

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:

For Random Matching Graph Generation Algorithm

Results of Selected1 Algorithm with Nodes = 1000				
Degree	Exp1	Exp2	Exp3	Average
2	11402	12836	13712	12650
3	86436	143717	86094	105416
4	586855	808174	492978	629336
5	3009503	3100439	2999080	3036341
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
3	10094	9547	10801	10147
4	28038	37139	28278	31152
5	94495	92801	99731	95676
6	183831	241157	227658	217549

Table2: Results of Selected2 Algorithm

Results of Selected3 Algorithm with Nodes = 1000

Degree	Exp1	Exp2	Exp3	Average
2	2910	3054	2742	2902
3	10189	9164	8807	9387
4	22682	30929	22930	25514
5	81007	58839	57692	65846
6	116471	157983	147919	140791

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
3	3158	3055	2884	3032
4	10011	12382	8622	10338
5	23467	30330	25884	26560
6	77122	49927	42974	56674

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.

Results of Selected1 Algorithm with 1000 Nodes

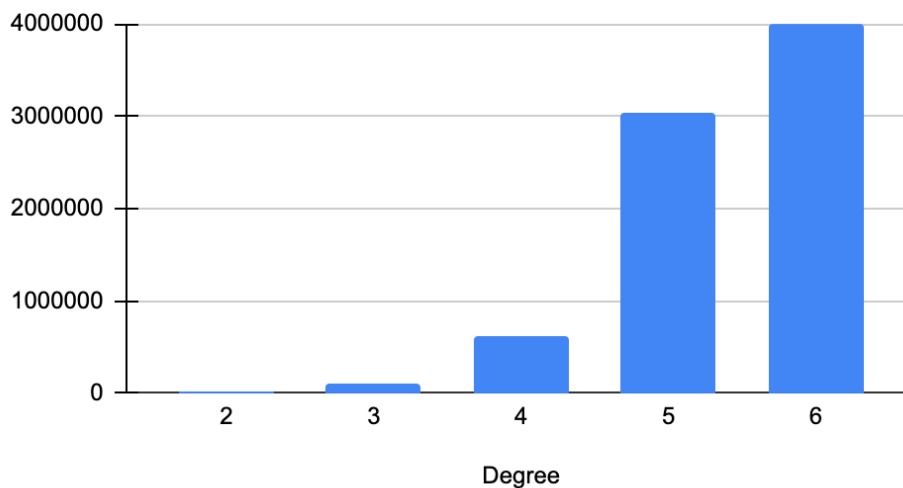


Figure1: Bar Graph of the number of recursive calls in Selected1 Algorithm as a function of vertex degree

Results of Selected2 Algorithm with 1000 nodes

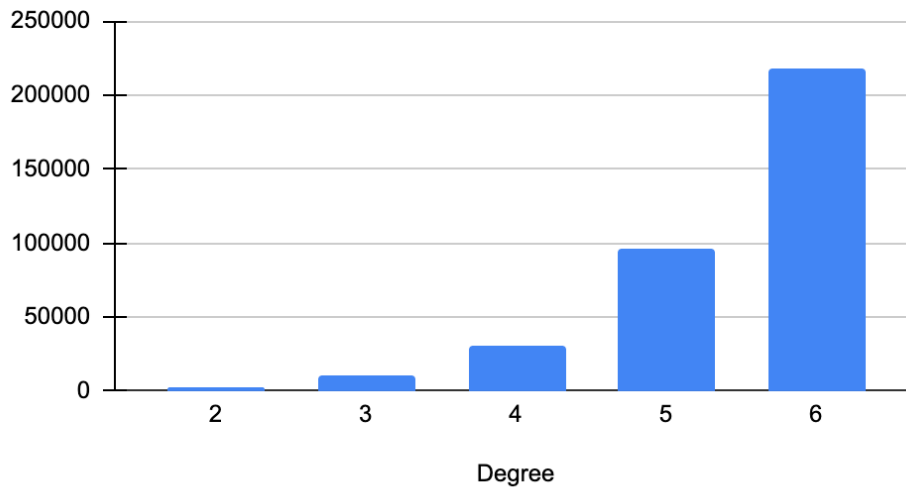


Figure2: Bar Graph of the number of recursive calls in Selected2 Algorithm as a function of vertex degree

