Report

Question 1

For the assignment purposes, I implemented the *selected1*, *selected2*, *selected3* algorithms as presented in the assignment document. For the *selected4* algorithm, which is my proposed variation of the previous algorithms, I follow the same pattern as *selected1* with the modification that all the adjacency lists have been precomputed into a adjacency matrix, so at each call all I have to do is to retrieve the proper adjacency list based on the proper index. For testing my implementations over the 4 different selected algorithms I created a small graph that is given below as well as the output results I am receiving over a run of the selected algorithms having a random edge as an input. All algorithm implementations share the same random numbers r_e.

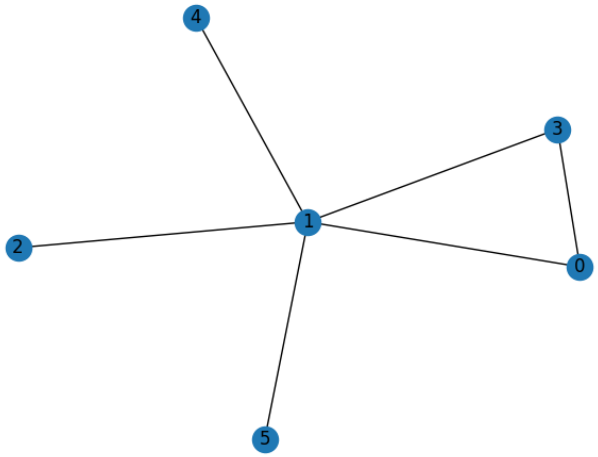


Figure 1: Question 1: Sanity Check Graph

Figure 2: Question 1: Sanity Check Results

Question 2

As a sanity check, I am reporting the results of my two generative methods given as parameters $n = 6$ and $d = 2$ (6 nodes with roughly degree of 2). The implementation 2a outputs a graph with 6 nodes and exactly 2 as the degree of each node, while the implementation 2b outputs a graph with 6 nodes and roughly 3 as the degree of each node (some of them might have 3).

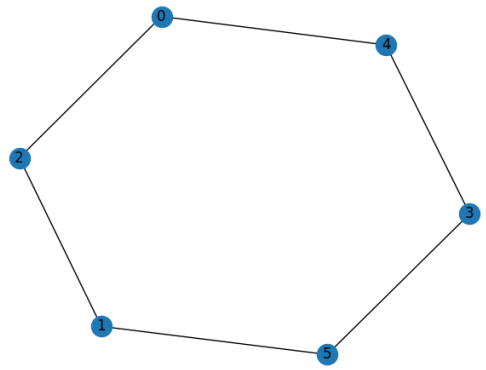


Figure 3: Implementation 2a) n=6, d=2

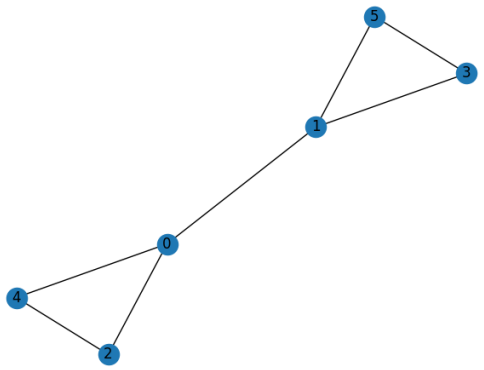


Figure 4: Implementation 2b) n=6, d=2

Question 3

I run my experiments with applying the exploration methods of question 1 to the generative methods of question 2 with the following parameters. The total number of nodes is n = 30 to make sure that the number of vertices, is sufficiently large so you don't see any unusual behavior due to the size of the graph. We repeat the exploration methods *repetitions* = 100 with different r_e numbers and different random edges at each repetition. The results for each graph generative method is described below in the following tables 1 and 2 respectively. We notice that as the degree increases the average recursive calls increases as well. Also, in comparison of *selected1*, *selected2* and *selected3* we observe that the number of average recursive calls is decreasing. **My variation named *selected4* is identical with *selected1*** since it is identical to *selected1* with the only difference that I have precomputed all adjacency list beforehand. This variation do not affect the number of average recursive calls leading to the same outputs for *selected1* and *selected4* exploration methods.

