CS4033 – Assignment 2 Analysis – Hybrid Sort

Rashi Loni, Nathaniel Poppe, Nailah Sall

The paths and cost from each algorithm from the following three cities to Bucharest: Oradea, Timisoara, Neamt.

```
?- bfs(oradea, Path, Cost).
Path = [oradea, sibiu, fagaras, bucharest],
Cost = 461 .

?- dfs(oradea, Path, Cost).
Path = [oradea, sibiu, fagaras, bucharest],
Cost = 461 .

?- astar(oradea, Path, Cost).
Path = [oradea, sibiu, rimnicu_vilcea, pitesti, bucharest],
Cost = 429.

?- bfs(timisoara, Path, Cost).
Path = [timisoara, arad, sibiu, fagaras, bucharest],
Cost = 568 .

?- dfs(timisoara, Path, Cost).
Path = [timisoara, lugoj, mehadia, drobeta, craiova, rimnicu_vilcea, pitesti, bucharest],
Cost = 720 .

?- astar(timisoara, Path, Cost).
Path = [timisoara, arad, sibiu, rimnicu_vilcea, pitesti, bucharest],
Cost = 536.

?- bfs(neamt, Path, Cost).
Path = [neamt, iasi, vaslui, urziceni, bucharest],
Cost = 406 .

?- dfs(neamt, Path, Cost).
Path = [neamt, iasi, vaslui, urziceni, bucharest],
Cost = 406 .

?- astar(neamt, Path, Cost).
Path = [neamt, iasi, vaslui, urziceni, bucharest],
Cost = 406.
```

Discussion of correctness and efficiency

The three algorithms, DFS, BFS, and A* found a path from the start city to the goal city Bucharest for each given case. If no path exists, the algorithms will return an empty path. In some cases, each algorithm found a different path, but the result was a valid continuous route to Bucharest. For example, when the start city is Neamt, all algorithms found the same path but for the start city of Timisoara, each algorithm found a different path. Each implementation of the algorithms resulted in a valid route using the Romanian road map data and the cost was also calculated correctly. Therefore, we can reasonably conclude that each algorithm has been implemented correctly.

The DFS, BFS, and A* algorithms deferred in terms of efficiency when finding the shortest path or the least expensive route. Among the cases we tested for, BFS tended to find the

shortest path with the least number of cities visited. This means it searched in a way that created the fewest edges. For example, when the start cities were Oradea and Timisoara, BFS found the path with the least number of cities visited as compared to DFS and A*. We also observed that A* tended to find the most cost-efficient and shortest distance path. A* uses the SLD heuristic and this is an advantage because it doesn't overestimate the cost. If any path exists it will find it making it complete. In the test case of Oradea, even though it visited more cities than BFS and DFS, A* found the path with the least cost. For BFS and DFS, the cost was 461 and A* resulted in a cost of 429. This means the A* algorithm tends to visit the least number of nodes as compared to DFS and BFS. DFS varied more with its performance. In some cases, it found an optimal path but in other cases it was highly inefficient and created a longer path with a much higher cost as compared to A* and DFS. It expands deeply first, which may cause it to go down the wrong branches.

When calculating the efficiency of an algorithm, we have to take into account the number of nodes visited and the total path cost. From our observations on the performance of the algorithms for the given test cases, we can reach certain conclusions about the efficiency of each algorithm. BFS explored the nodes for each level, which made it more reliable but can cause it to expand more nodes than is necessary to reach the goal city. We saw this in the results, as BFS guaranteed a shorter path in terms of the number of hops but not in terms of distance. The BFS algorithm can be a problem when it is used in larger graphs because although it goes level by level trying to find the minimum hops, it can get stuck and take a longer path than it needs to. DFS explored more deeply along a certain path, but this can cause it to go through longer and more inefficient routes. A* was the most efficient overall because it uses the path cost and the information provided by the SLD to Bucharest heuristic to make informed decisions about which nodes to visit. It visits fewer nodes and finds the path that takes the least distance traveled and to improve the algorithm, it would have a limit on the number of time it can revisit a city. A* has the least amount of nodes visited and the lowest total path cost. This proves that with the use of the SLD it aids in the efficiency of the algorithm. In conclusion, DFS is the least effective algorithm with the highest cost, BFS is the second most efficient coming a close second but has slightly more nodes visited, and in first place A* is the most efficient algorithm with the lowest cost and number of nodes visited and its use of the SLD heuristic.