AI Assignment 2

Question 2 – Exploring Avenue – Investigating modified greedy colouring algorithm performance across graphs of different sizes and types

- Thomas Meehan – 20425946
- Robert Smith – 20337801

Avenue/Problem Definition and Outline

For question 2 we decided to modify an existing colouring algorithm and investigate its performance over multiple different settings

These included adjusting the number of nodes in a graph, using different types of graphs and also different algorithm settings.

We then gathered resulting data from this regarding the performance of the algorithm, such as number of colours used, collisions detected and the relationship between different types of graphs and the algorithms performance.

Greedy Colouring Algorithm and modifications

The Greedy Colouring Algorithm is a simple colouring algorithm that iterates through nodes and assigns each new node with the smallest colour that has not been used by any of its neighbours.

We implemented a simple check to ensure the greedy colouring algorithm functioned optimally by assigning the new node with the lowest colour available that did not collide with any of its neighbours, for example a node with neighbours 4,4,7 it would verify the colour 1 was valid and then use this colour to populate the node.

This change allowed the greedy colouring algorithm to efficiently and consistently use lower numbers of colours to fill in these graphs when scaled.

Greedy Colouring Algorithm Settings

We decided to test this modified algorithm using different starting nodes and evaluating the performance of each of these settings on standardised graphs alongside a random control algorithm that iterated through the nodes randomly and assigned new nodes random colours if they did not collide with their neighbours.

These 3 algorithms were then tested on the same standardised graphs and their performances logged on each, allowing for easy comparison between our two modified algorithms and that of our control.

Graph Structures and Types

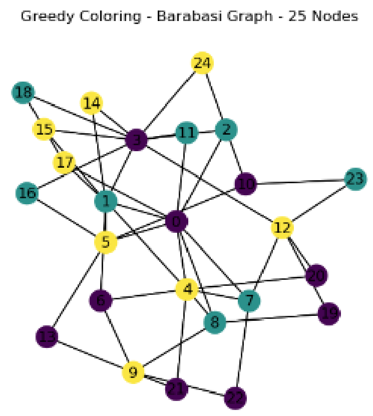
We tested the 3 algorithms across a total of 14 unique graphs with varying numbers of nodes.

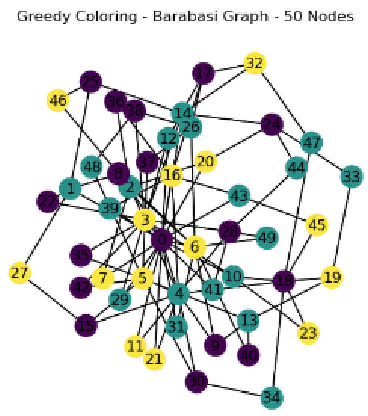
Our first type of graph was a Barabasi-Albert graph which generates random scale free networks which are like many real-world networks. The Barabasi-Albert graph uses a parameter m to connect each new node to m existing nodes when building the network. This would allow us to investigate how well each algorithm would work on a network similar to that of one in the real world.

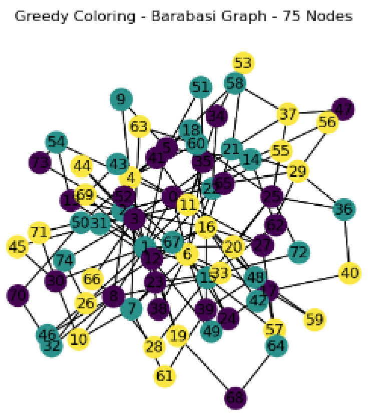
We also used Random graphs which created random network structures with a node connection possibility of p, meaning there was a p chance of each new node connecting to existing nodes. This would allow us to investigate the algorithms performance when dealing with random and unpredictable networks.

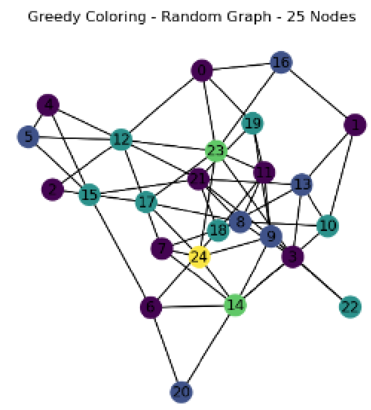
We also insured that parameters for each of these graphs were consistent to allow fair testing, this involved setting both graphs to have identical number of nodes and altering the edge generation parameters to ensure there were a reasonably similar number of edges in both types of graphs.

