CS2040S: Data Structures and Algorithms

Discussion Group Problems for Week 9

For: March 16–March 20

Problem 1.
(Priority queue.)

There are situations where, given a data set, we want to know the top k highest value elements. A possible solution is to store all n elements first, sort the data set in $O(n \log n)$, then report the rightmost k elements. Give an algorithm to: (i) find the top k largest elements better than $O(n \log n)$; (ii) find the top k largest elements as the elements are streaming in (for each new element that is given to you, your data structure must be ready to answer queries for the top k largest elements efficiently), and the algorithm runs faster than $O(n \log n)$.

Problem 2.
(Horseplay.)
(Relevant Kattis Problem: https://open.kattis.com/problems/bank)

There are m bales of hay that need to be bucked and n horses to buck them.

The i^{th} bale of hay has a weight of k_i kilograms.

The i^{th} horse has a strength s_i—the maximum weight, in kilograms, it can buck—and can be hired for d_i dollars.

Bucking hay is a very physically demanding task, so to avoid the risk of injury every horse can buck at most one bale of hay.

Determine, in $O(n \log n + m \log m)$, the minimum amount, in dollars, needed to buck all bales of hay. If it is not possible to buck them all, determine that as well.

Problem 3.
(DFS/BFS.)

Recall that when performing DFS or BFS, we may keep track of a `parent` pointer that indicates the very first time that a node was visited. Explain why these edges form a tree (i.e., why there are no cycles).

Problem 4.
(Graph components.)
(Relevant Kattis Problem: https://open.kattis.com/problems/countingstars)

Given an undirected graph $G = (V, E)$ as an adjacency list, give an algorithm to: (i) determine if the graph is connected; (ii) return the number of connected components (CC) in the graph.

Problem 5.
(Good students, bad students.)
(Relevant Kattis problem: https://open.kattis.com/problems/amanda)

There are good students and bad students[1]. And at the end of every year, we have to divide the students into two piles: G, the good students who will get an A, and B, the bad students who will get an F. (We only give two grades in this class.)

To help with this process, your friendly tutors have each created a set of notecards, each containing two names of students (from the entire class). They promise that for every notecard, one of the two names is a good student, and the other is a bad student. Unfortunately, they do not indicate which is which. Also, since the notecards come from thirty eight different tutors, it is not immediately certain that the cards are consistent. Maybe one tutor things that Humperdink is a good student, while another tutor thinks that Humperdink is a bad student. (And Humperdink may appear on several different cards.) In addition, they do not provide cards for every pair of students; they only provide a selection of pairs. Assume you can read the names on a card in $O(1)$ time.

Devise an algorithm for determining: (i) Are the notecards are consistent, i.e., is there *any* way that we can assign students to G and B that is consistent with the cards? (ii) Are the notecards sufficient, i.e., is there one or more than one way of assigning students to the sets G and B? (iii) Assuming there are more good students than bad students, and assuming that the notecards are consistent and sufficient, determine which set each student belongs in.

[1] No, not really. This sort of binary distinction is silly.

Problem 6.
(Word games.)
(Relevant Kattis problem: https://open.kattis.com/problems/sendmoremoney)

Consider the following two puzzles:

- Puzzle 1:

```
   S E N D
+  M O R E
----------
 M O N E Y
```

- Puzzle 2:

```
   F O R T Y
       T E N
+      T E N
------------
   S I X T Y
```

In each of these two puzzles, you can assign a digit (i.e., $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$) to each of the letters in the puzzle such that the given equation holds. (Each digit is only assigned to once letter.) The goal is to solve these puzzles. How should you model and solve these puzzles? What is the running time of your solution? Can you optimize your solution to find the answer more quickly, most of the time?

Problem 6.a. Explain how to model the problem as a graph search problem. What are the nodes? How many nodes are there? What are the edges? Where do you start? What are you looking for?

Problem 6.b. To solve this problem, should you use BFS or DFS? Why? How else can you make it run faster?

Problem 6.c. When does your search finish? Can you optimize the algorithm to minimize the amount of searching?

Problem 7.
(Graph modeling.)
(Relevant Kattis Problem: https://nus.kattis.com/sessions/qxixc8/problems/nus.cs3233mh)

Here are a bunch of problems. How would you model them as a graph? (Do not worry about solving the actual problem. Just think about how you would model it as a graph problem.) Invent some of your own problems that can be modeled as graph problems—the stranger, the better.

