CS2040 Tutorial 11

Week 13, starting 7 Nov 2022

Q1 Networks

There a **V** machines (e.g. x-ray, MRI, …) in Lion University, and you have information about the cost of linking **E** possible pairs of machines up into the same network (unlike real life, assume for simplicity that the cost of such linking is only dependent on the 2 machines in the pair and not on other machines in the same network).

You would like to link all **V** machines up into ONE network with cheapest total cost of links, but you are not sure whether this can be done. For example, machine **A** may not be compatible to be linked up with machine **B**, so (**A**, **B**) will not be in **E**.

To make things worse, you have a positive integer budget **B** which you cannot exceed. Still, you want to prioritize forming as few networks as possible, and secondarily, having the cheapest total cost of links to form those networks.

Design an algorithm to **efficiently** find and output the number of networks you will end up with (**V** networks in the worst case, 1 network in the best case), as well as the total cost of building such a network. What is the time complexity of your algorithm?

Q2 Minimax Path

Try solving Tutorial 11 Q1e using a solution that involves a minimum-spanning-tree algorithm in O(**RC**log(**RC**)) time.

Question 3 (Online Discussion) – BambooBear

You are working in BambooBear, which has *bases* in **K** cities and delivers bamboo to **V**-**K** different cities without *bases*. Transporting bamboo across city **A** to city **B** will incur some positive integer *overhead*, and the *overhead* from city **A** to city **B** is the same as the *overhead* from **B** to **A**. There is no *overhead* incurred for delivering bamboo within the same city.

Each of the **K** bases should be *responsible* for delivering bamboo to some set of cities. Some may only be *responsible* over its own city, while others are *responsible* for two or more cities – they deliver to one or more other cities without *bases*.

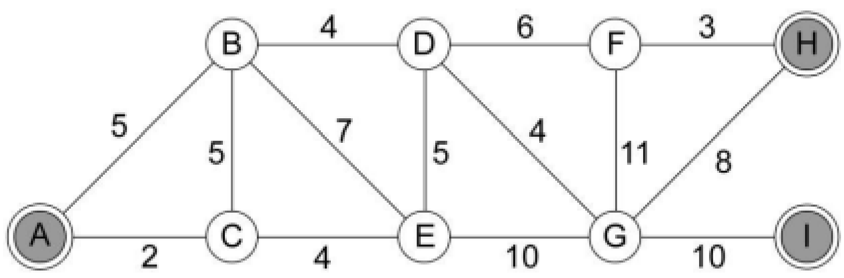
For each *base*, the *expense* for that *base* would be the sum of the *overheads* between roads that need to be travelled on among cities that the *base* is *responsible* for. The goal is to **minimize the total *expense*** of all *bases*.

You are given **V**, the ids of the **K** cities with *bases*, as well as **E** pieces of information about the *overhead* from one city to another. It is guaranteed that the 2 cities in one piece of information will be distinct among the **E** pieces of information.

Design an algorithm that **efficiently** finds and outputs, for each *base*, the *expense* of that *base*, as well as the number of cities it is *responsible* for, in O(**E** log **V**) time.

In the example below, there are:

- 9 *cities* A..H
- 3 *bases* A, H, I
- 14 pieces of information about *overheads* between cities



To minimize the total *expense*:

- only 6 out of the 14 possible roads should be utilized
- city A should be *responsible* for 6 cities, *expense* being 19
- city H should be *responsible* for 2 cities, *expense* being 3
- city I should be *responsible* for only 1 city (itself), 0 *expense*
- total *expense* is 22