Analysis of Various Graph Exploration Algorithms

Introduction

In this report, I have analyzed the experiment I worked which evaluated the performance of three Graph Exploration methods for finding the maximal matching. I also implemented two algorithms for Graph Generations, given the number of nodes and the desired vertex degree.

Background

As the name suggests, Graph Generation is the process of creating a graph, using the given information about the edges and the vertices of the desired graph. Graphs are an important concept in computer science and mathematics. They help visualize the relationship between various data points. They have applications in areas like social networks, transportation systems, and many more.

Matching is basically a list of a set of pairwise non-adjacent edges. A maximal matching is the one where we can not add any more edges to the matching. Maximal Matching has applications in areas like resource allocation and scheduling.

Implementation

I used the python library NetworkX to manipulate graphs in the implementation. NetworkX provides a wide range of tools for working and manipulating graphs, it can also be used to generate connected graphs, draw graphs, and find the number of edges and nodes.

For graph generation, I used two methods one where I took the union of d random matchings, each of size $n/2$, and in the second method I assigned a random point in Euclidean Square $(0,1)^2$ to every vertex and for each vertex v, I added edges from v to d vertices corresponding to the closest point in the square.

For the exploration method selected1, I simply loop over each adjacent edge and recursively check if that edge is blocked from entering the maximal matching by any of its adjacent edges which appear before it in the corresponding random permutation.

For selected2, I tried to optimize selected1 by stopping after discovering an adjacent edge in the maximal matching.

In selected3, I considered the adjacent edges in increasing order of their Re values. The idea behind this is that edges with lower Re values should be more likely to be in the maximal matching.

(Answer to the rationale behind my variation of exploration algorithm)
For my variation, I implemented a hybrid of selected1 and selected3. The idea behind this algorithm is to first sort the edges adjacent to the starting edge in increasing order of their random values (as done in

selected3), and then explore them one by one, checking whether each edge is selected (as done in selected1). However, to improve efficiency, I stored the selected status of each edge in a dictionary, so that I never have to recompute the solution for an edge if I visit it again. The rationale behind using this algorithm is that it is a recursive implementation of the graph exploration algorithm and uses memoization to avoid recomputing the same result again. This can improve the performance of the algorithm for large graphs as it reduces the number of recursive calls and prevents stack overflow errors.

Experimental Results

I ran the experiments on graphs with n = 1000 and changed the degrees to d = 2, 3, 4, 5, 6. For each combination, I computed the average recursive calls and also, repeated the experiment three times for each combination with different starting edges and different settings of Re values.

** One thing to note in these experiments, I set the maximum limit on recursive calls as 4000000

(Answer to Q3)
The following tables contain the results obtained from the experiments:

Results of Selected1 Algorithm with Nodes = 1000				
Degree	Exp1	Exp2	Exp3	Average
2	11402	12836	13712	12650.00
3	86436	143717	86094	105415.67
4	586855	808174	492978	629335.67
5	3009503	3100439	2999080	3036340.67
6	> 4000000	> 4000000	> 4000000	4000000

Table1: Results of Selected1 Algorithm

Results of Selected2 Algorithm with Nodes = 1000				
Degree	Exp1	Exp2	Exp3	Average
2	2967	2882	3551	3133.33
3	10094	9547	10801	10147.33
4	28038	37139	28278	31151.67
5	94495	92801	99731	95675.67
6	183831	241157	227658	217548.67

Table2: Results of Selected2 Algorithm

Results of Selected3 Algorithm with Nodes = 1000				
Degree	Exp1	Exp2	Exp3	Average
2	2910	3054	2742	2902.00
3	10189	9164	8807	9386.67
4	22682	30929	22930	25513.67
5	81007	58839	57692	65846.00
6	116471	157983	147919	140791.00

Table3: Results of Selected3 Algorithm

Results of My variation of Graph Exploration Algorithm with Nodes = 1000				
Degree	Exp1	Exp2	Exp3	Average
2	1030	923	1071	1008.00
3	3158	3055	2884	3032.33
4	10011	12382	8622	10338.33
5	23467	30330	25884	26560.33
6	77122	49927	42974	56674.33

Table4: Results of My variation of Graph Exploration Algorithm

(Answer to Q4)

I created the following visualizations to better understand the relationship between vertex degree and the number of recursive calls for the four graph exploration algorithms.