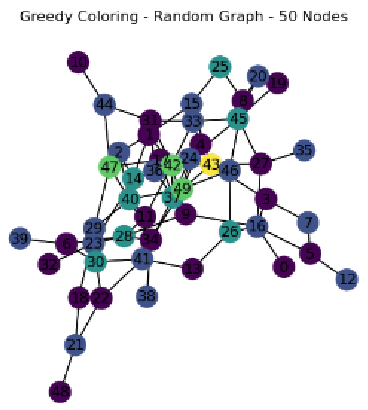
Our initial hypothesis for each of these graphs was that the algorithms would perform better on the Barabasi-Albert graphs due to their more structured style.

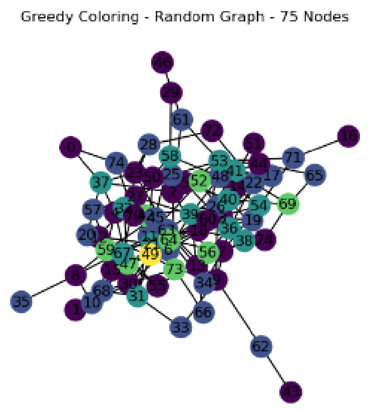










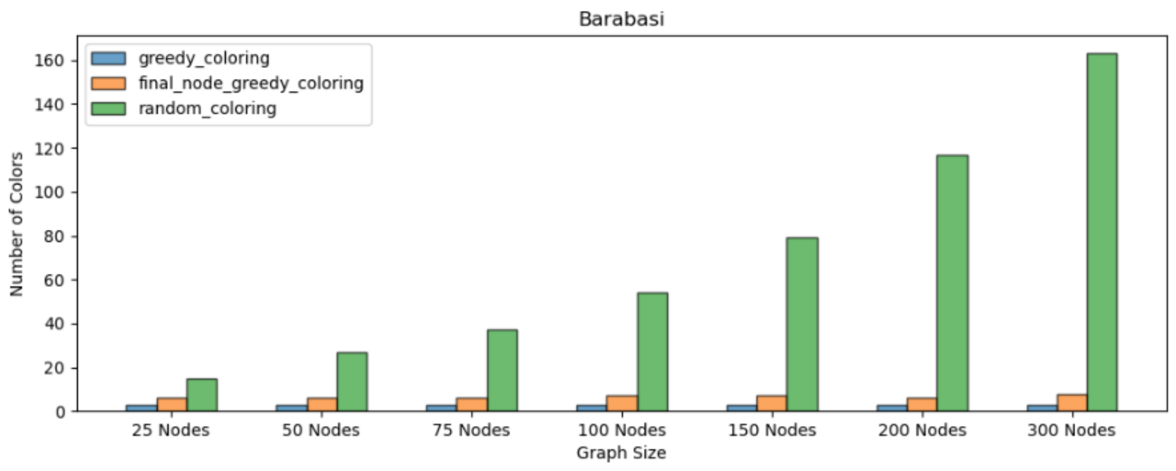


Colouring Functions Results – Colours used across networks of different sizes

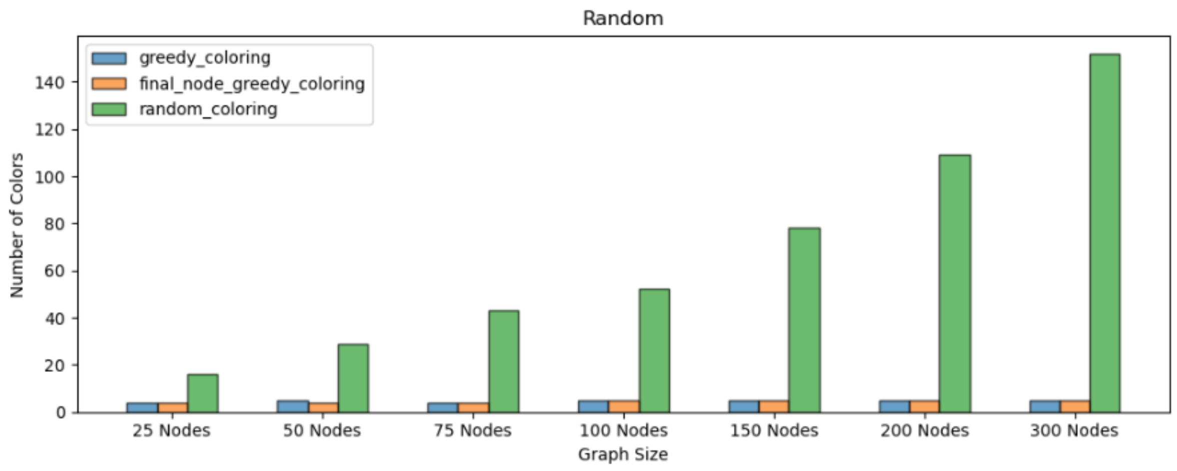
We then ran the algorithms across all the Barabasi-Albert and Random graphs and evaluated the results.

