IT5003 Semester 2 2024/2025

Data Structures and Algorithms

Tutorial 09
SSSP + Wrap-up
For Week 12 (Sat)/13 (Mon)

Document is last modified on: April 1, 2025

1 Introduction and Objective

In this tutorial, we will discuss the last examinable topic for IT5003 (and second-last for CS2040S): Single-Source Shortest Paths (SSSP) problem and continue talking about the 'graph modeling' soft skill, i.e., the ability to model a seemingly random (non-explicit-graph) problem into a graph problem (specifically the SSSP problem for this tutorial).

We will use `https://visualgo.net/en/sssp` during our discussion in this tutorial.

SSSP problem is quite easily found in many real-life applications and it is the source of many interesting Computer Science problems, as you can see in this tutorial. Again, we recommend that you put some thoughts on them before discussing the potential solutions with your tutor.

2 Tutorial Questions

Standard Stuffs

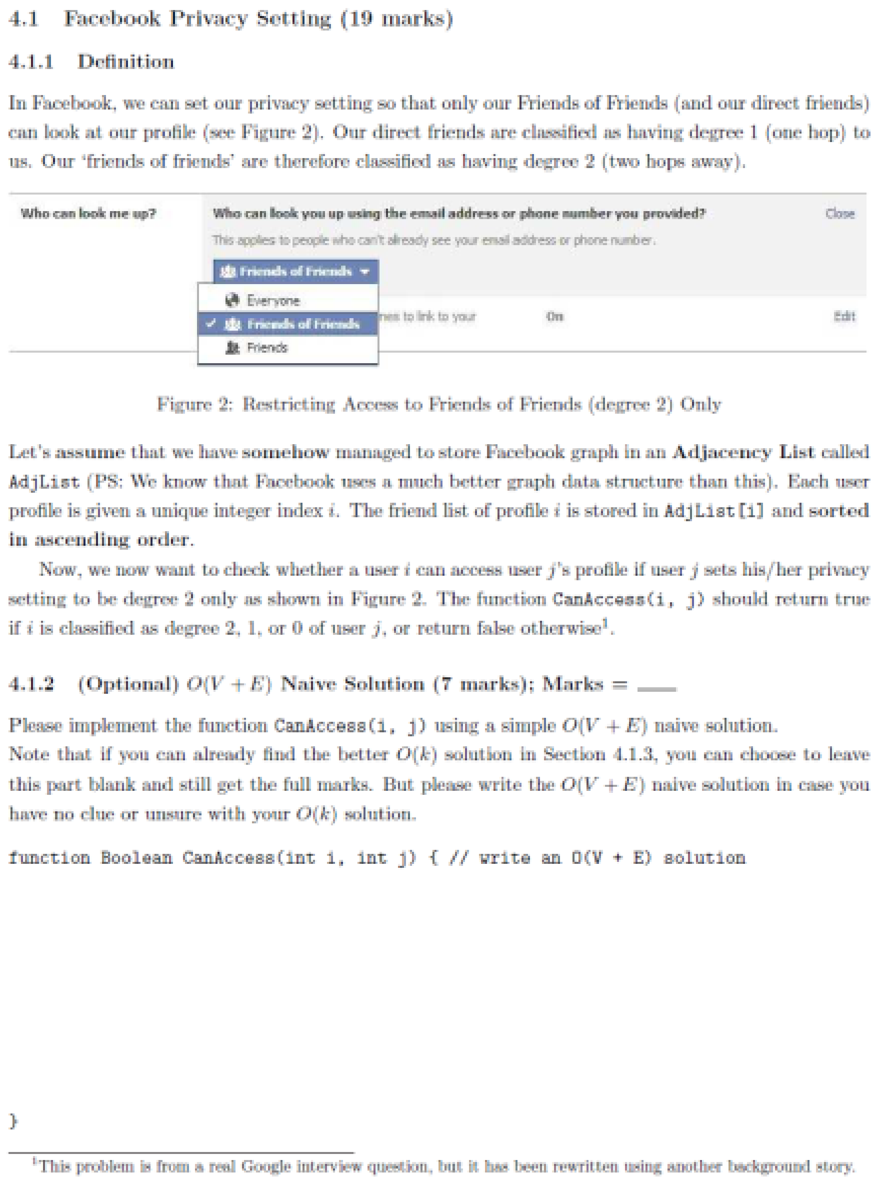
So far, you have been presented with these SSSP algorithms: Bellman-Ford algorithm (without the proofs), BFS (only for unweighted graph), Dijkstra's algorithm (the original that needs balanced BST and the modification - so that we can use built-in min PQ data structure, especially for IT5003), DFS/BFS (for Tree), and Topological Sort/DP (for DAG).

