Name:

Student ID #:

- You are expected to work on this assignment on your own
- Use pseudocode, Python-like or English to describe your algorithms. Absolutely no C++/C/Java
- When designing an algorithm, you are allowed to use any algorithm or data structure we explained in class, without giving its details, unless the question specifically requires that you give such details
- Always remember to analyze the time complexity of your algorithms
- Homework has to be submitted electronically on Gradescope by the deadline. No late assignments will be accepted
Problem 1. (25 points)

Consider the Activity Selection problem we discussed in class. Suppose that instead of always selecting the first activity to finish, we instead select the last activity to start (as long as it is compatible with all the previously selected activities). In other words, instead of considering the activities by “earliest finish”, we consider them by “latest start”. Explain why this approach is a greedy algorithm, analyze its time complexity, and prove that it yields an optimal solution (greedy choice and optimal substructure).

Answer:
Problem 2. (25 points)

A server has \( n \) customer waiting to be served. The service time required by each customer is known in advance: it is \( t_i \) minutes for customer \( i \). So if, for example, the customers are served in order of increasing \( i \), then the \( i \)-th customer has to wait \( \sum_{j=1}^{i} t_j \) minutes. We want to minimize the total waiting time:

\[
T = \sum_{i=1}^{n} (\text{time spent waiting by customer } i)
\]

Give a greedy (efficient) algorithm for computing the optimal order in which to process the customers. Prove why your algorithm is correct (i.e., always returns an optimal solution).

Answer:
Problem 3. (25 points)
Assume that you are given two arrays $A = \{a_1, a_2, \ldots, a_n\}$ and $B = \{b_1, b_2, \ldots, b_n\}$ of $n$ real numbers.

1. Describe a greedy algorithm to determine an ordering of the elements of $A$ and $B$ such that $W = \sum_{i=1}^{n} |a_i - b_i|$ is minimized.

2. Analyze the time complexity of your algorithm.

3. State and prove the greedy-choice property of your algorithm.

4. State and prove the optimal substructure property of your algorithm.

**Hint:** consider using the following fact. Given real numbers $x_1 \leq x_2$ and $y_1 \leq y_2$, then

$$|x_1 - y_1| + |x_2 - y_2| \leq |x_1 - y_2| + |x_2 - y_1|.$$

**Answer:**
Problem 4. (25 points)

Given an undirected graph $G = (V,E)$, an independent set in $G$ is any set $I \subseteq V$ of vertices such that no two vertices in $I$ are connected by an edge. In the maximum independent set problem (MIS), for a given graph $G$, we want to find an independent set of maximum size. Here is our proposed greedy algorithm: (1) Set $I \leftarrow \emptyset$; (2) Repeat (3-4) until no nodes are left; (3) Choose a vertex $v$ in $G$ of minimum degree (breaking ties arbitrarily). (4) Add $v$ to $I$ and remove from $G$ vertex $v$ and all its neighbors. Does this greedy algorithm always return the optimal solution? If you think it does, give a proof for the greedy choice property. If you think it does not, give a counterexample in which it fails.

Answer: