The search algorithm will traverse the tree differently depending on the queuing function that it takes. We have implemented all the six algorithms.

In the search algorithm the initial node is the first node that gets popped from the queue. The node is then expanded using the expand method inside the HelpR2D2 class. The expand method returns an array of possible child nodes by applying all the possible operators on the given node. The way those nodes are inserted inside the queue depends on the search algorithm that will be used which defines how the tree will be traversed. Let me explain it one by one.

BFS:

In Breadth First Search the tree is traversed level by level. So we always enqueue the returned nodes from the expand method at the end of the queue so that the parents’ nodes are always explored first.

DFS:

In Depth First Search, It is quiet the opposite from BFS. The tree is traversed branch by branch. If a branch ends, the algorithm starts backtracking. Thus, the returned nodes from the expand method are added at the beginning of the queue.

IDS:

In Iterative Deepening, a limited depth first algorithm is run repeatedly and every time the depth is incremented by one till a solution is found. To do this, we implemented two methods; *IDS(Deque<Node> nodes, Node intialNode)* and *IDS(Deque<Node> nodes, Node node, int depth).* The first one takes the nodes queue and the initial node and keeps calling the second method inside a loop till a solution is found. The second method here is the limited depth first search algorithm. Each time this method runs till the depth becomes less than the child nodes depth.

UC:

When the child nodes are returned from expanding the parent node, the nodes are inserted inside the queue and then the queue is sorted by the path cost. In this problem, since all operations yields 1 as a cost, the uniform cost works the same as the breadth first search.

Greedy:

In a greedy algorithm, all the child nodes returned from expanding the parent node are instead in the queue and then sorted by their heuristic function. The algorithm takes an int n; when set to 0 it denotes to the first heuristic function and when set to 1 it denotes to the second one.

AStar:

AStar algorithm is very similar to the Greedy one, except that the nodes are sorted by the heuristic function added to the path cost of the node.