Coding Assignment 5

ECS 122A Algorithm Design and Analysis

Finding a House in ByteLand

Problem Description

Note: This coding assignment includes many subtasks that admit much simpler solutions than the official full-score solution. Therefore, even if you cannot solve the full problem, you may still solve the subtask to get partial credit. See the subtasks section for information.

ByteLand just got redesigned. Now, it is a board consisting of $R \times C$ cells, organized in R rows and C columns. For instance, for a chess board, we have $R = C = 8$. The rows are indexed 1 through R, and the columns are indexed 1 through C. Cell (i, j) refers to the cell at the i-th row and the j-th column.

There is a kid ADA – some friends also call her "Algorithm-Design-Analysis" but, you know, it is a somewhat cumbersome name – who is located at cell $(1, 1)$. Her home is at position (R, C), and she really really wants to get home as soon as possible for her favorite show – believe it or not, finally, after the ages, the show is going to reveal a new chapter, called "The Winds of Winter"!!! It is instructive to think of cell $(1, 1)$ as the most north-west cell and of (R, C) as the most south-east cell.

From cell (i, j) ADA can move to cells $(i + 1, j)$, $(i - 1, j)$, $(i, j + 1)$ or $(i, j - 1)$, but there are some constraints, as is elaborated next.

Navigating through the redesigned ByteLand is not that easy. There are several restrictions and obstacles. An obvious restriction is that ADA can never leave ByteLand – okay, to be frank, she hypothetically can leave, but she does not want to because she says so! That is, at any time, she has to be on one of the $R \times C$ cells.

Buuuuut, that is not all. Some cells are easily passable, and it takes 1 second to reach them *regardless of the type of cell ADA is previously in.* We call these cells empty – as empty as ADA's hopes that any chapter of "A Dream of Spring" will ever be released.

Some cells have rocks, and at this point, ADA does not wear proper shoes to walk over, let alone climb, rocks. Some cells are muddy. For each muddy cell, it takes ADA between 2 and 5 seconds to reach them – interestingly enough, that number of seconds is always an integer.

Luckily, not each cell slows you down. There are "teleport" cells that have portals. Those portals allow her to teleport to a specifically chosen teleport cell in a second. More precisely, assume that ADA stepped on teleport cell (i, j). Then, she has one of the following five options:

- Do not use the teleport portal.
- Use the portal and, in 1 second, move to **teleport** cell (k, j) with the maximum possible k; i.e., move to the south-most teleport cell in column j.
- Use the portal and, in 1 second, move to **teleport** cell (k, j) with the minimum possible k; i.e., move to the north-most teleport cell in column j.
- Use the portal and, in 1 second, move to **teleport** cell (i, k) with the maximum possible k; i.e., move to the east-most teleport cell in row i.

- Use the portal and, in 1 second, move to **teleport** cell (i, k) with the minimum possible k; i.e., move to the west-most teleport cell in row i.

Weeeell, that's not all. There are also spiky cells. ADA's shoes have limited stamina. You know those are maybe not the highest-quality shoes she wears, as when she bought them, she had already overspent on her dress (;__;). So, she can walk over at most h spiky cells. If she steps on the $(h+1)^{th}$ spiky cell, her feet will start hurting so much that she will just sit down and keep crying for so long that she will miss her show. ADA takes only 1 second to reach a spiky cell.

If you help ADA find the fastest path, i.e., the path taking the least time, from $(1, 1)$ to (R, C), she will help you solve your other assignments ;). In particular, she just needs for each cell (i, j), the shortest amount of time needed to reach (i, j). She already knows how to use this information to construct all shortest paths and – perhaps more importantly – a random shortest path. All of the shortest path are equally important in her eyes, and she does not want to trust anyone (let alone your computer program) to be unbiased!

Input and Output

Input

Each input file contains several test instances. The first line contains a single integer I, representing the number of test instances. The description of the I instances follows.
Each test instance consists of multiple lines:

- The first line contains three space-separated integers R, C, h. R subsequent lines follow.
- The next R lines contain C space-separated integers denoting the types of the cells in row-major order. The j^{th} integer in the i^{th} row is the type of the cell (i, j). We denote it by $\mathsf{grid}(i, j)$.
- If $\mathsf{grid}(i, j) = 1$, then cell (i, j) is an empty cell,
- If $\mathsf{grid}(i, j) \in [2, 5]$ then it is a muddy cell that takes ADA $\mathsf{grid}(i, j)$ seconds to reach,
- If $\mathsf{grid}(i, j) = 0$, then it is a rocky cell,
- If $\mathsf{grid}(i, j) = -1$, then it is a teleport cell,
- If $\mathsf{grid}(i, j) = -2$, then it is a spiky cell.

