

CE2001 : Algorithms

Project 2

SEP2

Group 6

Done By:

(U1921023K) Mervyn Chiong Jia Rong

(U1921062L) Ang Kai Jun

(U1920248G) Lee Zhe Ren

(U1923071B) Harsh Doddannavar

**Introduction**

Our group debated between using either BFS(Breadth-First Search) or DFS(Depth-First Search). Ultimately deciding to use BFS. This is purely because the problem posed is to locate the shortest path to a certain location(hospital) in an undirected unweighted graph G. DFS is not applicable in this case because it would traverse more edges than it is required to do from destination to endpoint.

**Analysis of Breadth First Search Method**

**Assumptions:**

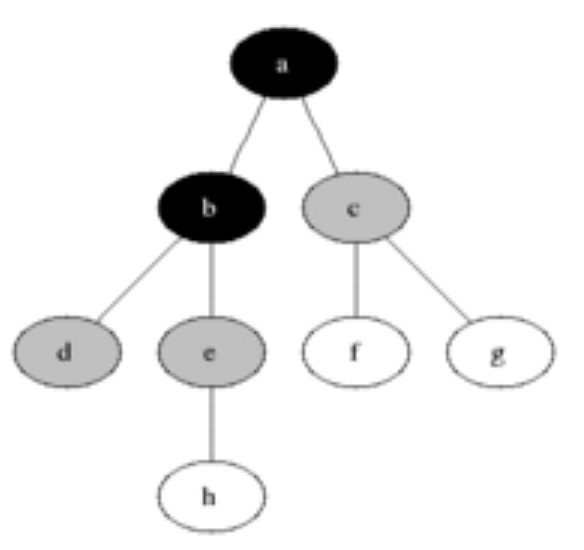
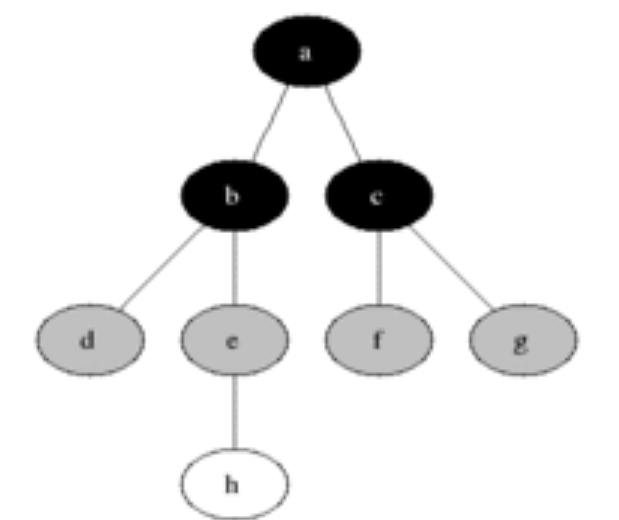
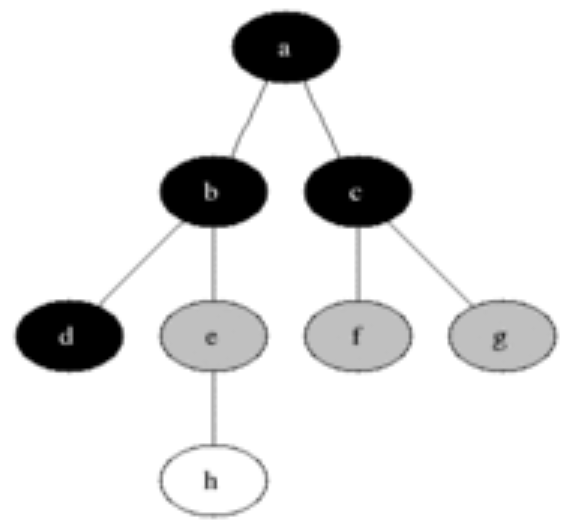
**Graph is G**

**Every edge is taken as a single unit(unweighted graph), the total sum would be E**

**Any other node, S**

**Source(root) node is the vertex,V**

BFS starts off from a root node(something like the top of a large tree diagram) and move to its adjacent nodes, then to all nodes that are connected to the root node and it will then move on to the next set but will go back to do the other nodes from the original source executing the same logic.

