGURE 36: Result Category Graph when large categories are excluded. Category size below 10 removed



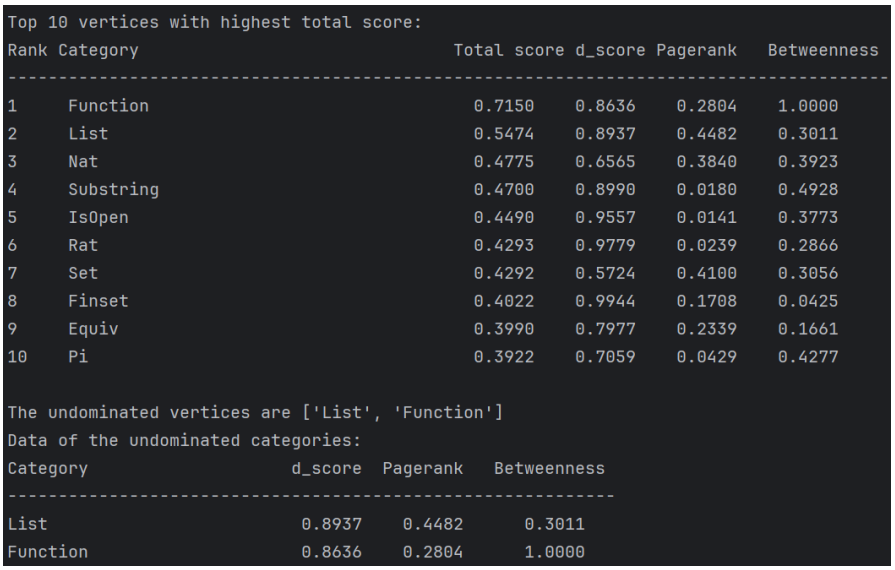
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Function                               0.7424    0.9418    0.2846    1.0000
2     List                                   0.5319    0.9056    0.4160    0.2749
3     Eq                                     0.4560    0.0030    0.7510    0.6134
4     Nat                                    0.4550    0.6842    0.3343    0.3468
5     Set                                    0.4440    0.6280    0.3972    0.3073
6     IsOpen                                 0.4415    0.9836    0.0127    0.3285
7     Substring                              0.4262    0.9038    0.0179    0.3570
8     Equiv                                  0.3961    0.8273    0.2114    0.1502
9     Rat                                    0.3944    0.9587    0.0217    0.2033
10    Finset                                 0.3812    0.9489    0.1523    0.0433

The undominated vertices are ['List', 'Function']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-----------------------------------------------------------
List                       0.9056    0.4160      0.2749
Function                   0.9418    0.2846      1.0000
```

FIGURE 37: Result Category Graph when large categories are excluded. Category size below 15 removed



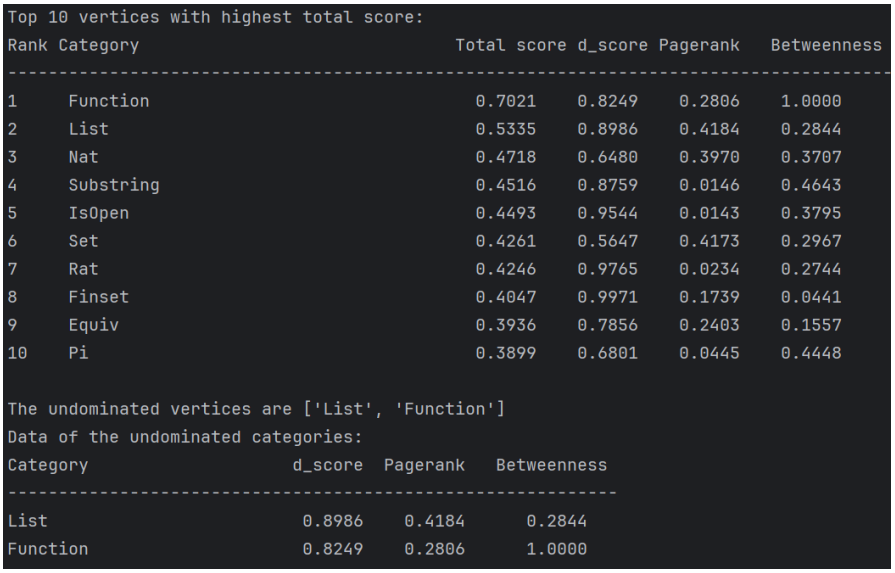
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Function                               0.7185    0.8839    0.2707    1.0000
2     List                                   0.5371    0.8935    0.4298    0.2889
3     Nat                                    0.4666    0.6587    0.3633    0.3779
4     Substring                              0.4604    0.9062    0.0176    0.4573
5     Eq                                     0.4521    0.0030    0.7631    0.5898
6     IsOpen                                 0.4454    0.9613    0.0138    0.3613
7     Set                                    0.4402    0.5949    0.4192    0.3070
8     Rat                                    0.4200    0.9784    0.0233    0.2587
9     Equiv                                  0.4025    0.8033    0.2261    0.1787
10    Finset                                 0.3935    0.9814    0.1611    0.0391

The undominated vertices are ['List', 'Function']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-----------------------------------------------------------
List                       0.8935    0.4298      0.2889
Function                   0.8839    0.2707      1.0000
```

