Design of Algorithms - Homework IV

K. Subramani
LCSEE,
West Virginia University,
Morgantown, WV
{ksmani@csee.wvu.edu}

1 Instructions

(i) The homework is due on May 3, in my office.

(ii) Attempt as many problems as you can. You will get partial credit, as per the policy discussed in class.

2 Problems

1. You are given a set of n tasks $S = \{a_1, a_2, \ldots, a_n\}$, with task a_i requiring p_i units of time to complete. At any point in time, only one task can be executed. Furthermore, the tasks are non-preemptive, in that once task a_i commences, it must run uninterrupted for p_i units of time. Design an algorithm that schedules the tasks so as to minimize the *average completion time.* Argue the correctness of your algorithm. Note that if you have two tasks a_1 and a_2 with processing times 3 and 5 respectively and a_1 is scheduled before a_2, then the average completion time is $\frac{1}{2}(3 + (3 + 5))$.

2. Consider a set of n jobs $S = \{a_1, a_2, \ldots, a_n\}$, with task a_i having processing time t_i, profit p_i and a deadline d_i. There is only one machine to schedule the jobs and the jobs are non-preemptive. If job a_i completes before its deadline d_i, you receive a profit p_i; otherwise, your profit is 0. Design an algorithm to schedule the jobs so that the profit obtained is maximized. You may assume that all processing times are integers in the set $\{1, 2, \ldots, n\}$.

3. In the Fractional Knapsack problem, you are given n objects $O = \{o_1, o_2, \ldots, o_n\}$ with respective weights $W = \{w_1, w_2, \ldots, w_n\}$ and respective profits $P = \{p_1, p_2, \ldots, p_n\}$. The goal is to pack these objects into a knapsack of capacity M, such that the profit of the objects in the knapsack is maximized, while the weight constraint is not violated. You may choose a fraction of an object, if you so decide; if $\alpha_i,\ 0 \leq \alpha_i \leq 1$ of object o_i is chosen, then the profit contribution of this object is $\alpha_i \cdot o_i$ and its weight contribution is $\alpha_i \cdot w_i$. Design an algorithm for this problem and argue its correctness.

4. Consider an ordinary binary min-heap data structure that supports the instructions INSERT() and EXTRACT-MIN() in $O(\log n)$ worst-case time. Design a potential function Φ such that the amortized cost of INSERT() is $O(\log n)$ and the amortized cost of EXTRACT-MIN() is $O(1)$. Prove that your potential function is valid, over any sequence of n operations.

5. Design a data structure that supports the following two operations for a set S of integers: INSERT(S, x) inserts x into set S, and DELETE-LARGER-HALF(S) deletes the largest $\lceil \frac{|S|}{2} \rceil$ elements from S.

   Explain how to implement the data structure so that any sequence of m operations runs in $O(m)$ time.