CS 473: Fundamental Algorithms, Fall 2014

Discussion 6

October 7, 2014

6.1 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 $0 \leq m \leq k$ and

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

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.

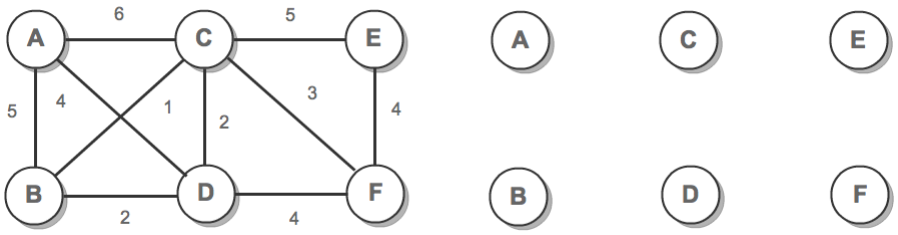
6.2 Weighted Scheduling.

We have n jobs $J_1, J_2, \ldots, J_n$ which we need to schedule on a machine. Each job J_i has a processing time t_i and a weight w_i. A schedule for the machine is an ordering of the jobs. Given a schedule, let C_i denote the finishing time of job J_i. For example, if job J_j is the first job in the schedule, its finishing time C_j is equal to t_j; if job J_j follows job J_i in the schedule, its finishing time C_j is equal to $C_i + t_j$. The weighted completion time of the schedule is $\sum_{i=1}^{n} w_i C_i$.

(A) For the case when $w_i = 1$ for all i, show that choosing the shortest job first is optimal.

(B) Give an efficient algorithm that finds a schedule with minimum weighted completion time given arbitrary weights.

6.3 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.

6.4 Borůvka's algorithm.

Borůvka'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 Borůvka'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 Borůvka's algorithm decreases the number of vertices by two at each iteration.
(C) Conclude that Borůvka's algorithm takes $O((n + m) \log n)$ time in the worst case. Why is the running time not $O(n \log n + m)$?