FIGURE 38: Result Category Graph when large categories are excluded. Category size below 20 removed



```
Top 10 vertices with highest total score:
Rank Category                                Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Function                                  0.7150    0.8636    0.2804    1.0000
2    List                                      0.5474    0.8937    0.4482    0.3011
3    Nat                                       0.4775    0.6565    0.3840    0.3923
4    Substring                                 0.4700    0.8990    0.0180    0.4928
5    IsOpen                                    0.4490    0.9557    0.0141    0.3773
6    Rat                                       0.4293    0.9779    0.0239    0.2866
7    Set                                       0.4292    0.5724    0.4100    0.3056
8    Finset                                    0.4022    0.9944    0.1708    0.0425
9    Equiv                                     0.3990    0.7977    0.2339    0.1661
10   Pi                                        0.3922    0.7059    0.0429    0.4277

The undominated vertices are ['List', 'Function']
Data of the undominated categories:
Category                   d_score  Pagerank   Betweenness
-----------------------------------------------------------
List                        0.8937    0.4482      0.3011
Function                    0.8636    0.2804      1.0000
```

FIGURE 39: Result Category Graph when large categories are excluded. Category size below 25 removed



```
Top 10 vertices with highest total score:
Rank Category                                Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Function                                  0.7021    0.8249    0.2806    1.0000
2    List                                      0.5335    0.8986    0.4184    0.2844
3    Nat                                       0.4718    0.6480    0.3970    0.3707
4    Substring                                 0.4516    0.8759    0.0146    0.4643
5    IsOpen                                    0.4493    0.9544    0.0143    0.3795
6    Set                                       0.4261    0.5647    0.4173    0.2967
7    Rat                                       0.4246    0.9765    0.0234    0.2744
8    Finset                                    0.4047    0.9971    0.1739    0.0441
9    Equiv                                     0.3936    0.7856    0.2403    0.1557
10   Pi                                        0.3899    0.6801    0.0445    0.4448

The undominated vertices are ['List', 'Function']
Data of the undominated categories:
Category                   d_score  Pagerank   Betweenness
-----------------------------------------------------------
List                        0.8986    0.4184      0.2844
Function                    0.8249    0.2806      1.0000
```

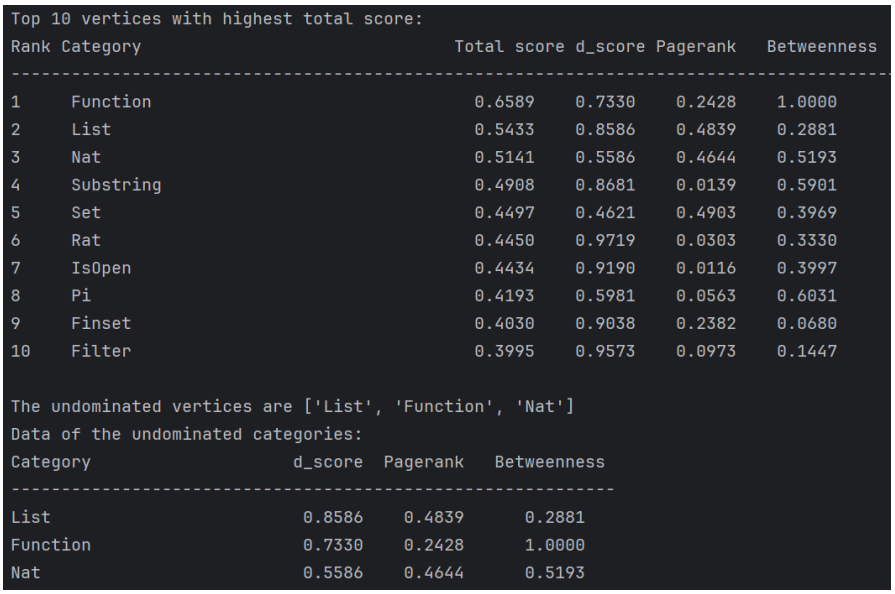
FIGURE 40: Result Category Graph when large categories are excluded. Category size below 30 removed

```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Function                                0.7029    0.7948    0.3131    1.0000
2    List                                    0.5355    0.8931    0.4346    0.2797
3    Nat                                     0.4795    0.6188    0.4080    0.4118
4    Substring                               0.4507    0.8669    0.0140    0.4710
5    IsOpen                                  0.4474    0.9529    0.0103    0.3794
6    Rat                                     0.4335    0.9980    0.0261    0.2769
7    Set                                     0.4235    0.5479    0.4163    0.3067
8    Finset                                  0.4055    0.9749    0.1901    0.0525
9    Pi                                      0.3882    0.6540    0.0485    0.4619
10   Equiv                                   0.3843    0.7678    0.2538    0.1319

The undominated vertices are ['List', 'Function']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
List                       0.8931    0.4346      0.2797
Function                   0.7948    0.3131      1.0000
```

