CMPT 307 — Data Structures and Algorithms

Exercises on Dynamic Programming and Network Flows. Due: Friday, December 6th

Reminder: the work you submit must be your own. Any collaboration and consulting outside resources must be explicitly mentioned on your submission.

1. The residents of the underground city of Zion defend themselves through a combination of kung fu, heavy artillery, and efficient algorithms. Recently they have become interested in automated methods that can help fend off attacks by swarms of robots.

   Here is what one of these robot attacks looks like.

   - A swarm of robots arrives over the course of n seconds; in the i-th second, x_i robots arrive. Based on remote sensing data, you know this sequence $x_1, x_2, \ldots, x_n$ in advance.
   - You have at your disposal an *electromagnetic pulse* (EMP), which can destroy some of the robots as they arrive; the EMP's power depends on how long it has been allowed to charge up. To make it precise, there is a function $f(\cdot)$ so that if j seconds have passed since the EMP was last used, then it is capable of destroying up to $f(j)$ robots.
   - So specifically, if it is used in the k-th second, and it has been j seconds since it was previously used, then it destroys $\min(x_k, f(j))$ robots. (After this use, it will be completely drained.)
   - We will also assume that EMP starts off completely drained, so if it is used for the first time in the j-th second, then it is capable of destroying up to $f(j)$ robots.

   The problem is: Given the data on robot arrivals $x_1, x_2, \ldots, x_n$, and given the recharging function $f(\cdot)$, choose the points in time at which you are going to activate the EMP so as to destroy as many robots as possible.

   (a) Show that the following algorithm does not correctly solve this problem, by giving an instance on which it does not return the correct answer.

   ```
   Schedule-EMP(x_1, ..., x_n)
      Let j be the smallest number for which f(j) ≥ x_j
         (If no such j exists then set j = n)
      Activate the EMP in the j-th second
      If n − j ≥ 1 then
         Continue recursively on the input x_{j+1}, ..., x_{n−j}
            (i.e. invoke Schedule-EMP(x_{j+1}, ..., x_{n−j})
   ```

   In your example, say, what the correct answer is and also what the algorithm above finds.

   (b) Give an efficient algorithm that takes the data on robot arrivals $x_1, \ldots, x_n$, and the recharging function $f(\cdot)$, and returns the maximum number of robots that can be destroyed by a sequence of EMP activations.

2. Consider a firm that trades shares in n different companies. For each pair $i \neq j$, they maintain a trade ratio r_{ij}, meaning that one share of i trades for r_{ij} shares of j. Here we allow the rate to be fractional; that is $r_{ij} = 2/3$ means that you can trade 3 shares of i for 2 shares of j.

   A *trading cycle* for a sequence of shares $i_1, i_2, \ldots, i_k$ consists of successively trading shares in company i_1 for shares in company i_2, then shares in company i_2 for shares i_3, and so on,

finally trading shares in i_k back to shares in company i_1. After such a sequence of trades, one ends up with shares in the same company i_1 that one starts with. Trading around a cycle is usually a bad idea, as you tend to end up with fewer shares than you started with. But occasionally, for short periods of time, there are opportunities to increase shares. We will call such a cycle an *opportunity cycle*, if trading along the cycle increases the number of shares. This happens exactly if the product of the ratios along the cycle is above 1. In analyzing the state of the market, a firm engaged in trading would like to know if there are any opportunity cycles.

Give an efficient algorithm that finds such an opportunity cycle, if one exists.

3. Let $G = (V, E)$ be an undirected graph with n vertices. Recall that a subset of the vertices is called an independent set if no two of them are joined by an edge. Finding large independent sets is difficult in general; but here we'll see that it can be done efficiently if the graph is simple enough.

   Call a graph $G = (V, E)$ a path if its vertices can be written as $v_1, v_2, \ldots, v_n$ with an edge between v_i and v_j if and only if the numbers i and j differ by exactly 1. With each vertex v_i, we associate a positive integer weight w_i.

   The goal is to solve the following problem: Find an independent set in a path G whose total weight is as large as possible.

   (a) Give an example to show that the following algorithm does not always find an independent set of maximum total weight.

