CMPT 310 Assignment 2 Report

Joshua Campbell

301266191

In this assignment, I constructed a backtracking program that can be used to solve a colouring for a graph.

Backtracking Implementation:

In order to solve the colouring for a graph, I utilized the Backtracking Search algorithm. This algorithm starts with a definition for a graph and attempts to colour the graph such that no adjacent vertices share the same colour. The algorithm accomplishes this by cycling through available colours and attempting to apply one of these colours to the current vertex. If the algorithm finds an inconsistent colouring (two adjacent vertices sharing the same colour), the algorithm will detect this and remove the last attempted colouring of a vertex. It will then attempt to colour the current vertex with a different colour. The algorithm will do this recursively until either a solution is found or all possible colourings of the vertices have been exhausted (the failure state).

Backtracking Heuristics Implementation:

Since this is an exhaustive search, this algorithm can be quite slow. In order to speed up the algorithm, I made use of the Minimum Remaining Values heuristic (using the Degree heuristic as a tie breaker) as well as the Least Constraining Value heuristics. The Minimum Remaining Value heuristic was implemented such that it would search through a list of unassigned vertices until it found the vertex with the fewest legal colours remaining (based on the colours used by its adjacent vertices. In the case of a tie, the Degree heuristic was used such that it would return the vertex with the most edges. For the Least Constraining Value heuristic, it was implemented such that it the algorithm would look at the current vertex's neighbours as well as the legal colours available to the current vertex. In order to choose order of the colours to use, the algorithm would compare the current vertex's legal colours against the legal colours of its adjacent vertices. Based on the legal colours of the current vertex, the algorithm would test each one against the vertex's neighbours, taking the sum of all of their remaining legal colours when a colour is used. Once this task was completed, the Least Constraining Value heuristic would return the current vertex's list of legal remaining colours ordered by the maximum number of legal remaining colours in all of its neighbours.

Program Testing:

When running the program on the provided graph and no heuristics enabled, I found that the program would solve the graph in roughly 0.000047 seconds. With the Minimum Remaining Value heuristic enabled, the program solved the graph in 0.000108 seconds. With both the Minimum Remaining Value heuristic and the Least Constraining Value heuristic enabled, the program solved the graph in 0.000135 seconds. As we can see from these run times, for smaller graphs such as the graph provided, the overhead searching that these heuristics need has a negative impact on how long it takes to solve a graph.

When running the program on a more complex graph, such as the queen8_12 graph (96 vertices, 1368 edges), with no heuristics enabled, the program solved the graph in 536.5 seconds. With the Minimum Remaining Value heuristic enabled, the program solved the graph in 0.41458 seconds. With both the Minimum Remaining Value heuristic and the Least Constraining Value heuristic enabled, the program would solve the graph in 0.04204 seconds. In the previous example (with the provided graph), the overhead generated by these heuristics resulted in the run time being longer than just exhaustively solving the graph. However, for this larger graph we can see that by using these heuristics the program was able to solve the graph exponentially faster.

Table of Run Times for Various Graphs:

	Asst2.data Graph	**Queen6_6.data Graph**	**Queen8_12.data Graph**	**Queen8_8.data Graph**
No Heuristics (only backtracking)	0.000047 seconds	2.3193 seconds	536.5 seconds	More than 5 minutes (manually aborted)
Backtracking + Minimum Remaining Value (and Degree Heuristic)	0.000108 seconds	0.07651 seconds	0.41458 seconds	More than 5 minutes (manually aborted)
Backtracking + Minimum Remaining Value (and Degree Heuristic) + Least Constraining Value	0.000135 seconds	0.10795 seconds	0.04204 seconds	8.631 seconds