-> ->

As such G=(V,E) , this will compute the shortest distance to any vertex as any path would corresponds to shortest path from root,v, to any other node S

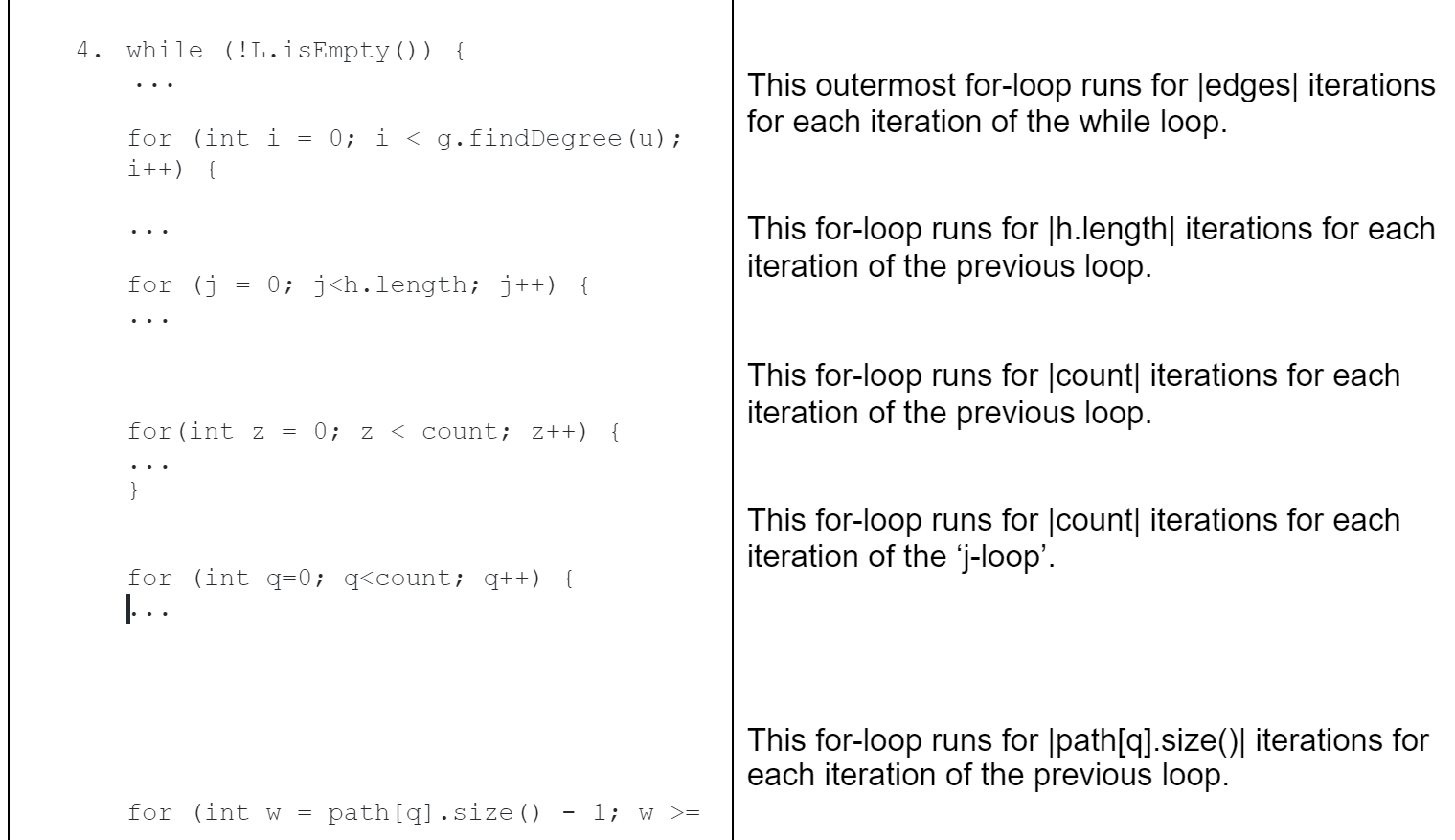
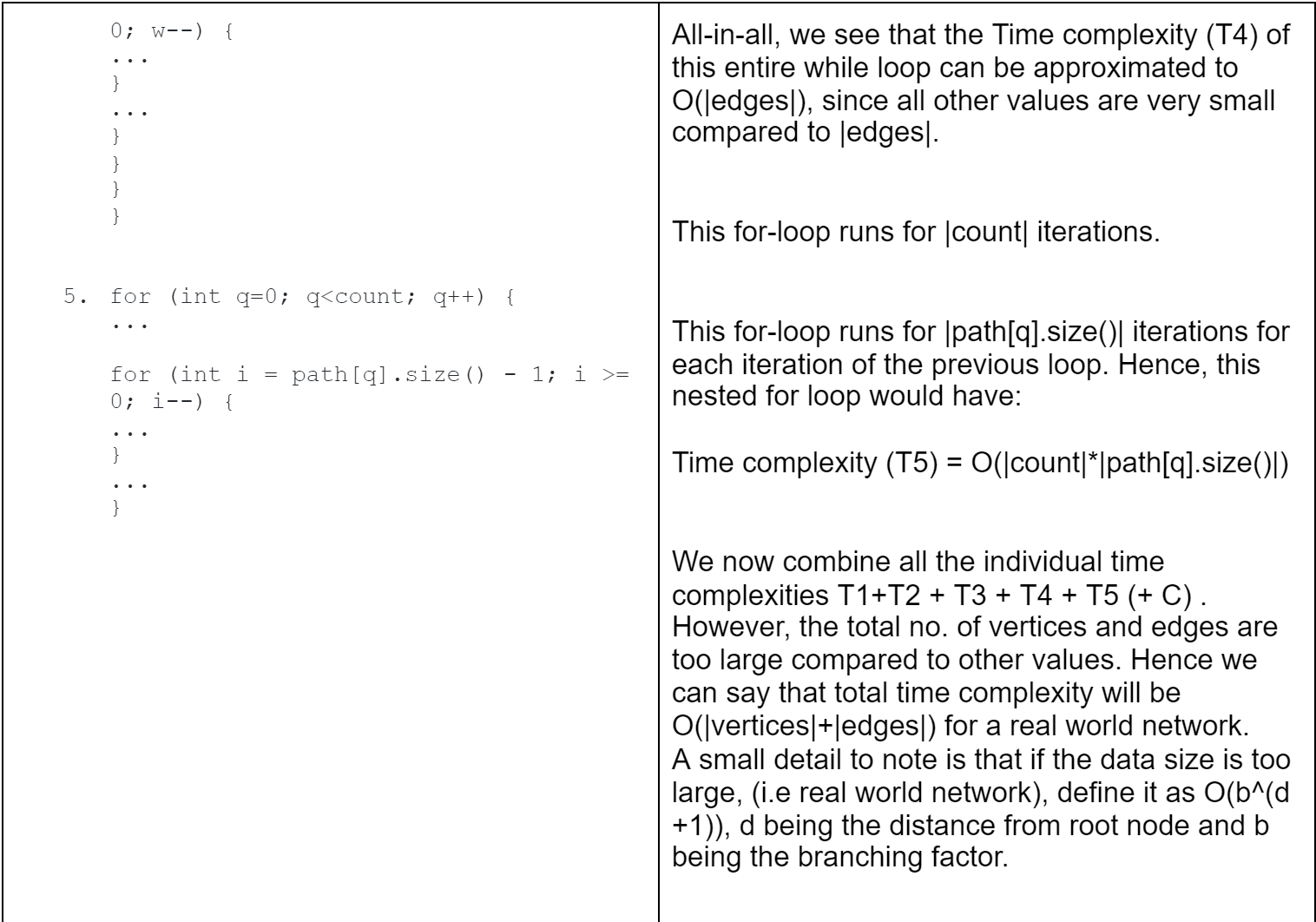
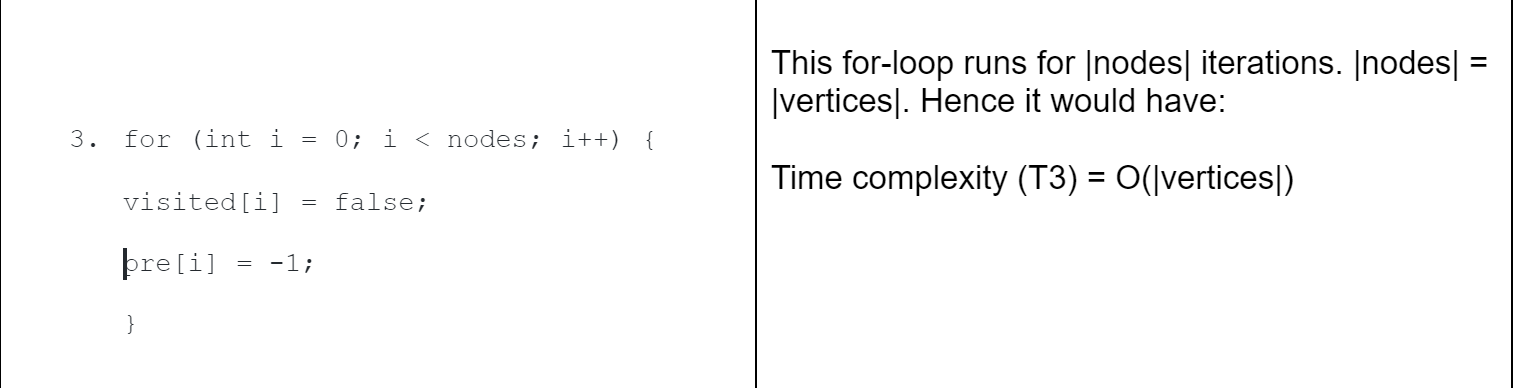
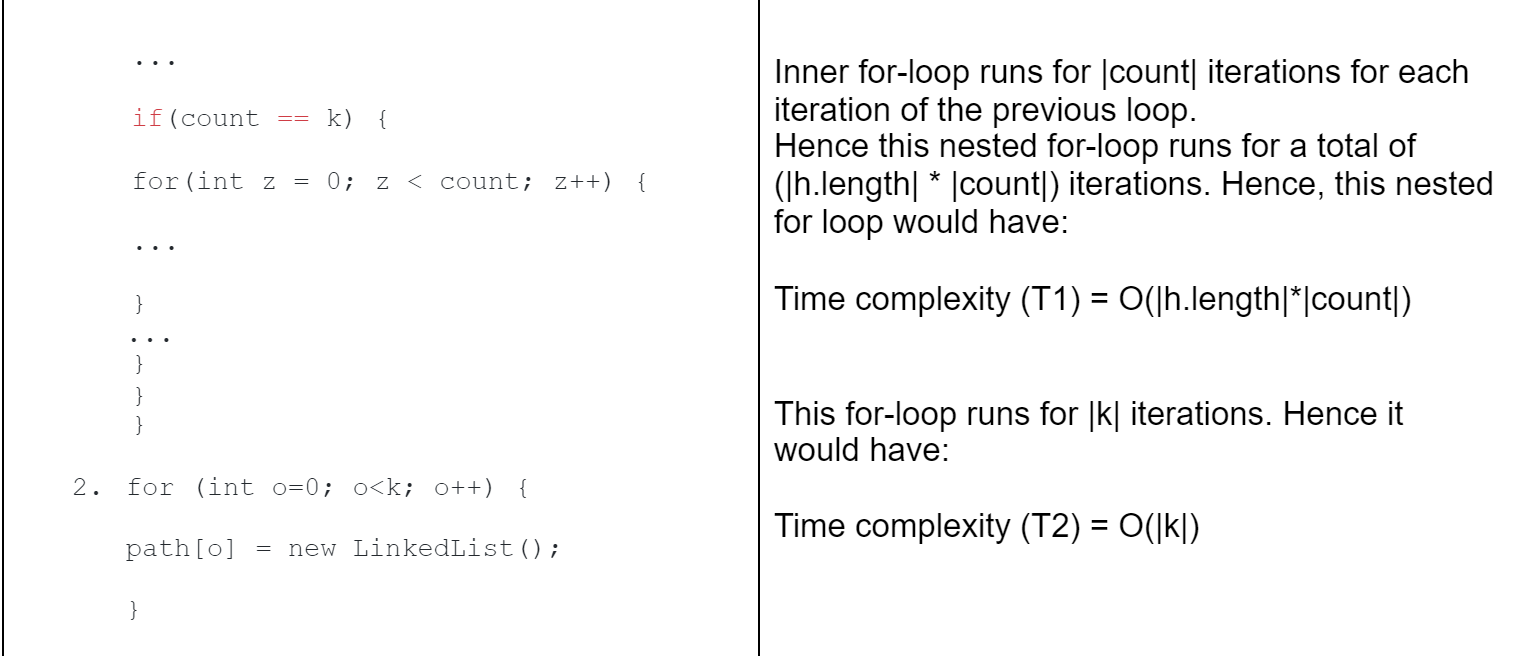
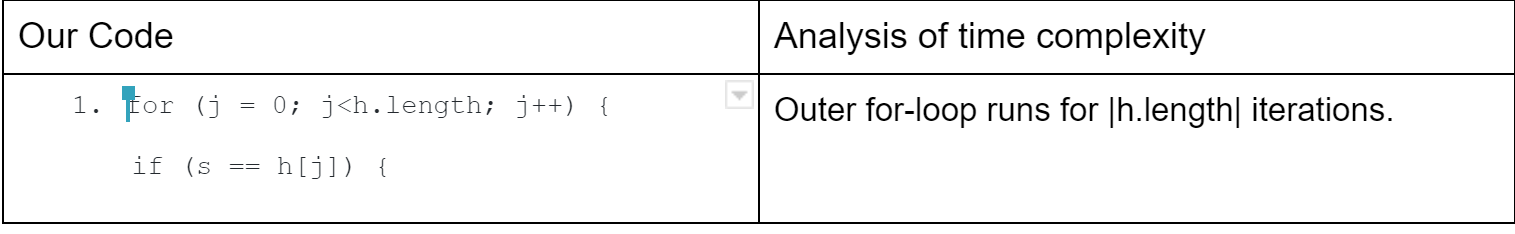
**Our code**

