Marathon Match 126

Challenge TCO21 Marathon Match Recommended THRIVE Articles

Key Information

1st

$1

Unregister

Submit

Next Deadline: Review | 0s until current deadline ends

Show Deadlines

Challenge Overview

Important Links

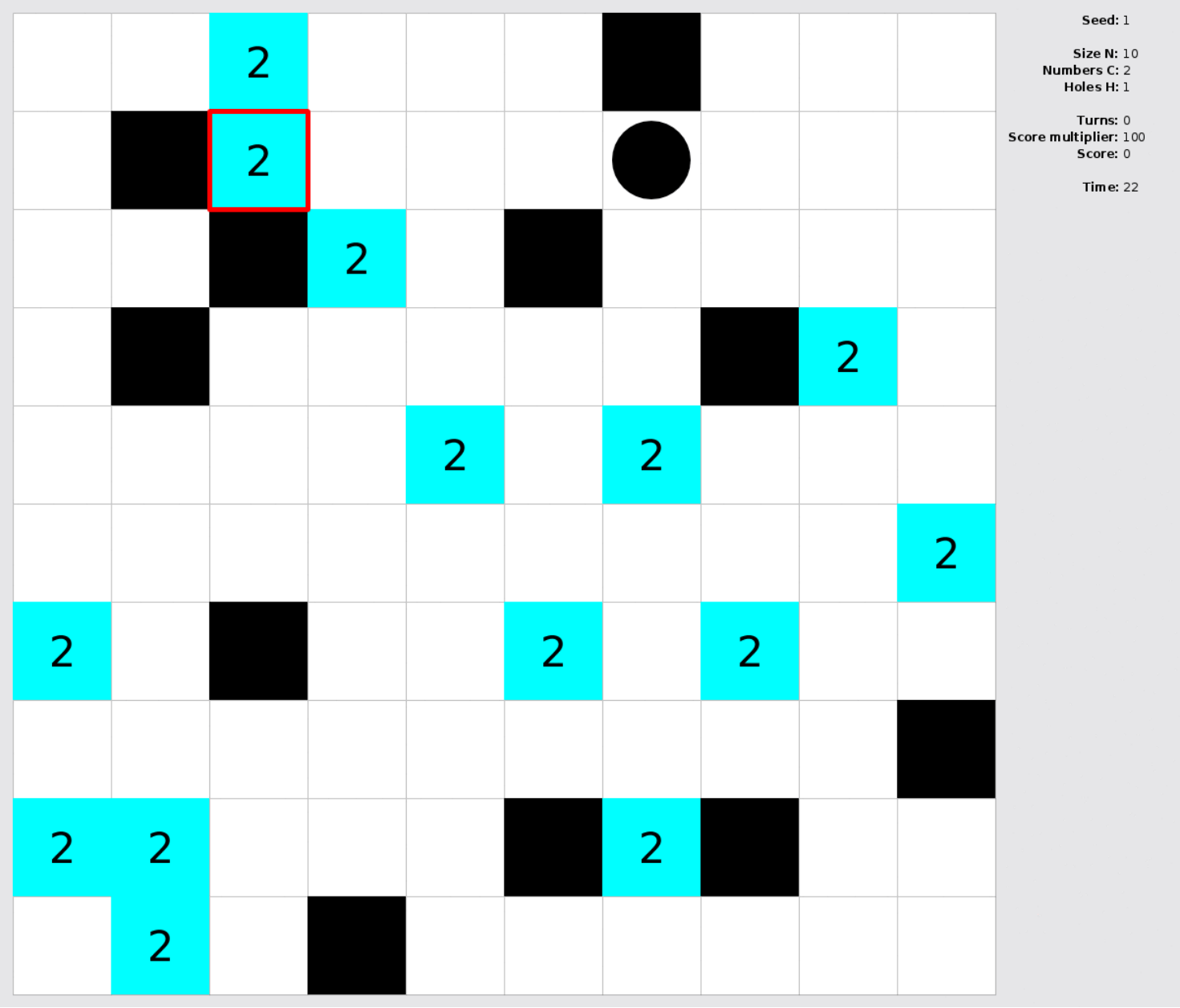
- Submission-Review You can find your submissions *artifacts* here. Artifacts will contain output.txt's for both example test cases and provisional test cases with stdout/stderr and individual test case scores.
- Other Details For other details like Processing Server Specifications, Submission Queue, Example Test Cases Execution.

Overview

Slider is a game played on a **N**x**N** grid. Each cell of the grid is either empty, contains a hole or contains a block with one of **C** colours. You play the game by moving and sliding the blocks into the holes. At the start of the game, a score multiplier **Z** begins with the value **N*N**. After each move or slide, **Z** decreases by one. Let **c** be the value of a block, you score **Z*(c-1)** points when the block disappears into a hole. You can move or slide a block up, down, left or right. A move (**M**) consists of moving the block in the specified direction l

cell. The block won't move if you try to move it onto another block or over the boundary. You can move a block into any hole. A slide (**S**) consists of sliding the block in the specified direction until it hits another block, the boundary of the grid, or falls into a hole. The game ends when the score multiplier **Z** reaches zero or when you decide to stop making moves. Your task is to score as many points as possible.

Here is an animation of a solution for seed=1



Input

Your code will receive as input the following values, each on a separate line:

- **N**, the size of the grid.
- **C**, the number of block colours.
- **H**, the number of holes.
- **N*N** lines describing the grid in row-major order. Each cell is either **0** (empty), **-1** (hole), or a number from **1** to **C**, representing the colour and value of the block in that cell. The first row is at the top of the grid.

