**Assignment 12**

**Name: DONGWOOK LEE**

**Problem 12.3** *Number Maze*

(c) **Bonus** (4 points)

```python
import random

INF = float('inf')


def node_edge_matrix(number_maze):
    return_adj_matrix = [[INF for col in range(len(number_maze)**2)]for row in range(len(number_maze)**2)]

    n = len(number_maze)
    m = len(return_adj_matrix)
    for i in range(0, n):
        for j in range(0, n):
            key = number_maze[i][j]
            return_adj_matrix[(n * i) + j][(n * i) + j] = 0
            if 0 <= i+key < n:                              # Down
                return_adj_matrix[(n * i) + j][(n * (i + key)) + j] = 1
            if 0 <= i-key < n:                              # Up
                return_adj_matrix[(n * i) + j][(n * (i - key)) + j] = 1
            if 0 <= j+key < n:                              # Right
                return_adj_matrix[(n * i) + j][(n * i) + (j+key)] = 1
            if 0 <= j-key < n:                              # Left
                return_adj_matrix[(n * i) + j][(n * i) + (j-key)] = 1

    return return_adj_matrix
```

```python
def number_maze_path(number_maze):
    adj_matrix = node_edge_matrix(number_maze)
    n = len(adj_matrix)
    start = 0
    target = n-1
    visited = list()

    my_matrix = [[0 for col in range(3)] for row in range(n)]
    for i in range(n):
        my_matrix[i][0] = adj_matrix[i][0]
        my_matrix[i][1] = INF
        my_matrix[i][2] = 0

    visiting_node = start
    next_node = 0
    my_matrix[visiting_node][1] = 0
    while True:
        visited.append(visiting_node)

        my_min = float('inf')
        for i in range(0, n):
            if i not in visited:
                if my_matrix[i][1] > my_matrix[visiting_node][1] + adj_matrix[visiting_node][i]:
                    my_matrix[i][1] = my_matrix[visiting_node][1] + adj_matrix[visiting_node][i]
                    my_matrix[i][2] = visiting_node

                if my_matrix[i][1] < my_min and my_matrix[i][0] != visiting_node:
                    my_min = my_matrix[i][1]
                    next_node = i
```

```
        if visiting_node == next_node:
            break
        else:
            visiting_node = next_node

    # print("Visited Tiles in order : ", end=' ')
    # for i in range(0, len(visited)):
    #     print(visited[i], end=' ')
    # print()

    # Now I need to back-track the path
    global path
    path = list()

    current_node = target
    while current_node != start:
        path.insert(0, current_node)
        current_node = my_matrix[current_node][2]
    path.insert(0, current_node)

    print("Path (", start, ", ", target, ") : ", end=' ')
    for i in range(0, len(path)):
        print(path[i], end=' ')
    print()

    if my_matrix[target][1] == INF:
        return -1
    else:
        return my_matrix[target][1]
```

**Test Case:**

```
4 1 1 4 4 1 1 5
3 3 7 6 6 4 1 7
6 7 4 6 3 4 7 1
1 3 4 4 2 5 7 2
1 7 7 2 3 6 7 4
4 6 1 6 4 6 7 5
5 1 2 1 5 1 2 1
1 2 3 1 5 4 2 1

Starting from [0] ...
0: UP
1: RIGHT
2: DOWN
3: LEFT
4: STOP and SOLVE
Current Position = 0 0
Direction: 4
Not Solved
Path ( 0 ,  63 ) :  0 63
-1
```

**No Solution**

```
3 3 6 7 4 3 3 7
7 4 3 7 3 4 4 4
3 1 7 2 4 2 2 4
7 2 1 7 5 5 4 7
6 5 7 3 1 2 2 6
6 3 3 5 7 3 6 5
3 2 7 4 5 2 3 5
3 2 5 1 5 6 5 2

Starting from [0] ...
0: UP
1: RIGHT
2: DOWN
3: LEFT
4: STOP and SOLVE
Current Position = 0 0
Direction: 4
Not Solved
Path ( 0 ,  63 ) :  0 3 59 58 63
4
```

**Solution Given**