ALGORITMI PARALELI ȘI DISTRIBUIȚI
Homework #2 Space Explorers and Federation HQ

Termen de predare: 24-Nov-2019 23:55

Responsabili Temă: **Cristian Chilipirea, Radu Ciobanu**

Objectives

Let there be a galaxy with **space explorers** and **Federation headquarters (HQs)**. The goal of the space explorers is to explore all the wormholes so that they have a complete view of the galaxy. The wormholes connect two star systems together, but entering them requires a perfect decoding of their sub-space-mambo-jumbonian frequencies. Once a wormhole is explored, the federation sends a special science vessel through, which, if it survives, scans the new solar system and provides the data about the next wormholes and the star systems they connect.

The Federation has multiple HQs, but they can be busy with other tasks and, as such, they are not always available. When a Federation HQ is available, it takes the decoded solution from the space explorers and sends the science vessel. If the decoding is wrong when the science vessel tries to enter the worm-hole, a great explosion destroys the galaxy, kills everyone, ends the program, and the student is given 0 points.

The solution should be implemented in **Java** using **threads**. You are only responsible for implementing the space explorers and the subspace communication channel between them and the Federation HQs.

Space Explorers and the Sub-space-mambo-jumbonian Frequencies

Each frequency decoding starts with data from the federation HQ, on which the science vessel uses an analyzatron and computes the frequency as a seemingly random string. The space explorer then needs to decode the sub-space-mambo-jumbonian frequencies, in order to enter a wormhole. To do this, the space explorer needs to compute the hash of the hash of the hash of ... of the hash of the frequency[1]. The number of times to repeat the hash operation is fixed and given as a parameter to the simulation.

The space explorers start the simulation at a given set of star systems which can be reached with boring slower-than-the-speed-of-light drives. Then, they get the list of wormholes from the federation HQs. Each wormhole has a frequency represented as a string sequence attached to it. An explorer chooses a wormhole and explores it by decoding the frequency and then sending the result to the Federation HQs, which verify its correctness by sending the science vessel.

The space explorers need to be implemented by you in the SpaceExplorer.java file found in the skeleton. You need to implement the constructor provided and you can use the given methods to decode the sub-space-mambo-jumbonian frequencies.

[1] A real-life distributed system that uses lots and lots of hash computations is Bitcoin. In Bitcoin, the processes that calculate all the hashes are called miners.

The Galaxy

The galaxy is composed of many solar systems. One solar system can be connected through wormholes with any number of other solar systems, whereas a few well-known solar systems can be accessed with a boring slower-than-the-speed-of-light drive. There is no guarantee that any system can be reached from any other system using this mesh of worm-holes. However, there is a guarantee that all interesting solar systems can be reached if all wormholes are used. You can imagine the galaxy as a graph, where the solar systems are the vertices and the wormholes are the edges that connect these vertices. The well-known solar systems are the vertices of the graph that can be accessed at the start of the simulation. The space explorers need to navigate through the wormholes and decode their frequencies in order to discover the entire galaxy.

It is possible to know where a wormhole will lead before it is decoded, so a science vessel can go through. However, in order to find the other wormholes from the new solar system, the science vessel needs to go there.

Federation HQs

Federation HQs are responsible not only for space exploration, but they each have a multitude of other tasks and experiments to complete. Because everyone at the Federation HQ is super important and super busy, it is possible to have large time periods when they simply do not answer. In this new society, bureaucracy has reached new, previously undreamed-of levels, so it is not clear when the HQs stop answering and when they start again.

When a Federation HQ does answer, it receives a wormhole's decoded sub-space-mambo-jumbonian frequency from a space explorer, along with the names of the two solar systems that are connected by said wormhole (i.e., the solar system that the space explorer has arrived from, and the solar system it currently is located in). An incorrect decoding result will trigger an explosion, because everyone at the headquarters is too busy to double check and they just send the science vessel through.

Once the science vessel returns, the Federation gives the list of new wormholes to the space explorers, so they can start work on the next decodings.

The Federation HQs are implemented by us in the HeadQuarter.java file found in the skeleton. You are not allowed to modify this file, since it will be overwritten by the checker when you submit your homework. However, you can use it to understand how a Federation HQ functions and how it interacts with the space explorers.

The Communication Channel

The space explorers and the Federation HQs communicate using a special bi-directional communication channel. For every decoded wormhole frequency, the space explorers give the Federation HQs the following information in a single message: new solar system, previous solar system, and the decoded string.