First, the tutor will (re-)demonstrate the executions of Bellman-Ford, BFS, Dijkstra's, DFS, and/or Topological Sort/DP algorithms on a small directed weighted graph using `https://visualgo.net/en/sssp` from a certain source vertex s. The tutor will re-explain when a certain algorithm can be used and when the same algorithm cannot be used. The tutor may invite some students to do this live demonstration using different source vertex s and/or using different graph.

Graph Modeling Exercises, via Past Paper Discussions

There are a few graph questions in recent final assessment papers. Let's discuss two of them (you can bet that there will be SSSP question(s) in the Final Assessment:

Q1). See Figure 1 and 2), Question 4.1, Facebook Privacy Setting



4.1 Facebook Privacy Setting (19 marks)

4.1.1 Definition

In Facebook, we can set our privacy setting so that only our Friends of Friends (and our direct friends) can look at our profile (see Figure 2). Our direct friends are classified as having degree 1 (one hop) to us. Our 'friends of friends' are therefore classified as having degree 2 (two hops away).

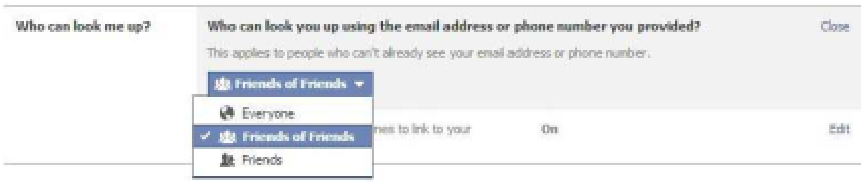

Who can look me up?
Who can look you up using the email address or phone number you provided?
Close
This applies to people who can't already see your email address or phone number.
Friends of Friends
Everyone
✓ Friends of Friends
Friends
nes to link to your
On
Edit

Figure 2: Restricting Access to Friends of Friends (degree 2) Only

Let's **assume** that we have **somehow** managed to store Facebook graph in an **Adjacency List** called `AdjList` (PS: We know that Facebook uses a much better graph data structure than this). Each user profile is given a unique integer index i. The friend list of profile i is stored in `AdjList[i]` and **sorted in ascending order**.

Now, we now want to check whether a user i can access user j's profile if user j sets his/her privacy setting to be degree 2 only as shown in Figure 2. The function `CanAccess(i, j)` should return true if i is classified as degree 2, 1, or 0 of user j, or return false otherwise[1].

