Challenge 5 - Dungeon Quest

You have been hired to implement a simple AI for a new action RPG game. The game designers have given you the following specfications:

The scenario of the game consists of a grid of **N x M** cells and your character can move forward, right or left but never go back to the previous cell (like the classic cell phone game, Snake). Your character moves at the speed of 1 cell per second. Once you beat the stage several gems appear on random cells in the scenario and disappear after **Z** seconds. There are 3 different types of gems with values of 1, 2 and 5. Your task is to implement an algorithm to find a path that maximizes the total value of gems collected in the **Z** seconds they remain visible.

For example, given the following 5 x 5 grid where the numbers indicate the value of the gems in the cells and the character **x** is the current position of your character:

```
2 - 2 - 2 - 0 - 0
|   |   |   |   |
0 - 2 - 2 - 0 - 0
|   |   |   |   |
0 - 0 - x - 5 - 0
|   |   |   |   |
0 - 0 - 0 - 5 - 0
|   |   |   |   |
0 - 0 - 0 - 0 - 0
```

A possible solution for a given time of 6 seconds would be:

```
2 - 2 - 2 - 0 - 0      2 - 2 - 2 - 0 - 0      2 - 2 - 2 - 0 - 0
|   |   |   |   |      |   |   |   |   |      |   |   |   |   |
0 - 2 - 2 - 0 - 0      0 - 2 - 2 - 0 - 0      0 - 2 - 2 - 0 - 0
|   |   |   |   |      |   |   |   |   |      |   |   |   |   |
0 - 0 - 0 - x - 0  =>  0 - 0 - 0 - 0 - 0  =>  0 - 0 - 0 - 0 - 0
|   |   |   |   |      |   |   |   |   |      |   |   |   |   |
0 - 0 - 0 - 5 - 0      0 - 0 - 0 - x - 0      0 - 0 - x - 0 - 0
|   |   |   |   |      |   |   |   |   |      |   |   |   |   |
0 - 0 - 0 - 0 - 0      0 - 0 - 0 - 0 - 0      0 - 0 - 0 - 0 - 0

2 - 2 - 2 - 0 - 0      2 - 2 - 2 - 0 - 0      2 - 2 - x - 0 - 0
|   |   |   |   |      |   |   |   |   |      |   |   |   |   |
0 - 2 - 2 - 0 - 0      0 - 2 - x - 0 - 0      0 - 2 - 0 - 0 - 0
|   |   |   |   |      |   |   |   |   |      |   |   |   |   |
0 - 0 - x - 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 - 0     0 - 0 - 0 - 0 - 0     0 - 0 - 0 - 0 - 0
```

With a final result of 14 in gem value

Input

The first line of the input is the number of test cases that follows. Each problem has 5 lines:

- **M, N**: width and height of the grid.
- **X, Y**: initial position on the grid.
- **Z**: number of seconds until the gems disappear.
- **G**: number of gems in the grid.
- **i0,j0,k0#i1,j1,k1#...#iT-1,kT-1,jT-1**: **iT, jT** are the coordinates of the Tth gem in the grid and **kT** is the value of the Tth gem.

Output:

The value of the gems collected for each test case, each one in a different line.

Sample input:

```
2
5,5
2,2
6
7
0,0,2#1,0,2#2,0,2#1,1,2#1,2,2#3,2,5#3,3,5
4,4
2,2
5
7
1,0,5#1,1,1#2,1,2#1,2,1#3,2,2#2,3,5#3,3,2
```

Sample output:

```
14
12
```

Limits:

- 3 ≤ **M** ≤ 100
- 3 ≤ **N** ≤ 100
- 2 ≤ **Z** ≤ 20

Submit & test your code

To test and submit code we provide a set of tools to help you. Download contest tools if you haven't already done that. You will then be able to test and submit your solution to this challenge with the challenge token.

```
Challenge token: dgNn8L3iJ6UAuoQfQrwS
```

To test your program

```
./test_challenge dgNn8L3iJ6UAuoQfQrwS path/program
```

A nice output will tell you if your program got the right solution or not. You can try as many times as you need.

To submit your program to the challenge

```
./submit_challenge dgNn8L3iJ6UAuoQfQrwS path/source_pkg.tgz path/program
```

Note that you first need to solve the test phase before submitting the code. During the submit phase, in some problems, we might give your program harder questions, so try to make your program failsafe.

Important: In this phase, you must provide the source code used to solve the challenge and, if necessary, a brief explanation of how you solved it.

Remember **you can only submit once!** Once your solution is submitted you won't be able to amend it to fix issues or make it faster, so please be sure your solution is finished before submitting it.

If you have any doubts, please check the info section.

Go ahead

I'm done! :)

Once you have submitted your code, hit refresh and continue to next challenge.

I'm stuck! :(

Be sure you follow the Tuenti Engineering twitter for updates and possible hints during the contest.

If this challenge is too hard and you are blocked, you will be able to skip it after two hours. Note that **you won't be able to complete it later**, and you have a limited number of challenges to skip.

Finally, if you run out of skips but are still really stuck with one problem, you will be able to skip it after 24 hours.

Challenge status:

Test case	Not done
Solution submitted	Not done
Skip	Skip this challenge :((You have 4 skips)

Refresh status

Tweet about this! #TuentiChallenge3

Share Follow @Tuentieng