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: In this problem we are sorting the activity by start time, and we schedule first the activity that starts last. In this way we leave as much time as possible for other activities to be scheduled. In this sense, this algorithm is greedy. The complexity of this algorithm for an input of \( n \) tasks is still \( O(n \log n) \) since we only modify the type of ordering compared to the algorithm we covered in class. The proof of optimality is almost identical to the one covered in the slides.

First, we need to prove that there is at least one optimal solution that contains the greedy choice (last start). \textbf{Proof:} Suppose \( A \) is an optimal solution for a set \( S \) of \( n \) tasks ordered by start time. Let’s also order the activities in \( A \) by start time. Let \( k \) be the last activity in \( A \). If \( k = n \), the schedule \( A \) begins with a greedy choice (latest start time). If \( k \neq n \), we show that there is another optimal solution \( B \) that begins with the greedy choice (activity \( n \)). Let \( B = (A - \{k\}) \cup \{n\} \). We need to show first that \( B \) is feasible: activities in \( B \) are non-conflicting because activities in \( A \) are non-conflicting, \( k \) is the last activity to start in \( A \) and \( s_k \leq s_n \). Since we have \( |B| = |A| \), \( B \) is optimal.

Second, we need to prove that if \( A' \) is optimal to \( S' = \{i \in S : f_i \leq s_n\} \) then \( A = A' \cup \{n\} \) is optimal to \( S \) (optimal substructure). \textbf{Proof:} By contradiction. If \( A \) is not optimal, we could a solution \( B = B' \cup \{n\} \) with more activities than \( A \). But \( B' \) is a solution to \( S' \) because the tasks in \( B' \) are non-overlapping. But \( |B'| > |A'| \) which contradicts the optimality of \( A' \).

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.

Answer: We simply proceed by a greedy strategy, by sorting the customers in the increasing order of service times and servicing them in this order. The running time is \( O(n \log n) \).

We prove the correctness directly (instead of using the greedy choice/optimal substructure proof). For any ordering of the customers, let \( \sigma(j) \) denotes the \( j \)-th customer in the ordering. Then:

\[
T = \sum_{i=1}^{n} \sum_{j=1}^{i} t_{\sigma(i)} = \sum_{i=1}^{n} (n - i + 1) t_{\sigma(i)}
\]
For any ordering, if \( t_{\sigma(i)} > t_{\sigma(j)} \) for \( i < j \), then swapping the positions of the two customers gives a better ordering (i.e., the value of \( T \) is decreased). Since we can generate all possible orderings by swaps, an ordering which has the property that \( t_{\sigma(1)} \leq \ldots \leq t_{\sigma(n)} \) must be the global optimum.

**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 an efficient 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

**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:** The greedy algorithm works as follows.

Sort both arrays \( A \) and \( B \). Suppose the new order is \( A = \{a_{\sigma(1)}, \ldots, a_{\sigma(n)}\} \) and \( B = \{b_{\sigma(1)}, \ldots, b_{\sigma(n)}\} \), where \( a_{\sigma(1)} \leq a_{\sigma(2)} \leq \ldots \leq a_{\sigma(n)} \), and \( b_{\sigma(1)} \leq b_{\sigma(2)} \leq \ldots \leq b_{\sigma(n)} \). Pick the pairs \((a_{\sigma(1)}, b_{\sigma(1)})\), \((a_{\sigma(2)}, b_{\sigma(2)})\), \ldots, \((a_{\sigma(n)}, b_{\sigma(n)})\). The algorithms runs in \( O(n \log n) \) time.

First, we have to show that a greedy choice at the first step results in an optimal solution. In other words, we prove that in the optimal solution \( O, (a_{\sigma(1)}, b_{\sigma(1)}) \) are paired. Let \( O' \) be another optimal solution where \( a_{\sigma(i)} \) is paired with \( b_{\sigma(i)} \) and \( b_{\sigma(1)} \) is paired with \( a_{\sigma(j)} \), where \( i \) and \( j \) cannot be 1. Since \( i > 1 \) and \( j > 1 \), and the arrays are sorted, then \( a_{\sigma(1)} \leq a_{\sigma(j)} \) and \( b_{\sigma(1)} \leq b_{\sigma(i)} \). But we know that given real numbers \( a_{\sigma(1)} \leq a_{\sigma(j)} \) and \( b_{\sigma(1)} \leq b_{\sigma(i)} \)
\[
| a_{\sigma(1)} - b_{\sigma(1)} | + | a_{\sigma(j)} - b_{\sigma(i)} | \leq | a_{\sigma(1)} - b_{\sigma(i)} | + | a_{\sigma(j)} - b_{\sigma(1)} |
\]
and therefore the solution in which \( (a_{\sigma(1)}, b_{\sigma(1)}) \) are paired is also optimal.

Second, we need to show that an optimal solution to the problem contains within optimal solutions to subproblems. More specifically, if \( Q \) is an an optimal solution to the subproblem that requires us to pair optimally \( A - \{a_{\sigma(1)}\}, B - \{b_{\sigma(1)}\} \) then \( Q \cup \{a_{\sigma(1)}, b_{\sigma(1)}\} \) is optimal for the original problem that require us to pair \( A, B \). Note that the objective function on \( P \) can be rewritten as follow
\[
\sum_{i=2}^{n} |a_{\sigma(i)} - b_{\sigma(i)}| + |a_{\sigma(1)} - b_{\sigma(1)}| = \sum_{i=1}^{n} |a_{\sigma(i)} - b_{\sigma(i)}| \leq \sum_{i=1}^{n} |a_i - b_i| \quad (1)
\]
Once (1) is established, the proof is by contradiction. If \( O = Q \cup \{a_{\sigma(1)}, b_{\sigma(1)}\} \) is not optimal for \( A, B \), then there is another solution \( O' = Q' \cup \{a_{\sigma(1)}, b_{\sigma(1)}\} \) for \( A, B \) such that \( \sum_{O'} |a_i - b_i| < \sum_{O} |a_i - b_i| \). Then, we could use \( Q' \) as a solution for \( (A - \{a_{\sigma(1)}\}, B - \{b_{\sigma(1)}\}) \) and \( \sum_{Q'} |a_i - b_i| < \sum_{Q} |a_i - b_i| \), which is a contradiction (because \( Q \) is optimal).
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.

Solution. This greedy algorithm does not always compute an optimal solution. Consider, for instance, the graph below.

The minimum degree is 2, so the algorithm could start, say, by choosing $A$, and remove $B$ and $H$. Then it is left with a cycle of length 5, $CDEFGC$, from which it will choose two more vertices, for the total of three vertices. But the maximum independent set has size 4, namely $\{B, D, F, H\}$. 