FIGURE 41: Result Category Graph when large categories are excluded. Category size below 35 removed

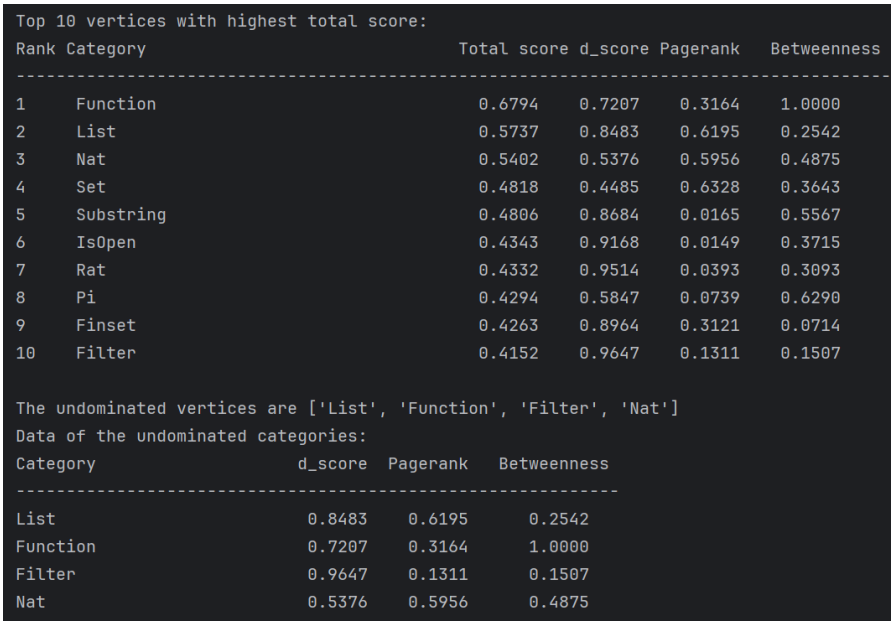
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    Function                                0.6701    0.7686    0.2408    1.0000
2    List                                    0.5665    0.8759    0.5080    0.3163
3    Nat                                     0.5101    0.5842    0.4570    0.4892
4    Substring                               0.4817    0.8661    0.0147    0.5642
5    IsOpen                                  0.4701    0.9353    0.0115    0.4635
6    Rat                                     0.4529    0.9926    0.0284    0.3381
7    Set                                     0.4481    0.5074    0.4736    0.3636
8    Pi                                      0.4110    0.6288    0.0554    0.5485
9    Finset                                  0.4099    0.9392    0.2296    0.0620
10   Equiv                                   0.3973    0.7471    0.2997    0.1459