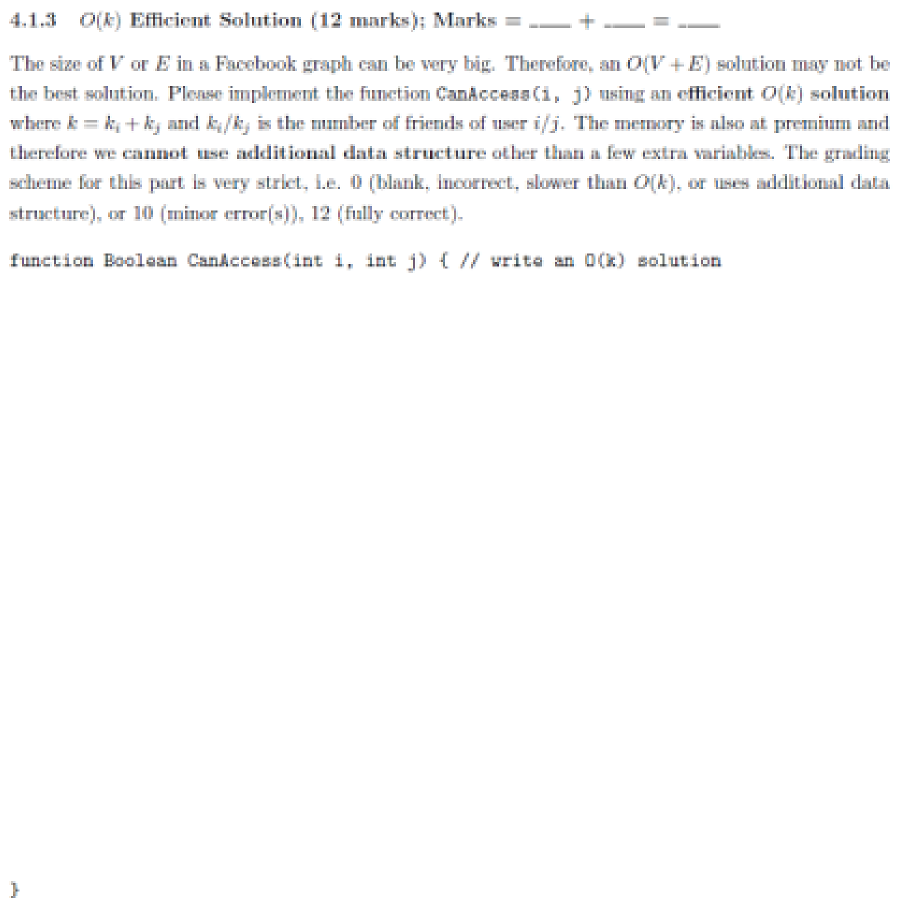
4.1.2 (Optional) $O(V + E)$ Naive Solution (7 marks); Marks = ___

Please implement the function `CanAccess(i, j)` using a simple $O(V + E)$ naive solution.
Note that if you can already find the better $O(k)$ solution in Section 4.1.3, you can choose to leave this part blank and still get the full marks. But please write the $O(V + E)$ naive solution in case you have no clue or unsure with your $O(k)$ solution.

```
function Boolean CanAccess(int i, int j) { // write an O(V + E) solution







}
```

[1]This problem is from a real Google interview question, but it has been rewritten using another background story.

Figure 1: Facebook 1



4.1.3 $O(k)$ **Efficient Solution (12 marks); Marks = ___ + ___ = ___**

The size of V or E in a Facebook graph can be very big. Therefore, an $O(V + E)$ solution may not be the best solution. Please implement the function `CanAccess(i, j)` using an **efficient** $O(k)$ **solution** where $k = k_i + k_j$ and k_i/k_j is the number of friends of user i/j. The memory is also at premium and therefore we **cannot use additional data structure** other than a few extra variables. The grading scheme for this part is very strict, i.e. 0 (blank, incorrect, slower than $O(k)$, or uses additional data structure), or 10 (minor error(s)), 12 (fully correct).

```
function Boolean CanAccess(int i, int j) { // write an O(k) solution






}
```

Figure 2: Facebook 2

Q2). Read `https://open.kattis.com/problems/emergency` and solve it!

Hands-on

TA will run the second half of this session with a few to do list:

- Speedrun sssp of VisuAlgo Online Quiz:
  `https://visualgo.net/training?diff=Medium&n=5&tl=5&module=sssp`
  Then, share any last minute tips for VA OQ preparation based on TA's experiences (all questions involving negative weight edges have been turned off)
- Hands-on: One task about SSSP.

Problem Set 6

We will end the tutorial with **high-level** discussion of PS6.
We can now discuss more ideas of PS6.

Remarks for IT5003

This is the last tutorial for IT5003.

Take class photo for momento.