Homework 3

Due: October 5, 2021 at 14:30 ET

This homework must be typed in LaTeX and handed in via Gradescope.

Please ensure that your solutions are complete, concise, and communicated clearly. Use full sentences and plan your presentation before you write. Except in the rare cases where it is indicated otherwise, consider every problem as asking you to prove your result.

Problem 1

Generalize Huffman's algorithm to ternary codewords (i.e., codewords using the symbols 0,1, and 2). Argue its correctness and optimality. Analyze its runtime.

Problem 2

a. Let $G = (V, E)$ be an undirected graph. Using the definition of a matroid, show that (E, l) is a matroid. Where $A \in l$ if and only if A is an acyclic subset of E.

b. Let $w : E \to R+$ be a function that assigns a non-negative weight to each element of E. Give an efficient algorithm to find an acyclic subset of E of maximum weight. Analyze runtime and argue correctness/optimality.

Problem 3

Suppose you are given a set $S = \{a_1, a_2, ..., a_n\}$ of tasks, where task a_i requires p_i units of processing time to complete, once it has started. You have on 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 the processing. Your goal is to minimize the average completion time, that is, to minimize $(1/n)\sum_{i=1}^{n} c_i$. 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 $(5 + 8)/2 = 6.5$.

a. Give an 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 is started, it must run continuously for p_i units of time. Prove that your algorithm minimizes the average completion time, and state the running time of your algorithm.

b. Suppose now that the tasks are not all available at once. That is, each task has a ***release time*** r_i before which it is not available to be processed. 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 $p_i = 6$ may start running at time 1 and be preempted at time 4. It can then resume at time 10 but be preempted at time 11 and 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. We say that the completion time of a_i is 15. Give an 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 state the running time of your algorithm.

Problem 4 - Multiple Classes Fractional Knapsack

Consider a variation of the Fractional Knapsack discussed in class, for which each item in S is assigned to a single *class* $\{1, 2, \ldots, k\}$. That is each item in S is a triple of values (b_i, w_i, c_i) where b_i is the benefit of the i-th item, w_i is its weight, and c_i is its class.

The goal of the problem is still to select items for our knapsack in a way that maximizes the overall benefit without exceeding a given allowable weight $W > 0$. However, only one item may be chosen among those assigned to the same subclass. Recall, that in this *fractional* variant, it is possible to select an arbitrary fraction of each item.

Consider the following algorithm (note that the algorithm returns a list of (element, fraction of element taken) tuples):

- For each item in S compute its value as $v_i = b_i/w_i$.
- For each of the k classes, sort the items in each class in non-increasing order of their value v_i.
- Let S_j, for $j \in \{1, 2, \ldots, k\}$ denote the item of maximum value among those in the class j. If there are multiple items in j with the same, maximum value pick the one with maximum weight.
- Invoke the procedure MCFknapsack$(\{S_1, S_2, \ldots, S_k\}, W)$.
    1. If $W = 0$, return $\emptyset$.
    2. Otherwise, pick the item with highest value among those in $\{S_1, S_2, \ldots, S_k\}$. Let e denote such element, let w_e denote its weight, and let j denote its class.
    3. return $\{(e, \min(1, W/w_e))\} \cup$ MCFknapsack$(\{S_1, S_2, \ldots, S_k\} \setminus S_j, \max\{0, W - w_e\})$

a. Is this a greedy algorithm? Motivate your answer.

b. Is this algorithm optimal? Motivate you answer.