CSE 421 Intro to Algorithms Winter 2020

Homework 2

Due Friday January 22, 6:00 pm PST

This problem set requires more thinking than the 1st problem set did, so it is especially important to start early.

Problem 1: In Mathematics and theoretical Computer Science, people proudly mention their *Erdős number* - how far they are away in the co-authorship graph from Paul Erdős - who was famous for the wide range of his collaborations. (He died in 1996 at age 83, but some of the more than 1500 papers on which he is a co-author have publication dates as recent as 2015.) Similarly, starting in the 1990's, people who noticed that Kevin Bacon was showing up in a lot of movies introduced the *Six Degrees of Kevin Bacon* game in which people would try to find a shortest path of co-appearances linking some arbitrary actor to Kevin Bacon. (There are still websites that can give you the answers.) In both of these situations there is a graph of people and edges between them if they are socially connected in some way. Of course a single short path between **v** and **w** does not necessarily describe a strong connection between them; it could be merely coincidental. However, if there are a lot of ways in which two people are connected by a shortest path then that suggests a much stronger connection. Suppose that the task instead is to compute the *number of shortest paths* between **v** and **w**.

It turns out that this can be done efficiently. Suppose that you are given a undirected graph **G=(V,E)** and two nodes **v** and **w** in **G**. Give an algorithm that computes the number of shortest paths between **v** and **w** in **G** in time **O(n+m)** where **n** is the number of vertices and **m** is the number of edges in **G**. (The algorithm will not be able to exactly list all such paths since there may be many more of them than the time bound.) Justify the running time bound of your algorithm.

Problem 2: Your company has been consulted by Homeland Security to do risk analysis for (undirected) computer networks. In particular, Homeland Security is interested in:

- determining whether there are single points of network failure **v** in the following sense, if node **v** stopped working then an emergency broadcast message sent from one of the nodes other than **v** would not make it to all the remaining nodes in the network, and

- finding all single points of failure exist, so that they can be corrected.

The issue seems familiar and you recall that your Algorithms instructor suggested that an extension of recursive depth-first search will do the job for problems like this.

Show how to modify the code for recursive depth-first search of undirected graphs to obtain an **O(n+m)** time algorithm that (i) assigns each node **v** a number, **dfsnum(v)**, indicating a sequence number for when it was first visited by depth-first search, and (ii) computes **smallest(v)** for each node **v**, the smallest dfsnum of any node that was encountered in the recursive call **DFS(v)**.

Show how to use this to determine whether or not a node **v** (other than the root of the DFS) is a single point of network failure in a connected network by comparing **dfsnum(v)** and **smallest(w)** for the children **w** of **v** in the DFS tree. Give a separate simpler condition for the root of the DFS.

Problem 3: You have one photocopier and you are given a set of **n** photocopy jobs from different customers to be run on your copier, each job **i** consisting of a number of original pages and a number of copies to be printed, and hence a predictable time $\mathbf{t_i}$ to run the job. You must service all **n** jobs. In order to make many of your customers happy, you decide want to produce a schedule to minimize the *average* finishing time over all **n** jobs.

> Find an efficient algorithm that does this, assuming that once you start a job, you have to finish it before you can start the next one.

Problem 4 (Extra Credit): As with many modern photocopiers, you can interrupt jobs and interleave the printing on those jobs if you need to (though the total processing time will be the same for each job). How would a solution to Problem 3 change if you were allowed to switch between jobs and switching took 0 extra time? Why?

Problem 5 (Extra Credit): A *proper coloring* of an undirected graph $\boldsymbol{G = (V, E)}$ is an assignment of colors to the vertices of G so that no edge has both endpoints of the same color. In class we saw how to properly color a graph with 2 colors (red and blue), or to tell whether that is impossible, in $\boldsymbol{O(n + m)}$ time. Later in the course, we will show that the problem for 3 colors is NP-complete and so seems to be hard. In this problem, your task is simpler: You are promised that the graph can be colored with 3 colors, but you don't need to find a coloring with 3 colors; you are allowed many more colors. In particular, you will be allowed $\boldsymbol{O(\sqrt{n})}$ colors. Find an efficient algorithm to find such a coloring for any 3-colorable graph.

Hint: Figure out how to use a small number of colors to color any vertex together with its neighboring vertices, and apply this to the neighborhoods of high degree vertices in $\boldsymbol{G}$, one at a time. Then finish up the rest of the coloring.