Recitation 6

P1) A thief robbing a jewelry store finds n items. The i-th item is worth v_i dollars and weighs w_i pounds, where v_i and w_i are integers. The thief wants to take as valuable a load as possible, but he can carry at most W pounds, for some integer W. Using advanced melting tools, the thief can choose to take a fraction of an item. For example, he can take $\%42$ of a gold necklace if he wants to. Design an algorithm that, given the thief's maximum carrying capacity W and the list of n items along with their value and weight, computes the most valuable load the thief can take.

Answer

Greedy Algorithm is as follows:

1. Sort items decreasingly based on the ratio of their v_i to w_i, $(\frac{v_1}{w_1})$ is the greatest one.
2. $F = 0, i = 1, V = 0.$
3. While $F < W$
    a. If $F + w_i \leq W$,

        take the whole item i,

        $F = F + w_i, i = i + 1, V = V + v_i$.

    b. Otherwise, take $\frac{W-F}{w_i}$amount of item i,

        $F = W, i = i + 1, V = V + (\frac{W-F}{w_i})v_i.$

        Break the loop.

In order to prove the correctness of the algorithm, apply contradiction argument. Assume that the optimal solution has a greater value V, sorting items in optimal solution, find the smallest index j which the j-th item in optimal solution is different from the j-th item in greedy algorithm solution. Using an exchange argument, show that in optimal solution by replacing the j-th item with j-th item greedy algorithm solution, the value of items is strictly higher that optimal value, that is a contradiction.

P2) Let G be an arbitrary connected, undirected graph with a distinct cost $c(e)$ on every edge e. Suppose e^* is the cheapest edge in G; that is, $c(e^*) < c(e)$ for every edge $e \neq e^*$. Then there is a minimum spanning tree T of G that contains the edge e^*.

Answer

We prove that any spanning tree always contains edge e^*. Assume that there exists a spanning tree T that does not include e^*. Adding edge e^* to tree T forms a cycle C. Since e^* has the smallest cost, any other edge in the cycle has strictly greater cost. Thus, remove an arbitrary edge e other

than e^* from C. $T + e^* - e$ is a new spanning tree with a strictly lower cost than T. This is a contradiction that T is a minimum spanning tree. Thus any minimum spanning tree must have edge e^*.

P3) Suppose you are given a set $S = \{a_1, a_2, \dots, a_n\}$ 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 $\frac{1}{n}\sum_{i=1}^{i=n} c_i$. 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 $\frac{5+8}{2} = 6.5$. If task a_1 runs first, however, then $c_1 = 3$, $c_2 = 8$, and the average completion time is $\frac{3+8}{2} = 5.5$.

> Design an algorithm that schedules the tasks 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 state the running time of your algorithm.

Answer

In problems which ask to minimize/maximize the average over a fixed set of items, it is enough to minimize/maximize the total value. In this problem, we use an intuitive greedy algorithm that sorts the tasks increasingly based on their processing time. Each task with a smaller processing time will be scheduled sooner than task with greater processing time. Without loss of generality, assume that $p_1 < p_2 < \cdots < p\_n$. And the algorithm order for schedule is: tasks $1, 2, \dots, n$.

In order to prove the correctness of the greedy algorithm, again we use contradiction and exchange arguments. Let $a_1, a_2, \dots, a_n$ be the order that optimal solution schedules the tasks. Assuming optimal solution has strictly lower completion time than greedy algorithm, compare the i-th task scheduled in greedy and optimal solution, find the smallest index i such that these two are different ($a_i \neq i$). From the way greedy algorithm is defined, $p_i < p_{a_i}$. It is the case that task i has been scheduled some time later in optimal solution. Let j ($j > i$) be the order number that task i is scheduled in optimal solution. Using an exchange argument, show that if in optimal solution you swap task i and task a_i, we come up with a new solution which the completion time is strictly lower that optimal solution. This is a contradiction that optimal solution has the lowest completion time.