**Solving part(b),** time complexity should not depend on the total number of hospitals h

**Solving part(c)(controlling k input value).**finding the distances to top-2 nearest hospitals from each node. Revise your algorithm in (b) to accommodate this new requirement.

**Time Complexity**

Regardless, the time complexity will be O(|V|+|E|) as the function will explore every single edge. |V| being the total number of vertices and |E| being the total number of edges in the graph. Hence it may vary if you have a lot of nodes between O(1) and O(|V^2|). A small detail to note is that if the data size is too large, (i.e real world network), define it as *O*(*bd* + 1), d being the distance from root node and b being the branching factor.

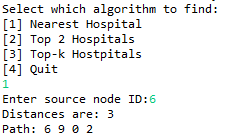
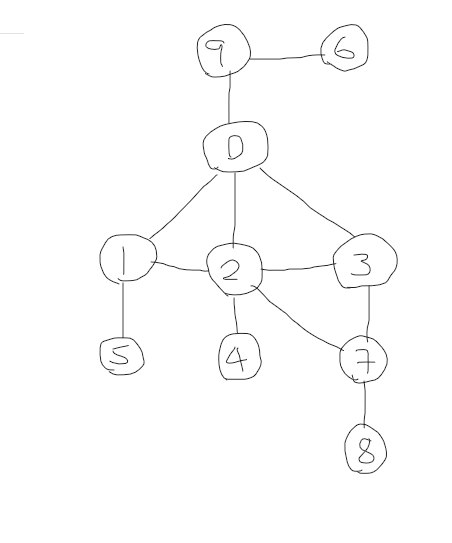


**Number of hospitals and locations:**

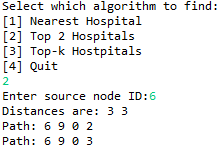
**Using the same set,for all the cases.**

