Comp 202 - Fall 2021
Homework #4
Due date - 23:59 02/01/2022

In this homework, you will implement a method that returns possible smartphone lock screens with starting point and lock length provided for you. You must implement this method by using the **Depth First Search** algorithm. Figure 1 below is an example of our smartphone screen lock. We will be naming nodes as A, B, C, D, E, F, G, H, and I, respectively.

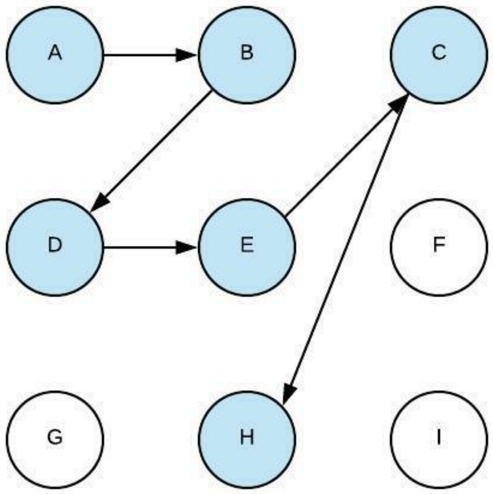

Figure 1

The structure of the graph is already provided for you. Connections of each node are given in ArrayLists. You can access them from the HashMap that is called “adjacency_list”. There is a print_result method that takes “ArrayList<String>” as input and prints the corresponding lock screens. Please do not change this method to have the same output format for auto-grading. Also, make sure that the main function is the same as the beginning when you submit your homework.

You will implement the possible_locks method. You can add helper functions. You might find implementing a “Depth First Search” method useful. We will give the starting node, length of the input, and the adjacency_list as input to the possible_locks. For Figure 1, the starting point is A, and the length is 6. We count the number of nodes as length. If length is not a valid value, return null. The adjacency list provided for you has connections in alphabetical order, so keep in mind that your traversal with the DFS algorithm will yield alphabetical results. See test cases in the main class and an example output for more details.

Your possible locks functions should return each possible lock in an ArrayList<String>. You need to represent a lock with letters appended to each other. For instance, Figure 1 must be represented by “ABDECH” in this ArrayList<String>. Locks with the starting point C and length two are represented by “[CB, CD, CE, CF, CH]”. See the “print_result” method and example outputs for more details. Make sure that you added comments to your code for critical points of the DFS algorithm so that we can understand how you implemented it.

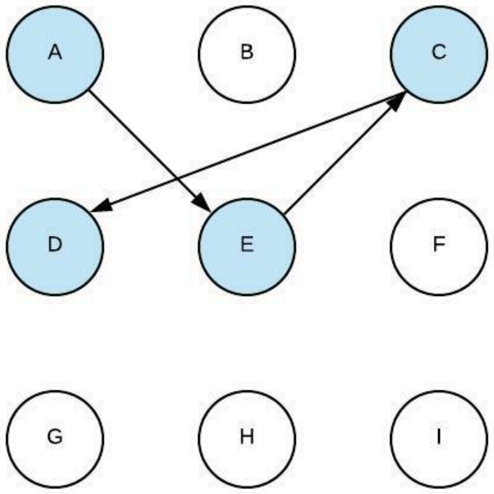

Figure 2

Your lines can intersect, as shown in Figure 2. So the corresponding lock is “A -> E -> C -> D,” and it is a valid lock.

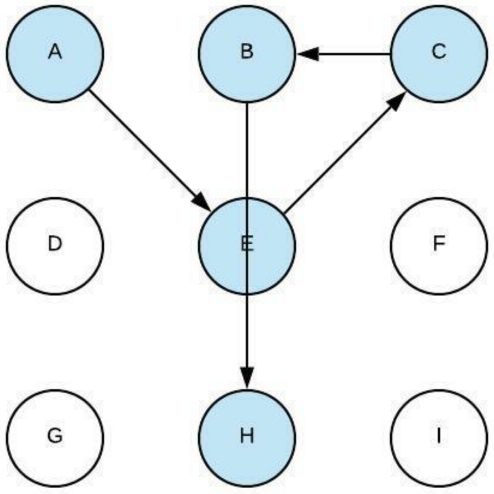

Figure 3

However, Figure 3 is not a valid lock. Your lines can not pass through already visited nodes. Connections of node B are given as "A", "C", "D", "E", "F", "G", and "I". So even though real-life phone screens allow connecting B directly to H after E is used, we will not allow it for the sake of simplicity. So once again, “A -> E -> C -> B -> H” is not a valid lock, and you will not include it when we ask for locks with the starting point A and length 5.

Submission Details

Usage of github classroom and submission details:

1. Accept the invitation using this link https://classroom.github.com/a/yGJTPmbz.
2. Choose your student ID from the list.
3. A personal repository with the starter code will be created. You can directly edit the code on github's page. You can clone it to your local storage and edit it there. Ensure that your repositories are private.
4. Make sure your changes are committed and pushed to the repository.
5. Check if you have successfully passed the tests in the "Actions" tab of your repository. Note that these automatic tests do help you understand whether or not your solution works up to some level, but passing them does not guarantee that you will receive a full grade.
6. To make sure your code is received and avoid any potential problems on github's side, submit a copy of java files on Blackboard as well. Submit all files as a single .zip file, named as: "ID_NAME_SURNAME.zip"

Grading Criteria

You are going to receive all the credits from one part. We may deduct points if your code does not have necessary comments. But this does not mean you need to write very long comments. Simply explain your DFS implementation.

For your further questions about the homework send an email to: bbiner21@ku.edu.tr