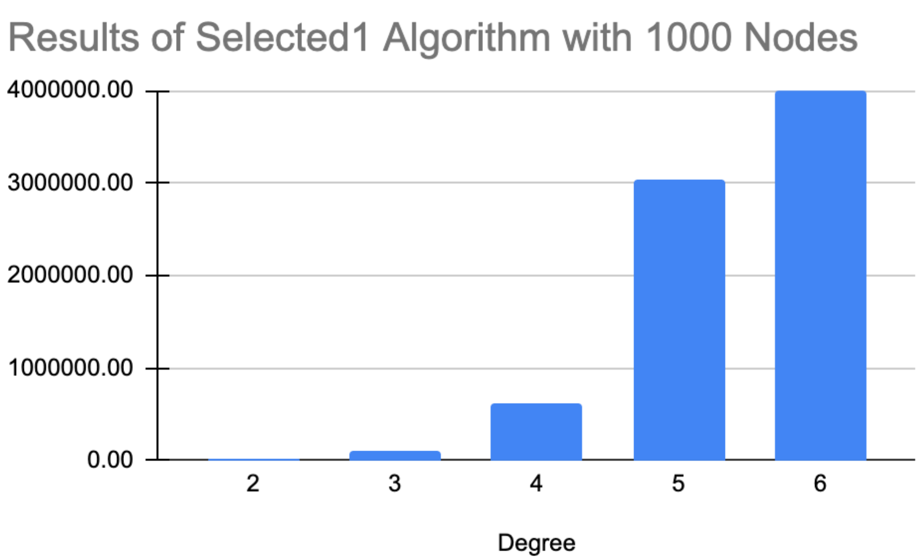

Figure1: Bar Graph of the number of recursive calls in Selected1 Algorithm as a function of vertex degree

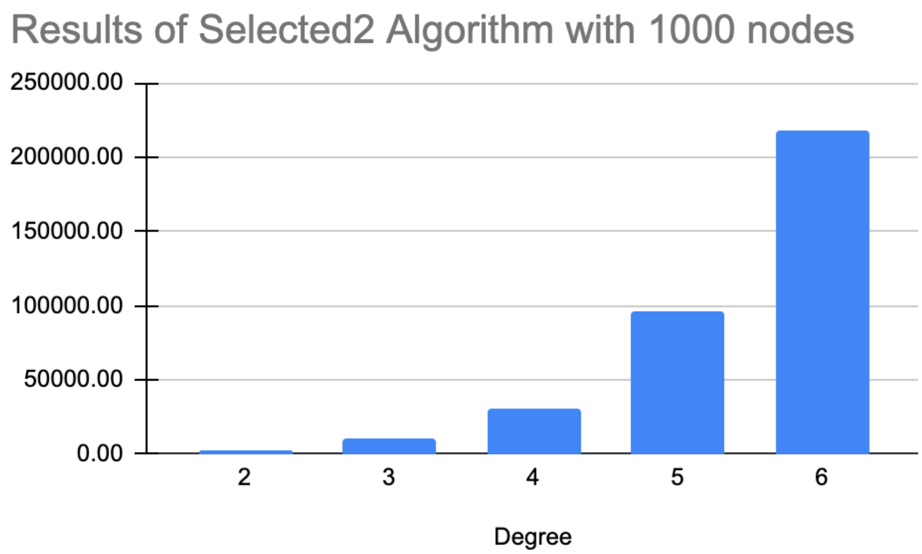

Figure2: Bar Graph of the number of recursive calls in Selected2 Algorithm as a function of vertex degree