**Assuming nodes 3,7,8,2 are hospitals**

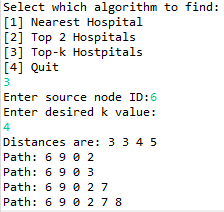
**Case(1a): (in example)**

<-this is our code result<- visual representation

**Case(1c):**

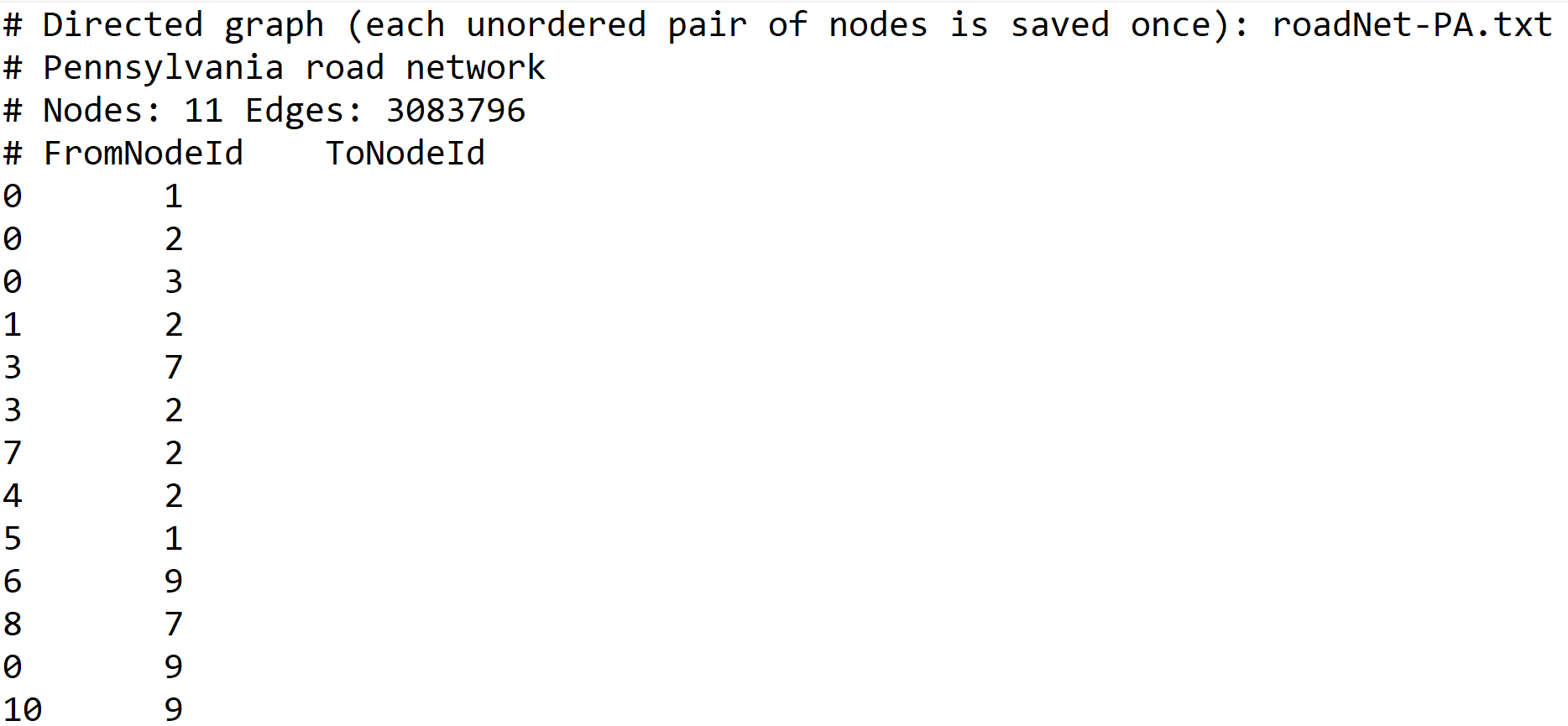
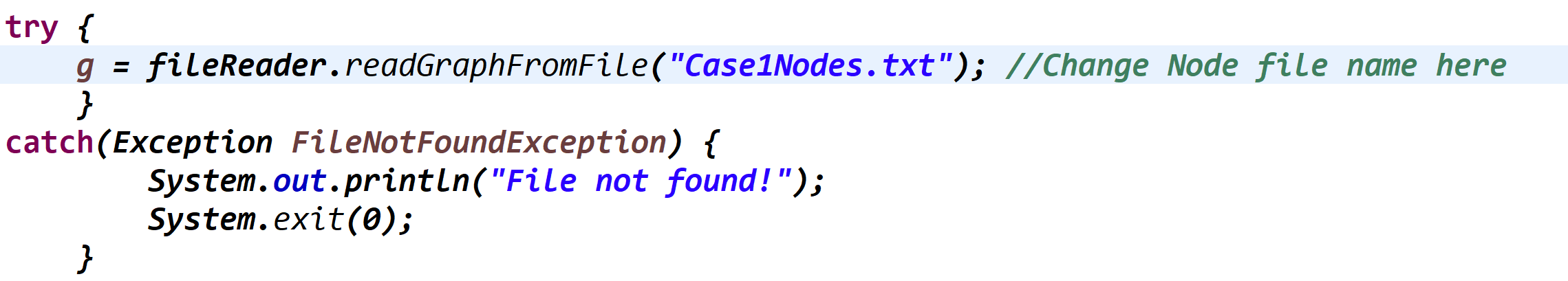
Top 2 nearest hospitals distance

**Case(1d):**

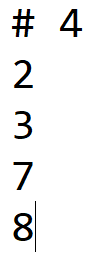
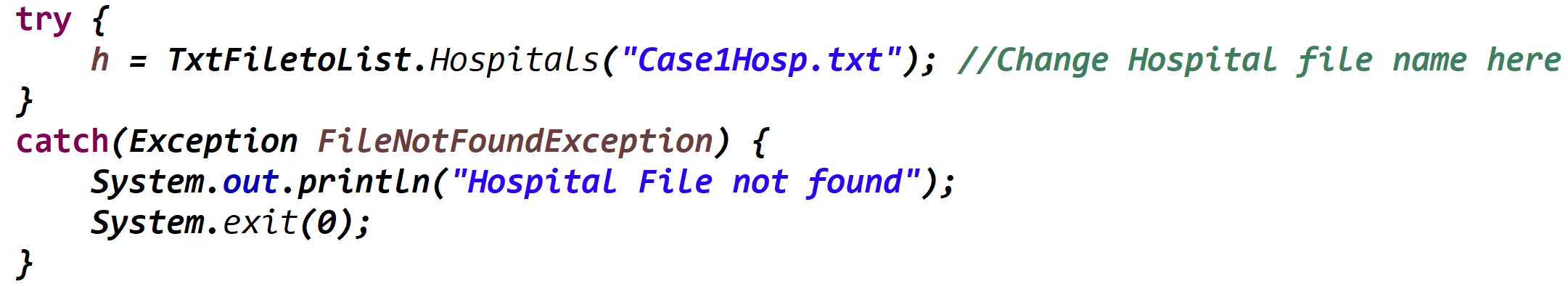
Top-K Hospitals all h nodes detected