Results of Selected3 Algorithm with 1000 nodes

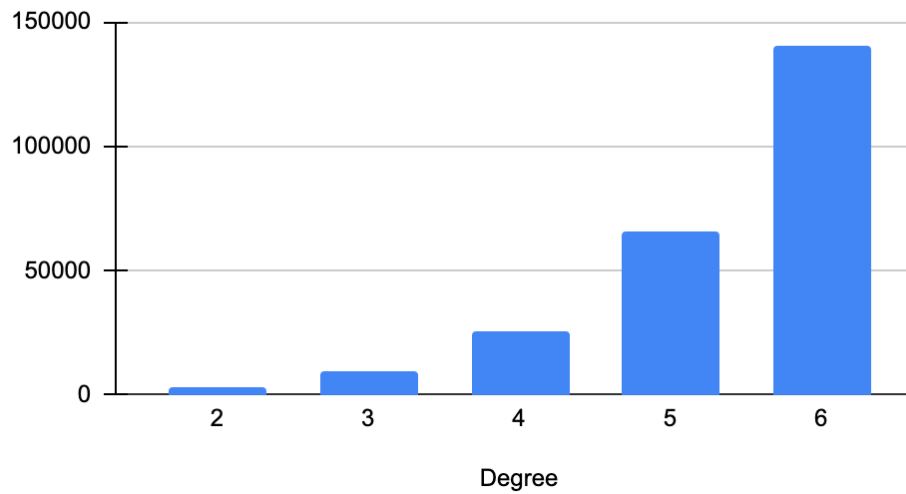


Figure3: Bar Graph of the number of recursive calls in Selected3 Algorithm as a function of vertex degree

Results of My Variation of Graph Exploration Algorithm with 1000 nodes

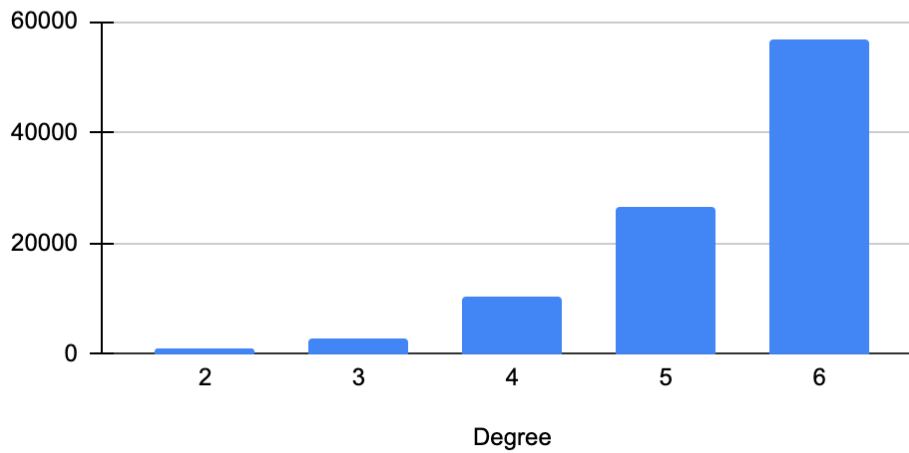


Figure4: Bar Graph of the number of recursive calls in My Variation of Graph Exploration Algorithm as a function of vertex degree

For Euclidean Square Graph Generation Algorithm

Results of Selected1 Algorithm with Nodes = 1000

Degree	Exp1	Exp2	Exp3	Average
2	6997	6376	5961	6445
3	56589	51171	60328	56029
4	361964	405133	358312	375136
5	3051496	2817634	2511653	2793594
6	> 4000000	> 4000000	> 4000000	4000000

Table5: Results of Selected1 Algorithm

Results of Selected2 Algorithm with Nodes = 1000

Degree	Exp1	Exp2	Exp3	Average
2	3200	2987	3115	3101
3	10848	10994	10009	10617
4	25977	27569	33597	29048
5	77538	85865	96801	86735
6	277982	210594	317500	268692

Table6: Results of Selected2 Algorithm

Results of Selected3 Algorithm with Nodes = 1000

Degree	Exp1	Exp2	Exp3	Average
2	2946	2996	2906	2949
3	9761	9758	10612	10044
4	30103	28199	28152	28818
5	66412	65871	66502	66262
6	197280	178918	175064	183754

Table7: Results of Selected3 Algorithm

Results of My variation of Graph Exploration Algorithm with Nodes = 1000

Degree	Exp1	Exp2	Exp3	Average
2	1171	1114	1121	1135
3	3718	3721	3798	3746
4	10508	8861	8952	9440
5	23013	19646	18727	20462
6	52780	62120	101703	72201

Table8: Results of My variation of Graph Exploration Algorithm

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.