The undominated vertices are ['List', 'Function', 'Nat']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
List                       0.8759    0.5080      0.3163
Function                   0.7686    0.2408      1.0000
Nat                        0.5842    0.4570      0.4892
```

FIGURE 42: Result Category Graph when large categories are excluded. Category size below 40 removed



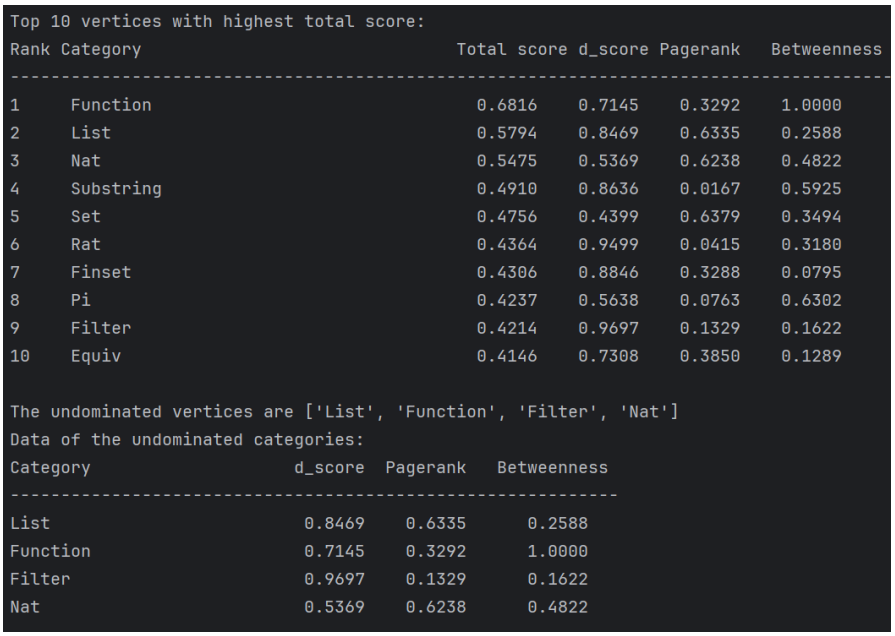
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Function                               0.6589    0.7330    0.2428    1.0000
2     List                                   0.5433    0.8586    0.4839    0.2881
3     Nat                                    0.5141    0.5586    0.4644    0.5193
4     Substring                              0.4908    0.8681    0.0139    0.5901
5     Set                                    0.4497    0.4621    0.4903    0.3969
6     Rat                                    0.4450    0.9719    0.0303    0.3330
7     IsOpen                                 0.4434    0.9190    0.0116    0.3997
8     Pi                                     0.4193    0.5981    0.0563    0.6031
9     Finset                                 0.4030    0.9038    0.2382    0.0680
10    Filter                                 0.3995    0.9573    0.0973    0.1447

The undominated vertices are ['List', 'Function', 'Nat']
Data of the undominated categories:
Category                   d_score  Pagerank   Betweenness
-----------------------------------------------------------
List                        0.8586    0.4839      0.2881
Function                    0.7330    0.2428      1.0000
Nat                         0.5586    0.4644      0.5193
```

FIGURE 43: Result Category Graph when large categories are excluded. Category size below 45 removed



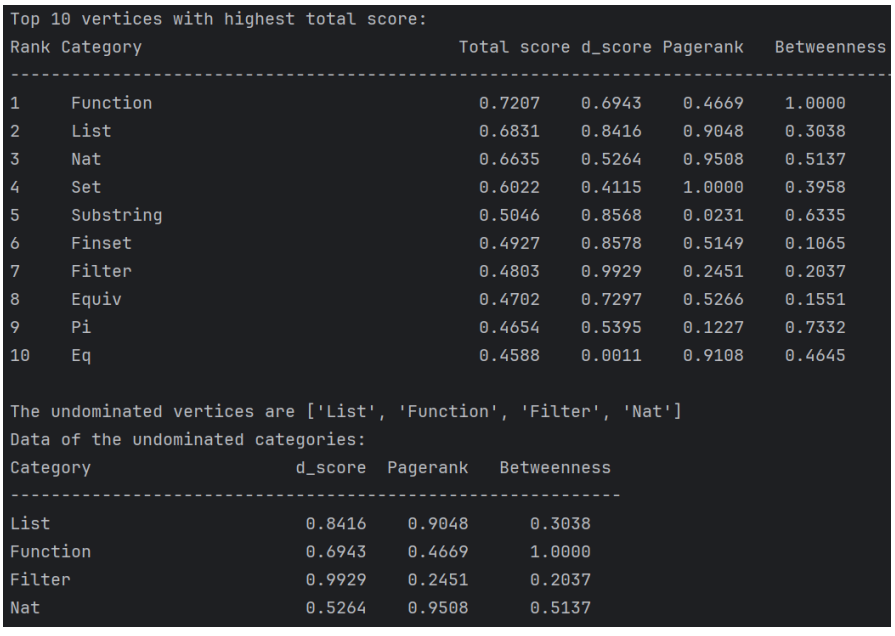
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Function                               0.6794    0.7207    0.3164    1.0000
2     List                                   0.5737    0.8483    0.6195    0.2542
3     Nat                                    0.5402    0.5376    0.5956    0.4875
4     Set                                    0.4818    0.4485    0.6328    0.3643
5     Substring                              0.4806    0.8684    0.0165    0.5567
6     IsOpen                                 0.4343    0.9168    0.0149    0.3715
7     Rat                                    0.4332    0.9514    0.0393    0.3093
8     Pi                                     0.4294    0.5847    0.0739    0.6290
9     Finset                                 0.4263    0.8964    0.3121    0.0714
10    Filter                                 0.4152    0.9647    0.1311    0.1507

