Marathon Match

Problem Statement

Contest: 2013 TCO Marathon Round 1

Normal view

Problem: SnowCleaning

Problem Statement

IMPORTANT: This problem is used for two simultaneous matches: TCO'13 Marathon Round 1 and Marathon Match 79. You can compete in TCO'13 R1 only if you are eligible for TCO'13. You can compete in MM 79 if you are not eligible for TCO'13 or if you would like to skip TCO'13 R1 by some reason. Competing in both matches is *not allowed*. Doing so will lead to *disqualification*.

You've decided to start a snow cleaning business. You've signed a contract with your native city according to which your company is responsible for cleaning all snow in the city during the next 2,000 days.

The city can be represented as a rectangular board with **boardSize** rows and columns. Both rows and columns are numbered 0 to **boardSize**-1, where row 0 is the topmost row, row **boardSize**-1 is the bottommost row, column 0 is the leftmost column and column **boardSize**-1 is the rightmost column.

Each cell of the city may either be clean or it may contain snow. If a cell contains snow, its amount is not tracked and is irrelevant in this problem. Before the first day all cells are clean. Then, during each day there may be snowfalls in some cells. If a cell is clean and there is a snowfall in this cell, it will contain snow after the snowfall.

In order to clean snow, you can hire workers. You are allowed to have up to 100 workers. A worker can't be fired. When you hire a worker, you can choose in which cell of the city it should appear. The worker will appear in this cell and clean all snow from it by the end of the current day (if there is snow in the cell). During each of the next days, you can request your worker either to stay in his current cell or to move into one of 4 neighboring cells. Two cells are neighboring if they share a side. Once completing your request, the worker will clean all snow from his/her current cell (if there is snow in the cell). Note that if you request your worker to move, he/she will first move and then will clean snow, so if the worker's original cell contains snow, it will remain uncleaned. Multiple workers are allowed to share the same cell at any time.

In the end of each day, each your worker needs to get a salary of **salary**. Additionally, for each uncleaned cell the city will request you to pay a fine of **snowFine**. Your contract with the city assumes a fixed size reward for cleaning snow during all the 2,000 days. Therefore, if you want to maximize income, you need to minimize expenses. Your task in this problem is to operate the business so that the sum of paid salaries and fines is as small as possible.

Implementation

You will need to implement two methods: `init` and `nextDay`.

`init` will be called only once and before all `nextDay` calls. It serves to give your solution the values of **boardSize**, **salary** and **snowFine**. The return value from this method will be ignored.

`nextDay` will be called 2,000 times -- once for each day. Its input parameter **snowFalls** contains exactly 2*K elements where K is the number of snowfalls at this day. The elements are *Row*[0], *Col*[0], *Row*[1], *Col*[1], ..., *Row*[K-1], *Col*[K-1], respectively. Here *Row*[i] and *Col*[i] are the row and column of the cell where the i-th snowfall takes place. The cells are listed in row-major order (sorted in increasing order of rows, ties are breaked by increasing order of columns).

The return value from `nextDay` should be a list of commands that you issue for your workers. Each element must contain a single command. There are two types of commands:

- "`H <ROW> <COL>`". This command hires a new worker and places him at row `<ROW>` and column `<COL>`. Each worker gets

an ID depending when it was hired. Chronologically first hired worker has an ID of 0, second hired worker has an ID of 1, and so on. If several workers were hired during the same day, they get IDs depending on where the commands used to hire them are located in the return value (earlier standing command means lower ID).

- "M <ID> <DIR>". This command moves a worker with ID = <ID> using the direction specified by <DIR>. <DIR> is a single character 'U' (up), 'D' (down), 'L' (left) or 'R' (right). It is not possible to move a worker outside the board and to move worker more than once during the same day. If you hired a worker, you can't move it at the same day.

If you don't issue any command for some worker, he will stay in his current cell.

There are many different events happening during the same day. Here is the order in which they occur:

- First, some cells may experience snowfalls.
- Then, each your worker will execute his command.
- Finally, the salaries and fines will be updated according to the current number of workers and uncleaned cells.

Scoring

For each test case we will calculate your raw and normalized scores. If you were not able to operate the business successfully during all 2,000 days (due to time limit, memory limit, crash, invalid return value, etc.), then your raw score is -1 and the normalized score is 0. Otherwise, the raw score will be the sum of all salaries and fines you paid. The normalized score for each test is 1,000,000.0 * BEST / YOUR, where BEST is the lowest non-negative score currently obtained on this test case (i.e., considering the last submission from each competitor). Finally, your total score is equal to the arithmetic average of normalized scores on all test cases.

You can see your raw scores on each example test case when you are making an example submit. You can also see total scores of all competitors on provisional test set in the match standings. No other information about scores is available during the match.

Test case generation

Each test case generated using the same algorithm (but a different seed for random number generator). The algorithm is described in this section of the problem statement. For shorter explanation, unless otherwise specified, each occurrence of "chosen" means "chosen uniformly, at random".

boardSize is chosen between 20 and 50, inclusive. **salary** and **snowFine** are each (independently) chosen between 10 and 100, inclusive.

