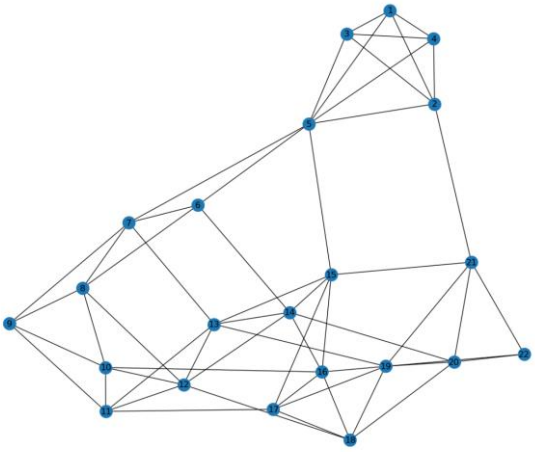
AI Assignment 2

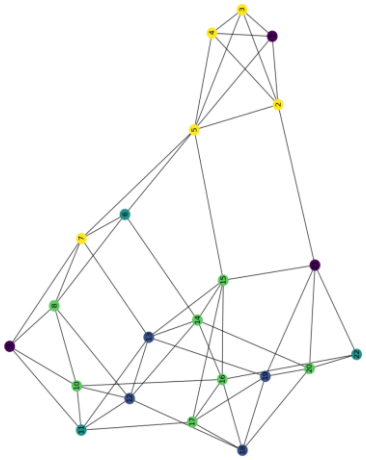
Question One – Correct Colouring of Graph with Conflict Detection

- Thomas Meehan – 20425946
- Robert Smith – 20337801

1.1 – Initially we were tasked with choosing a network topology that would be suitable for our experiment. We decided to choose a 22 node 55 edge undirected graph as we felt this could allow our algorithm ample exploration space to discover patterns and different ways of solving the colouring problem.



1.2 – We then randomly assigned colours to each node, ensuring that no more than 5 colours were used in the whole network, we were then able to view the network topology before applying our colouring solution. This allowed us to ensure that the colouring solution worked as intended.

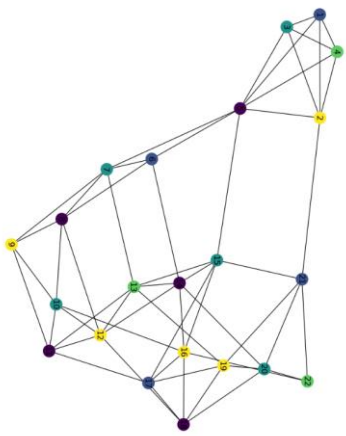


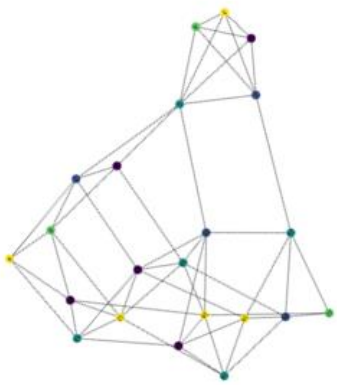
1.3 – We then defined our colouring solution which checked neighbouring nodes for colour conflicts and chose the next available colour if a conflict was present, if there was no conflict on the selected node, the colour remained unchanged. Seeing as there was only 5 colours for the algorithm to choose from this ensured the algorithm worked towards an optimum solution and didn't just use a different colour for each node.

1.4 – Following this we implemented the updateColors function inside a loop that ran for 1000 iterations updating one conflicting node per iteration, this allowed us to ensure the nodes were being changed as intended and allowed us a better insight into how the algorithm worked as opposed to just changing all conflicting nodes within a single iteration, we also printed each remaining conflict to console during each iteration showing us the algorithm in action and resolving one conflict per iteration. We also created another script that completed this using the same function within a single iteration.

```
Detected conflicts in iteration 1 : [1, 2, 3, 5, 6, 7, 9, 15, 16, 18, 20, 22]
Conflicts resolved in iteration 1 : 1
Detected conflicts in iteration 2 : [1, 2, 3, 7, 9, 15, 16, 18, 20, 22]
Conflicts resolved in iteration 2 : 1
Detected conflicts in iteration 3 : [1, 2, 3, 7, 9, 18, 20]
Conflicts resolved in iteration 3 : 1
Detected conflicts in iteration 4 : [1, 2, 3, 18, 20]
Conflicts resolved in iteration 4 : 1
Detected conflicts in iteration 5 : [2, 3, 18, 20]
Conflicts resolved in iteration 5 : 1
Detected conflicts in iteration 6 : [18, 20]
Conflicts resolved in iteration 6 : 1
Detected conflicts in iteration 7 : []
Conflicts resolved in iteration 7 : 0
```

