Problem Set 11

Data Structures and Algorithms, Fall 2022

Due: December 8, in class.

Problem 1

Not just any greedy approach to the activity-selection problem produces a maximum-size set of mutually compatible activities. For each of the following greedy strategies, either prove it is correct (i.e., produces an optimal solution), or give a counterexample.

(a) Select the activity of least duration from among those that are compatible with previously selected activities, then recurse.

(b) Select the last activity to start that is compatible with all previously selected activities, then recurse.

(c) Select the compatible activity that overlaps the fewest other remaining activities, then recurse.

(d) Select the compatible remaining activity with the earliest start time, then recurse.

Problem 2

Suppose you are given a set $S = \{a_1, a_2, \cdots, a_n\}$ of tasks, where task a_i requires p_i units of processing time to complete, once it has started. You have one computer on which to run these tasks, and the computer can run only one task at a time. Let c_i be the completion time of task a_i, that is, the time at which task a_i completes processing. Your goal is to minimize the average completion time, that is, to minimize $(1/n) \cdot \sum_{i=1}^{n} c_i$. For example, suppose there are two tasks, a_1 and a_2, with $p_1 = 3$ and $p_2 = 5$, and consider the schedule in which a_2 runs first, followed by a_1. Then $c_2 = 5$, $c_1 = 8$, and the average completion time is $(5 + 8)/2 = 6.5$. If task a_1 runs first, however, then $c_1 = 3$, $c_2 = 8$, and the average completion time is $(3 + 8)/2 = 5.5$.

(a) Devise an efficient algorithm that schedules the tasks so as to minimize the average completion time. Each task must run non-preemptively, that is, once task a_i starts, it must run continuously for p_i units of time. Prove that your algorithm minimizes the average completion time, and analyze the running time of your algorithm.

(b) Suppose now that the tasks are not all available at once. That is, each task cannot start until its release time r_i. Suppose also that we allow preemption, so that a task can be suspended and restarted at a later time. For example, a task a_i with processing time $p_i = 6$ and release time $r_i = 1$ might start running at time 1 and be preempted at time 4. It might then resume at time 10 but be preempted at time 11, and it might finally resume at time 13 and complete at time 15. Task a_i has run for a total of 6 time units, but its running time has been divided into three pieces. In this scenario, a_i's completion time is 15. Devise an efficient algorithm that schedules the tasks so as to minimize the average completion time in this new scenario. Prove that your algorithm minimizes the average completion time, and analyze the running time of your algorithm.

Problem 3

Trimedia Disks Inc. has developed "ternary" hard disks. Each cell on a disk can now store values 0, 1, or 2 (instead of just 0 or 1). To take advantage of this new technology, provide a modified Huffman

algorithm for compressing sequences of characters from an alphabet of size n, where the characters occur with known frequencies $f_1, f_2, \cdots, f_n$. Your algorithm should encode each character with a variable-length codeword over the values 0, 1, 2 such that no codeword is a prefix of another codeword and so as to obtain the maximum possible compression. Prove that your algorithm is correct.

Problem 4

Let X be a set of n intervals on the real line. A proper coloring of X assigns a color to each interval, so that any two overlapping intervals are assigned different colors. Devise an efficient greedy algorithm to compute the minimum number of colors needed to properly color X. Assume that your input consists of two arrays $L[1 \cdots n]$ and $R[1 \cdots n]$, representing the left and right endpoints of the intervals in X. You do *not* need to argue the correctness of your algorithm, but you need to analyze its time complexity.

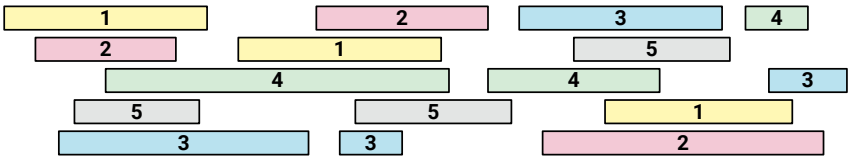


Figure 1: A proper coloring of a set of intervals using five colors.

Bonus Problem

Consider the following generalized set cover problem: Given a universe U of n elements, a collection of subsets of U, $\mathcal{S} = \{S_1, \cdots, S_k\}$, and a cost function $c : S \to \mathbb{Q}^+$, find a minimum cost sub-collection of $\mathcal{S}$ that covers all elements of U.

This problem is suspected to be hard, in the sense that there might not exist a polynomial (with respect to the length of the input) time algorithm that can solve the problem exactly. However, good approximation algorithms do exist for this problem. In particular, if the optimal solution incurs a cost of OPT, we can efficiently find a solution that costs at most $O(OPT \cdot \ln n)$.

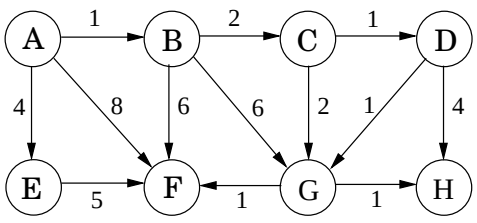
(a) Devise one such algorithm. Prove your algorithm indeed gives a solution that costs at most $O(OPT \cdot \ln n)$, and the runtime of your algorithm is polynomial with respect to the input length.

(b) Prove that the approximation ratio is asymptotically tight. That is, give a problem instance in which your algorithm picks at least $\Omega(OPT \cdot \ln n)$ subsets.

Problem 5

(a) Give a weighted, directed graph $G = (V, E)$ with no negative-weight cycles. Use this graph to demonstrate minimum spanning trees and shortest path trees are not necessarily identical.

(b) Suppose Dijkstra's algorithm is run on the following graph, starting at node A. (I) Draw a table showing the intermediate distance values of all the nodes at each iteration of the algorithm. (II) Show the final shortest-path tree.



Problem 6

You are given a set of cities, along with the pattern of highways between them, in the form of an undirected graph $G = (V, E)$. Each stretch of highway $e \in E$ connects two of the cities, and you know its length in miles, l_e. You want to get from city s to city t. There's one problem: your car can only hold enough gas to cover L miles. There are gas stations in each city, but not between cities. Therefore, you can only take a route if every one of its edges has length $l_e \leq L$.

(a) Given the limitation on your car's fuel tank capacity, show how to determine in $O(|V| + |E|)$ time whether there is a feasible route from s to t. You do *not* need to prove the correctness of your answer.

(b) You are now planning to buy a new car, and you want to know the minimum fuel tank capacity that is needed to travel from s to t. Give an $O((|V| + |E|) \log |V|)$ time algorithm to determine this. You do *not* need to prove the correctness of your answer.