Name Weiyi Zhang Name Yiliang Liu

Student Number: V00868237 Student Number: V00869672

1. Consider a board of size 2x2

|  |  |
| --- | --- |
| 1 | 2 |
| 2 | 1 |

Now consider a board of size 3x3

|  |  |  |
| --- | --- | --- |
| 1, 6 | 3, 4 | 2, 5 |
| 4, 5 | 1, 2 | 3, 6 |
| 3, 2 | 6, 5 | 1, 4 |

In a board of size 4x4

|  |  |  |  |
| --- | --- | --- | --- |
| 1, 2, 3, 4, 5, 6 | 7, 8, 13, 14, 19, 20 | 9, 11, 15, 17, 21, 23 | 10, 12, 16, 18, 22, 24 |
| 7, 8, 9, 10, 11, 12 | 1, 2 ,15, 16, 21, 22 | 3, 5, 13, 18, 19, 24 | 4, 6, 14, 17, 20, 23 |
| 13, 14, 15, 16, 17, 18 | 3, 4, 9, 10, 23, 24 | 1, 6, 7, 12, 20, 22 | 2, 5, 8, 11, 19, 21 |
| 19, 20, 21, 22, 23, 24 | 5, 6, 11, 12, 17, 18 | 2, 4, 8, 10, 14, 16 | 1, 3, 7, 9, 13, 15 |

The size of S for n rooks on n x n board = n!

1. Despite Queen’s move, it can attack at most three squares in any given column. The first queen will have n choices, and the second has n-3 choices, and the third has n-6 choices, etc

Thus, the space size S>= n x (n-3) x (n-6) …

S3 >= (n x (n-3) x (n-6) …)3

S >= 3√ (n x (n-3) x (n-6) …)3

S >= 3√ (n x n x n x (n-3) x (n-3) x (n-3) x (n-6) x (n-6) x (n-6) …)

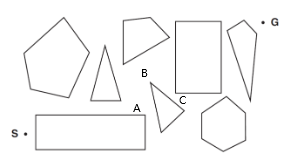
S >= 3√ 3

S >= 3√(n-3i)(n-3i-1)(n-3i-2)…

S >= 3√(n!)

1. There are infinite number of states and paths if we consider all positions (x, y)
2. The shortest distance between two points is a straight line. In this case, reaching from one polygon to another is connecting the vertex that is closest to each other. If there is a polygonal shape between two vertices, the shortest path is to warp the path around the obstacle while keeping it as a straight line.

In this case, AC is the shortest distance between two rectangles. However, the triangle is in between. The shortest path is to draw a straight to the vertex of the triangle B, then connect BC. Thus the shortest path is A-B-C



1. False. DFS only expands the first child node of the parent node until it reaches the end. Thus, it is possible that DFS can expands fewer nodes than A\* search
2. True. h\*(n) always >=0

H(n) = 0 <= h\*(n)

H(n) = 0 is admissible heuristics

1. True. BFS is complete at any step costs. It can find the goal as long as the goal exists.
2. False. The rook and the goal state are on the same column, and the Manhattan distance is 2. However, there are no obstacles between and the goal. Thus, Manhattan distance overestimate the number of move to reach the goal. Thus, it is not admissible heuristics

A\* Tree Search

Timisoara (g=0.0 h=329.0 f=329.0) order=0

Arad (g=118.0 h=366.0 f=484.0) order=3

Lugoj (g=111.0 h=244.0 f=355.0) order=1

Mehadia (g=181.0 h=241.0 f=422.0) order=2

Timisoara (g=222.0 h=329.0 f=551.0)

Drobeta (g=256.0 h=242.0 f=498.0) order=5

Lugoj (g=251.0 h=244.0 f=495.0) order=4

Zerind (g=193.0 h=374.0 f=567.0)

Sibiu (g=258.0 h=253.0 f=511.0) order=6

Timisoara (g=236.0 h=329.0 f=565.0)

Mehadia (g=321.0 h=241.0 f=562.0)

Timisoara (g=362.0 h=329.0 f=691.0)

Craiova (g=376.0 h=160.0 f=536.0) order=10

Mehadia (g=331.0 h=241.0 f=572.0)

Oradea (g=409.0 h=380.0 f=789.0)

Arad (g=398.0 h=366.0 f=764.0)

Fagaras (g=357.0 h=176.0 f=533.0) order=8

Rimnicu (g=338.0 h=193.0 f=531.0) order=7

Craiova (g=484.0 h=160.0 f=644.0)

Sibiu (g=418.0 h=253.0 f=671.0)

Pitesti (g=435.0 h=100.0 f=535.0) order=9

Bucharest (g=568.0 h=0.0 f=568.0)

Sibiu (g=456.0 h=253.0 f=709.0)

Craiova (g=573.0 h=160.0 f=733.0)

Bucharest (g=536.0 h=0.0 f=536.0)

Rimnicu (g=532.0 h=193.0 f=725.0)

Drobeta (g=496.0 h=242.0 f=738.0)

Rimnicu (g=522.0 h=193.0 f=715.0)

Pitesti (g=514.0 h=100.0 f=614.0)

A\* Graph Search

Timisoara (g=0.0 h=329.0 f=329.0) order=0

Arad (g=118.0 h=366.0 f=484.0) order=3

Lugoj (g=111.0 h=244.0 f=355.0) order=1

Mehadia (g=181.0 h=241.0 f=422.0) order=2

Timisoara (g=222.0 h=329.0 f=551.0)

Drobeta (g=256.0 h=242.0 f=498.0) order=4

Lugoj (g=251.0 h=244.0 f=495.0)