   ```
   start with S equal to the empty set
   while some vertex remain in G
         pick a vertex v_i of maximum weight
         add v_i to S
         delete v_i and its neighbors from G
   endwhile
   return S
   ```

   (b) Give an example to show that the following algorithm also does not always find an independent set of maximum total weight.

   ```
   let S_1 be the set of all v_i where i is an odd number
   let S_2 be the set of all v_i where i is an even number
   (Note that S_1 and S_2 are both independent sets.)
   determine which of S_1 or S_2 has greater total weight, and return this one
   ```

   (c) Give an algorithm that takes an n-node path G with weights and returns an independent set of maximum total weight. The running time should be polynomial in n, independent of the values of the weights.

4. Consider the sequence alignment problem over a four-letter alphabet $\{z_1, z_2, z_3, z_4\}$, with a given gap penalty and given mismatch costs. Assume that each of these parameters is a positive integer.

   Suppose you are given two strings $A = a_1 a_2 \ldots a_m$ and $B = b_1 b_2 \ldots b_n$ and a proposed alignment between them. Give an $O(mn)$ algorithm to decide whether this alignment is the unique minimum cost alignment between A and B.

5. The edge connectivity of an undirected graph is the minimum number k of edges that must be removed to disconnect the graph. For example, the edge connectivity of a tree is 1, and

the edge connectivity of a cycle is 2. Show how the edge connectivity of an undirected graph $G = (V, E)$ can be determined by running Ford-Falkerson algorithm on at most $|V|$ flow networks, each having $O(|V|)$ vertices and $O(|E|)$ edges.

6. Suppose we are given a directed network $G = (V, E)$ with a root node r and a set of terminals $T \subseteq V$. We would like to disconnect many terminals from r, while cutting relatively few edges.

   We make this trade-off precise as follows. For a set of edges $F \subseteq E$, let $q(F)$ denote the number of nodes $v \in T$ such that there is no $r - v$ path in the subgraph $(V, E - F)$. Give a polynomial-time algorithm to find a set F of edges that maximizes the quantity $q(F) - |F|$. (Note that setting F equal to the empty set is an option.)

7. Consider a set of mobile computing clients in a certain town who each need to be connected to one of several possible base stations. We will suppose there are n clients, with the position of each client specified by its (x, y) coordinates in the plane. There are also k base stations; the position of each of these is specified by (x, y) coordinates as well.

   For each client, we wish to connect it to exactly one of the base stations. Our choice of connections is constrained in the following ways. There is a range parameter r — a client can only be connected to a base station that is within distance r. There is also a load parameter L — no more than L clients can be connected to any single base station.

   Your goal is to design a polynomial time algorithm for the following problem. Given the positions of a set of clients and a set of base stations, as well as, the range and the load parameters, decide whether every client can be connected simultaneously to a base station, subject to the range and load conditions in the previous paragraph.

8. Your friends have written a very fast piece of maximum flow code based on repeatedly finding augmenting paths as in the class. However, after you have looked at a bit of output from it, you realize that it is not always finding a flow of maximum value. The bug turns out to be pretty easy to find; your friends had not really got into the whole backward edge thing when writing the code, and so their implementation builds a variant of the residual graph that only includes the forward edges. In other words, it searches for $s-t$ paths in a graph G_f consisting only of edges e for which $f(e) < c_e$, and it terminates when there is no augmenting path consisting entirely of such edges. We will call this the Forward-Edge-Only algorithm. (Note that we do not try to prescribe how this algorithm chooses its forward edge paths; it may choose them in any fashion it wants, provided it terminates only when there are no forward edge paths.)

   It is hard to convince your friends they need to reimplement the code. In addition to its blazing speed, they claim, in fact, that it never returns a flow whose value is less than a fixed fraction of optimal. Do you believe this? The crux of their claim can be made precise in the following statement: there is an absolute constant $k > 1$ (independent of the particular input flow network), so that on every instance of the Maximum Flow problem, the Forward-Edge-Only algorithm is guaranteed to find a flow of value at least $1/k$ times the maximum flow value (regardless of how it chooses its forward edge paths).

   Decide whether you think this statement is true or false, and give a proof of either the statement or its negation.