Degree d	**Selected 1**	**Selected 2**	**Selected 3**	**Selected 4**
2	**1.45**	**1**	**1.02**	**1.45**
3	**6.03**	**1.95**	**1.37**	**6.03**
4	**24.27**	**5.7**	**2.32**	**24.27**
5	**90.36**	**10.4**	**2.44**	**90.36**
6	**470.7**	**20.32**	**5.96**	**470.7**

Table 1: Average recursive calls for graph generative method 2a.

Degree d	**Selected 1**	**Selected 2**	**Selected 3**	**Selected 4**
2	**4.35**	**2.17**	**1.37**	**4.35**
3	**37.51**	**6.14**	**2.04**	**37.51**
4	**267.15**	**12.52**	**3.96**	**267.15**
5	**1721.72**	**34.07**	**4.29**	**1721.72**
6	**4789.37**	**33.48**	**5.38**	**4789.37**

Table 2: Average recursive calls for graph generative method 2b.

Question 4

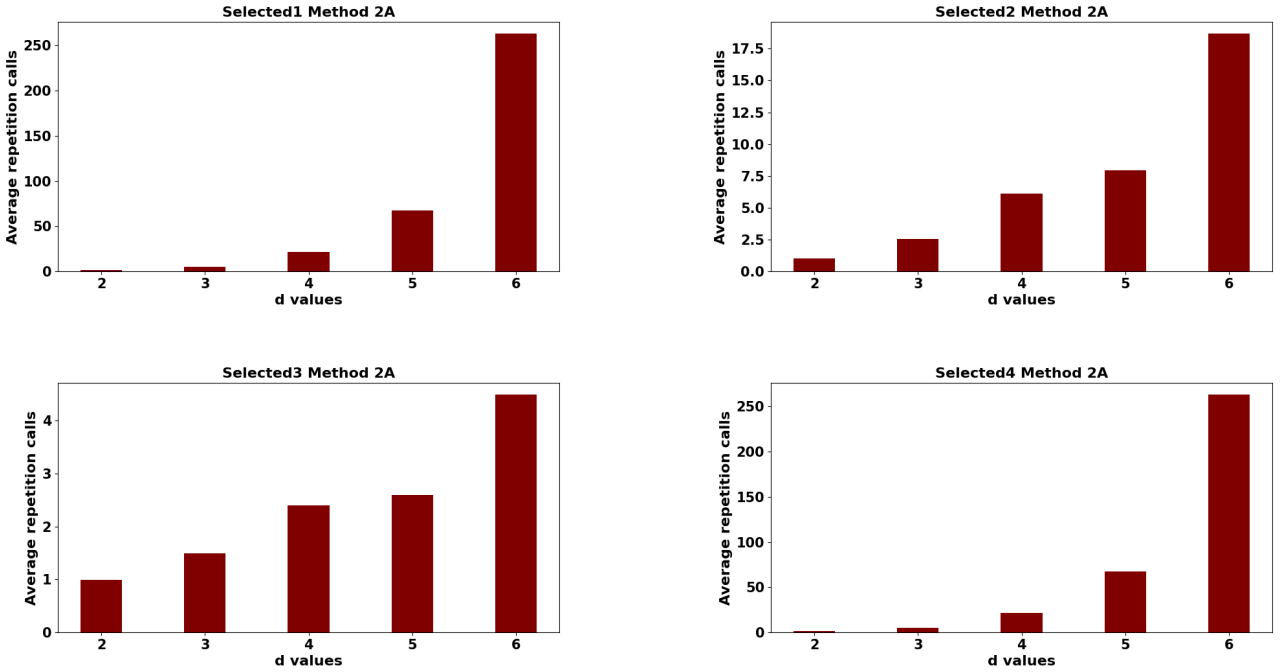


Figure 5: Results of all exploration methods on the graph generative method 2a

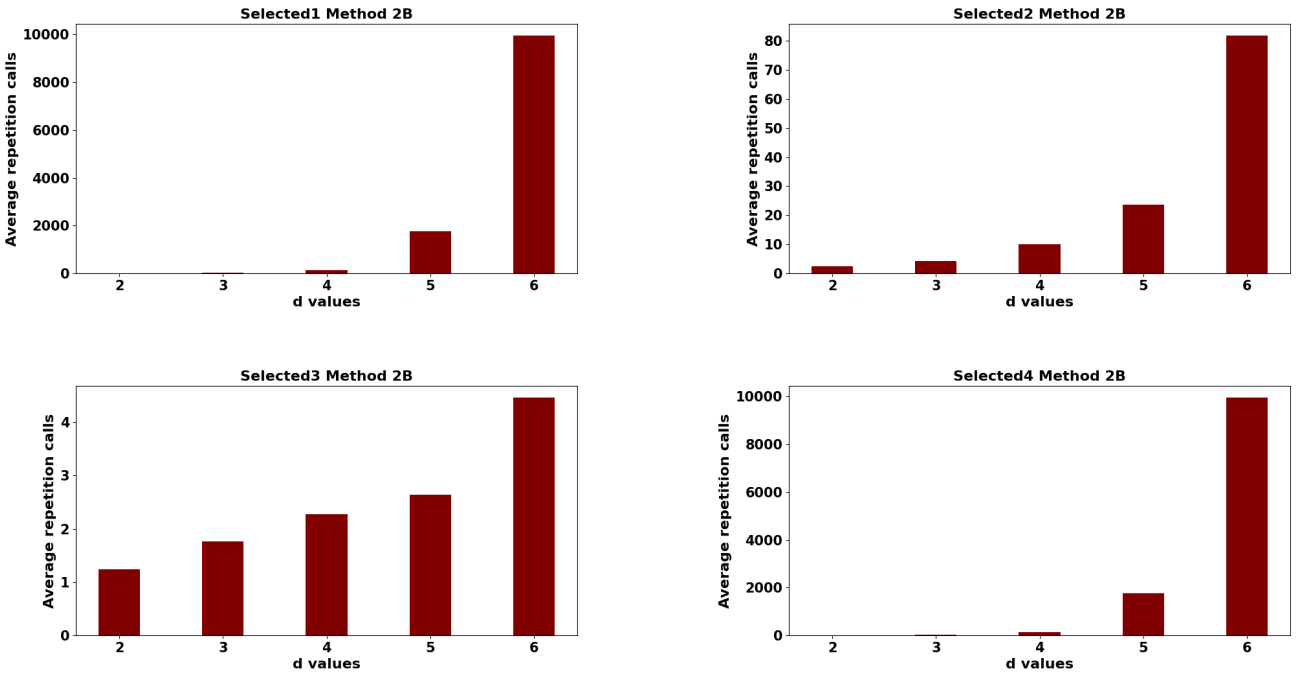
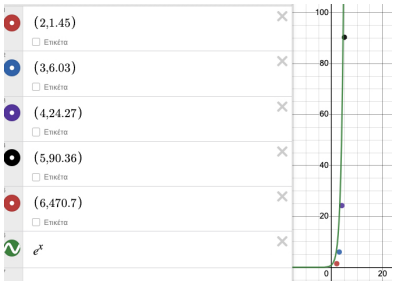
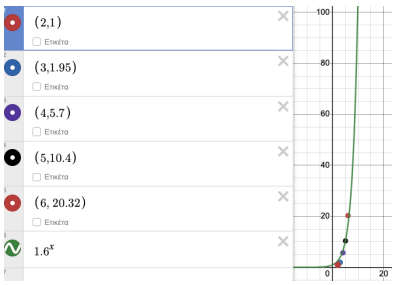


