

Problem 1: Suppose you have one machine and a set of \( n \) jobs \( a_1, a_2, ..., a_n \) to process on that machine. Each job \( a_j \) has a processing time \( t_j \), a profit \( p_j \) and a deadline \( d_j \). The machine can process only one job at a time, and job \( a_j \) must run uninterrupted for \( t_j \) consecutive time units. If job \( a_j \) is completed by its deadline \( d_j \), you receive profit \( p_j \), but if it is completed after its deadline, you receive a profit of 0. Give an algorithm to find the schedule that obtains the maximum amount of profit, assuming that all processing times are integers between 1 and \( n \). What is the running time of your algorithm?

Problem 2: Given a graph \( G = (V, E) \) a subset \( U \subseteq V \) of nodes is called a node cover if each edge in \( E \) is adjacent to at least one node in \( U \). Given a graph, we do not know how to find the minimum node cover in an efficient manner. But if we restrict \( G \) to be a tree, then it is possible. Give a greedy algorithm that finds the minimum node cover for a tree. Analyze its correctness and running time.

A tree is a connected graph with no cycles, i.e. where any pair of nodes is connected by a simple path.

Problem 3: Let’s revisit the binary search trees from the previous homework. They are augmented with the following information. At each node \( x \) we also store \( m(x) \): the number of nodes in the subtree rooted at \( x \) (including \( x \)). This time we relax our balance requirement to be that for every node \( x \) in the tree \( m(L(x)) \leq \alpha m(R(x)) \) or \( m(R(x)) \leq \alpha m(L(x)) \) for a constant \( 1/2 \leq \alpha < 1 \). We call these tree \( \alpha \)-balanced.

- Prove that \( h = O(\log n) \) where \( h \) is the height of a \( \alpha \)-balanced tree with \( n \) nodes.
- Show how to make an arbitrary binary search tree into an \( \alpha \)-balanced one for \( \alpha = 1/2 \)
- Start from a 1/2-balanced tree. Do insertion and deletions. When the tree is not \( \alpha \)-balanced anymore, bring it back to 1/2 balanced. Show that inserting and deleting cost \( O(\log n) \) time in an amortized sense.