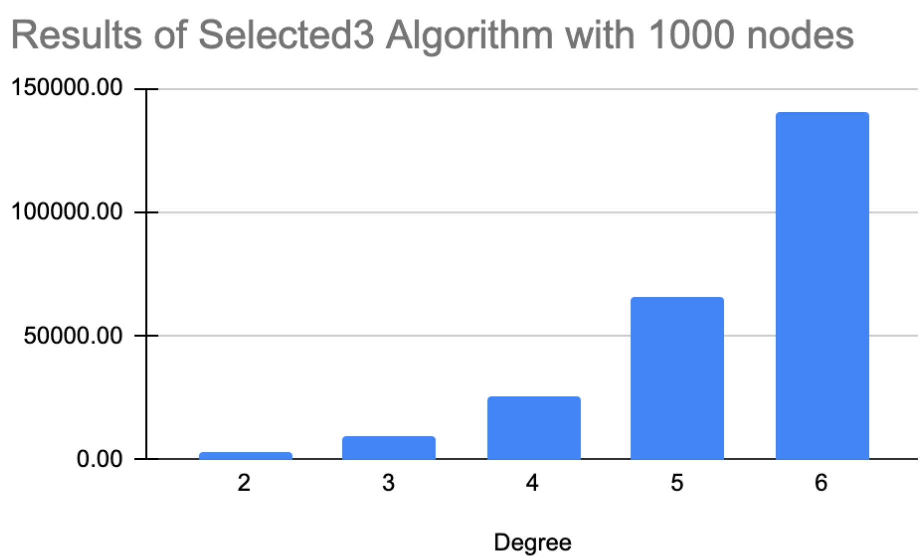

Figure3: Bar Graph of the number of recursive calls in Selected3 Algorithm as a function of vertex degree

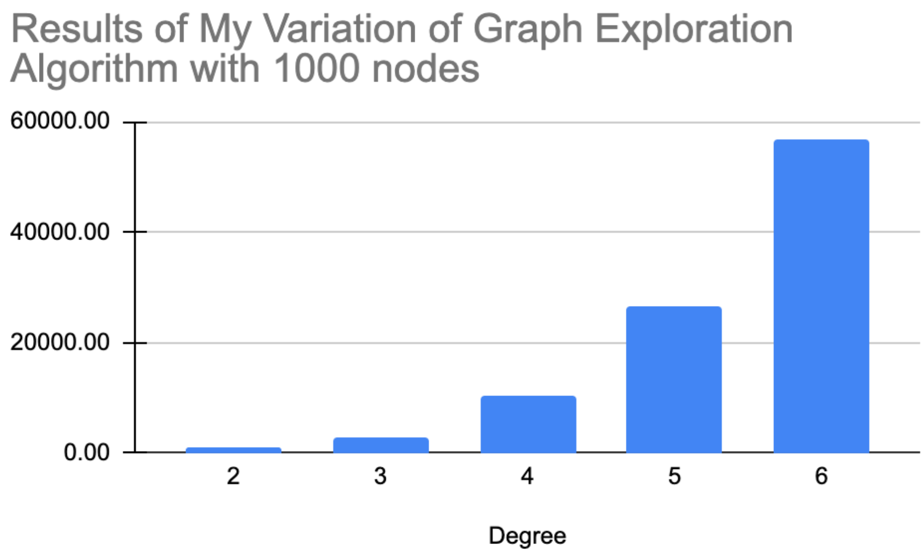

Figure4: Bar Graph of the number of recursive calls in My Variation of Graph Exploration Algorithm as a function of vertex degree

(Answer to Q5)
To better understand the relationship between the number of recursive calls and the vertex degree of the graph, I created the following two line graphs. Both graphs are showing the same thing, I just created the second one to better visualize the selected2, selected3, and my variation algorithms as the scale of the y-axis for these and the selected1 algorithm is very different.

