



Marathon Match 122

[Challenge](#)[TC021](#)[Algorithm](#)[Marathon Match](#)[Recommended THRIVE Articles](#)

Key Information

1st

\$0

2nd

\$0

[Unregister](#)[Submit](#)

The challenge is finished.

[Show Deadlines](#) [DETAILS](#)[REGISTRANTS \(162\)](#)[SUBMISSIONS \(872\)](#)[MY SUBMISSIONS \(0\)](#)[WINNERS \(2\)](#)[CHALLENGE FORUM](#)

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.

ELIGIBLE EVENTS:

[2021 Topcoder\(R\) Open](#)

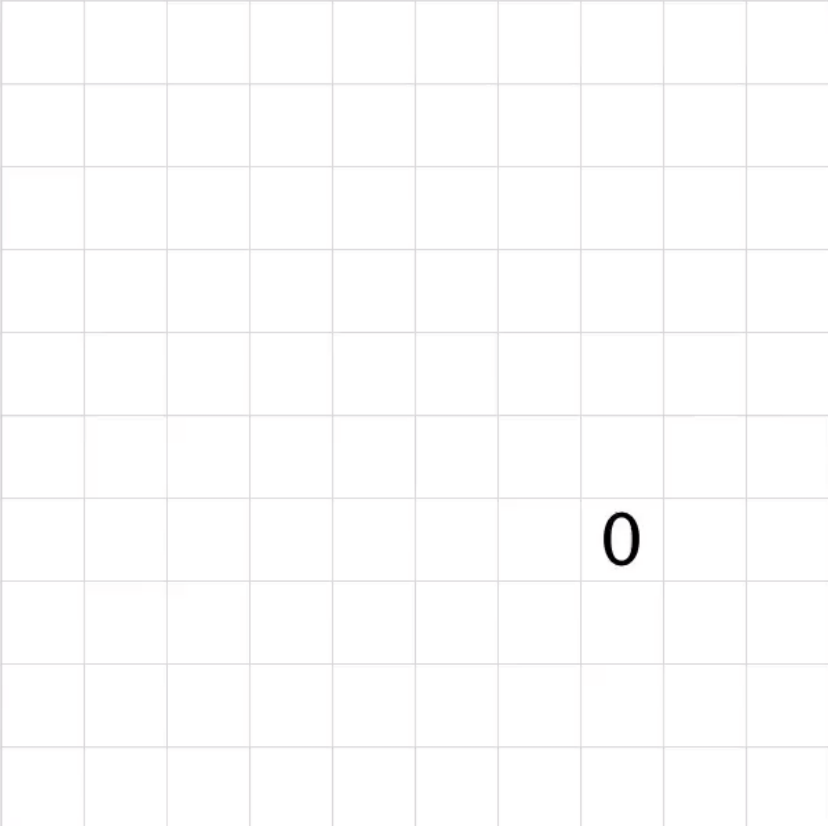
CHALLENGE LINKS:

[How To Compete in a Marathon Match](#)**CHALLENGE TF**[Standard Terms](#)[Competitions v2.2](#)**Support**

Overview

Super Minesweeper is a game where you must locate **M** mines that have been hidden somewhere on a **NxN** grid. Each cell of the grid is either a mine or a value. A value in cell X represents the number of mines in cells Y whose squared Euclidean distance from cell X is **D** or less. The game begins with all cells hidden except for a single cell with a zero whose location is provided. In each move you can uncover one cell of the grid. Unlike the classic Minesweeper, the game continues when you uncover (hit) a mine, but you lose points. Your task is to uncover as many values as possible, while minimizing the number of mines you have hit. In particular, your raw score is the percentage of values uncovered divided by the number of mines you have hit plus one.

Here is an animation of a solution for seed=1.



Seed: 1

Size N: 10
Mines M: 10
Distance D: 2

Flagged: –
Guessed: –
Mines Hit: –
Score: –

Time: –

TOOLBOX:

[Topcoder Extension](#) ?
[for VSCode](#)

SHARE:



ID: 30156479

Input and Output

This is an interactive problem, so your code needs to interact with the tester for each move. Initially your code will receive as input the following values, each on a separate line:

- **N**, the size of the grid.
- **M**, the number of mines in the grid.
- **D**, the distance threshold.
- A line containing the location of the initial zero, formatted as "row column" (without the quotes). Coordinates are 0-based.

The following game loop repeats until the game terminates, which occurs when you either uncover all the values or when you have decided to stop (see the stop command):

- Your solution sends commands by writing a line to output. After each command you will receive feedback from the tester, which you can use to update the state in your solution. There are three types of commands: stop, flagging or guessing command.
- The stop command is performed by writing the line "STOP" (without the quotes). This will terminate the game and you will receive your current score.
- The flagging command is performed by writing the line "F row column" (without the quotes), where coordinates are 0-based. This will place a flag in the corresponding cell. This command is used for visualisation purposes only and has no consequences for your score. The tester will send you an empty line as confirmation that the command was performed.
- The guessing command is performed by writing the line "G row column" (without the quotes), where coordinates are 0-based. There are two possible outcomes:
 - If you have hit a mine then the tester will send you a line formatted as "BOOM! runtime" (without the quotes), where runtime is the total time spent in your solution in milliseconds.
 - Otherwise you have uncovered a value and the tester will send you a line formatted as "value runtime" (without the quotes).

Scoring

The raw score is the percentage of values you have uncovered (relative to the total non-mine cells)

divided by the number of mines you have hit plus one. Thus the maximum score for a test case is achieved when you have uncovered all values without hitting any mines.

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

- Not formatting the commands correctly.
- Using coordinates that are out of bounds.
- Uncovering cells that have already been used.

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 **50**, inclusive.
- The number of mines **M** is chosen between **$0.1 \cdot N \cdot N$** and **$0.3 \cdot N \cdot N$** , inclusive. The location of mines is chosen at random.
- The distance threshold **D** is chosen from the set **$\{1, 2, 4, 5, 8, 9, 10\}$** .
- The location of the initial zero is chosen at random. Each grid is guaranteed to have at least one such zero.
- All values are chosen uniformly at random.

Notes

- The squared Euclidean distance between cells (x_1, y_1) and (x_2, y_2) is $(x_1 - x_2)^2 + (y_1 - y_2)^2$.
- 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 SuperMinesweeper.<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 - [SuperMinesweeper.java](#)
- C++ Source Code - [SuperMinesweeper.cpp](#)
- Python3.6 Source Code - [SuperMinesweeper.py](#)
- C# Source Code - [SuperMinesweeper.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

- [Tester source code](#)
- [Tester executable](#)

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\SuperMinesweeper.exe" or "~/topcoder/SuperMinesweeper".

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 SuperMinesweeper".

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.
- **-pause**. Starts visualizer in paused mode. See more information below.
- **-manual**. Play the game manually using the mouse. Left click uncovers cells and right click toggles a flag on/off.
- **-N** <N> Sets a custom grid size.
- **-M** <M> Sets a custom number of mines.
- **-D** <D> Sets a custom distance threshold.

The visualizer works in two modes. In *regular* mode, steps are visualized one after another with a delay