**Case(2)+(3):**

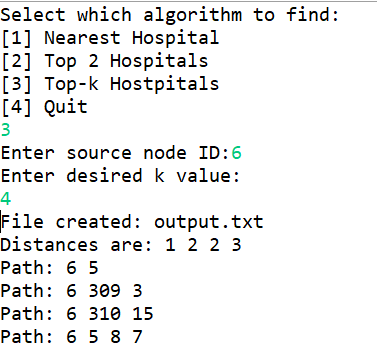
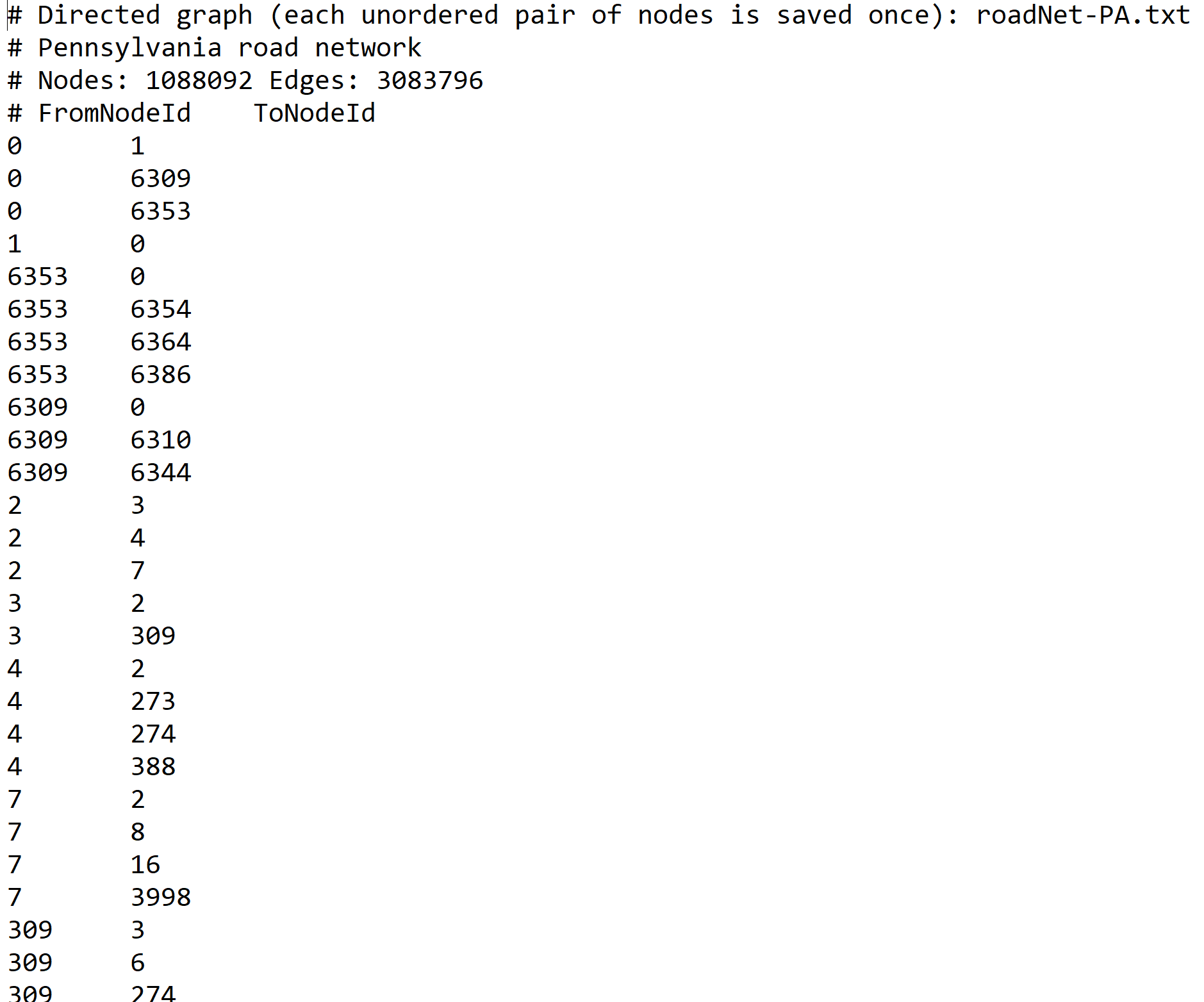
using a small input graph, the below image is for Case1Nodes.txt



The below images are the other file, fulfilling 2 file input



**Experiment(case 2):Testing with real world network:**

^snippet of the real time network output and a snippet of whats inside it

**Empirical Findings:**

**The question :** see the effects of h and k(input size) on the performance of various algorithms and conclude if the empirical findings are consistent with theoretic analysis.