Output

Your code should write to output the following:

- On the first line, **T**, the number of turns to perform.
- **T** lines, each representing a single turn. The format should be "r c type dir" (without the quotes), where **r** and **c** are the row and column coordinates of the block that you want to move or slide. These coordinates are 0-based. "type" should be **M** to move the block and **S** to slide the block. "dir" specifies the direction of the turn and must be one of **U** (up), **D** (down), **L** (left) or **R** (right).

Scoring

The raw score is the total score you achieved over all turns. Moving a block with value **c** into a hole, with the score multiplier **Z** at that turn, gives you **Z*(c-1)** points.

If your return was invalid, then your raw score on this test case will be -1. Possible reasons include:

- Not formatting the turns correctly.
- Using coordinates that are out of bounds.
- Trying to move empty cells or holes.
- Performing more than **N*N** allowed turns.
- Exceeding the time-limit

If your raw score for a test case is negative then your normalized score for that test case is 0. Otherwise, your normalized score for each test case is YOUR/MAX, where YOUR is your raw score and MAX is the largest positive raw score currently obtained on this test case (considering only the last submission from each competitor). Finally, the sum of all your test scores is normalized to 100.

Test Case Generation

Please look at the generate() method in visualizer's source code for the exact details about test case generation. Each test case is generated as follows:

- The grid size **N** is chosen between **10** and **30**, inclusive.
- The number of colours **C** is chosen between **2** and **9**, inclusive.
- The number of holes **H** is chosen between **1** and **10**, inclusive.
- The fill factor **F** is chosen between **0.2** and **0.8**, inclusive.
- The grid is initialized with **H** randomly placed holes.
- Each cell not containing a hole is filled with a randomly chosen block with the probability **F**.
- All values are chosen uniformly at random.

Notes

- The black blocks described by the value 1 do not score you any points because **Z*(1-1)=0**

- The time limit is 10 seconds per test case (this includes only the time spent in your code). The memory limit is 1024 megabytes.
- The compilation time limit is 30 seconds.
- There are 10 example test cases and **100** full submission (provisional) test cases. There will be 2000 test cases in the final testing.
- The match is rated.

Languages Supported

C#, Java, C++ and Python

Submission Format

Your submission must be a single ZIP file not larger than 500 MB, with your source code only.
Please Note: Please zip only the file. Do not put it inside a folder before zipping, you should directly zip the file.

Make sure you name your Source Code file as Slider.*<appropriate extension>*

SAMPLE SUBMISSIONS

Here are example solutions for different languages, modified to be executed with the visualizer. You may modify and submit these example solutions:

- Java Source Code - Slider.java
- C++ Source Code - Slider.cpp
- Python3.6 Source Code - Slider.py
- C# Source Code - Slider.cs

Tools

An offline tester is available below. You can use it to test/debug your solution locally. You can also check its source code for an exact implementation of test case generation and score calculation. You can also find links to useful information and sample solutions in several languages.

Downloads

- Visualizer Source - Slider_Source.zip
- Local Tester - tester.jar

Offline Tester / Visualizer

In order to use the offline tester/visualizer tool for testing your solution locally, you'll have to include in your solution the main method that interacts with the tester/visualizer via reading data from standard input and writing data to standard output.

To run the tester with your solution, you should run:

java -jar tester.jar -exec "<command>" -seed <seed>

Here, <command> is the command to execute your program, and <seed> is seed for test case generation. If your compiled solution is an executable file, the command will be the full path to it, for example, "C:\TopCoder\Slider.exe" or "~/topcoder/Slider". In case your compiled solution is to be run with the help of an interpreter, for example, if you program in Java, the command will be something like "java -cp C:\TopCoder Slider".

Additionally you can use the following options:

- **-seed** <seed> Sets the seed used for test case generation, default is seed 1.
- **-debug**. Print debug information.
- **-novis**. Turns off visualisation.
- **-delay** <delay>. Sets the delay (in milliseconds) between visualizing consecutive simulation steps, default is 200.
- **-fastMove**. Does not show intermediate steps when sliding blocks, so makes the animation much faster.
- **-pause**. Starts visualizer in paused mode. See more information below.
- **-manual**. Play the game manually using the mouse. Left click to select a block, and left click again on an empty cell to move or slide the block there. Clicking the same block again deselects it.
- **-N** <N> Sets a custom grid size.
- **-C** <C> Sets a custom number of block colours.
- **-H** <H> Sets a custom number of holes.
- **-F** <F> Sets a custom fill factor.

The visualizer works in two modes. In *regular* mode, steps are visualized one after another with a delay specified with the **-delay** parameter. In *paused* mode, the next move will be visualized only when you press any key. The space key can be used to switch between regular and paused modes. The default starting mode is regular. You can use the **-pause** parameter to start in paused mode. You can also play the game manually using the **-manual** parameter.

Marathon local testers have many useful options, including running a range of seeds with a single command, running more than one seed at time (multiple threads), controlling time limit, saving input/output/error and loading solution from a file. The usage of these options are described here.

Payments

Topcoder will compensate members in accordance with our standard payment policies, unless otherwise specified in this challenge. For information on payment policies, setting up your profile to receive payments, and general payment questions, please refer to Payment Policies and Instructions.