



## Marathon Match 124

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

### Key Information

1st  
\$02nd  
\$03rd  
\$0[Unregister](#)[Submit](#)

The challenge is finished.

[Show Deadlines](#) [DETAILS](#)[REGISTRANTS \(166\)](#)[SUBMISSIONS \(507\)](#)[MY SUBMISSIONS \(8\)](#)[SUBMISSION REVIEW](#)[WINNERS \(3\)](#)[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

#### ELIGIBLE EVENTS:

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

#### CHALLENGE LINKS:

[How To Compete in a Marathon Match](#)[Support](#)

#### CHALLENGE TERMS:

## Overview

You are a treasure hunter exploring a cave. The cave can be described as a simple undirected graph, composed of chambers (nodes) which may contain one or more treasures, and paths (edges) which connect those chambers. Unfortunately, because the cave is dark and there is little in the way of identifying information, you have become lost.

You know ahead of time how many total chambers are in the cave system (**numChambers**), but that is all you know. For any given chamber, you know how many paths leave that chamber, and how many treasures are in the chamber. While in a chamber, you can optionally take any number of the treasures that are in that chamber, up to a given maximum per step, before selecting a path to take to another chamber.

At any given time, you may choose to discontinue exploring, immediately warping out of the cave system with whatever treasures you have gathered so far.

Treasures have a value, **treasureValue**, describing how much a single treasure is worth. Exploring the cave takes time and energy, described in **stepCost**. The maximum number of treasures you may collect in any given step is described in **maxTreasurePickup**.

For each step, you are given **numTreasures** and **numPaths**, describing how many treasures are in the current chamber, and how many paths lead away from the chamber. You should then return the number of treasures you wish to take, or -1 to end exploring. Then, you should return a value 0..numPaths-1 indicating the index of the path to take next, where -1 means return to the chamber you were in previously.

## Implementation

Your code should read parameters from stdin, and write output to stdout, as outlined in the following pseudo-code:

### TOOLBOX:

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

### SHARE:



ID: 30169660

- **Initialization:**

```
ReadLine(treasureValue)
ReadLine(stepCost)
ReadLine(numChambers)
ReadLine(maxTreasurePickup)
```

- **For each step:**

```
ReadLine(numTreasures)
ReadLine(numPaths)
ReadLine(elapsedTimeInMs)
WriteLine(treasuresToTake)
WriteLine(nextPathIndex)
```

## Scoring

Your raw score for a test case will be calculated as:

**treasureValue** \* #treasuresCollected - **stepCost** \* #stepsTaken

Returning any invalid values, generating an error, or exceeding the time limit, will result in scoring -1 for that test case. If the calculated score is negative, you will score a 0 for that test case.

Your overall score across all test cases will be calculated as the sum of your relative scores on each test case where you received a positive score. Your relative score is calculated as YOURS / BEST, where YOURS is your score for that case, and BEST is the largest positive score anyone achieved on that case. Finally, the sum of all your test scores is normalized to 100.

## Test Case Generation

- **treasureValue** is selected between 10 and 100, inclusive, uniformly at random.
- **stepCost** is selected between 1 and **treasureValue**, inclusive, uniformly at random.

- A maximum number of chambers (**maxChambers**) is selected between 10 and 100, inclusive, uniformly at random.
- **maxTreasurePickup** is selected between 1 and 10, inclusive, uniformly at random.
- The number of treasures in each chamber is chosen between 0 and 50, inclusive, uniformly at random.

The graph is generated as follows:

- Initially queue a single chamber for generation.
- While there are chambers queued for generation, pull the next off the queue, and generate it if it has not already been generated.
- Generating a chamber consists of:
  - Select how many treasures are in the chamber.
  - Randomly connect to 2-4 other chambers, some of which may have already been generated.
  - Queue the connected chambers for generation.

## Notes

- The graph describing the cave is a single component, that is to say that every node is reachable from every other node.
- There are no nodes of degree 1.
- You may choose to take no treasure from a chamber.
- You may make a maximum of 1,000 steps.
- The graph will have a maximum of 100 nodes.
- The number of chambers actually present may be less than the selected **maxChambers**. ???

## 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**

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 `LostTreasureHunter.<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 - [LostTreasureHunter.java](#)
- C++ Source Code - [LostTreasureHunter.cpp](#)
- Python3.6 Source Code - [LostTreasureHunter.py](#)
- C# Source Code - [LostTreasureHunter.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 - [LostTreasureHunter\\_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\LostTreasureHunter.exe" or "~/topcoder/LostTreasureHunter".

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 LostTreasureHunter".

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.

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. 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](#).

## Recommended THRIVE Articles

[Explore THRIVE](#)

20 min read

### Power up C++ with the Standa..

 Mar 29, 2021

20 min read

## Power up C++ with the Standa..

📅 Mar 29, 2021

5 min

## List of awesome learning res..

📅 Mar 29, 2021



### COMPETE

All Challenges  
Competitive  
Programming  
Gig Work  
Practice

### TRACKS

Competitive  
Programming  
Data Science  
Design  
Development  
QA

### COMMUNITY

Blog  
Challenge Pipeline  
Events Calendar  
Forums  
Programs  
Statistics  
TCO  
Thrive

### HELP CENTER

Getting Paid  
FAQ  
General Info  
Website Help

### ABOUT

Admins  
Contact Us  
About Community  
Changelog  
Talk to Sales

### FOLLOW US