Results of Selected1 Algorithm with 1000 Nodes

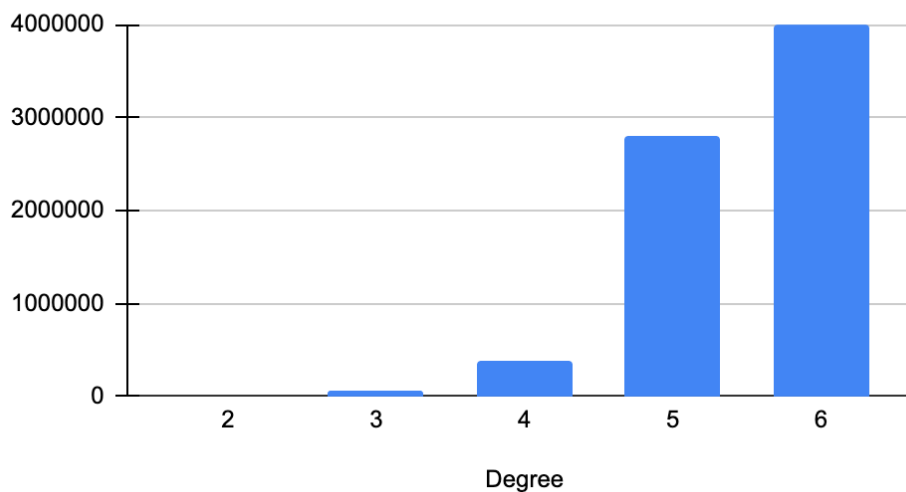


Figure5: Bar Graph of the number of recursive calls in Selected1 Algorithm as a function of vertex degree

Results of Selected2 Algorithm with 1000 nodes

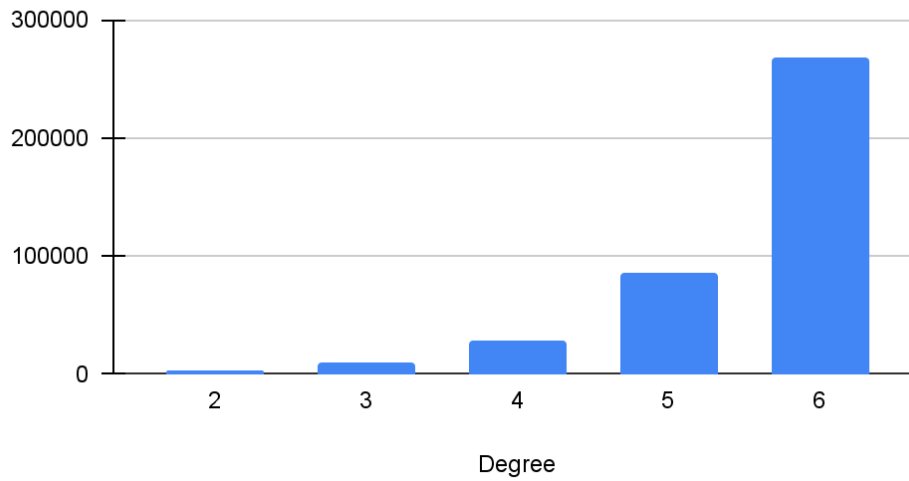


Figure6: Bar Graph of the number of recursive calls in Selected2 Algorithm as a function of vertex degree

