ME 8710: Assignment 7

Question 1:

Q1)

A. DFS

3	2			10			14	15	16	17	18	19	20
4	(S) 1	7	8	9	11	12	13						21
5	6												22
									(G) 31	30	29	28	23
													24
											27	26	25

Number of Steps: 21

Total Number of Nodes: 31

B. BFS

6	2			10			17	21	26	30	34	38	42
5	(S) 1	3	7	9	11	13	15	18	22	27	31	35	39
8	4			12			19						43
				14			23		(G) 46				45
41				16					44				
37	33	29	25	20	24	28	32	36	40				

Number of Steps: 15

Total Number of Nodes: 46

C. A*

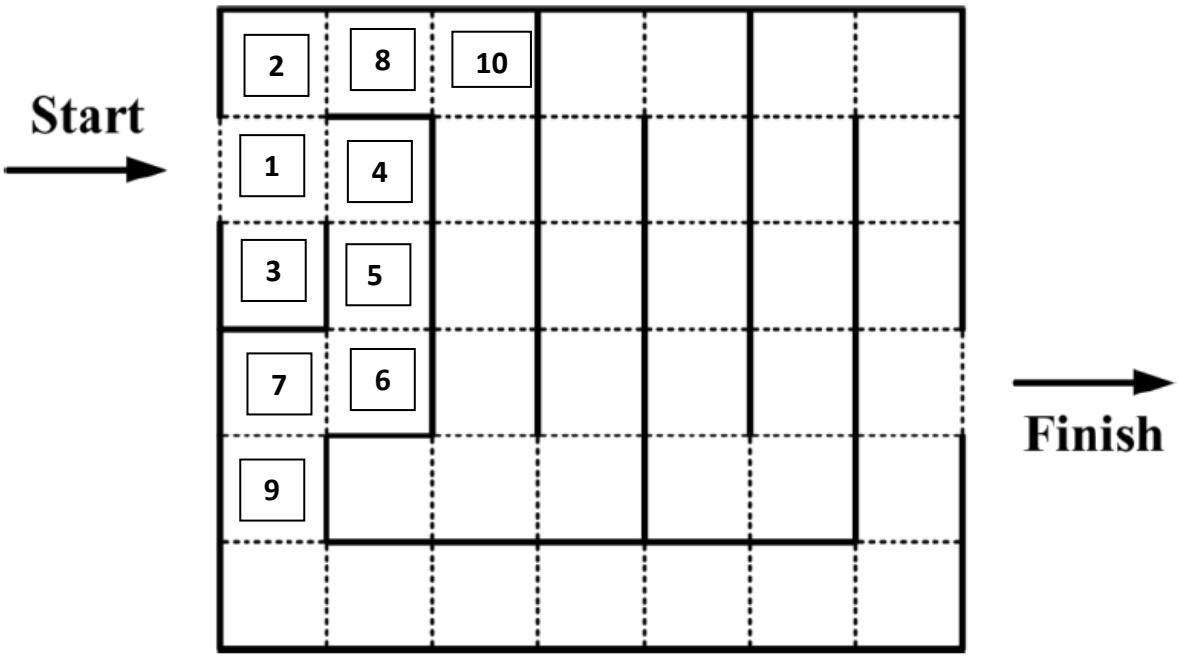
	2			9			16	19	22	26	30		
5	(S) 1	3	6	8	10	12	14	17	20	23	27	31	
7	4			11			18						
				13			21		(G) 37				
				15					36				
			29	25	28	32	33	34	35				

Number of Steps: 15

Total Number of Nodes: 37

node	g(n)	horizontal cells	vertical cells	h(n)	f(n)	
1	0	8	2	9.998	9.998	expand to node 2,3,4,5
2	1	8	3	10.997	11.997	expand to node 24
3	1	7	2	8.998	9.998	expand to node 6
4	1	8	1	8.999	9.999	expand to node 7
5	1	9	2	10.998	11.998	
6	2	6	2	7.998	9.998	expand to node 8
7	2	9	1	9.999	11.999	
8	3	5	2	6.998	9.998	expand to node 9,10,11
9	4	5	3	7.997	11.997	
10	4	4	2	5.998	9.998	expand to node 12
11	4	5	1	5.999	9.999	expand to node 13
12	5	3	2	4.998	9.998	expand to node 14
13	5	5	0	5.000	10.000	expand to node 15
14	6	2	2	3.998	9.998	expand to node 16,17,18
15	6	5	1	5.999	11.999	expand to node 25
16	7	2	3	4.997	11.997	
17	7	1	2	2.998	9.998	expand to node 19,20
18	7	2	1	2.999	9.999	expand to node 21
19	8	1	3	3.997	11.997	
20	8	0	2	1.998	9.998	expand to node 22,23
21	8	2	0	2.000	10.000	
22	9	0	3	2.997	11.997	expand to node 26
23	9	1	2	2.998	11.998	expand to node 27
24	2	9	3	11.997	13.997	
25	7	5	2	6.998	13.998	expand to node 28,29
26	10	1	3	3.997	13.997	expand to node 30
27	10	2	2	3.998	13.998	expand to node 31
28	8	4	2	5.998	13.998	expand to node 32
29	8	6	2	7.998	15.998	
30	11	2	3	4.997	15.997	
31	11	3	2	4.998	15.998	
32	9	3	2	4.998	13.998	expand to node 33
33	10	2	2	3.998	13.998	expand to node 34
34	11	1	2	2.998	13.998	expand to node 35
35	12	0	2	1.998	13.998	expand to node 36
36	13	0	1	0.999	13.999	expand to node 37
37	14	0	0	0.000	14.000	solution reached

