COMP 304 ASSIGNMENT ONE – REPORT

2a) The Heuristic value is given by f(n) = g(n) + h(n), where g(n) is the time taken to get from the initial root state to the that particular state, and h(n) is the heuristic, in our case it is the number of people on the RHS of the Bridge(The RHS is the starting side, of which all persons are initially situated). A Priority Queue is used, so it always selects the smallest f(n) values amongst expanded states.

It is an A\* Search, so f(n) needs to be admissible, and never over-estimate the cost path. g(n) is already a factual value related to the total cost, so only h(n) would be of concern to us to find out if f(n) is admissible. If there are n number of people on the RHS of any particular state, and since it’s true that it’s going to take more than n moves to get everyone over the bridge, since a torch needs to be return back to the RHS for more people can cross. It is also true that a person time has to be a minimum of one or more. Therefore h(n) value will therefore always be n for n number of people, and the actual cost from that state to the goal is will be therefore guaranteed to be more than n. Therefore f(n) will always be less than the actual path cost, and is thus admissible.

java –jar DCP.jar

2b)

Below is a table of number of nodes that each search expanded for different input examples

|  |  |  |  |
| --- | --- | --- | --- |
|  | (3, 5, 9) with 17 | (1, 2, 5, 10) with 17 min time | (1, 2, 5, 10, 12) with 25 min time |
| Breadth First | C:\Users\Avron\Pictures\304\shortBFS.png | C:\Users\Avron\Pictures\304\1-2-5-10BFS.png | C:\Users\Avron\Pictures\304\Long BFS.png |
| Depth First | C:\Users\Avron\Pictures\304\shortDFS.png | C:\Users\Avron\Pictures\304\1-2-5-10DFS.png | C:\Users\Avron\Pictures\304\Long DFS.png |
| Greedy Best First | C:\Users\Avron\Pictures\304\ShortGreedy.png | C:\Users\Avron\Pictures\304\1-2-5-10Greedy.png | C:\Users\Avron\Pictures\304\Long Greedy.png |
| A\* | C:\Users\Avron\Pictures\304\shortAstar.png | C:\Users\Avron\Pictures\304\1-2-5-10AStar.png | C:\Users\Avron\Pictures\304\LongAStar.png |

A number of factors may affect performance, but most importantly when it comes to searches, more nodes visited, results in a decrease in performance, therefore we want to find a search that will deliver the goal with the least number of nodes visited. As evident in the above table, our problem search with most number of nodes visited is always the Breadth First Search, for all 3 examples, and the search with the least number of nodes visited is always the Greedy Best First, and the A\* search.

The reason why BFS visits the most number of nodes before reaching the goal state, is because the goal states are always the leaf nodes in the tree, and the nature of the BFS is to traverse the Tree level by level, therefore it the leaf nodes are visited last, thus BFS is not optimal performance wise.

With DFS, an optimal is dependent on where the goal state is situated. If it’s closer to the side that the DFS beginning to expand and visit, then the search will be highly optimal, else if the first goal state is on the opposite end from where the DFS beginning, then most nodes will be expanded and visited, before the goal state is found.

With Greedy Best & A\*, the performance is similar due to the number of nodes expanded being similar for this problem. The h(n) for Greedy Best Search is the same h(n) used to calculate f(n) for A\*, and g(n) is the actual cost from root to that particular node, therefore the number of nodes visited are similar values for both these searches. Therefore it can be concluded that the 2 Informed searches are more optimal performance wise.