Algorithms

Assignment: Greedy Algorithms

Name: .................................................................
Id: .................................................................

<table>
<thead>
<tr>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

Good Luck!
1. **Scheduling n jobs on one machine:**

- There are \( n \) jobs: \( J_1, \ldots, J_n \).
- Job \( J_i \) is represented by the time interval: \([s_i, f_i)\); \( f_i > s_i \geq 0 \) for all \( 1 \leq i \leq n \).
- Job \( J_i \) is associated with a profit value \( p_i > 0 \).
- At any point of time \( t \geq 0 \), the machine can serve at most one job (*Mutual Exclusion*).
- The goal is to schedule some of the jobs while optimizing a given objective function.

**A greedy scheme:**

- Order the jobs according to some greedy criterion.
- Scan the jobs according to 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.

**Remark:** If possible, illustrate your answers to the following problems.

(a) **The profit of a job is its length:** \( p_i = f_i - s_i \). The goal is to maximize the profit. This is equivalent to finding a set of jobs whose intervals "covers" the time interval the most. Find an instance for which ordering the jobs by their lengths (from the longest to the shortest) is not optimal.
   If you can, show that for infinite values of \( n \), there are instances for which this greedy algorithm produces a solution whose profit is almost one third of the optimal profit.
(b) **The activity selection problem**: The profit of each job is 1. The goal is to maximize the profit. This 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 solution.

(i) Find an instance for which ordering the jobs by a non-decreasing order of their starting times does not produce an optimal solution. If you can, show that for an infinite values of \( n \), there are instances for which this greedy algorithm produces a solution whose profit is approximately \( 1/n \) of the optimal profit.

(ii) Find an instance for which ordering the jobs by their lengths (from the shortest to the longest) does not produce an optimal solution. If you can, show that for infinite values of \( n \), there are instances for which this greedy algorithm produces a solution whose profit is approximately half of the optimal profit.

(iii) 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 job by their degrees (from the smallest to the largest) does not produce an optimal solution.
Arbitrary profits: The profit of a job is arbitrary (shorter jobs may be more profitable than longer jobs). The goal is to maximize the profit. This is equivalent to finding a set of jobs whose total profit is the largest.

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

If you can, show that for infinite values of $n$, there are instances for which this greedy algorithm produces a solution whose profit is approximately $1/n$ of the optimal profit.

(ii) 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.

If you can, show that for infinite values of $n$, there are instances for which this greedy algorithm produces a solution whose profit is very small compared to the optimal profit.
(d) **Scheduling all the \( n \) jobs on several machine:** There are \( n \) jobs: \( J_1, \ldots, J_n \). Job \( J_i \) is represented by the time interval: \( [s_i, f_i); f_i > s_i \geq 0 \) for all \( 1 \leq i \leq n \). 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 following a greedy criterion and scan them according to 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.

Show an instance for which ordering the jobs by a non-decreasing order of their finish times does not produce an optimal solution.

Prove that ordering the jobs by a non-decreasing order of their starting times produces an optimal solution.
2. 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, if, for each item, its strength 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 $s_i$.

(a) Consider the following three items: (i) a loaf of bread whose weight is 5 and whose strength is 6; (ii) 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 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 that this strategy might not produce a safe packing order, even if one exists.

(d) Assume that we have a safe packing order in our 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 we swap $X_i$ and $X_j$ we still have a safe packing order.
(e) Design a greedy algorithm to produce a safe packing order if one exists. Justify your answer