CS170 Midterm 2 Review

Overview

- Material: up to (and including) chapter 6
- Shortest Paths
- Greedy algorithms
- Dynamic programming

Problems

Problem 1 *Given a weighted graph $G = (V, E)$, find the maximum capacity path between all pairs of vertices. For each path, define the capacity of the path as the minimum edge weight on the path. Assume all edge weights are greater than 0, so for unconnected nodes, the maximum capacity would be 0.*

Subproblems Define subproblem $C(i, j, k)$ as the maximum capacity of a path from i to j that only uses nodes 1 through k. There are $O(n^3)$ subproblems.

Algorithm and Recursion The recursion is as follows: $C(i, j, k) = max\{min\{C(i, k, k-1), C(k, j, k-1)\}, C(i, j, k-1)\}$. The algorithm will initialize $C(i, j, 0) = w_{ij}$ for all pairs (i, j) for which there is an edge between i and j and 0 otherwise.

Running Time There are $O(n^3)$ subproblems and each takes constant time to solve, so the algorithm takes $O(n^3)$ time.

Problem 2 *Given a binary tree $G = (V, E)$ with root v_0, find the cost of the minimum spanning tree T that contains v_0 and has $k-1$ other vertices (k vertices total).*

Subproblems Define subproblem $C(v, i)$ is the cost of the optimal tree that starts at v and has i vertices.

Algorithm and Recursion $C(v, i) = min\{c(v_L, i-1) + w_{v,v_L}, c(v_R, i-1) + w_{v,v_R}, min_{1 \le x \le i-2}\{C(v_L, x), C(v_R, i-x-1)\} + w(v, v_R) + w(v, v_L)\}$ where v_L is the left child of v and v_R is the right child of v. The algorithm will initialize $C(v, 1) = 0$ for all v. We can update in order of increasing i.

Running Time There are $O(nk)$ subproblems and each takes $O(k)$ time to solve, so the run time is $O(nk^2)$.

Problem 3 *Alice wants to eat exactly 1 tomato every day. On day i, the cost of one tomato is c_i but a tomato will spoil after d days. Given the prices for the next n days, you want to come up with the optimal cost to purchase tomatoes for this period.*

We will use a greedy algorithm. The idea of the algorithm is to determine when we should buy a tomato for each given day by looking at the d days before. Let $B(i)$ be the number of tomatoes to buy on day i and initialize $B(i)$ to 0 for all i. For all i from 1 to n, look at $c_{i-d'+1}, c_{i-d'+2}, ..., c_i$ and find the minimum cost c_j, where $d' = min(d, i)$. Then $B(j) = B(j) + 1$. The run time here is $O(nd)$.

Problem 4 *You have n seats in a restaurant, and you want to use as many of those seats as possible. You can seat a group if the size of the group is less than or equal to the number of remaining seats. Say k groups show up, and $g_1, ..., g_k$ represent the sizes of these groups. Let OPT be the number of people seated in the optimal strategy. Let S be the number of people seated when we seat the groups in decreasing order (so the largest group first). Show that $S \geq \frac{OPT}{2}$.*

We will prove the following statement: With the strategy corresponding to S, either we seat all groups of size $\leq n$ or we seat more than $\frac{n}{2}$ people. This statement will finish the question, since the first case means that $S = OPT$ and the second case shows that $S \geq \frac{OPT}{2}$. The proof follows. Assume we do not seat all groups of size $\leq n$. Let i be the first group with $g_i \leq n$ that is not seated. Now there are two cases. In the first case, $g_i \leq \frac{n}{2}$. In the second case, $\frac{n}{2} < g_i \leq n$. In the first case, the number of empty seats is $< \frac{n}{2}$, which means that we seat more than $\frac{n}{2}$. In the second case, since group i is not seated, we know that a group with larger size is seated because of the seating strategy we are using, so we must seat more than $\frac{n}{2}$ people.

Problem 5 *We are given a matrix A, where $A[i,j]$ represents the element in the i^{th} row and j^{th} column, such that for all rows $A[i,j] = \infty$ for all but 10 entries. We have an operation $P(i,x)$ such that for all j, $P(i,x)$ maps $A[i,j]$ to $A[i,j]+x$ and $A[j,i]$ to $A[j,i]-x$. Is there a sequence of operations such that $A[i,j] \geq 0$ for all i,j?*

First note that $P(i,x)$ followed by $P(i,y)$ is the same as $P(i,x+y)$, and $P(i,x)$ followed by $P(j,y)$ is the same as $P(j,y)$ followed by $P(i,x)$. So our sequence, if it exists, can be written as $P(1,x_1)$, $P(2,x_2)$, ..., $P(n,x_n)$. This sequence will map $A[i,j]$ to $A[i,j]+x_i-x_j$, and we want to find numbers x_i and x_j such that $A[i,j]+x_i-x_j \geq 0$. Rearranging the previous inequality, we obtain $x_j \leq x_i + A[i,j]$. This inequality holds if we think of x_j as the shortest path from a node v to a node j, and let $A[i,j]$ represent the weight of the edge between i and j. Build a directed graph G with $V = \{1,...,n\}$ and with edge weights given by $A[i,j]$. Add a vertex v_0 with edges to each vertex in v and $w_{v_0,i} = 0$ for all i. We can use Bellman Ford to find the shortest path from v_0 to all vertices in V, and we will then obtain x_i for all i. The x_i we obtain will obey the relationship determined by the above inequality. If there is no negative cycle in G, the sequence will exist.