We ran these experiments on graphs of node sizes 25,50,75,100, 150, 200 and 300. As expected, the random algorithm performed the poorest on each graph, with the number of colours used directly correlating and increasing with the number of nodes. This occurred in both graph types.

We then ran both the Greedy Colouring Algorithm (GCA) that iterated increasingly through the nodes and also the Greedy Colouring Algorithm that iterated in reverse and retrieved results for these.



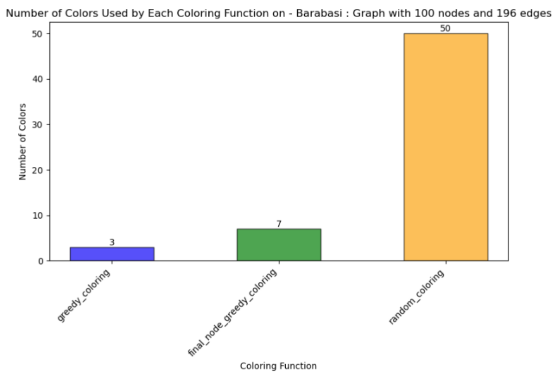
From the Barabasi-Albert results we can see that both GCAs performed similarly on the Barabasi-Albert type graphs, with the original GCA marginally outperforming the reversed across all network sizes. We can also see that both GCAs do not appear to have a direct correlation with number of colours used and number of nodes present. With a slim increase in the reversed GCAs colours used and a consistent number of colours used across all the normal GCAs graphs.

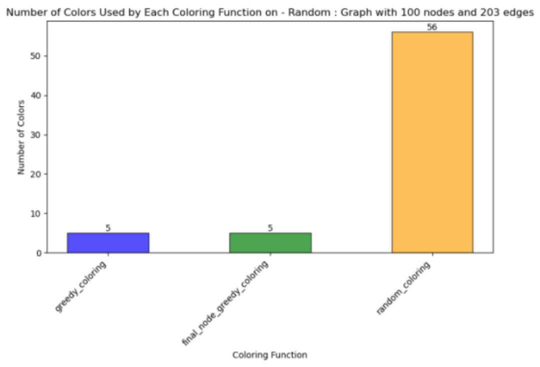


When observing the results from the Random graphs we can again see that the Random algorithm performed the poorest out of the 3 again a correlation between the number of colours used and the number of nodes in the network is evident in the random algorithms results

Again, both GCAs performed well and consistently with each other, however we can see that the original GCA performed worse on the randomly constructed graphs than it did on the Barabasi-Albert graph. The reversed GCA did not display any shift when compared to its performance on the Barabasi-Albert graphs.

We then collected data for a graph of each type with 100 nodes to be better able to identify the difference between all 3 algorithms on each graph.