Results of Selected3 Algorithm with 1000 nodes

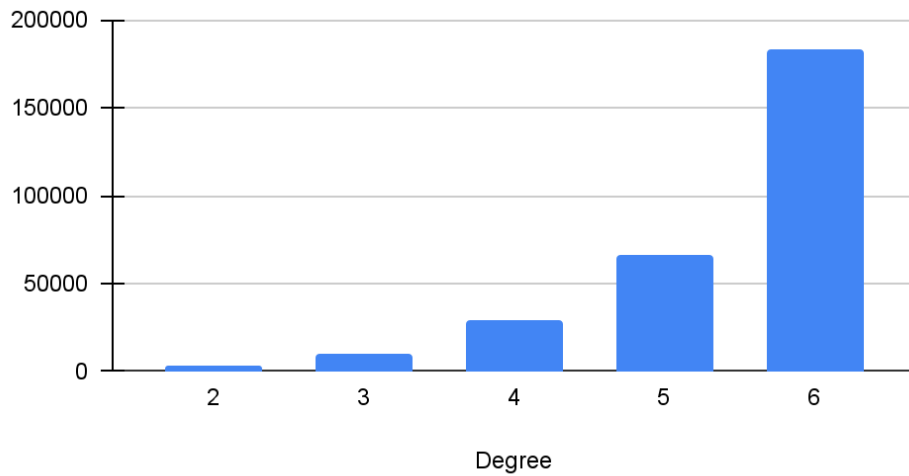


Figure7: Bar Graph of the number of recursive calls in Selected3 Algorithm as a function of vertex degree

Results of My Variation of Graph Exploration Algorithm with 1000 nodes

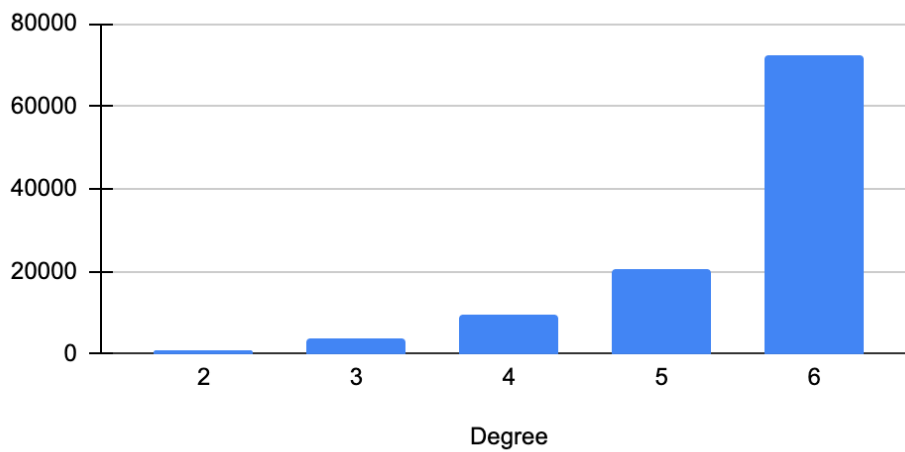


Figure8: Bar Graph of the number of recursive calls in My Variation of Graph Exploration Algorithm as a function of vertex degree

Analyzing the Results

For Random Matching Graph Generation Algorithm

In the following table, I have combined the results of all the graph exploration algorithms:

Combined Results of all Graph Exploration Algorithms with 1000 Nodes				
Degree	Selected1	Selected2	Selected3	My Variation
2	12650	3133	2902	1008
3	105416	10147	9387	3032
4	629336	31152	25514	10338
5	3036341	95676	65846	26560
6	4000000	217549	140791	56674

Table9: Combined Results of all four Graph Exploration Algorithms

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.

Result of all Graph Exploration Algorithms

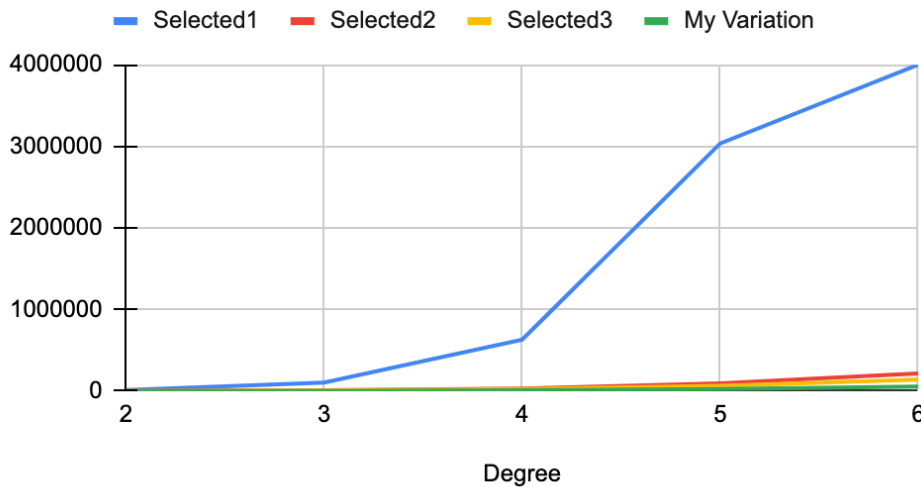


Figure9: Line Graph of the number of recursive calls for multiple values of vertex degree for all algorithms

