1. [30 pt] Suppose you are choosing between the following three algorithms:

(a) Algorithm A solves problems by dividing them into five subproblems of half the size, recursively solving each subproblem, and then combining the solutions in linear time.

(b) Algorithm B solves problems of size \( n \) by recursively solving two subproblems of size \( n - 1 \) and then combining the solutions in constant time.

(c) Algorithm C solves problems of size \( n \) by dividing them into nine subproblems of size \( n/3 \), recursively solving each subproblem, and then combining the solutions in \( O(n^2) \) time.

What are the running times of each of these algorithms (in big-O notation), and which would you choose?

2. [40 pt] You are given a set of \( n \) jobs. Associated with job \( i \) is an integer deadline \( d_i > 0 \) and a profit \( p_i > 0 \). For any job \( i \), the profit \( p_i \) is earned if the job is completed by its deadline. To complete a job, one has to process the job on a machine for one unit of time. Only one machine is available for processing jobs. A feasible solution for this problem is a subset \( J \) of jobs such that each job in this subset can be completed by its deadline. The value of a feasible solution \( J \) is the sum of the profits of the jobs in \( J \), or \( \sum_{i \in J} p_i \). An optimal solution is a feasible solution with maximum value.

Develop the optimal greedy algorithm by following the steps we used for developing the solution for activity selection problem, showing your work at each step.

3. [30 pt] A binary tree is full if all of its vertices have either zero or two children. Let \( B_n \) denote the number of full binary trees with \( n \) vertices.

(a) By drawing out all full binary trees with 3, 5, or 7 vertices, determine the exact values of \( B_3 \), \( B_5 \), and \( B_7 \). Why have we left out even number of vertices, like \( B_4 \)?

(b) For general \( n \), derive a recurrence relation for \( B_n \).

(c) Show by induction that \( B_n \) is \( 2^{\Omega(n)} \).