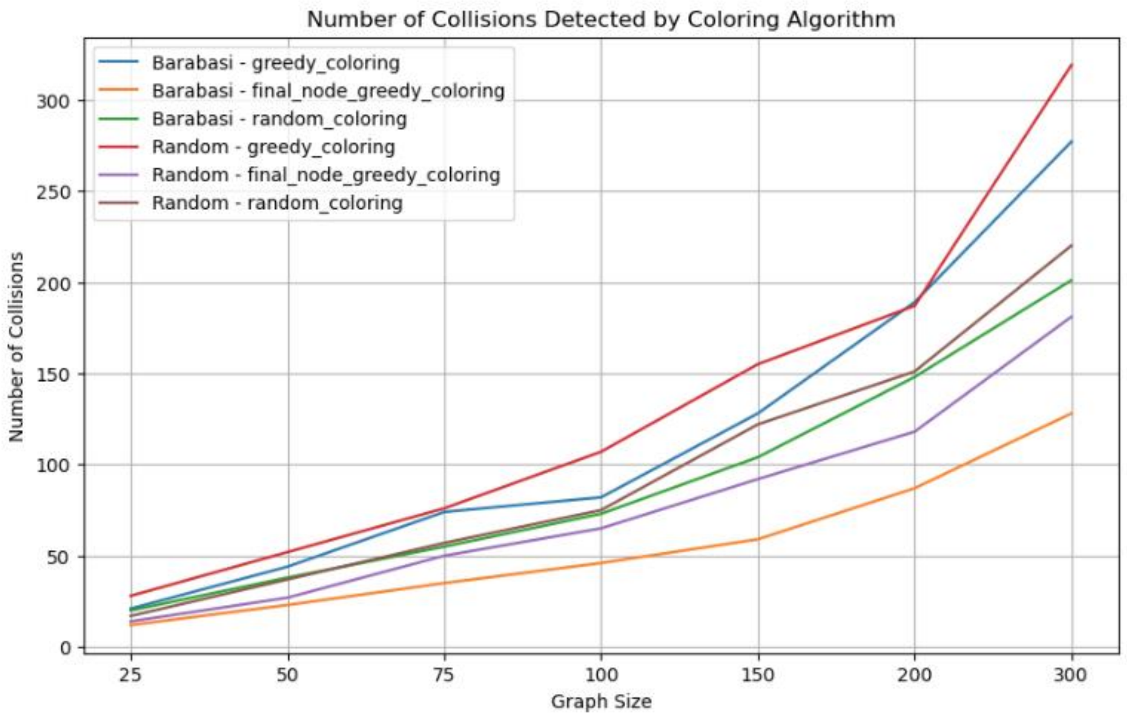




One potential explanation for the slightly poorer performance from the original GCA could be due to the randomness of the random graph, with each nodes number of edges being completely random, this contrasts with the more structured Barabasi-Albert graph.

Number of Collisions logged across graphs and different functions.

We also logged the number of collisions in each colouring function across all the different types of graphs and presented this as a line plot, allowing us to view the number of collisions in each algorithm and see how different graphs and node lengths affected the number of collisions.



We can see from this graph that final_node_greedy_coloring performed best on both graphs in this context returning lower collision numbers for both the Barabasi-Albert and Random graphs.

We can also see that the initial greedy colouring algorithm registered the most collisions on both graphs and that the random registered more consistent and average number of collisions across both graphs.

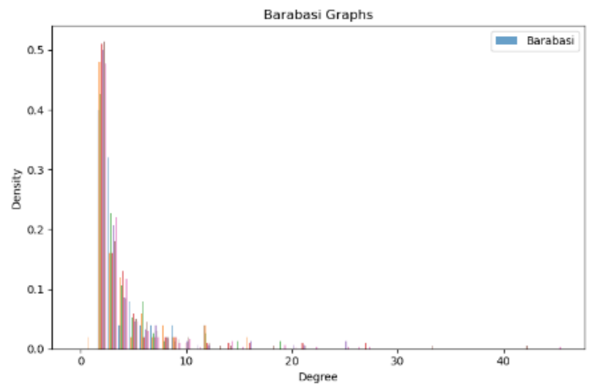
The number of collisions also correlates with the number of nodes in the graph with an increase in number of nodes leading to more collisions across all 6 instances of our experiment.

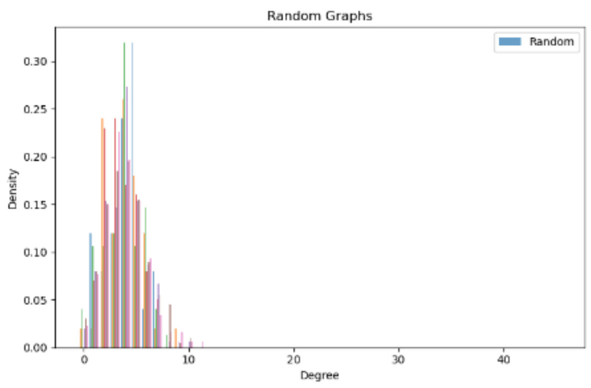
The Barabasi-Albert graph also registered less collisions when compared to the random graph, meaning the algorithms were better able to discover an official solution on the more structured graph as opposed to the random graph, this could be due to the distribution of nodes and their degrees.

Distribution of Nodes and their number of Degrees.

We investigated the differences between the 2 types of graphs and how they were constructed to better understand why the algorithms would perform better on the Barabasi compared to the random and also why we could observe more collisions in the random graphs than the Barabasi ones.

One explanation for this could be due to the distribution of nodes and their degrees in both graphs, with random graphs containing various nodes with many degrees compared to the Barabasi graph which followed a much more obvious pattern.