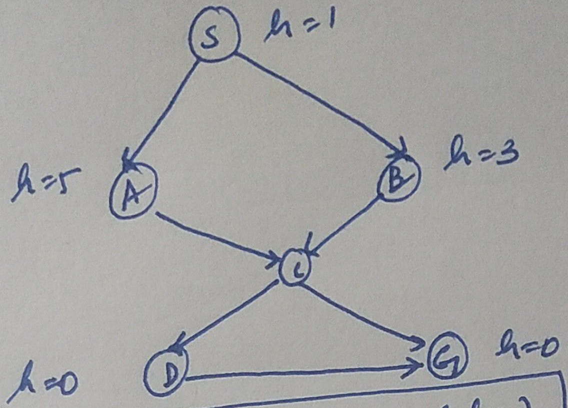
Q2)



a). A* will guarantee the solution for the above problem.

node	g(n)	horizontal cells	vertical cells	h(n)	f(n)	
1	0	6	2	8.000	8.000	expand to node 2,3,4
2	1	6	3	9.000	10.000	expand to node 8
3	1	6	1	7.000	8.000	x
4	1	5	2	7.000	8.000	expand to node 5
5	2	5	1	6.000	8.000	expand to node 6
6	3	5	0	5.000	8.000	expand to node 7
7	4	6	0	6.000	10.000	expand to node 9
8	2	5	3	8.000	10.000	expand to node 10
9	5	6	1	7.000	12.000	
10	3	4	3	7.000	10.000	

Ans 3: →



(S, f=1)
(B, f=5) (A, f=6)
(C, f=7) via (A, f=6)
C (f=6) via (B, f=5)
D (f=5), (G, f=7)
(G, f=6) via (D, f=5)

$$f(n) = g(n) + h(n)$$ ✓

Node	g(n)	h(n)	f(n)
S	0	1	1
A	1	5	6
B	2	3	5
C (via A)	5	2	7
C (via B)	4	2	6
G	7	0	7
D	5	0	5
G via D	6	0	6

so, the column g(n) represents the transition cost

Since, the evaluation funcn: —

$f = g + h$ We get: →

$g = f - h$

The optimal path is: →

(S → B → C → G) ✓

```
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
0  1  0  0  0  0  22 21 20 0  22 0  0  0  0  0  0  0  0  0
0  2  0  0  0  22 21 20 19 20 21 22 0  0  0  0  0  0  0  0
0  3  0  0  0  0  0  0  18 0  22 0  0  0  0  0  0  0  0  0
0  4  0  0  0  0  0  0  17 0  0  0  0  0  0  0  0  0  0  0
0  5  0  0  0  0  0  0  16 0  0  0  0  0  0  0  0  0  0  0
0  6  7  8  9  10 11 0  15 0  0  0  0  0  0  0  0  0  0  0
0  0  0  9  10 11 12 13 14 0  0  0  0  0  0  0  0  0  0  0
0  0  0  10 11 12 13 14 15 0  0  0  0  0  0  0  0  0  0  0
0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
[(2, 5), (2, 6), (2, 7), (2, 8), (3, 8), (4, 8), (5, 8), (6, 8), (7, 8), (7, 7), (7, 6), (6, 6), (6, 5), (6, 4), (6, 3), (6,
2), (6, 1), (5, 1), (4, 1), (3, 1), (2, 1), (1, 1)]
```

DFS Implementation Output

```
            # try new direction
            current_path.append((new_row, new_col))
            if dfs(current_path):  # recursive call
                return True
            else:
                current_path.pop()  # backtrack


# result a list of coordinates which should be taken in order to reach the goal
result = [(0, 0)]
if dfs(result):
    print("Success!")
    print(result)
else:
    print("Failure!")
```

BFS Implementation Output

```
Success!
[(0, 0), (0, 1), (0, 2), (0, 3), (1, 3), (2, 3), (2, 2), (3, 2), (3, 1), (3, 0), (4, 0), (5, 0), (5, 1), (6, 1), (6, 2), (6,
3), (6, 4), (5, 4), (4, 4), (3, 4), (3, 5), (2, 5), (1, 5), (0, 5), (0, 6), (0, 7), (1, 7), (2, 7), (3, 7), (4, 7), (5, 7), (6,
7), (7, 7)]
```

```
    path = search(maze,cost, start, end)
    print(path)
```

```
[[0, -1, -1, -1, -1, -1], [1, 2, 3, 4, 5, -1], [-1, -1, -1, -1, 6, 7], [-1, -1, -1, -1, -1, 8], [-1, -1, -1, -1, -1, 9]]
```

```
]: print('\n'.join([''.join(["{:" ">3d}".format(item) for item in row])
      for row in path]))
```

```
  0 -1 -1 -1 -1 -1
  1  2  3  4  5 -1
 -1 -1 -1 -1  6  7
 -1 -1 -1 -1 -1  8
 -1 -1 -1 -1 -1  9
```

A Star Implementation Output

```
bi_directional_search(input_1, 2, 4)
```

```
[2, 0, 1, 4]
```

Bi Directional Output