The undominated vertices are ['List', 'Function', 'Filter', 'Nat']
Data of the undominated categories:
Category                   d_score  Pagerank   Betweenness
-----------------------------------------------------------
List                        0.8483    0.6195      0.2542
Function                    0.7207    0.3164      1.0000
Filter                      0.9647    0.1311      0.1507
Nat                         0.5376    0.5956      0.4875
```

FIGURE 44: Result Category Graph when large categories are excluded. Category size below 50 removed



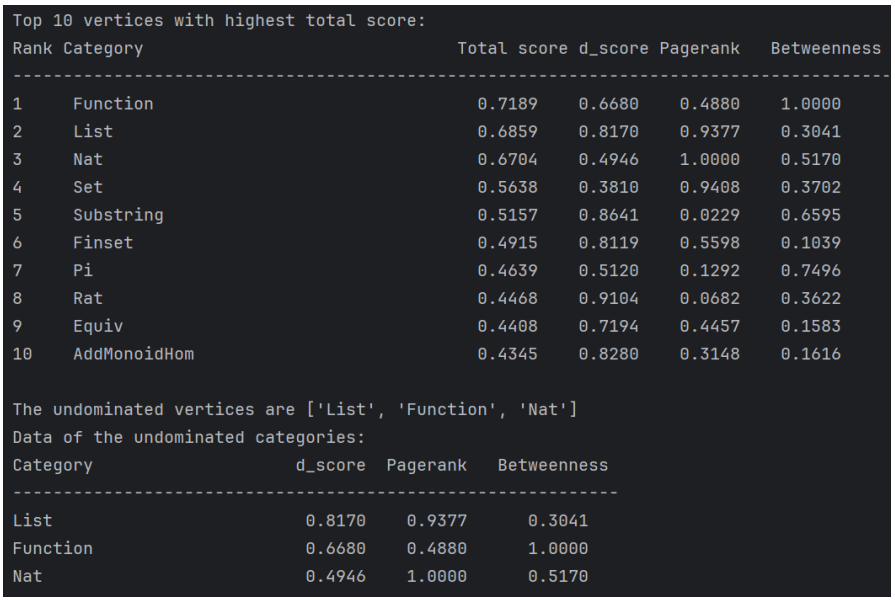
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Function                               0.6816    0.7145    0.3292    1.0000
2     List                                   0.5794    0.8469    0.6335    0.2588
3     Nat                                    0.5475    0.5369    0.6238    0.4822
4     Substring                              0.4910    0.8636    0.0167    0.5925
5     Set                                    0.4756    0.4399    0.6379    0.3494
6     Rat                                    0.4364    0.9499    0.0415    0.3180
7     Finset                                 0.4306    0.8846    0.3288    0.0795
8     Pi                                     0.4237    0.5638    0.0763    0.6302
9     Filter                                 0.4214    0.9697    0.1329    0.1622
10    Equiv                                  0.4146    0.7308    0.3850    0.1289

The undominated vertices are ['List', 'Function', 'Filter', 'Nat']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
List                       0.8469    0.6335      0.2588
Function                   0.7145    0.3292      1.0000
Filter                     0.9697    0.1329      0.1622
Nat                        0.5369    0.6238      0.4822
```

FIGURE 45: Result Category Graph when large categories are excluded. Category size below 55 removed



