Answer all questions. Give informal (at least) proofs for all answers. Grading will be on completeness and logical correctness, and if applicable, efficiency, as well as correctness.

Analyzing loops-10pts

Here is an algorithm that given a directed graph \( G \) with \( n \) vertices in adjacency list format (with lists \( N(x) \) for each node \( x \) of the nodes reachable from edges coming out of the \( x \)), computes an array \( \text{Indegree}[1..n] \) of the in-degrees of each vertex, i.e., the number of edges to each node.

Give a time analysis, up to \( \Theta \), in terms of both \( n \) and \( m \) the number of edges of \( G \).

\[
\text{InDegrees}[G=(\{1,..n\},E)]: \text{directed graph in adjacency list format}
\]

1. Initialize \( \text{InDegrees}[1..n] \) to all 0.
2. FOR \( I = 1 \) to \( n \) do:
   3. FOR \( J \in N(x) \) do: \( \text{InDegrees}[J]++ \).
4. Return \( \text{InDegrees} \).

Greedy algorithms, correctness proofs, and using data structures

The following problem has three parts. Answer all parts.

You are designing software for a tour company. The tour company will be given a list of \( n \) tourists, each with a positive minimum hotel space requirement \( s_1, .. s_n \). The tour company will also have a list of \( m > n \) available hotel rooms \( R_1, .. R_m \) with each room \( R_j \) having an area \( a_j \) and a price \( p_j \). The company wants to assign each tourist \( i \) a distinct room \( room[i] \) with \( a_{room[i]} \geq s_i \) and of minimum total price, i.e., minimize \( \sum_i p_{room[i]} \).

Counter-example, 10 pts

Here is a greedy strategy that does not always solve the problem optimally.

Give an instance where it fails.

Greedy strategy A: Take the tourist with the smallest space requirement. Assign that tourist the cheapest room that meets the requirement. Remove that tourist and the assigned room and repeat until all tourists are assigned rooms. (If no such room exists, output “no legal assignments”).

Correctness proof, 10 points

Here is a greedy strategy that always solves the problem optimally.

Greedy strategy B: Take the tourist with the largest space requirement. Assign that tourist the cheapest room that meets the requirement. Remove that tourist and the assigned room and repeat until all tourists are assigned rooms. (If no such room exists, output “no legal assignments”).

Either give a proof that this strategy is correct, or fill in the missing case in the proof supplied at the end of the exam.

Data structures and efficiency, 10 points:

Give an efficient algorithm implementing greedy strategy B above, and a time analysis. Specify clearly the data structures and preprocessing used, and give pseudo-code or a clear description of all steps in terms of these data structure operations. Give an informal explanation for why your algorithm follows the given strategy. Give a complete time analysis of your algorithm. Some of your grade will be based on the efficiency of your algorithm.

A proof of correctness with one case missing.

The proof follows the transformation method.

Lemma: Let \( T_1 \) be the tourist with largest space requirement \( s_1 \), and let \( R_j \) be the room with \( a_j \geq s_1 \) of minimal price \( p_j \). If there is any assignment that meets the space requirements, then there is an optimal assignment that assigns \( T_1 \) room \( R_j \).

Proof: Assume there is an assignment that meets the space requirements, and let \( room' \) be such an assignment of minimal total price. We show that there is an optimal assignment \( room'' \) assigning \( T_1 \) to room \( R_j \), i.e., \( room''[1] = j \).

Case 1: If \( room' \) assigns \( T_1 \) to room \( R_j \), we let \( room'' \) be \( room' \).

Case 2: If \( room' \) does not assign any tourist to \( R_j \), we let \( room''[i] = room'[i] \) for \( i = 2..n \) and let \( room''[1] = j \). Since only \( T_1 \) has been reassigned, and \( R_j \) meets the space requirement for \( T_1 \), \( room'' \) meets
all space requirements. Since \( \text{room}'[1] \) meets the space requirement, and \( R_j \) is the cheapest room to meet the requirement for \( T_1 \), \( P_{\text{room}'[1]} \leq p_j \). Thus, the total price for \( \text{room}'' \) is at most that for \( \text{room}' \), so \( \text{room}'' \) is also optimal.

Case 3: Assume \( \text{room}' \) assigns some \( T_i, i \neq 1, \) to \( R_j \). (PROVIDE PROOF)

Thus is all cases, \( \text{room}'' \) is an optimal legal assignment that assigns \( T_1 \) to \( R_j \).

Theorem: If there is a legal assignment, the greedy strategy finds one of minimal cost.

We prove this by induction on \( n \). If \( n = 1 \), there is one tourist, and we place that tourist into the cheapest room that meets the requirement, so if any room meets the requirement, we find the one of minimal cost, which is also the total cost.

Assume the strategy is optimal for any set of \( n - 1 \) tourists and any set of rooms, and let \( T_1,..T_n \) be a set of \( n \) tourists (with \( s_1 \) being the largest space requirement and \( R_j \) the cheapest room that meets this requirement). The greedy assignment \( \text{rooms} \) assigns 1 to \( j \) and then recursively uses the same strategy to assign \( T_2...T_n \) into the rooms except for \( R_j \). By the induction hypothesis, \( \text{rooms}[2...n] \) is an optimal assignment for \( T_2..T_n \) in this set of rooms. If there is a legal solution, by the lemma, there is an optimal solution \( \text{rooms}''[1..n] \) that assigns \( T_1 \) room \( R_j \). Then \( \text{rooms}''[2..n] \) assigns \( T_2..T_n \) into rooms not including \( R_j \), so by the optimality of the greedy recursive solution, the total prices for \( \text{rooms}''[2]...\text{rooms}''[n] \) is at least that for \( \text{rooms}[2]...\text{rooms}[n] \). Since both \( \text{rooms} \) and \( \text{rooms}'' \) have the same first room, the same is true for their total prices. Thus, the total price for \( \text{rooms} \) is at most that for \( \text{rooms}'' \) which is the minimum possible, so \( \text{rooms} \) also has minimum possible total costs.

Thus, by induction on \( n \), the greedy strategy always finds a solution of minimal total costs (if any solution exists.)