Zerind (g=193.0 h=374.0 f=567.0)

Sibiu (g=258.0 h=253.0 f=511.0) order=5

Timisoara (g=236.0 h=329.0 f=565.0)

Craiova (g=376.0 h=160.0 f=536.0) order=9

Mehadia (g=331.0 h=241.0 f=572.0)

Oradea (g=409.0 h=380.0 f=789.0)

Arad (g=398.0 h=366.0 f=764.0)

Fagaras (g=357.0 h=176.0 f=533.0) order=7

Rimnicu (g=338.0 h=193.0 f=531.0) order=6

Craiova (g=484.0 h=160.0 f=644.0)

Sibiu (g=418.0 h=253.0 f=671.0)

Pitesti (g=435.0 h=100.0 f=535.0) order=8

Bucharest (g=568.0 h=0.0 f=568.0)

Sibiu (g=456.0 h=253.0 f=709.0)

Craiova (g=573.0 h=160.0 f=733.0)

Bucharest (g=536.0 h=0.0 f=536.0)

Rimnicu (g=532.0 h=193.0 f=725.0)

Drobeta (g=496.0 h=242.0 f=738.0)

Rimnicu (g=522.0 h=193.0 f=715.0)

Pitesti (g=514.0 h=100.0 f=614.0)



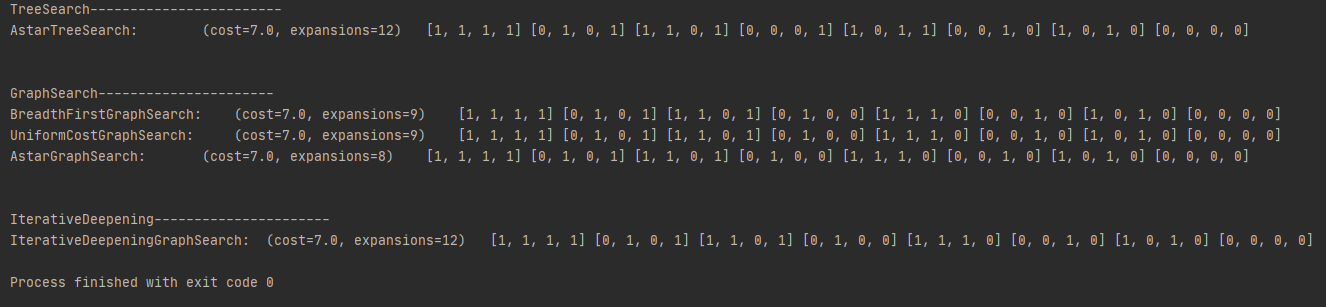
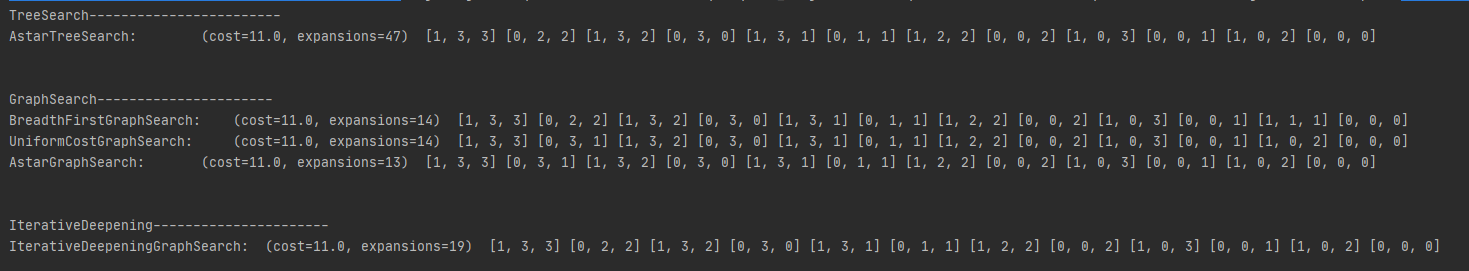
public String IterativeDeepeningTreeSearch() {

//*TODO*  
 int limit = 0;  
 while(true){  
 String result = TreeSearchDepthLimited(new FrontierLIFO(),limit);  
 if(result!=null) return result;  
 limit++;  
 }  
}

public String IterativeDeepeningGraphSearch() {  
 //*TODO* int limit = 0;  
 while(true){  
 String result = GraphSearchDepthLimited(new FrontierLIFO(),limit);  
 if(result!=null) return result;  
 limit++;  
 }  
  
}

private String TreeSearchDepthLimited(Frontier frontier, int limit) {  
 //*TODO* cnt=0;  
 initialNode = MakeNode(problem.initialState);  
 frontier.insert( initialNode );  
 while(true){  
 if(frontier.isEmpty()) return null;  
 Node node = frontier.remove();  
 if(problem.goal\_test(node.state)) return Solution(node);  
 if(node.depth < limit){  
 frontier.insertAll(Expand(node,problem));  
 }  
 cnt++;  
 }  
}

private String GraphSearchDepthLimited(Frontier frontier, int limit) {  
 //*TODO* cnt=0;  
 initialNode = MakeNode(problem.initialState);  
 frontier.insert( initialNode );  
 Set<Object> explored = new HashSet<Object>(); //empty set  
 while(true){  
 if(frontier.isEmpty()) return null;  
 Node node = frontier.remove();  
 if(problem.goal\_test(node.state)) return Solution(node);  
 if(!explored.contains(node.state)&&node.depth<limit){  
 explored.add(node.state);  
 frontier.insertAll(Expand(node,problem));  
 }  
 cnt++;  
 }  
}

2. 
3. 
4. 