**Assumptions:**

We take the same root node for all cases

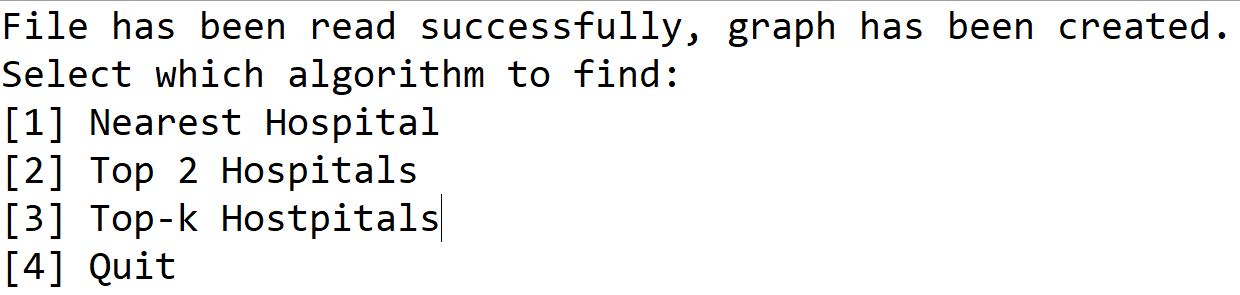
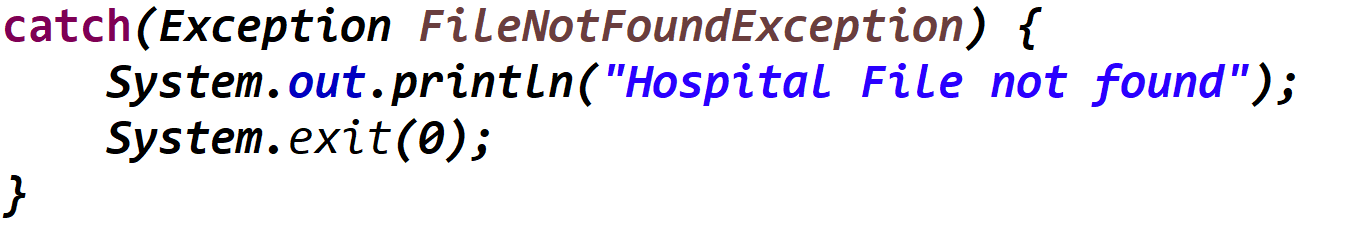
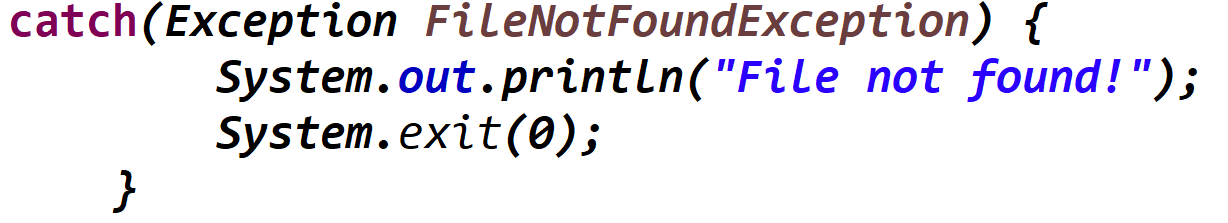
And the total number of nodes increases

**Found result:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Cases** | **h(hospitals)** | **K(top nearest number of hospital)** | **Empirical study BFS(total distance to h)** | **Various Algorithms(i.e BFS)(taken from online)** | **Theoretical Result** |
| **1** | **5** | **1** | **53** | **50** | **53** |
| **2** | **5** | **2** | **53, 54** | **50,51** | **53, 54** |
| **3** | **5** | **3** | **53, 54, 55** | **50,51,52** | **53, 54, 55** |
| **4** | **5** | **4** | **53, 54, 55, 55** | **50,51,52,53** | **53, 54, 55, 55** |
| **5** | **5** | **5** | **53, 54, 55, 55, 55** | **50,51,52,53,54** | **53, 54, 55, 55, 55** |

**Example of user-friendly interface:**

lab supervisor can test on arbitrary input graphs.



**References:**

<https://www.researchgate.net/publication/221302072_BFS_Solution_for_Disjoint_Paths_in_P_Systems/download>

<https://www.geeksforgeeks.org/depth-first-search-or-dfs-for-a-graph/>

<https://www.geeksforgeeks.org/shortest-path-unweighted-graph/>

Contributions:

Harsh worked on the code to read data from the hospital file. Kai Jun worked on the graph class and the code to read and create the graph from the node file. Zhe Ren and Kai Jun mainly implemented the BFS Algorithm, while everyone worked together on debugging it. Time complexity of the algorithm was derived by Harsh. Testing and documentation of the results were carried out by Mervyn.