Results of Selected2, Selected3 and My Variation of Graph Exploration Algorithms

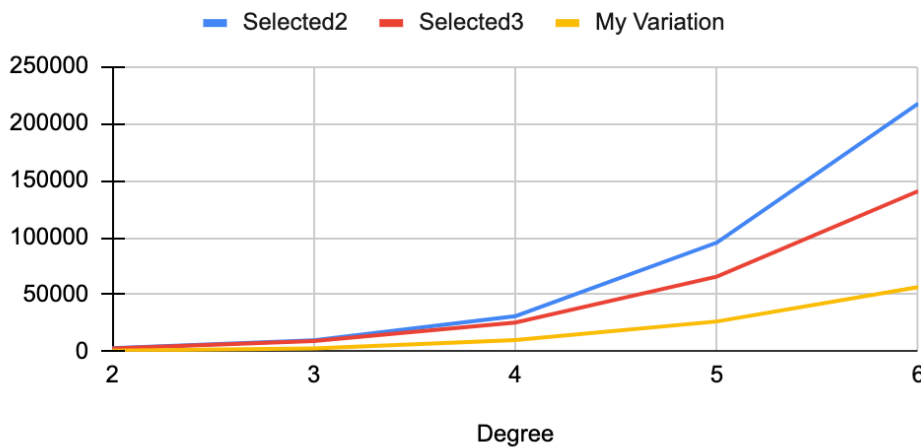


Figure10: Line Graph of the number of recursive calls for multiple values of vertex degree for selected2, selected3, and my variation algorithms

(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.

For Euclidean Square Graph Generation Algorithm

In the following table, I have combined the results of all the graph exploration algorithms:

Combined Results of all Graph Exploration Algorithms with 1000 Nodes				
Degree	Selected1	Selected2	Selected3	My Variation
2	6445	3101	2949	1135
3	56029	10617	10044	3746
4	375136	29048	28818	9440
5	2793594	86735	66262	20462
6	4000000	268692	183754	72201

Table10: Combined Results of all four Graph Exploration Algorithms

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 show 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.

Result of all Graph Exploration Algorithms

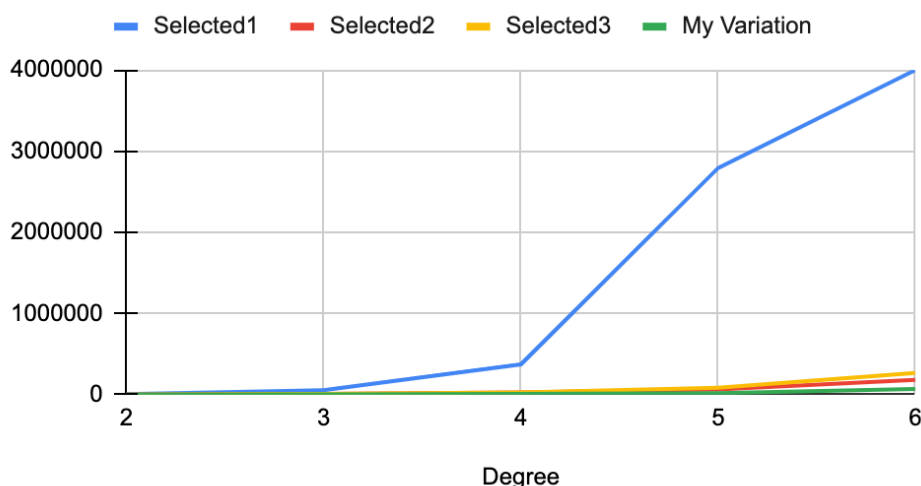


Figure9: Line Graph of the number of recursive calls for multiple values of vertex degree for all algorithms