Output

Compute result, an $R \times C$ matrix where $\mathsf{result}(i, j)$ denotes the minimum amount of time ADA needs to reach cell (i, j) from $(1, 1)$. If (i, j) **cannot** be reached from $(1, 1)$, then output `-1` instead. Print the matrix as R lines where each line contains C space separated integers. These integers should be the elements of the result matrix in row-major order.

Constraints

Each input file contains at most 11 test instances. In a single input file, the total number of cells across all test instances is at most $4 \cdot 10^6$. The time limit (for all test instances in a single file) is 3 seconds for C/C++ and 5 seconds for Python.

Each test instance satisfies the following additional constraints:

- Cells $(1, 1)$ is empty, i.e., $\mathsf{grid}(1, 1) = 1$
- $1 \leq R, C \leq 1000$
- $0 \leq h \leq 10$
- Sum of $R \cdot C$ across all instances is at most $4 \cdot 10^6$

Test Cases

Your program will be evaluated on 10 test files. Passing all the instances within a file will earn you 0.8 points. Note that if your program outputs an incorrect answer for even one of the instances within a file, you will receive 0 points for that file.

Subtasks

- (Subtask 1). In six test files, i.e., in 60% of test files, each cell is empty or rock. Moreover, in two of those test files, R and C are at most 10 for each test instance.
- (Subtask 2). In two other test files, each cell is empty, rock, muddy, or teleport. That is, there are no spiky cells in those two test files.
- (Subtask 3). In the remaining two test files, all types of cells are present.

Sample Input and Output

Note: In the samples below, extra spaces and new lines are added to improve readability. In the actual cases, there are no extra new lines between test cases or extra spaces in the grid output to make sure the cell values line up in a single column.

Sample Input 1:

```
6

2 5 0
1 0 1 1 1
1 1 1 0 1

2 5 0
1 5 1 1 1
1 1 1 5 1

2 5 1
1 -2  5  1  1
3 -2  1 -2  1

2 5 2
1 -2  5  1  1
3 -2  1 -2  1

2 5 0
1 2 -1 0 -1
1 1  1 5  1

3 5 0
1 2 -1 0 -1
5 5  5 0  0
5 5  5 5  1
```

Sample Output 1:

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

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

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

0 1 6 7 8
3 2 3 6 7

0 2 3 -1 4
1 2 3  8 5

0   2  3 -1  4
5   7  8 -1 -1
10 12 13 18 19
```

Sample Explanation

In the following, we provide explanations on the shortest path needed to get to the bottom-rightmost cell, i.e., (R, C).

Test Case 1: There is a unique path to every cell from $(1, 1)$ in this case. ADA is forced to avoid the rocky cells and so the unique path is represented in the output where i^{th} cell on the path has distance exactly $i - 1$ (with the first cell being $(1, 1)$).

Test Case 2: The optimal path to the bottom right cell is the same as the previous case since the rocky cells are replaced with muddy cells with high running time.

Test Case 3: Since ADA can only afford to travel to a single spiky cell, she is forced to travel via the first row in order to reach the bottom rightmost cell.

Test Case 4: This case is similar to Test Case 3, except that ADA can afford to travel two spiky cells. By using the bottom row, she can avoid the muddy cell with value 5 at $(2, 5)$.

Test Case 5: Note that we can teleport even if there is a rocky cell in between the teleport cells.

Test Case 6: The teleport cell cannot be used to avoid the muddy cells if we wish to reach $(3, 5)$.

Submission Guideline

Write your program in either C, C++, or Python **in a single file**. Submit the file on Gradescope. The time limit on Gradescope is 3 seconds for C/C++ and 5 seconds for Python. You can make at most 10 submission attempts. You may refer to `sample-cpp.cpp` or `sample-py.py` for sample code that takes input and writes output.