Algorithms

Assignment: Greedy Algorithms

Name: .................................................................

Id: .................................................................

Grade

Good Luck!
1. **Scheduling $n$ jobs on one machine:** There are $n$ jobs: $J_1, \ldots, J_n$ and one machine. For $1 \leq i \leq n$, job $J_i$ requests the service of the machine during the time interval: $[s_i, f_i)$ ($f_i > s_i \geq 0$). For $1 \leq i \leq n$, the profit of serving (scheduling) job $J_i$ is $p_i > 0$. At any point of time $t \geq 0$, the machine can serve at most one job (Mutual Exclusion). The goal is maximize the combined profit of all the scheduled jobs.

**A greedy scheme:** Order the jobs according to some greedy criterion and scan them following this order. Accept a job if it can be scheduled without violating the Mutual Exclusion constraint. Otherwise reject it. Decisions are irreversible: a rejected job cannot be scheduled later and an accepted job cannot be preempted.

(a) **The profit of a job is its length:** $p_i = f_i - s_i$. Intuitively, maximizing the profit is equivalent to finding a set of jobs whose combined time intervals “cover” the most the time interval $[\min_{1 \leq i \leq n} s_i, \max_{1 \leq i \leq n} f_i]$.

Find an instance for which ordering the jobs by their lengths (from the longest to the shortest) is not optimal.

For any $n \geq 4$, find an instance for which the greedy algorithm that follows this order produces a schedule whose profit is very close to only $\frac{1}{3}$ of the optimal profit.
(b) **The activity selection problem:** The profit of each job is 1. In this variant, maximizing the profit is equivalent to finding the largest set of jobs that can be scheduled without violating the Mutual Exclusion constraint. Note, that ordering the jobs by a non-decreasing order of the finish times produces an optimal schedule.

(i) **Ordering by the starting times:**
Find an instance for which ordering the jobs by a non-decreasing order of their starting times is not optimal.

For any \( n \geq 3 \), find an instance for which the greedy algorithm that follows this order produces a schedule whose profit is \( 1/(n - 1) \) of the optimal profit.
(ii) **Ordering by lengths:**
Find an instance for which ordering the jobs by their lengths from the shortest to the longest is not optimal.

For any \( n \geq 3 \), find an instance for which the greedy algorithm that follows this order produces a schedule whose profit is half of the optimal profit.
(iii) Ordering by degrees: The degree of a job $J_i$ is the number of jobs whose time intervals intersect with the time interval of $J_i$. Find an instance for which ordering the jobs by their degrees from the smallest to the largest is not optimal.
(c) **Arbitrary profits:** The profit of any job is arbitrary. In particular, shorter jobs may be more profitable than longer jobs.

**(i) Ordering by profits:**
Find an instance for which ordering the jobs by their profits from the most profitable to the least profitable is not optimal.

For any $n \geq 4$, find an instance for which the greedy algorithm that follows this order produces a schedule whose profit is very close to only $1/(n - 1)$ of the optimal profit.
(ii) Ordering by profits and lengths:
Find an instance for which ordering the jobs by their profit over length ratios from the largest ratio to the smallest ratio is not optimal.

For any $n \geq 2$, find an instance for which the greedy algorithm that follows this order produces a schedule whose profit could be only a very small fraction of the optimal profit.
2. **Scheduling all the $n$ jobs on several machines:** There are $n$ jobs: $J_1, \ldots, J_n$ and many machines. For $1 \leq i \leq n$, job $J_i$ must be served on one of the machines during the time interval: $[s_i, f_i)$ ($f_i > s_i \geq 0$). At any point of time $t \geq 0$, a machine can serve at most one job (Mutual Exclusion). The goal is to schedule all the jobs on as few as possible machines.

**A greedy scheme:** Order the jobs according to some greedy criterion and scan them following this order. The current considered job is scheduled on the first possible machine (first-fit-greedy). When none of the existing machines can serve this job without violating the mutual exclusion constraint, a new machine is added to serve this job.

(i) Show an instance for which ordering the jobs by a non-decreasing order of their finish times does not produce an optimal solution.
(ii) Prove that ordering the jobs by a non-decreasing order of their starting times produces an optimal solution.
3. You need to pack several items into your shopping bag without squashing anything. The items are to be placed one on top of the other. Each item has a weight and a strength, defined as the maximum weight that can be placed above that item without it being squashed. A packing order is safe if no item in the bag is squashed. That is, the strength of each item is at least the combined weight of what is placed in the bag above it.

Notations: There are \( n \) items \( X_1, X_2, \ldots, X_n \). For \( 1 \leq i \leq n \), the weight of \( X_i \) is \( w_i \) and the strength of \( X_i \) is \( s_i \). Denote item \( X_i \) by the pair \((w_i, s_i)\).

(a) Consider the following three items: (i) a loaf of bread whose weight is 5 and whose strength is 6; (ii) a carton of eggs whose weight is 4 and whose strength is 4; and (iii) a bottle of milk whose weight is 12 and whose strength is 9. Which of the six possible orderings of the bread, eggs, and milk are safe and which are unsafe.
(b) Consider packing the $n$ items in weight order, with the heaviest at the bottom. Show an example that this strategy might not produce a safe packing order, even if one exists.

(c) Consider packing the $n$ items in strength order, with the strongest at the bottom. Show an example for which this strategy does not produce a safe packing order even if one exists.
(d) Assume that you have a safe packing order in your bag in which \( X_j \) is placed directly on top of \( X_i \). Suppose also that:

\[
s_i - w_j \leq s_j - w_i.
\]

Show that if you swap \( X_i \) and \( X_j \) you still have a safe packing order.
(e) Design a greedy algorithm that produces a safe packing order if one exists. Prove that the output of your algorithm is a safe packing order.