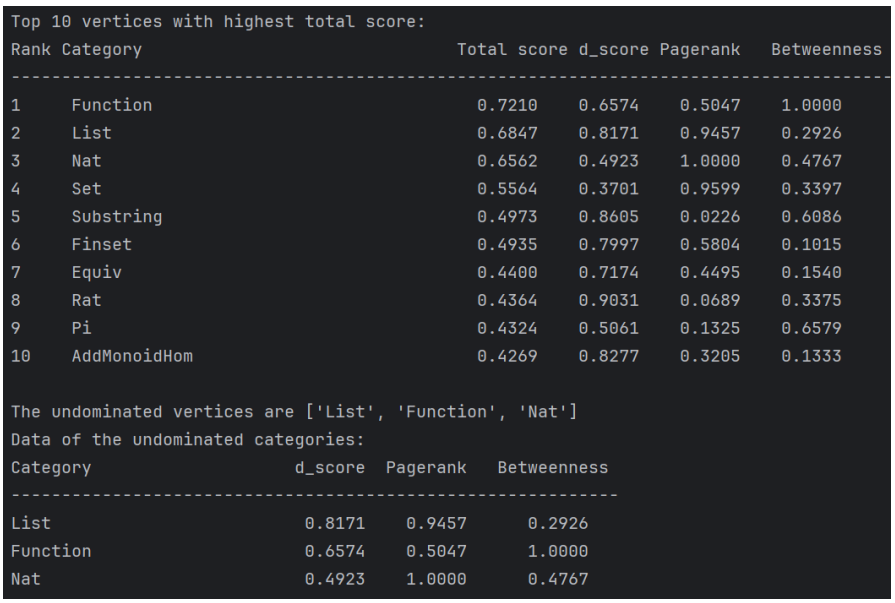
```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Function                               0.7207    0.6943    0.4669    1.0000
2     List                                   0.6831    0.8416    0.9048    0.3038
3     Nat                                    0.6635    0.5264    0.9508    0.5137
4     Set                                    0.6022    0.4115    1.0000    0.3958
5     Substring                              0.5046    0.8568    0.0231    0.6335
6     Finset                                 0.4927    0.8578    0.5149    0.1065
7     Filter                                 0.4803    0.9929    0.2451    0.2037
8     Equiv                                  0.4702    0.7297    0.5266    0.1551
9     Pi                                     0.4654    0.5395    0.1227    0.7332
10    Eq                                     0.4588    0.0011    0.9108    0.4645

The undominated vertices are ['List', 'Function', 'Filter', 'Nat']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
List                       0.8416    0.9048      0.3038
Function                   0.6943    0.4669      1.0000
Filter                     0.9929    0.2451      0.2037
Nat                        0.5264    0.9508      0.5137
```

FIGURE 46: Result Category Graph when large categories are excluded. Category size below 60 removed



```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Function                               0.7189    0.6680    0.4880    1.0000
2     List                                   0.6859    0.8170    0.9377    0.3041
3     Nat                                    0.6704    0.4946    1.0000    0.5170
4     Set                                    0.5638    0.3810    0.9408    0.3702
5     Substring                              0.5157    0.8641    0.0229    0.6595
6     Finset                                 0.4915    0.8119    0.5598    0.1039
7     Pi                                     0.4639    0.5120    0.1292    0.7496
8     Rat                                    0.4468    0.9104    0.0682    0.3622
9     Equiv                                  0.4408    0.7194    0.4457    0.1583
10    AddMonoidHom                           0.4345    0.8280    0.3148    0.1616

The undominated vertices are ['List', 'Function', 'Nat']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
List                       0.8170    0.9377      0.3041
Function                   0.6680    0.4880      1.0000
Nat                        0.4946    1.0000      0.5170
```

FIGURE 47: Result Category Graph when large categories are excluded. Category size below 65 removed



```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Function                               0.7210    0.6574    0.5047    1.0000
2     List                                   0.6847    0.8171    0.9457    0.2926
3     Nat                                    0.6562    0.4923    1.0000    0.4767
4     Set                                    0.5564    0.3701    0.9599    0.3397
5     Substring                              0.4973    0.8605    0.0226    0.6086
6     Finset                                 0.4935    0.7997    0.5804    0.1015
7     Equiv                                  0.4400    0.7174    0.4495    0.1540
8     Rat                                    0.4364    0.9031    0.0689    0.3375
9     Pi                                     0.4324    0.5061    0.1325    0.6579
10    AddMonoidHom                           0.4269    0.8277    0.3205    0.1333

The undominated vertices are ['List', 'Function', 'Nat']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-------------------------------------------------------
List                       0.8171    0.9457      0.2926
Function                   0.6574    0.5047      1.0000
Nat                        0.4923    1.0000      0.4767
```