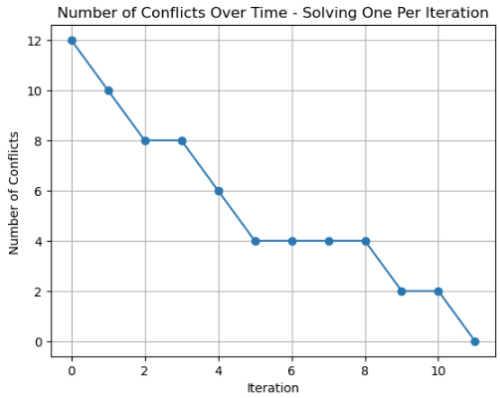
1.5 – Upon completion of all iterations and confirmation that no conflicts remained unresolved we were able to output the topology of the network after the colour correction had been applied, visually confirming that the algorithm functioned as intended and as its outputs per iteration suggested. Both the one conflict resolution per iteration and all conflicts resolved in one iteration scripts functioned as intended and output graphs that showed colour correction had been applied to both



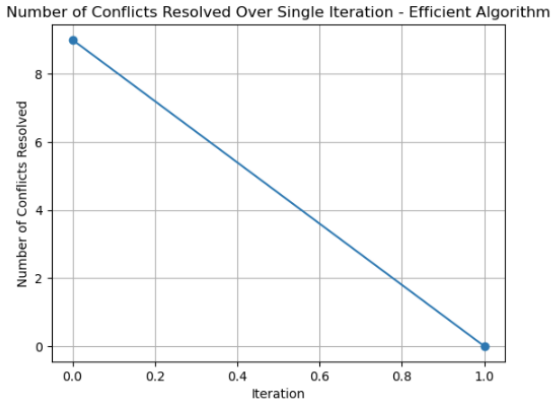


Output for 1 Resolution per iteration. *Output for Full Solution in 1 iteration.*

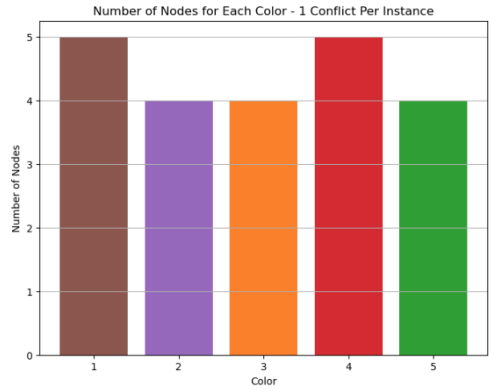
1.6 – Following this we were able to graph the performance of the colouring algorithm over its iterations, the version solving one conflict per iteration showed us that although it a conflict each iteration this also had the potential to replace one conflict with another as demonstrated by the graph below, this graph offers us greater insights into how the algorithm works as compared to the more efficient version which solves all conflicts in one iteration



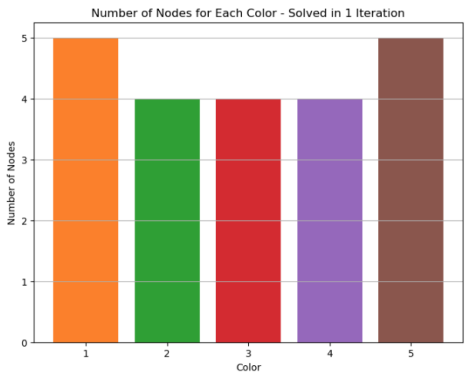
Algorithm Solving 1 Conflict Per Iteration



Algorithm Solving All Conflicts in 1 Iteration



Bar Chart Display of Distribution of colour across nodes - 1 Conflict per iteration.



Bar Chart Display of Distribution of colour across nodes – Solved in 1 iteration.

Summary Of Problem One findings

Overall, our findings for this problem gave us an insight into how the colouring algorithm, worked, by identifying conflicts and resolving them, repeating this until no more conflicts had been found, altering the number of colours used when finding the solution also showed us how the algorithm behaved differently when assigned more colours essentially being “lazy” and using non optimum numbers of colours. In this

case the optimum number of colours was 5 so setting the maximum number of colours that could be used as 5 allowed the algorithm to find an appropriate and optimum solution for the colouring problem.

References –

- https://www.geeksforgeeks.org/graph-coloring-applications/
- https://plotly.com/python/network-graphs/
- https://www.baeldung.com/cs/graphs-vertex-colouring
- https://www.geeksforgeeks.org/types-of-network-topology/
-