Figure 6: Results of all exploration methods on the graph generative method 2a

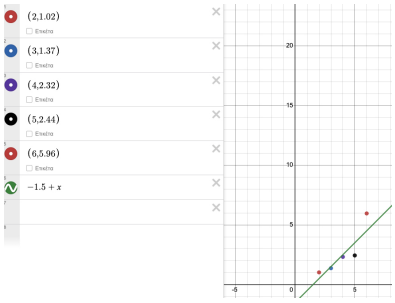
Question 5



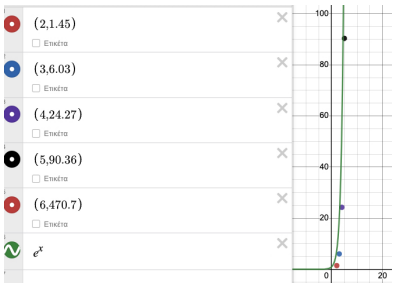
(a) Exponential Growth of Selected 1



(b) Exponential Growth of Selected 2

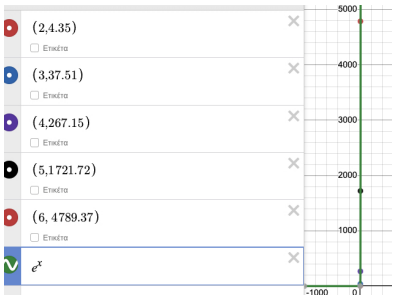


(c) Linear Growth of Selected 3

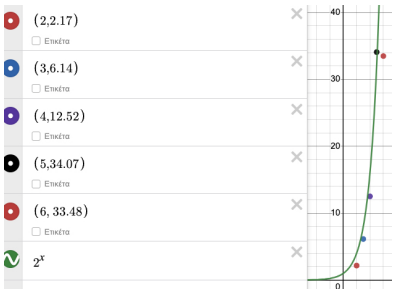


(d) Exponential Growth of Selected 4

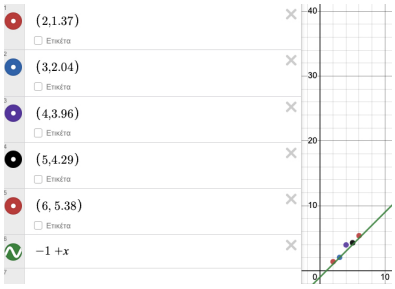
Figure 7: Growth plot of all exploration methods for generative method 2a



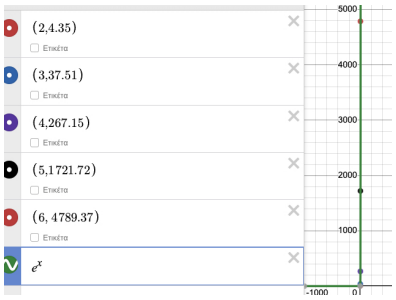
(a) Exponential Growth of Selected 1



(b) Exponential Growth of Selected 2



(c) Linear Growth of Selected 3



(d) Exponential Growth of Selected 4

Figure 8: Growth plot of all exploration methods for generative method 2b

Code Reproducibility

The code was written and tested on Ubuntu 20.04 operating system, with Python 3.9 installed. The only python packages required to be installed to run the code are the following: *numpy, matplotlib, networkx*. To install them under pip you need to follow the command:

- pip install numpy networkx matplotlib

After unzipping the .zip folder and accessing it, the following lines provide you a description of how to run the subquestions of the exercise.

For question 1, please run the *q1* script as follows: **python q1.py**

For question 2a), please run the *q2a* script as follows: **python q2a.py**

For question 2b), please run the *q2b* script as follows: **python q2b.py**

For questions 3 and 4, please run the *q3-4* script as follows: **python q3-4.py**

All the plots will be generated under the plots folder.