Problem 7.a. Imagine you have a population in which some few people are infected with this weird virus. For any two patients, you want to decide whether the infection might have spread from one to the other via some intermediate asymptomatic patients. You suspect that if such a path exists it isn't too long (since most cases are not asymptomatic.)

Problem 7.b. Imagine you have a population in which some few people are infected with this weird virus. You also have a list of locations that each of the sick people were in during the last 14 days. Determine if any of the sick people ever met.

Problem 7.c. You are given a set of jobs to schedule. Each job j starts at some time s_j an ends at some time t_j. Many of these jobs overlap. You want to efficiently find large collections of non-overlapping jobs so that you can assign each collection to a single server.

Problem 7.d. An English professor complains that students in their class are cheating. The professor suspects that the cheating students are all copying their material from only a few different sources, but does not know where they are copying from. Students that are not cheating, on the other hand, all submit fairly different solutions. How should we catch the cheaters?

Problem 7.e. There are n children and n presents, and each child has told you which presents they want. How do we assign presents to children?

Problem 8.
(A problem lovely as a tree.)
(Relevant Kattis Problem: https://open.kattis.com/problems/flyingsafely)

Assume you are given a connected graph with n nodes and m edges as an adjacency list. (You are given n but not m; assume each adjacency list is given as a linked list, so you do not have access to its size.)

Give an algorithm to determine whether or not this graph is a *tree*. Recall that a tree is a connected graph with no cycles. (What if you want to find a cycle?)

Your algorithm should run in $O(n)$; particularly, it should be independent of m. Assume $O(n + m)$ is too slow.

Problem 9.
(Gone viral.)

There are n students in the National University of Singapore. Among them, there are $n - 1$ friendships. (In case you are unfamiliar with the concept of friendship, it means that two people are friends with each other.) Note that friendship is a symmetric relation, but it is not necessarily transitive.

Any two people in the National University of Singapore are either directly or indirectly friends. Formally, between any two different people x and y, either x is friends with y or there exists a sequence $q_1, q_2, \ldots, q_k$ such that x is friends with q_1, q_i is friends with q_{i+1} for all $i < k$ and q_k is friends with y.

It was discovered today that **two** people were found to have coronavirus in the National University of Singapore.

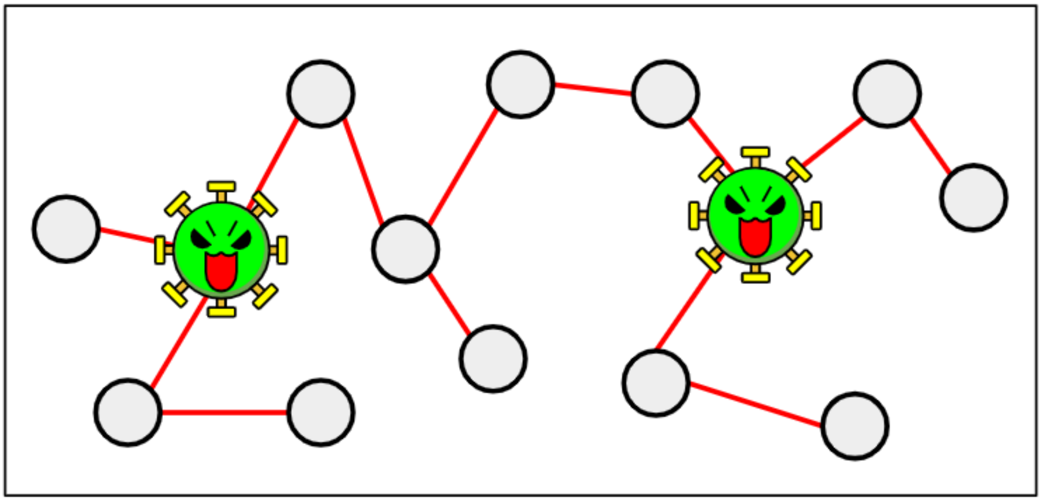

Figure 1: *Gone Viral.* (Matthew Ng Zhen Rui)

Every day, every person can meet with **at most one friend.** When these two people meet, if exactly one of them has the coronavirus, it will be transmitted to the other.

Give an $O(n \log^2 n)$ algorithm to determine the minimum possible number of days before it is possible that *everyone* has the coronavirus.