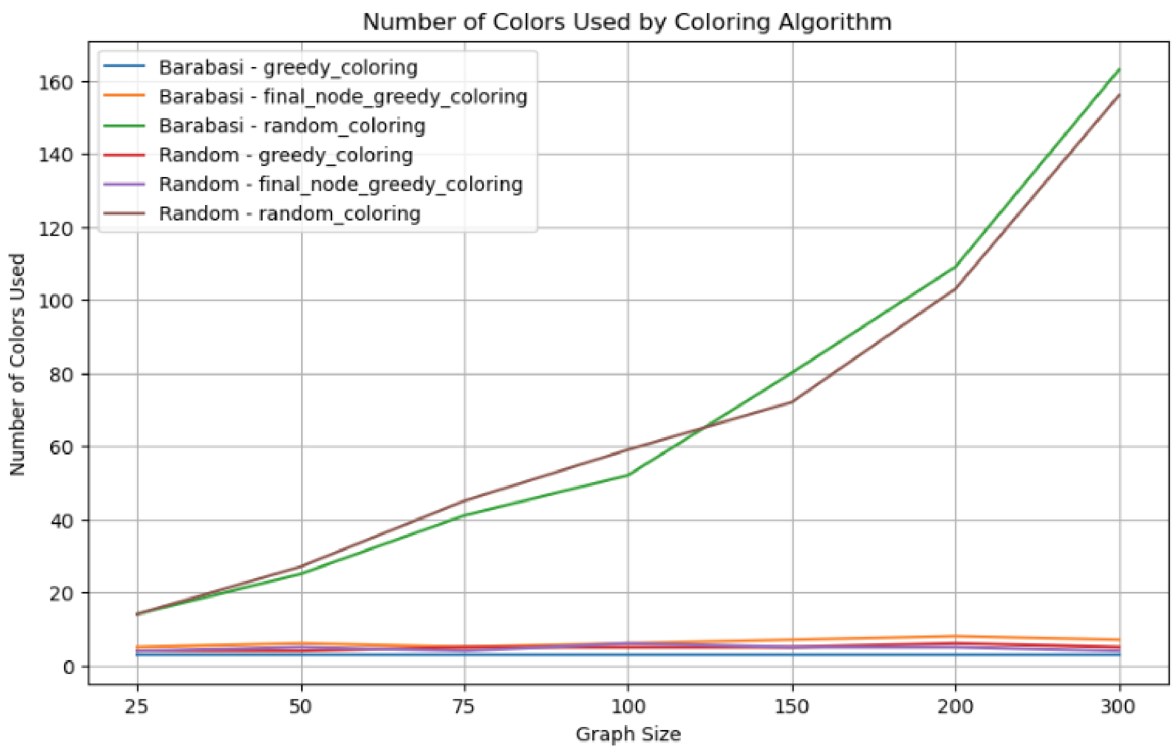
We can see from the above graphs that the random graphs tended to contain numerous nodes with fewer 10 connections yet these nodes all seemed to range between 4-8 connections each, this is in contrast to the Barabasi graph which contains a small number of nodes with high connections (hubs) and most other nodes fall in the region of having 2-3 connections each, this density in terms of the degrees leads to fewer colours being used by the GCA as the vast majority of the Barabasi graphs nodes have few neighbours allowing fewer colours to be used without conflicting. The random graph shows us that most nodes have more neighbours increasing the number of colours used by our algorithms.

Comparison of both GCA algorithms

Overall, it is fair to say that both of these algorithms performed consistently across all of our experiments. With the original GCA performing slightly better on Barabasi graphs and the final node GCA performing marginally better on the random graphs.

This suggests that altering the order in which the nodes are processed does not significantly affect our algorithms performance. However, depending on the graph used each algorithm could be more suited to one than the other. In terms of structured graphs, we see the original GCA outperform the reversed GCA and vice versa in the random graphs.

It is also evident that GCA algorithms outperform our random algorithm significantly with this random algorithm performing very poorly colouring about 66% of the total nodes used across both graphs. While our GCAs used consistently low numbers of colours with the number of nodes having little influence on the number of colours used



This is evident from this line plot of the number of colours used by all functions and graph types across each node size.

Conclusion

Overall when investigating the efficiency of our modified GCA we found it performed overwhelmingly better than a random colouring technique and tended to perform slightly better on average on structured graphs than random ones, however we also discovered that it was more than capable of performing to a high standard on random graphs of various sizes.

References:

- https://en.wikipedia.org/wiki/Barab%C3%A1si%E2%80%93Albert_model
- https://mathworld.wolfram.com/RandomGraph.html#:~:text=A%20random%20graph%20is%20a,edge%20probabilities%20distributed%20uniformly%20in%20.
- https://en.wikipedia.org/wiki/Greedy_coloring
- https://en.wikipedia.org/wiki/Graph_coloring
-