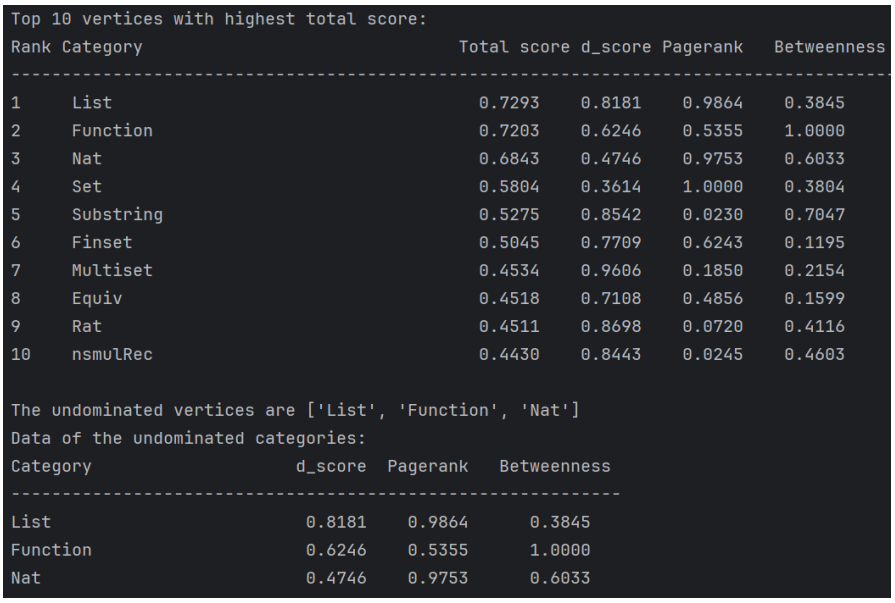
FIGURE 48: Result Category Graph when large categories are excluded. Category size below 70 removed

```
Top 10 vertices with highest total score:
Rank Category                                Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Function                                 0.7190    0.6561    0.4999    1.0000
2     List                                     0.6847    0.8183    0.9357    0.3012
3     Nat                                      0.6616    0.4889    1.0000    0.4964
4     Set                                      0.5544    0.3703    0.9514    0.3423
5     Finset                                   0.4938    0.7977    0.5783    0.1067
6     Substring                                0.4899    0.8566    0.0220    0.5908
7     Equiv                                    0.4400    0.7180    0.4471    0.1556
8     Rat                                      0.4348    0.9004    0.0706    0.3337
9     AddMonoidHom                             0.4301    0.8277    0.3222    0.1412
10    Multiset                                 0.4251    0.9355    0.1667    0.1738

The undominated vertices are ['List', 'Function', 'Nat']
Data of the undominated categories:
Category                   d_score  Pagerank   Betweenness
-----------------------------------------------------------
List                        0.8183    0.9357      0.3012
Function                    0.6561    0.4999      1.0000
Nat                         0.4889    1.0000      0.4964
```

FIGURE 49: Result Category Graph when large categories are excluded. Category size below 75 removed

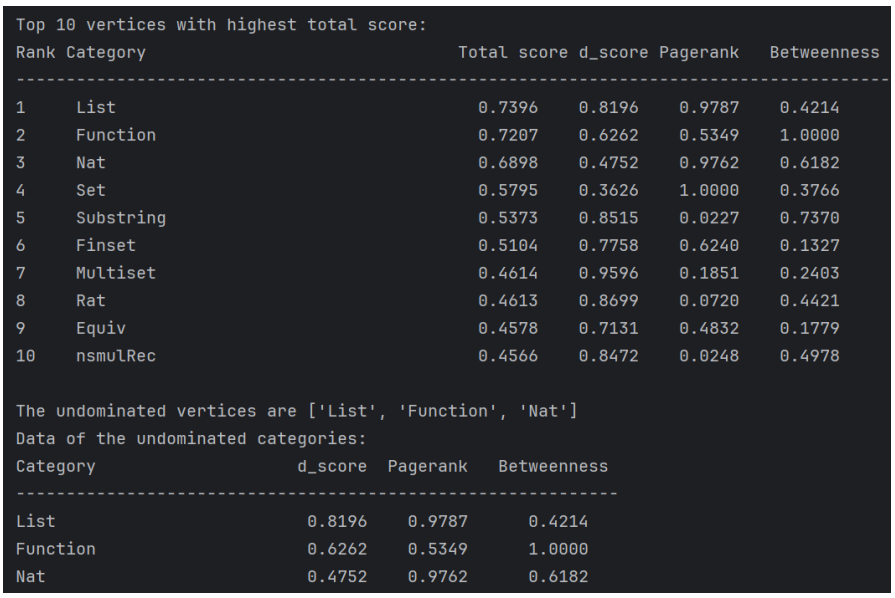
```
Top 10 vertices with highest total score:
Rank Category                                Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     Function                                 0.7179    0.6545    0.4983    1.0000
2     List                                     0.7045    0.8187    0.9328    0.3630
3     Nat                                      0.6985    0.4878    1.0000    0.6081
4     Set                                      0.5569    0.3697    0.9413    0.3602
5     Substring                                0.5234    0.8556    0.0221    0.6920
6     Finset                                   0.5020    0.7985    0.5789    0.1297
7     Rat                                      0.4672    0.8961    0.0720    0.4335
8     Pi                                       0.4611    0.5030    0.1332    0.7463
9     Equiv                                    0.4480    0.7190    0.4486    0.1772
10    AddMonoidHom                             0.4378    0.8273    0.3239    0.1631

The undominated vertices are ['List', 'Function', 'Nat']
Data of the undominated categories:
Category                   d_score  Pagerank   Betweenness
-----------------------------------------------------------
List                        0.8187    0.9328      0.3630
Function                    0.6545    0.4983      1.0000
Nat                         0.4878    1.0000      0.6081
```