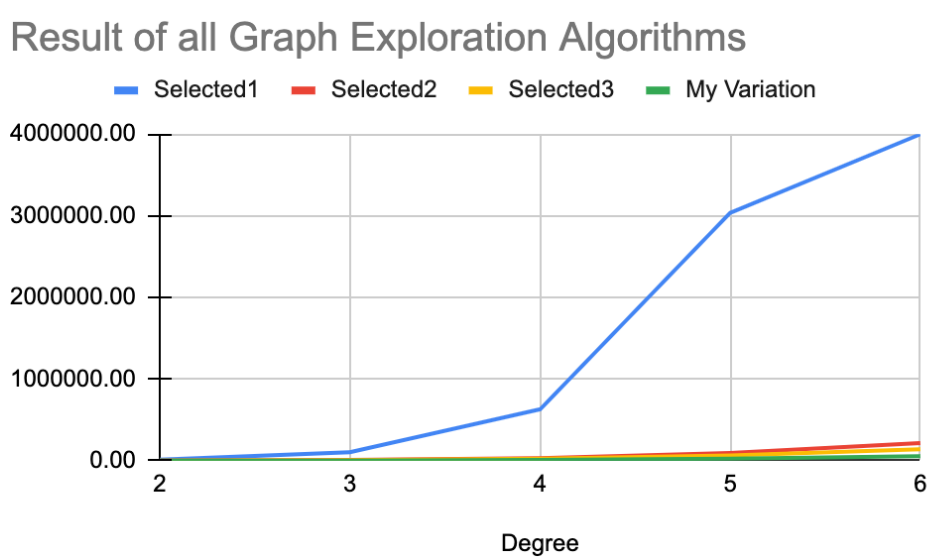

Figure5: Line Graph of the number of recursive calls for multiple values of vertex degree for all algorithms

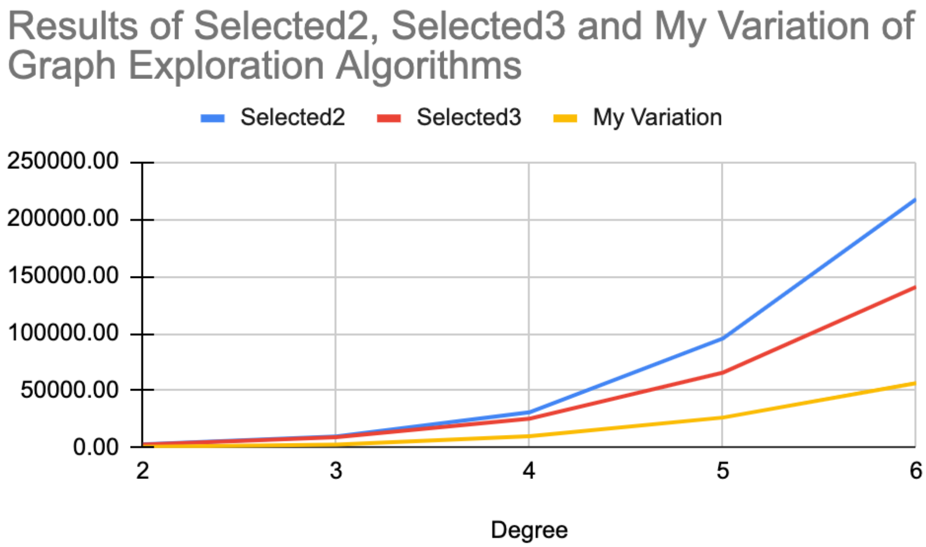

Figure6: Line Graph of the number of recursive calls for multiple values of vertex degree for selected2, selected3, and my variation algorithms

Discussion

(Answer to Q5)

According to the above graphs, we can see that the number of recursive calls increases **exponentially** as the vertex degree increases for all four graph exploration algorithms used in this experiment.

We can also notice that the selected1 algorithm requires the highest number of recursive calls almost 10 times selected2. The reason behind this could be the fact that it explores the graphs like in a DFS algorithm, it does not include any optimization or randomness in algorithm. This leads to the increased possibility of exploring the paths even if they won't lead to maximal matching.

Also, we can see that my variation of the exploration algorithm performed the best as it included memoization.

Conclusion

All in all, the results of this experiment show that the average number of recursive calls per edge for the given exploration algorithms increases **exponentially** with vertex degree.

Link to the Excel Sheet where, I collected and analyzed data