Answers to Task 2 – C1803744

Question 1).

a). The largest number of nodes that need to be stored in the frontier at any point using breadth first search is b^d. This is because the standard worst-case space complexity for a breadth first search would is b^d.

b). The largest number of nodes that need to be stored in the frontier is b*d if using iterative deepening for the search problem.

c). The largest number of nodes that would be stored in both frontiers is $2b^{d/2}$. This is because in each frontier the largest number of nodes that would be stored is $b^{d/2}$ so therefore as there are two frontiers then it gets multiplied by 2.

Question 2).

The reason that it is not possible to use iterative deepening for both the forward and backwards search is because that the chances of them both having the same nodes in their frontiers is very low so they are very unlikely to have a non-empty intersection. This is due to the fact the iterative deepening searches only keep limited nodes in memory so the chances of both frontiers having the same node for an intersection is slim. The searches would more likely find the goal node or start node before they found each other so there would be no point in using it for bidirectional searches.

Question 3).

It is not guaranteed that this solution is optimal.

The fringe case: The fringe case for bidirectional uniform cost search is that the two uniform cost searches do not meet in the middle and both find paths too the start/goal. As by definition the uniform cost search will paths returned will be optimal, this intern makes the bidirectional search also optimal as it has two optimal paths.

The normal case: The solution might not optimal if we stop the search when the uniform cost searches have a non-empty intersection. To see why, if we take off the forward frontier node N(a) with the cost D and the node N(b) with the cost D+7. At the same time, we take off the backwards frontier node N(b) with cost J and N(a) with cost J+2. We then have two paths, one with cost D + (J + 2) and (D + 7) + J. As you can see the path (D+7)+J is bigger than the D+(J+2) but we have already removed the nodes from the frontiers meaning that the path is not optimal.

To make bidirectional search work with uniform cost search we would have to change the goal state of the bidirectional search algorithm to let the searches continue partially after a non-empty intersection.