FIGURE 50: Result Category Graph when large categories are excluded. Category size below 80 removed



```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     List                                   0.7293    0.8181    0.9864    0.3845
2     Function                               0.7203    0.6246    0.5355    1.0000
3     Nat                                    0.6843    0.4746    0.9753    0.6033
4     Set                                    0.5804    0.3614    1.0000    0.3804
5     Substring                              0.5275    0.8542    0.0230    0.7047
6     Finset                                 0.5045    0.7709    0.6243    0.1195
7     Multiset                               0.4534    0.9606    0.1850    0.2154
8     Equiv                                  0.4518    0.7108    0.4856    0.1599
9     Rat                                    0.4511    0.8698    0.0720    0.4116
10    nsmulRec                               0.4430    0.8443    0.0245    0.4603

The undominated vertices are ['List', 'Function', 'Nat']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-----------------------------------------------------------
List                       0.8181    0.9864      0.3845
Function                   0.6246    0.5355      1.0000
Nat                        0.4746    0.9753      0.6033
```

FIGURE 51: Result Category Graph when large categories are excluded. Category size below 85 removed



```
Top 10 vertices with highest total score:
Rank Category                              Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1     List                                   0.7396    0.8196    0.9787    0.4214
2     Function                               0.7207    0.6262    0.5349    1.0000
3     Nat                                    0.6898    0.4752    0.9762    0.6182
4     Set                                    0.5795    0.3626    1.0000    0.3766
5     Substring                              0.5373    0.8515    0.0227    0.7370
6     Finset                                 0.5104    0.7758    0.6240    0.1327
7     Multiset                               0.4614    0.9596    0.1851    0.2403
8     Rat                                    0.4613    0.8699    0.0720    0.4421
9     Equiv                                  0.4578    0.7131    0.4832    0.1779
10    nsmulRec                               0.4566    0.8472    0.0248    0.4978

The undominated vertices are ['List', 'Function', 'Nat']
Data of the undominated categories:
Category                  d_score  Pagerank   Betweenness
-----------------------------------------------------------
List                       0.8196    0.9787      0.4214
Function                   0.6262    0.5349      1.0000
Nat                        0.4752    0.9762      0.6182
```

FIGURE 52: Result Category Graph when large categories are excluded. Category size below 90 removed

```
Top 10 vertices with highest total score:
Rank Category                                Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    List                                      0.7438    0.8204    0.9641    0.4477
2    Function                                  0.7174    0.6209    0.5305    1.0000
3    Nat                                       0.6931    0.4749    0.9578    0.6469
4    Set                                       0.5851    0.3538    1.0000    0.4020
5    Substring                                 0.5217    0.8468    0.0218    0.6960
6    Finset                                    0.5041    0.7645    0.6078    0.1411
7    Rat                                       0.4691    0.8836    0.0719    0.4520
8    nsmulRec                                  0.4666    0.8489    0.0232    0.5274
9    Equiv                                     0.4649    0.7124    0.4832    0.2000
10   Multiset                                  0.4631    0.9567    0.1839    0.2494

The undominated vertices are ['List', 'Function', 'Nat']
Data of the undominated categories:
Category                   d_score  Pagerank   Betweenness
-----------------------------------------------------------
List                        0.8204    0.9641      0.4477
Function                    0.6209    0.5305      1.0000
Nat                         0.4749    0.9578      0.6469
```

FIGURE 53: Result Category Graph when large categories are excluded. Category size below 95 removed

```
Top 10 vertices with highest total score:
Rank Category                                Total score d_score Pagerank   Betweenness
-------------------------------------------------------------------------------------
1    List                                      0.7676    0.8123    1.0000    0.4913
2    Nat                                       0.6977    0.4612    0.9304    0.7016
3    Function                                  0.6529    0.6173    0.5125    0.8283
4    Set                                       0.5680    0.3438    0.9261    0.4346
5    Substring                                 0.5085    0.8440    0.0198    0.6613
6    Rat                                       0.5034    0.9003    0.0628    0.5470
7    Finset                                    0.4927    0.7533    0.5701    0.1557
8    nsmulRec                                  0.4876    0.8504    0.0213    0.5908
9    Multiset                                  0.4651    0.9624    0.1771    0.2563
10   Equiv                                     0.4612    0.7096    0.4437    0.2312

The undominated vertices are ['List', 'Function', 'Nat']
Data of the undominated categories:
Category                   d_score  Pagerank   Betweenness
-----------------------------------------------------------
List                        0.8123    1.0000      0.4913
Function                    0.6173    0.5125      0.8283
Nat                         0.4612    0.9304      0.7016
```

FIGURE 54: Result Category Graph when large categories are excluded. Category size below 100 removed