Results of Selected2, Selected3 and My Variation of

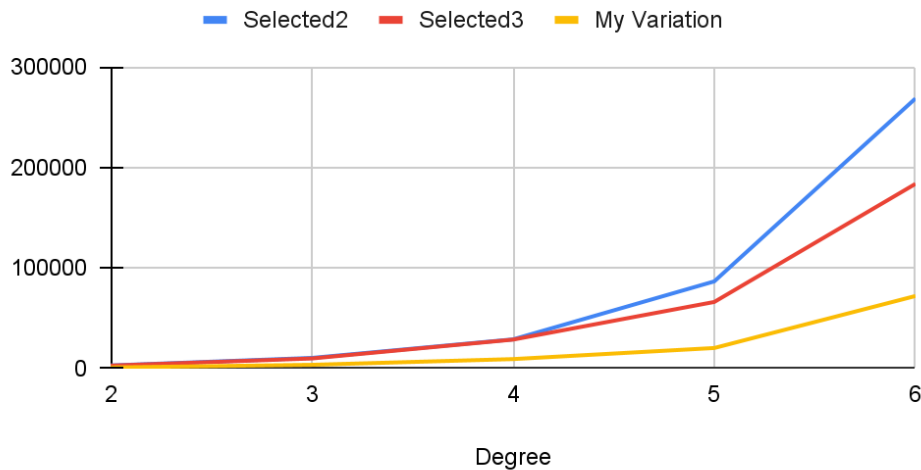


Figure10: Line Graph of the number of recursive calls for multiple values of vertex degree for selected2, selected3, and my variation algorithms

(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.

Comparing the results of two Graph Generation algorithms

Selected1 Algorithm

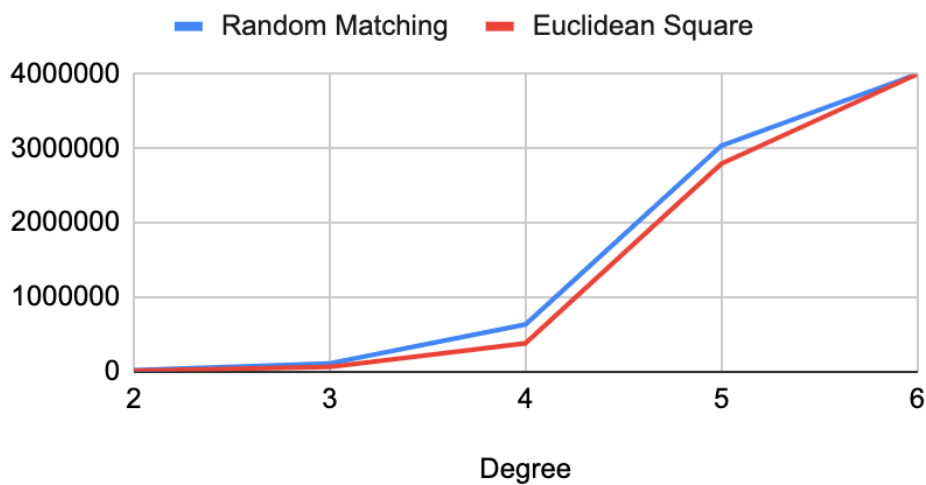


Figure11: Performance of Selected1 Algorithm for graphs generated using both the Graph Generation Algorithms

