CS 473: Fundamental Algorithms, Spring 2013

Discussion 6

February 18, 2013 Version: **1.11**

6.1. Longest Common Subsequence.

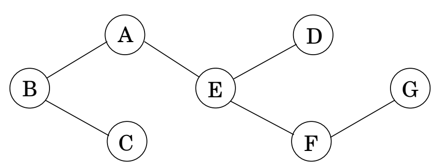
Let $X[1...m]$ and $Y[1...n]$ be two arbitrary arrays. A ***common subsequence*** of X and Y is another sequence that is a subsequence of both X and Y. Describe an efficient algorithm to compute the length of the *longest* common subsequence of X and Y.

A subsequence is anything obtained from a sequence by extracting a subset of elements, but keeping them in the same order; the elements of the subsequence need not be contiguous in the original sequence. For example, the strings `C`, `DAMN`, and `YAIOAI`, and `DYNAMICPROGRAMMING` are all subsequences of the sequence `DYNAMICPROGRAMMING`.

6.2. Minimum Weight Vertex Cover in Trees.

Given a graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$, a ***vertex cover*** of G is a subset $S \subseteq \mathsf{V}$ of vertices such that, for each edge $e = uv$ in G, u or v is in S. That is, the vertices in S ***covers*** all the edges. It is known that finding the minimum size vertex cover is NP-Hard in general graphs but it can be solved in trees using dynamic programming.

This is the goal of this problem. Given a tree $T = (\mathsf{V}, \mathsf{E})$ and a non-negative weight $w(v)$ for each vertex $v \in \mathsf{V}$, give an algorithm that computes the minimum weight vertex cover of T. In the tree below, $\{B, E, G\}$ is a vertex cover while $\{C, E, F\}$ is not a vertex cover. It is helpful to root the tree.



6.3. Covering points by intervals.

Consider the problem of covering numbers by intervals. Specifically, assume that you are given a set P of n points/numbers on the real lines, and a set of intervals $\mathcal{F}$. The purpose is to find the minimum weight set of intervals that covers all the points of P.

Suppose all intervals have the same weight. Design a greedy algorithm for this special case. How fast can you implement this algorithm? Prove the correctness of your algorithm.

How would you solve the weighted version? How fast is your algorithm?

6.4. Loopy loop cycles in a graph.

Given a parameter k, and a directed graph $\mathsf{G} = (\mathsf{V}, \mathsf{E})$ with weights on the edges (the weights might be negative), describe an algorithm for deciding if the graph has a negative (simple) cycle of length k or less. What is the running time of your algorithm?

Describe an algorithm for the case where the graph is undirected. What is the running time of your algorithm in this case?