CS 170 HW 3

Due on 2017-02-11, at 11:59 pm

1 (★) Study Group

Please note that this homework is due 24 hours earlier than usual.

List the names and SIDs of the members in your study group.

2 (★★★) Disrupting a Network of Spies

Let $G = (V, E)$ denote the "social network" of a group of spies. In other words, G is an undirected graph where each vertex $v \in V$ corresponds to a spy, and we introduce the edge $\{u, v\}$ if spies u and v have had contact with each other. The police would like to determine which spy they should try to capture, to disrupt the coordination of the group of spies as much as possible. More precisely, the goal is to find a single vertex $v \in V$ whose removal from the graph splits the graph into as many different connected components as possible. This problem will walk you through the design of a linear-time algorithm to solve this problem. In other words, the running time will be $O(|V| + |E|)$. In the following, let $f(v)$ denote the number of connected components in the graph obtained after deleting vertex v from G. Also, assume that initial graph G is connected (before any vertex is deleted) and is represented in adjacency list format.

Prove your answer to each part (some parts are simple enough that the proof can be a brief justification; others will be more involved).

(a) Perform a depth-first search starting from some vertex $r \in V$. How could you calculate $f(r)$ from the resulting depth-first search tree in an efficient way?

(b) Suppose $v \in V$ is a node in the resulting DFS tree, but v is not the root of the DFS tree (i.e., $v \neq r$). Suppose further that no descendant of v has any non-tree edge to any ancestor of v. How could you calculate $f(v)$ from the DFS tree in an efficient way?

(c) Definition: If w is a node in the DFS tree, let $\mathsf{up}(w)$ denote the depth of the shallowest node y such that there is some graph edge $\{x, y\} \in E$ where either x is a descendant of w or $x = w$. We'll define $\mathsf{up}(w) = \infty$ if there is no edge $\{x, y\}$ that satisfies these conditions.

Now suppose v is an arbitrary non-root node in the DFS tree, with children $w_1, \ldots, w_k$. Describe how to compute $f(v)$ as a function of k, $\mathsf{up}(w_1), \ldots, \mathsf{up}(w_k)$, and $\mathsf{depth}(v)$.

Hint: Think about what happened in part (b); think about what changes when we can have non-tree edges that go up from one of v's descendants to one of v's ancestors; and think about how you can detect it from the information provided.

(d) Design an algorithm to compute $\mathsf{up}(v)$ for each vertex $v \in V$, in linear time.

(e) Describe how to compute $f(v)$ for each vertex $v \in V$, in linear time.

3 (★★★) Inverse FFT

Recall that in class we defined M_n, the matrix involved in the Fourier Transform, to be the following matrix:

$$M_n = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega & \omega^2 & \dots & \omega^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{n-1} & \omega^{2(n-1)} & \dots & \omega^{(n-1)(n-1)} \end{bmatrix}$$

For the rest of this problem we will refer to this matrix as $M_n(\omega)$ rather than M_n.

(a) Define

$$M_n(\omega^{-1}) = \begin{bmatrix} 1 & 1 & 1 & \dots & 1 \\ 1 & \omega^{-1} & \omega^{-2} & \dots & \omega^{-(n-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \omega^{-(n-1)} & \omega^{-2(n-1)} & \dots & \omega^{-(n-1)(n-1)} \end{bmatrix}$$

Recall that $\omega^{-1} = 1/\omega = \bar{\omega} = \exp(-2\pi i/n)$.

Show that $(1/n)M_n(\omega^{-1})$ is the inverse of $M_n(\omega)$, i.e. show that

$$(1/n)M_n(\omega^{-1})M_n(\omega) = I$$

where I is the $n \times n$ identity matrix– the matrix with all ones on the diagonal and zeros everywhere else.

(b) Suppose we have a polynomial $C(x)$ of degree at most $n-1$ and we know the values of $C(1), C(\omega), \dots, C(\omega^{n-1})$. Explain how we can use $M_n(\omega^{-1})$ to find the coefficients of $C(x)$.

(c) Show that $M_n(\omega^{-1})$ can be broken up into four $n/2 \times n/2$ matrices in almost the same way as $M_n(\omega)$. Specifically, suppose we rearrange columns of M_n so that the columns with an even index are on the left side of the matrix and the columns with an odd index are on the right side of the matrix (where the indexing starts from 0). Show that after this rearrangement, $M_n(\omega^{-1})$ has the form:

$$\begin{bmatrix} M_{n/2}(\omega^{-1}) & \omega^{-j}M_{n/2}(\omega^{-1}) \\ \\ M_{n/2}(\omega^{-1}) & -\omega^{-j}M_{n/2}(\omega^{-1}) \end{bmatrix}$$

As in class, the notation $\omega^{-j}M_{n/2}(\omega^{-1})$ is used to mean the matrix obtained from $M_{n/2}(\omega^{-1})$ by multiplying the j^{th} row of this matrix by ω^{-j} (where the rows are indexed starting from 0). You may assume that n is a power of two.

4 (★★) The Greatest Roads in America

Arguably, one of the best things to do in America is take a great American road trip. And in America there are some amazing roads to drive on (think Pacific Crest Highway, Route 66, etc). An intrepid traveller has choosen to set course across America in search of some amazing driving. What is the length of the shortest path that hits at least k of these amazing roads?

Assume, that the roads in America can be expressed as a directed weighted graph $G = (V, E, d)$ and that our traveller wishes to drive across at least k roads from the subset $R \subset E$ of 'amazing' roads. Furthmore, assume that the traveller starts and ends at her home $h \in V$. Also, you can assume that the traveller is OK with repeating roads from R i.e. the k roads she chooses from R do not need to be unique.

Provide a 4 part solution with runtime in terms of $n = |V|, m = |E|, k$, and $r = |R|$.

Hint: First try out $k = 1$. How can G be modified so that we can use a 'common' algorithm to solve the problem?

5 (★★★) Activity Selection

Assume there are n activities each with its own start time a_i and end time b_i such that $a_i < b_i$. All these activities share a common resource (think computers trying to use the same printer). A feasible schedule of the activities is one such that no two activities are using the common resource simultaneously. Mathematically, the time intervals are disjoint: $(a_i, b_i) \cap (a_j, b_j) = \emptyset$. The goal is to find a feasible schedule that maximizes the number of activities k.

Here are two potential greedy algorithms for the problem.

Algorithm A: Select the earliest-ending activity that doesn't conflict with those already selected until no more can be selected.

Algorithm B: Select items by shortest duration that doesn't conflict with those already selected until no more can be selected.

(a) Show that Algorithm B can fail to produce an optimal output.

(b) Show that Algorithm A will always produce an optimal output.

(c) (Extra Credit) Show that Algorithm B will always produce an output greater than 1/2 the optimal output.

6 (Extra Credit Problem) Matrix Filling

(This is an *optional* challenge problem. It is not the most effective way to raise your grade in the course. Only solve it if you want an extra challenge.)

Consider the following problem: We are given an $n \times n$ matrix where some of the entries are blank. We would like to fill in the blanks so that all pairs of columns of the matrix are linearly dependent (two vectors v and u are linearly dependent if there exists some constant c such that $cv = u$) or report that there is no such way to fill in the blanks. Formulate this as a graph problem and design an $O(n^2)$ algorithm to solve it. You may assume that all the non-blank entries in the matrix are nonzero.