Snow is assumed to be produced by snow clouds. There can be different types of clouds. The number of types is chosen between 1 and 10, inclusive. A cloud type has the following parameters:

- *R* -- chosen between 1 and 3, inclusive. The cloud is basically a (2**R*+1)x(2**R*+1) square of cells. The cell in row *R*, column *R* is called cloud's center.
- *T* -- chosen between 10 and 25, inclusive. This is the amount of days the cloud remains active once it appeared.
- *GlobalP* (real value) -- chosen between 0.0 (inclusive) and 1.0 (exclusive). The meaning of this and subsequent parameters is explained below.
- *LocalP* (matrix of real values) -- each value is chosen (independently) between 0.0 (inclusive) and 1.0 (exclusive).
- *MoveP* (matrix of 4 integers). Each value is generated as *Ceil*(100**x***x*), where *Ceil* rounds a number up towards the nearest integer and *x* is a real number chosen between 0.0 (inclusive) and 1.0 (exclusive).

Each cloud of the given type works as follows during each day. With probability of (1 - GlobalP) it does not generate any snowfalls during the day. Assuming it does generate showfalls, let its center be currently at row *CenterRow* and column *CenterCol*. For each (*LocalRow*, *LocalCol*), 0 <= *LocalRow*, *LocalCol* <= 2**R*, if cell at row *CenterRow* + *LocalRow* - *R*,

column *CenterCol* + *LocalCol* - *R* exists within the city, a snowfall at this cell is generated with probability of *LocalP*[*LocalRow*][*LocalCol*]. In other words, *GlobalP* gives a global probability for a cloud to snow at a given day and *LocalP* gives probabilities for a cloud to snow at each particular cell around its center. Once all snowfalls are generated, the cloud's center will move into one of its 4 consecutive cells. The probabilities to move in each of 4 directions are proportional to *MoveP*. A cloud is allowed to move completely or partially outside the city and it's also allowed to return back to city after that (we assume here that a city is just a part of an infinite rectangular grid).

Once cloud types are generated for a test case, the total number of clouds to appear is chosen between 50 and 200, inclusive. Then each of these cloud is simulated. Its appearance day will be chosen between 0 and 1,999, inclusive, and its type will be chosen uniformly, at random. Then cloud's activity is simulated according to its type and cloud's behaviour algorithm described in the previous paragraph. We assume that day 1,999 is the last day when you operate the business and it is possible that a cloud will generate some snowfalls after this day. All such snowfalls can just be ignored. It is possible that two or more snow clouds will generate a snowfall at the same cell at the same day. This will still be reported as one snowfall to `nextDay` method.

Tools

An offline tester/visualizer is available. You can use it to test/debug your solution locally. You can also check its source code for exact implementation of test case generation and score calculation.

Definition

Class:	SnowCleaning
Method:	init
Parameters:	int, int, int
Returns:	int
Method signature:	int init(int boardSize, int salary, int snowFine)

Method:	nextDay
Parameters:	vector <int>
Returns:	vector <string>
Method signature:	vector <string> nextDay(vector <int> snowFalls)

(be sure your methods are public)

Notes

- The time limit is 20 seconds (this includes only the time spent in your code). The memory limit is 1024 megabytes.
- There is no explicit code size limit. The implicit source code size limit is around 1 MB (it is not advisable to submit codes of size close to that or larger). Once your code is compiled, the binary size should not exceed 1 MB.
- The compilation time limit is 30 seconds. You can find information about compilers that we use and compilation options here.
- There are 10 example test cases and 100 full submission (provisional) test cases.

Examples

0)

```
Board size = 47
```

```
Snow fine = 85
Salary = 54
Cloud types = 6
Snowfalls = 9752
```

1)

```
Board size = 22
Snow fine = 24
Salary = 50
Cloud types = 6
Snowfalls = 7509
```

2)

```
Board size = 30
Snow fine = 83
Salary = 29
Cloud types = 9
Snowfalls = 12426
```

3)

```
Board size = 39
Snow fine = 79
Salary = 83
Cloud types = 3
Snowfalls = 12213
```

4)

```
Board size = 22
Snow fine = 36
Salary = 59
Cloud types = 8
Snowfalls = 5139
```

5)

```
Board size = 37
Snow fine = 65
Salary = 36
Cloud types = 6
Snowfalls = 14471
```

6)

```
Board size = 23
```

```
Snow fine = 46
Salary = 54
Cloud types = 7
Snowfalls = 10470
```

7)

```
Board size = 41
Snow fine = 69
Salary = 30
Cloud types = 9
Snowfalls = 5758
```

8)

```
Board size = 37
Snow fine = 83
Salary = 74
Cloud types = 3
Snowfalls = 8396
```

9)

```
Board size = 45
Snow fine = 84
Salary = 41
Cloud types = 2
Snowfalls = 13858
```
