1. Problem 3
In order that the greedy algorithm for box packing is optimal, we need show that the schedule (i.e. packing the boxes in the order they arrived and once the track reaches its weight start another truck) created by this algorithm always "stays ahead" of any other solution. Show that if greedy algorithm fits boxes \( b_1, b_2, \ldots, b_j \) into first \( k \) trucks and the other algorithm fits boxes \( b_1, \ldots, b_i \) into first \( k \) trucks then \( i \leq j \). Prove by induction on \( k \).

2. Problem 4
Given an algorithm that takes two sequences of events \( S \) and \( S' \), determine in \( O(m + n) \) time whether \( S' \) is a subsequence of \( S \), each sequence can possibly contain event more than once. \( S = s_1s_2\ldots s_n \) and \( S' = s'_1s'_2\ldots s'_{m} \). Sketch of the algorithm. Find first element in \( S \) which is equal \( s'_1 \). Increment pointer \( j \) of \( S' \). Then keep on incrementing the pointer in \( S \) until you find element equal \( s'_2 \). Keep on going until one of the pointers reaches the end. Each element will be visited only once and comparison operation is \( O(1) \) hence the running will be \( O(n + m) \).

3. Problem 13
An optimal algorithm is to schedule the jobs in decreasing order of \( w_i/t_i \). Prove the optimality by an "exchange argument".

4. Problem 14
The algorithm goes as follows:

1. Organized all the processes in the sequence \( S \) by non-decreasing order of their finish times.
2. While some process in \( S \) is still not covered: Insert a \textit{status check} right at the finish time of the first uncovered process.

The algorithm wo’n’t stop until all the processes are covered and it terminates since in every iteration at least one process is covered.
Next you can show that the set of \textit{status checks} computed by the algorithm has the minimum possible size, using the "stay ahead" type of argument.