## Name: Implementation of Iterative Deepening Search (IDS) algorithm to solve Weighted Maze search problem.

Description:

You will be given a 2D Maze/Grid as input in this problem. You will also be given a source/initial cell and a destination/goal cell. The maze’s each cell will contain a value that denotes the cost/risk that is accumulated by reaching that cell. **You need to write a program that will implement the idea of the Iterative deepening search algorithm to reach the destination cell from the source cell and calculate the cost that is accumulated in the whole path**. For example, if your starting cell is (1, 2) and you use cells (2,2) -> (2,3)-> (3,3)-> (4,3) to reach the goal cell (4,3), then summing up all the costs of these cells will accumulate the total cost of the path to reach the final cell from the goal cell.

During traversing from a particular cell (i,j) there can be only four possible moves (i+1, j), (i, j+1), (i, j-1), and (i-1, j) if it falls under the boundary of the grid. **Also, there exist priorities during the traversal**, which cells need to be discovered first from a particular cell, the order is,

1) (i+1, j),

2) (i, j+1)

3) (i, j-1), and

4) (i-1, j).

**1) indicates the most and 4) indicates the least priority**. Also, there are **some blocked cells** that can not be used. Such cells’ cost value will be **-1**.

In the first line, you will be given two integers M (1<=M<=40) and N (1<=N<=40) denoting the number of rows and columns of the given grid. In the following M lines, you will have N values denoting that particular row’s each cell’s cost value. So, ith (1<=i<= M) line will have j (1<=j<=N) values where each value c[i][j] (c[i][j] = -1/ blocked cell or 0<=c[i][j] <=100) represents the cost of cell (i,j). After this, you will get two sets of values (a, b) and (c,d) denoting the cell information of start and goal state or cell. Cells’ information will follow 0-based indexing. The leftmost and uppermost cell is (0, 0). Each line of input will have values as single-space-separated.

As output, in the first line, you will **print the cost** that will be calculated by applying the IDS algorithm and maintaining all the constraints. In the following line, you will print the number of cells that have been used to reach the goal state from the start state including both of them. Let the number be K, and **print K**. Then **in the following K lines, you will print the cells’ information** in the formation of row number and column number. The output of each cell will follow 0-based indexing. **If the solution or goal state can never be reached print only “-1”**. You can fix the maximum search level to 5000.

Please look at the input and output section for more clarification.

| Input | Output |
| --- | --- |
| 3 3  0 2 -1  -1 3 2  -1 1 3  0 0 2 2 | 9  5  0 0  0 1  1 1  2 1  2 2 |
| 4 4  1 2 0 -1  2 -1 1 2  1 2 -1 3  1 2 2 0  0 2 3 3 | 6  5  0 2  1 2  1 3  2 3  3 3 |
| 15 15  4 -1 0 6 -1 7 2 3 3 8 5 5 5 9 9  3 5 -1 0 -1 6 6 2 -1 4 2 3 2 6 1  8 4 2 0 4 -1 2 0 0 8 1 4 7 2 7  8 6 8 8 5 9 3 8 9 9 1 8 4 2 0  1 5 1 5 3 0 4 3 1 -1 0 -1 8 1 7  6 -1 4 -1 9 1 5 6 1 3 -1 9 7 8 0  7 9 1 -1 -1 1 -1 -1 8 2 5 1 3 3 1  6 9 5 2 6 9 4 1 5 8 -1 8 3 -1 3  0 5 3 8 8 6 3 7 5 4 -1 6 -1 8 1  -1 2 -1 -1 1 1 3 7 7 0 4 7 6 6 9  2 2 -1 3 2 7 9 7 7 3 1 3 8 -1 6  5 0 1 -1 2 7 5 7 5 6 0 0 2 0 4  5 1 5 7 9 0 4 7 6 6 3 8 1 8 1  8 9 7 6 5 0 3 3 3 3 3 4 4 5 9  6 9 7 4 1 6 1 3 6 3 5 6 1 8 8  0 5 13 12 | 102  21  0 5  1 5  1 6  2 6  3 6  4 6  5 6  5 7  5 8  6 8  7 8  8 8  9 8  10 8  11 8  11 9  11 10  12 10  12 11  13 11  13 12 |
| 20 23  1 4 4 3 -1 4 5 6 7 8 1 3 1 2 3 7 9 3 6 5 5 1 4  7 3 5 4 3 2 1 3 2 3 8 5 6 9 6 -1 8 9 1 9 4 -1 6  -1 7 6 7 4 6 8 2 7 -1 2 -1 8 6 5 1 4 6 5 9 4 3 -1  7 1 -1 1 9 8 6 7 9 4 9 4 8 1 4 6 -1 3 2 3 -1 6 6  2 9 6 6 -1 5 -1 6 6 7 5 3 5 -1 5 4 8 9 7 9 6 5 2  1 6 8 9 8 5 3 7 5 4 8 3 4 8 1 6 8 3 6 1 1 2 2  4 6 -1 -1 4 5 8 3 9 5 9 7 4 6 5 3 1 4 1 8 9 4 1  5 7 -1 1 -1 3 9 7 8 -1 1 9 5 3 5 -1 1 5 3 9 9 2 5  6 -1 3 5 6 2 9 7 1 9 1 5 -1 1 1 7 -1 8 9 1 6 -1 6  3 4 -1 7 2 4 6 2 8 8 3 4 6 5 -1 9 8 5 -1 9 9 4 4  -1 1 -1 8 6 2 3 7 2 -1 1 2 8 -1 7 9 9 5 1 8 8 -1 4  9 -1 5 8 6 5 6 2 8 6 -1 1 8 7 6 1 5 3 -1 6 3 8 -1  -1 7 6 4 -1 4 -1 2 2 5 6 5 2 6 7 6 3 2 7 6 8 8 8  8 6 7 -1 -1 6 -1 7 9 7 5 9 4 3 6 7 4 6 3 3 9 2 1  3 3 2 5 4 8 1 7 4 7 3 5 4 2 8 4 8 9 5 1 3 9 -1  6 2 6 8 2 3 2 6 4 4 9 9 -1 9 8 8 5 4 -1 9 3 -1 6  3 7 7 5 4 2 1 6 8 6 7 3 1 2 1 6 2 9 4 6 4 -1 7  6 8 1 3 4 9 3 4 6 2 5 6 3 1 9 7 3 2 5 3 3 4 7  7 7 7 9 6 -1 4 4 4 4 2 9 -1 6 9 8 5 5 3 6 8 9 1  -1 8 2 2 8 7 2 6 7 7 5 8 9 5 8 4 5 5 6 4 -1 6 8  1 10 18 11 | 96  19  1 10  2 10  3 10  4 10  5 10  6 10  7 10  8 10  9 10  10 10  10 11  11 11  12 11  13 11  14 11  15 11  16 11  17 11  18 11 |
| 3 3  1 -1 3  -1 -1 6  7 8 9  0 0 2 2 | -1 |

## Simulation

1st Test Case: **start Goal intermediate**

Cost: 0 + 2 + 3 + 1 + 3 = 9

Path: (0 0) (0 1) (1 1) (2 1) (2 2)

| 0 | 2 | -1 |
| --- | --- | --- |
| -1 | 3 | 2 |
| -1 | 1 | 3 |