Selected2 Algorithm

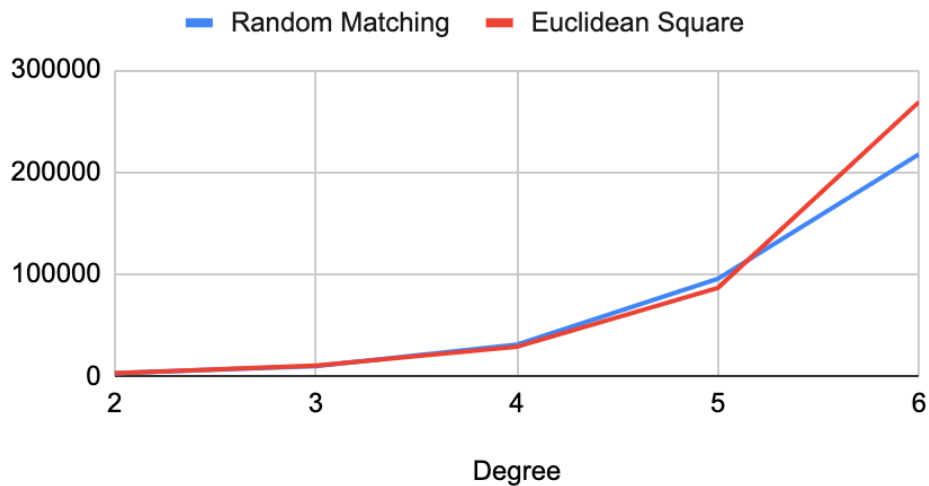


Figure12: Performance of Selected2 Algorithm for graphs generated using both the Graph Generation Algorithms

Selected3 Algorithm

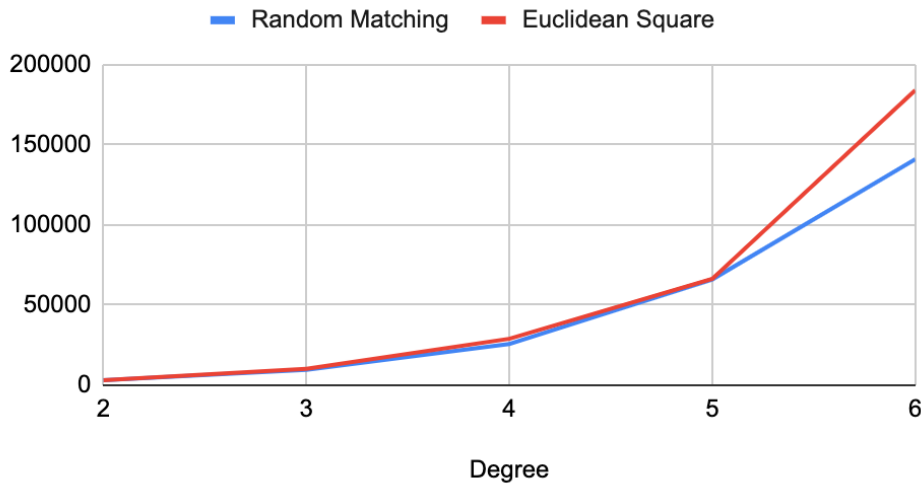


Figure13: Performance of Selecte31 Algorithm for graphs generated using both the Graph Generation Algorithms

My Variation

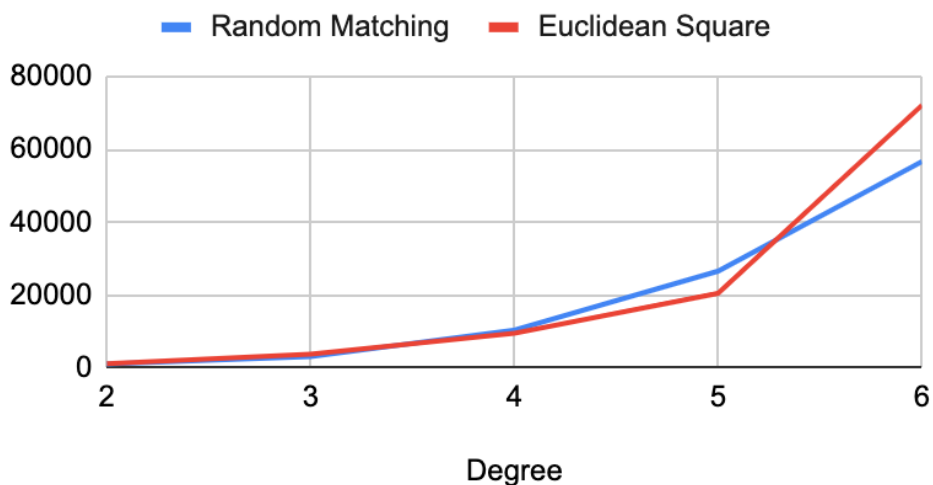


Figure14: Performance of My Variation Algorithm for graphs generated using both the Graph Generation Algorithms

One thing is pretty evident in the above graphs, that all the exploration algorithms perform equally for graph generated using both the graph generation algorithms, but as the degree increases we notice that exploration algorithms perform better on graph generated using Random matching algorithm.

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