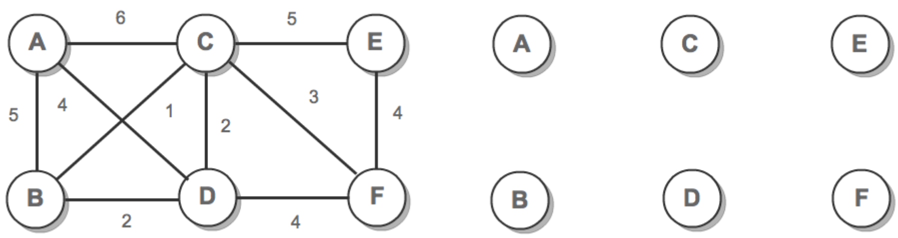
CS 473: Fundamental Algorithms, Fall 2011

Discussion 7

March 8, 2011

7.1 Minimum Spanning Tree. Consider the following graph:



(A) Draw the edges in the Minimum Spanning Tree for the following graph.

(B) Given G and MST T, suppose you decrease the weight of an edge e not in T. Give an algorithm to recompute the MST in $O(n)$ time.

7.2 Borøuvka's algorithm.

Borøuvka's algorithm computes the MST of a graph $G = (V, E)$, by repeatedly picking for every vertex in the graph the cheapest edge adjacent to it. Let $F \subseteq E$ be the set of edges picked by this process. The Borøuvka's algorithm then collapse every connected component of (V, F) into a single vertex. It continues this process iteratively till remaining with a single vertex. The set of edges picked formed the required MST.

Formally, the collapsing of the graph is done as follows: An edge in the original graph that connects two vertices in the same connected component disappears in the new graph. And edge of the original graph that connects two different connected components, now connects the two respective connected components. Naturally, if there are several edges connecting the same pair of connected components, we remember only the cheapest one.

(A) Show how to compute the collapsed graph in linear time (i.e., $O(|V| + |E|)$), for any set of edges $F \subseteq E$.

(B) Show that Borøuvka's algorithm decreases the number of vertices by two at each iteration.

(C) Conclude that Borøuvka's algorithm takes $O((n+m)\log n)$ time in the worst case. Why is the running time not $O(n \log n + m)$?

7.3 Stock Picking.

You have a group of investor friends who are looking at n consecutive days of a given stock at some point in the past. The days are numbered. $i = 1, 2, \ldots, n$. For each day i, they have a price $p(i)$ per share for the stock on that day.

For certain (possibly large) values of k, they want to study what they call *k-shot strategies*. A k-shot strategy is a collection of m pairs of days $(b_1, s_1), \ldots, (b_m, s_m)$, where $0 \le m \le k$ and

$$1 \le b_1 < s_1 < b_2 < s_2 \cdots < b_m < s_m \le n.$$

We view these as a set of up to k nonoverlapping intervals, during each of which the investors buy 1,000 shares of the stock (on day b_i) and then sell it (on day s_i). The *return* of a given k-shot strategy is simply the profit obtained from the m buy-sell transactions, namely,

$$1000 \cdot \sum_{i=1}^{m} (p(s_i) - p(b_i)).$$

(A) Design an efficient algorithm that determines, given the sequence of prices, the k-shot strategy with the maximum possible return. Since k may be relatively large, your running time should be polynomial in both n and k.

(B) Now, modify your algorithm to only use $O(n)$ space.

7.4 Set cover and the greedy algorithm.

Given a set U, and a family of subsets $\mathcal{F}$, the ***set cover*** problem asks for the minimum number of sets in $\mathcal{F}$ that fully cover U. For example, here is an instance of set cover, with the ground set being

$$U = \{1, 2, 3, 4\}.$$

and the family of sets being

$$\mathcal{F} = \Big\{ A = \{1\}, B = \{1, 3\}, C = \{2, 4\}, D = \{2\}, E = \{3\}, F = \{4\} \Big\}.$$

Valid covers of U might be A, D, E, F as $A \cup D \cup E \cup F = U$, and it is in this case of size 4. In this specific case, the best set cover is B, C as it is made of two sets of $\mathcal{F}$.

The greedy algorithm for set cover always picks the set covering the largest number of elements not covered yet into the set cover. Show an example, where this greedy algorithm fails. Show an example where for a set of size n, the greedy algorithm outputs a set of size $\Theta(\log n)$, by the optimal cover is made out of two sets!