Algorithms Under Uncertainty : Homework 1
Full Marks : 40
Due : 11/09/2023

Solutions must be done in teams of at most two. Clearly mention the name of your partner at the top. Also, both the partners are required to submit their individual solution copies on Google Classroom. All solutions to be written in latex. No other format is acceptable. You are free to consult online resources, but must mention them. If found hiding your sources, it will be considered an act of plagiarism.

Problem 1. (10 points) Consider the following generalization of the ski rental problem. Instead of only renting or buying, you have multiple options. You are given purchase options $(d_1, p_1), \ldots, (d_k, p_k)$, where (d_i, p_i) means that you can rent a ski for d_i days by paying p_i amount of money. Assume that there is a renting option $(1, 1)$ which would mean that you can rent it for 1 day and pay 1 unit of money. You can also assume that there is a renting option (∞, B), which would mean buying it by paying B units of money.

Now the input remains the same as in ski-rental: it will snow for T days and then stop snowing thereafter. Give an efficient polynomial time algorithm to solve the off-line problem (the goal is to minimize the cost incurred by you assuming that you would want to ski on every day it snows). Give a c-competitive algorithm for this problem, where c is a constant. If it helps, you can assume that the purchase options obey economies of scale: if $d_1 < d_2 < \cdots < d_k$, then $p_1/d_1 > p_2/d_2 > \cdots > p_k/d_k$. (Hint : First try to show that it is ok to assume that $p_{i-1} < p_i/2$ for all i - in the sense that otherwise we can simply ignore the choice $d(p_{i-1}, d_{i-1})$ and still do good)

Problem 2. (10 points) Consider the same setting as above. However, the input looks different now. On every day, it may or may not snow (so it no longer looks like a sequence of days of snowing followed by a dry run). But the renting options depend on the number of days irrespective of whether it snows or not. For example, suppose you rent a ski for 3 days. Then you will need to return it after 3 days even if it does not snow on each of these three days.

Give an optimal offline algorithm for this problem. Give an online $O(k)$-competitive algorithm for this problem (where k is the number of different renting options). (Hint: first try with small values of k).

Problem 3. (5 points) Consider the following online algorithm for the list update problem: whenever an element is accessed, we swap it with the preceding element (i.e., move it one place closer to the front). Show that this algorithm has unbounded competitive ratio, i.e., for any number L, the competitive ratio of this algorithm is more than L.

Problem 4. (10 points) Consider the following algorithm for the list update problem: whenever an element is accessed, we move it half-way to the front, i.e., if it is at position k from the front, we move it to position $\lfloor k/2 \rfloor$ (assume that the first node is at position 0). What can you say about the competitive ratio (upper bound) of this algorithm ?

Problem 5. (5 points) In class, we proved that for online load balancing under restricted assignments and unit-sized jobs, the natural greedy algorithm is $\mathcal{O}(\log m)$-competitive, where m is the number of machines. Prove that the same algorithm is $\mathcal{O}(\log m)$-competitive even when size of job j is any arbitrary non-negative number p_j.