CS 170 HW 3

Due on 2018-09-09, at 9:59 pm

1 (★) Study Group

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), has at least two vertices, and is represented in adjacency list format.

For each part, prove that your answer is correct (some parts are simple enough that the proof can be a brief justification; others will be more involved).

(a) Let T be a tree produced by running DFS on G with root $r \in V$. (In particular, $T = (V, E_T)$ is a spanning tree of G.) Given T, find an efficient way to calculate $f(r)$.

(b) Let $v \in V$ be some vertex that is not the root of T (i.e., $v \neq r$). Suppose further that no descendant of v in T has any non-tree edge (i.e. edge in $E \setminus E_T$) to any ancestor of v in T. How could you calculate $f(v)$ from T in an efficient way?

(c) For $w \in V$, let $D_T(w)$ be the set of descendants of w in T including w itself. For a set $S \subseteq V$, let $N_G(S)$ be the set of *neighbours* of S in G, i.e. $N_G(S) = \{y \in V : \exists x \in S \text{ s.t. } \{x, y\} \in E\}$. We define $\mathsf{up}_T(w) := \min_{y \in N_G(D_T(w))} \mathsf{depth}_T(y)$, i.e. the smallest depth in T of any neighbour in G of any descendant of w in T.

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

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 which, on input G, T, computes $\mathsf{up}_T(v)$ for all vertices $v \in V$, in linear time.

(e) Given G, describe how to compute $f(v)$ for all vertices $v \in V$, 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:

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

where ω is a primitive n-th root of unity.

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

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

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

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

(b) Let A be a square matrix with complex entries. The *conjugate transpose* $A^\dagger$ of A is given by taking the complex conjugate of each entry of A^T. A matrix A is called *unitary* if its inverse is equal to its conjugate transpose, i.e. $A^{-1} = A^\dagger$. Show that $\frac{1}{\sqrt{n}}M_n(\omega)$ is unitary.

(c) 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 $C(x)$.

(d) 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:

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

The notation $\omega^{-j} M_{n/2}(\omega^{-2})$ is used to mean the matrix obtained from $M_{n/2}(\omega^{-2})$ 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 (★★) Vandermonde matrix

Recall that the Vandermonde matrix $V_n(x_1, \ldots, x_n)$ is given by

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \ldots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \ldots & x_2^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \ldots & x_n^{n-1} \end{bmatrix}$$

(a) Let $p(x) = x^3 + 2x + 1$. Use a Vandermonde matrix to find $p(0), p(1), p(4), p(7)$.

(b) Use a Vandermonde matrix to find the polynomial q of minimal degree such that $q(0) = 2, q(1) = 0, q(4) = 6, q(7) = 30$.

(c) Find $x_1, \ldots, x_n \in \mathbb{C}$ such that the FFT matrix $M_n(\omega) = V_n(x_1, \ldots, x_n)$, where ω is a primitive n-th root of unity.

5 (★★) Connectivity vs Strong Connectivity

(a) Prove that in any connected undirected graph $G = (V, E)$ there is a vertex $v \in V$ such that removing v from G gives another connected graph.

(b) Give an example of a strongly connected directed graph $G = (V, E)$ such that, for *every* $v \in V$, removing v from G gives a directed graph that is not strongly connected.

(c) Let $G = (V, E)$ be a connected undirected graph such that G remains connected after removing any vertex. Show that for every pair of vertices u, v where $(u, v) \notin E$ there exist two different u-v paths.

6 (★★★) Finding Clusters

We are given a directed graph $G = (V, E)$, where $V = \{1, \ldots, n\}$, i.e. the vertices are integers in the range 1 to n. For every vertex i we would like to compute the value $m(i)$ defined as follows: $m(i)$ is the smallest j such that vertex j is reachable from vertex i. (As a convention, we assume that i is reachable from i.) Show that the values $m(1), \ldots, m(n)$ can be computed in $O(|V| + |E|)$ time.

7 (★★★) Arbitrage

Shortest-path algorithms can also be applied to currency trading. Suppose we have n currencies $C = \{c_1, c_2, \ldots, c_n\}$: e.g., dollars, Euros, bitcoins, dogecoins, etc. For any pair i, j of

currencies, there is an exchange rate $r_{i,j}$: you can buy $r_{i,j}$ units of currency c_j at the price of one unit of currency c_i. Assume that $r_{i,i} = 1$ and $r_{i,j} \geq 0$ for all i, j.

The Foreign Exchange Market Organization (FEMO) has hired Oski, a CS170 alumnus, to make sure that it is not possible to generate a profit through a cycle of exchanges; that is, for any currency $i \in C$, it is not possible to start with one unit of currency i, perform a series of exchanges, and end with more than one unit of currency i. (That is called *arbitrage*.)

More precisely, arbitrage is possible when there is a sequence of currencies $c_{i_1}, \ldots, c_{i_k}$ such that $r_{i_1,i_2} \cdot r_{i_2,i_3} \cdot \cdots \cdot r_{i_{k-1},i_k} \cdot r_{i_k,i_1} > 1$. This means that by starting with one unit of currency c_{i_1} and then successively converting it to currencies $c_{i_2}, c_{i_3}, \ldots, c_{i_k}$ and finally back to c_{i_1}, you would end up with more than one unit of currency c_{i_1}. Such anomalies last only a fraction of a minute on the currency exchange, but they provide an opportunity for profit.

We say that a set of exchange rates is arbitrage-free when there is no such sequence, i.e. it is not possible to profit by a series of exchanges.

(a) Give an efficient algorithm for the following problem: given a set of exchange rates $r_{i,j}$ which is *arbitrage-free*, and two specific currencies s, t, find the most advantageous sequence of currency exchanges for converting currency s into currency t.

Hint: represent the currencies and rates by a graph whose edge weights are real numbers.

(b) Oski is fed up of manually checking exchange rates, and has asked you for help to write a computer program to do his job for him. Give an efficient algorithm for detecting the possibility of arbitrage. You may use the same graph representation as for part (a).

8 (★★★★) Bounded Bellman-Ford

Modify the Bellman-Ford algorithm to find the weight of the lowest-weight path from s to t with the restriction that the path must have at most k edges.

9 (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.