For every decoded solution, the federation HQs give the space explorers the list of solar systems connected through wormholes with the newly-accessible solar system. If the federation HQ gets an incorrect decode or an incorrect previous solar system, the science vessel enters the wormhole incorrectly and the entire galaxy explodes and implodes at the same time.

When there are no more wormholes to explore, the federation HQ puts EXIT messages in the communication channel, one for each space explorer. They also trigger the end of the program.

After a wormhole's frequency is correctly decoded, the Federation HQs give the space explorers the list of new solar systems in separate messages that arrive in the following order:

- *discovered solar system*
- *adjacent undiscovered solar system 1*
- *discovered solar system*
- *adjacent undiscovered solar system 2*
- *...*
- *END.*

If two federation HQs give the space explorers this information at the same time, it might confuse them. Each of the adjacent solar system messages contains the string to be hashed by the space explorers representing the sub-space-mambo-jumbonian frequencies.

The shape of the messages exchanged by the Federation HQs and the space explorers are provided by us in the Message.java file from the skeleton. This file should not be modified, since it will be overwritten by the checker. A message from a space explorer to an HQ contains the previous solar system, the current solar system, and the decoded frequency (in the "data" field). A message from an HQ to a space explorer contains a solar system and its undecoded frequency (in the "data" field).

The bi-directional communication channel needs to be implemented by you in the CommunicationChannel.java file found in the skeleton. You need to implement the constructor and the methods provided.

Scalability

It should be obvious by now that, if we had more space explorers decoding, the solar systems would be discovered faster. Because of this, the number of space explorers should match the number of threads and the execution time should scale with the number of brave space explorers.

Running, Testing, and Uploading

Your program should be be run as follows (where the Homework.java file containing the main method is provided in the skeleton and should not be modified, since it will be overwritten by the checker):

```
java Homework <wormHoles> <hashCount> <federationHQCount> <spaceExplorerCount>
```

The first parameter specifies the files used for input, the second one represents the number of times the hash function should be applied when decoding a frequency, the third parameter is the number of Federation HQ threads, while the final parameter specifies the number of space explorer threads.

As an example, the program in the skeleton can be run as shown below:

```
java Homework test_cases/test01 100000 4 4
```

In order to help with the testing, we have provided you with one set of input files, located in the folder called *test_cases*. The file *test01_answer.txt* contains the correct answers for each wormhole (which should be decoded by the space explorers and verified by the Federation HQs), *test01_data.txt* contains the values to be hashed (i.e., undecoded sub-space-mambo-jumbonian frequencies), while *test01_graph.txt* contains the layout of the galaxy. All test files should follow the same naming format (e.g., if your first parameter when running the homework is "mytest", then you should have the files *mytest_answer.txt*, *mytest_data.txt* and

mytest_graph.txt). The files are read only by the Federation HQs, whereas the space explorers only receive data from the HQs.

Galaxy Generator and Wormhole Frequency Decoder

To help you with your testing, we have added a Python program that can generate a galaxy (i.e., a graph) and the frequencies for the wormholes (i.e., the values to be hashed by the explorers). It is located in the folder called *generator*, and can be run as follows (where we create a graph with 100 nodes and 150 edges, stored in files *test01_graph.txt* and *test01_data.txt*):

```
python generate_graph.py -f test01 -s 100 -e 150
```

In the homework archive, you can also find a tool for computing consecutive hashes on some input strings, which can be used to generate the input files with decoded frequencies that the HQs use to verify the correctness of the values received from the explorers. The HQ implementation in the skeleton uses those same hash functions to verify the correctness of the explorers' computations.

With these two tools, you can create your own complete tests and verify your program even before uploading it to the checker.

Upload

You have to upload a .zip archive which contains only the CommunicationChannel.java and SpaceExplorer.java files. The archive should not contain any other files or directories. **All extra files will be deleted or overwritten.**

Your solution needs to compile and run on the checker, where currently Java 1.7 is installed.

Any attempt to trick the checker system or falsify the results (including the speedup), would result in a penalty that would remove all points for the 3 homework assignments.

It is forbidden to have any I/O in the submitted solution. No reads, no writes, to standard output or files.

Grading

40 points - Correctness
60 points - Speedup

For the correctness tests on the checker, 6 tests of 2 points each are run on a galaxy with 100 systems, 4 tests of 3 points each are run for a galaxy with 50 systems, and 4 tests of 4 points each are run for a 500-system galaxy. If the duration of a simulation is higher than 30 seconds, the test automatically fails. If at least one correctness test fails, the scalability tests become irrelevant and thus are not graded.

A solution that doesn't compile or that does not pass any tests will be awarded 0 points.