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 $\text{InDegrees}$.

Initializing an array of length $n$ takes $\Theta(n)$ time. In the two nested loops, for each $I = 1..n$, the inner loop takes $\Theta(1 + \text{deg}(I))$ time; the 1 is because we need to verify that the list is empty even when there are no neighbors. Thus, the total time for the nested loops is $\Theta(\sum_{I=1}^{n} 1 + \text{deg}(I)) = \Theta(n + m)$ because each edge contributes one to the (out) degree on one node. Thus the total time is $\Theta(n + m)$. Since this is true on every input, we don’t need a worst-case.

**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 $s_1, s_2, \ldots, s_n$. The tour company will also have a list of $m > n$ available hotel rooms $R_1, R_2, \ldots, R_m$ 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”).

Let there be two tourists, $T_1$ requiring space 1 and $T_2$ requiring space 2, and three rooms: room 1 has space 2 and price 1, room 2 has space 1 and price 2 and room 3 has space 3 and price 4. Then the greedy algorithm above places tourist 1 in the cheapest room, room 1, and then must place tourist 2 in the only remaining room that meets their needs, room 3, for a total cost of 1+4=5. A better solution is to put tourist 1 in room 2, and tourist 2 in room 1, for a total cost of 1+2=3 < 5.

**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. (See end for missing case.)

**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. structures used.
We’ll sort the tourists by space required (largest to smallest) $T_1...T_n$, and rooms $R_1...R_n$ by size (largest to smallest). Let Available be the set of rooms that meet the current tourist’s needs and have not yet been assigned. We need to find the member of Available of smallest cost, and assign it to the current tourist. Then to update Available for the next tourist, we need to delete the used room, and add in all rooms of size between the two tourists’ requirements. We can keep track of which rooms to add just by having a pointer to where we left off in the sorted array of rooms. Then we need to keep track of Available by price, find and delete the min, and insert these new elements. These operations point to a min-heap for Available (of pairs $(j,p_j)$ ordered by $p_j$). The algorithm becomes:

AssignRooms($T[1...n]$, $R[1...m]$)
1. IF $n > m$ return “Impossible”
2. Initialize Available, a min-heap of pairs $(j,p_j)$ ordered by second co-ordinate.
3. Sort $T$ by size required, largest to smallest.
4. Sort $R$ by size, largest to smallest.
5. $J \leftarrow 1$.
6. FOR $I = 1$ to $n$ do:
7. While $J \leq n$ AND $a_j \leq s_j$ do:
8. InsertAvailable($(j,p_j)$)
9. $J++$
10. IF Available is empty, return “Impossible”
11. $(K,p_K) \leftarrow \text{MinAvailable}$
12. room[$I$] $\leftarrow K$.
14. Return room[1...n].

Since the heap is one of rooms, the size is at most $m$. Insert and delete both take $O(\log m)$ time. The times to sort are $O(n \log n)$ and $O(m \log m)$ respectively. Note that, since we never decrement $J$, each room is inserted at most once, and so we perform $m$ inserts total, and the While loop executes at most $m$ times total. Thus, the total cost of the While loop is $O(m \log m)$. Then the rest of the FOR loop involves constant time and one $\text{Delete}$ operation, so it is $O(n \log m)$ time. Thus the total time is $O(n \log n + m \log m + m \log m + n \log m) = O(m \log m)$ since we are told $m \geq n$ (and we immediately halt and reject otherwise.)

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 room’[1] meets the space requirement, and $R_j$ is the cheapest room to meet the requirement for $T_1$, $p_{room’[1]} \leq p_j$. Thus, the total price for room” is at most that for room’, so room’ is also optimal.

Case 3: Assume room’ assigns some $T_i$, $i \neq 1$, to $R_j$. PROOF PROVIDED BELOW.

Let room”[1] = $j$, let room”[i] = room’[1], and otherwise room”[k] = room’[k] for $k \neq 1, i$. In other words, we have tourist 1 and tourist $i$ switch rooms, and leave the others alone. Since we use the same set of rooms overall, this assignment is also of minimal cost. Since $T_1$ had the largest requirement, and room” must meet $T_1$’s requirement $a_{room”[1]} \geq s_1 \geq s_i$, so room”[1] meets Tourist $i$’s requirement. Since $R_i$ meets tourists $i$’s requirement, and all other tourists are in the same rooms as before, all tourist requirements are met by the
assignment $rooms''$. Thus $rooms''$ is an optimal assignment that meets all requirements and puts tourist 1 in room $i$, as claimed.

Thus is all cases, $room''$ 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 $rooms$ 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, $rooms[2..n]$ 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 $rooms''[1..n]$ that assigns $T_1$ room $R_j$. Then $rooms''[2..n]$ assigns $T_2, .., T_n$ into rooms not including $R_j$, so by the optimality of the greedy recursive solution, the total prices for $rooms''[2..n]$ is at least that for $rooms[2..n]$. Since both $rooms$ and $rooms''$ have the same first room, the same is true for their total prices. Thus, the total price for $rooms$ is at most that for $rooms''$ which is the minimum possible, so $rooms$ 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.)