HOMEWORK 1 REPORT

Ekin Nohutçu
150116067

Algorithm for BFS:

1. I created a graph by using adjacency matrix and put distance values into matrix rather than 1 or 0's.
2. In BFS method, it takes 3 parameters. First one is for adjacency matrix, second one is source and third one is destination.
3. First, I created an Arraylist of integers to store the path from source laptop to destination path.
4. Then I created a Queue of paths.
5. For visited information, I created an array which holds boolean values.
6. I added a source into path and add path into paths queue and visited value for the source laptop is become true.
7. While there is a path in queue of paths:
    7.1) Remove the path from queue of paths.
    7.2) Create an integer to hold current node.
    7.3) If current node and destination node are the same, return the Arraylist of path.
    7.4) Else, traverse nodes that have connection with current node. If their distances are greater than 0 and the node is not visited, add this node into new path and add this path into the queue of paths and make their visited value as true.
8. If a node is not reachable from source, remove every node inside path.
9. Return Arraylist of nodes (integers).

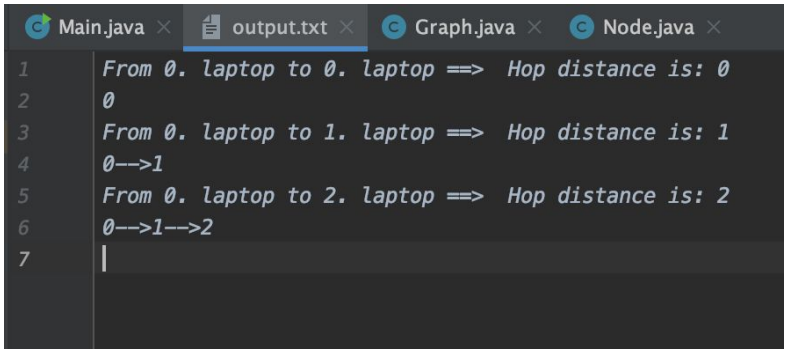
Time and Space Complexity:

- For time complexity: It is equal to the BFS algorithm time complexity. So this is Θ(n).

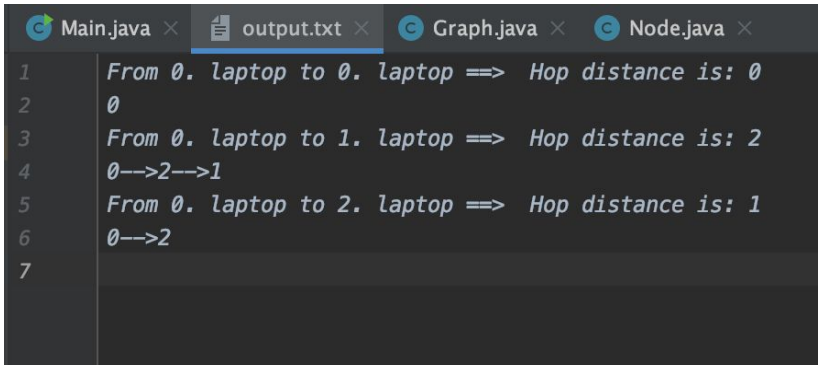
- For space complexity: I am using adjacency matrix so the space complexity is n^2

OUTPUTS:

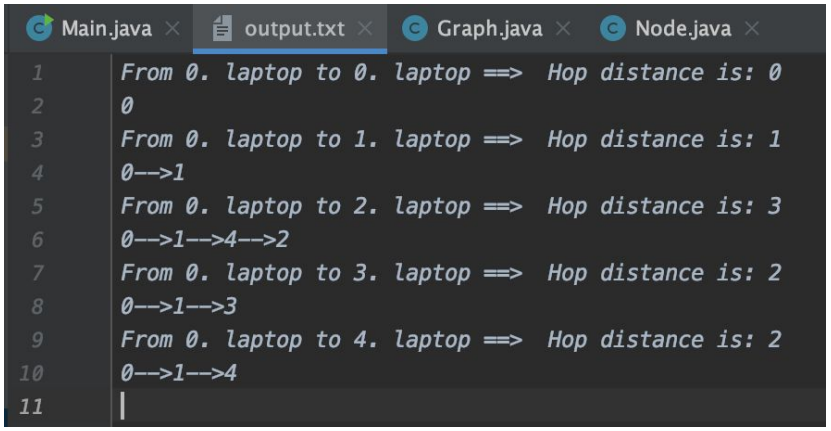
Output for the test1.txt



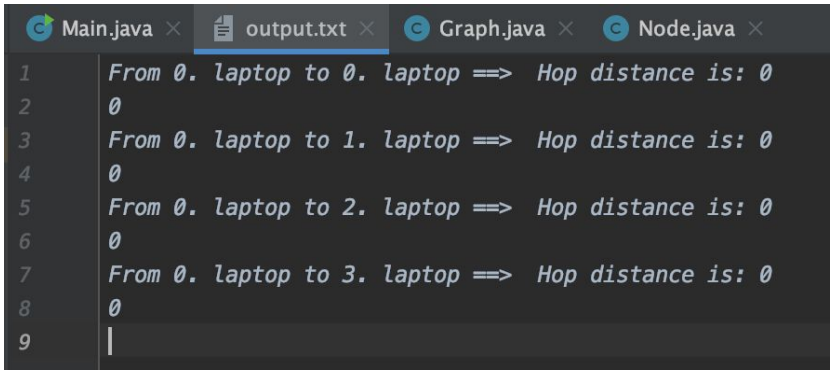
Output for the test2.txt



Output for the test3.txt



Output for the test4.txt



Output for the test5.txt

Main.java | output.txt | Graph.java | Node.java

```
From 0. laptop to 0. laptop ==>  Hop distance is: 0
0
From 0. laptop to 1. laptop ==>  Hop distance is: 1
0-->1
From 0. laptop to 2. laptop ==>  Hop distance is: 2
0-->1-->2
From 0. laptop to 3. laptop ==>  Hop distance is: 3
0-->1-->2-->3
From 0. laptop to 4. laptop ==>  Hop distance is: 3
0-->1-->2-->4
From 0. laptop to 5. laptop ==>  Hop distance is: 4
0-->1-->2-->4-->5
From 0. laptop to 6. laptop ==>  Hop distance is: 0
0
From 0. laptop to 7. laptop ==>  Hop distance is: 0
0
From 0. laptop to 8. laptop ==>  Hop distance is: 4
0-->1-->2-->4-->8

```

Output for the test6.txt

Main.java | output.txt | Graph.java | Node.java

```
From 0. laptop to 0. laptop ==>  Hop distance is: 0
0
From 0. laptop to 1. laptop ==>  Hop distance is: 1
0-->1
From 0. laptop to 2. laptop ==>  Hop distance is: 2
0-->1-->2
From 0. laptop to 3. laptop ==>  Hop distance is: 3
0-->1-->2-->3
From 0. laptop to 4. laptop ==>  Hop distance is: 4
0-->1-->2-->3-->4
From 0. laptop to 5. laptop ==>  Hop distance is: 2
0-->1-->5
From 0. laptop to 6. laptop ==>  Hop distance is: 3
0-->1-->5-->6
From 0. laptop to 7. laptop ==>  Hop distance is: 3
0-->1-->2-->7
From 0. laptop to 8. laptop ==>  Hop distance is: 5
0-->1-->2-->3-->4-->8
From 0. laptop to 9. laptop ==>  Hop distance is: 4
0-->1-->5-->6-->9
From 0. laptop to 10. laptop ==>  Hop distance is: 5
0-->1-->5-->6-->9-->10
From 0. laptop to 11. laptop ==>  Hop distance is: 6
0-->1-->2-->3-->4-->8-->11
From 0. laptop to 12. laptop ==>  Hop distance is: 7
0-->1-->2-->3-->4-->8-->11-->12
From 0. laptop to 13. laptop ==>  Hop distance is: 0
0
From 0. laptop to 14. laptop ==>  Hop distance is: 8
0-->1-->2-->3-->4-->8-->11-->12-->14

```