After Lecture 17 & 18 Practice Problems (all taken from previous exams)

1. Which of the following problems is equivalent to the 0–1 Knapsack problem?

   a) You are given a bag that can carry a maximum weight of W. You are given N items which have a weight of $\{w_1, w_2, w_3, \ldots, w_n\}$ and a value of $\{v_1, v_2, v_3, \ldots, v_n\}$. You can break the items into smaller pieces. Choose the items in such a way that you get the maximum value.

   b) You are studying for an exam and you have to study N questions. The questions take $\{t_1, t_2, t_3, \ldots, t_n\}$ time(in hours) and carry $\{m_1, m_2, m_3, \ldots, m_n\}$ marks. You can study for a maximum of T hours. You can either study a question or leave it. Choose the questions in such a way that your score is maximized.

   c) You are given infinite coins of denominations $\{v_1, v_2, v_3, \ldots, v_n\}$ and a sum S. You have to find the minimum number of coins required to get the sum S.

   d) You are given a suitcase that can carry a maximum weight of 15kg. You are given 4 items which have a weight of $\{10, 20, 15, 40\}$ and a value of $\{1, 2, 3, 4\}$. You can break the items into smaller pieces. Choose the items in such a way that you get the maximum value.

2. A greedy algorithm can be used to solve all the dynamic programming problems.

   a) True.

   b) False.

3. All optimization problems exhibit optimal substructure.

   a) True.

   b) False.

4. Given a value N, if we want to make change for N cents, and we have an infinite supply of each of $S = \{S_1, S_2, \ldots, S_m\}$ valued coins, how many ways can we make the change? The order of coins doesn't matter, so different permutations of the same coin sets are ignored. Prove the necessary traits of the poblem to determine an algorithm.

   For example, for $N = 4$ and $S = \{1, 2, 3\}$, there are four solutions $\{1, 1, 1, 1\}$, $\{1, 1, 2\}$, $\{2, 2\}$, $\{1, 3\}$. So output should be 4. So the output should be 4. For $N = 10$ and $S = \{2, 5, 3, 6\}$, there are five solutions: $\{2, 2, 2, 2, 2\}$, $\{2, 2, 3, 3\}$, $\{2, 2, 6\}$, $\{2, 3, 5\}$, and $\{5, 5\}$. So the output should be 5.**Optimal Substructure** To count total number solutions, we can divide all set solutions in to two sets.

   1) Solutions that do not contain mth coin (of S_m).

   2) Solutions that contain at least one S_m.

   Let COUNT($S[]$, m, N) = COUNT($S[]$, $m-1$, N) + COUNT($S[]$, m, $N - S_m$).
   The problem has optimal substructure property as the problem can be solved using optimal solutions to subproblems. The order of the coins in $S[]$ is not relevant.

   Overlapping Subproblems It should be notes that the above function computes the same

subproblems again and again. See the following recursion tree $S = \{1, 2, 3\}$ and $n = 5$. The function COUNT($\{1\}$, 3) is called two times. If we draw the complete tree, then we can see that there are many subproblems being called more than once.

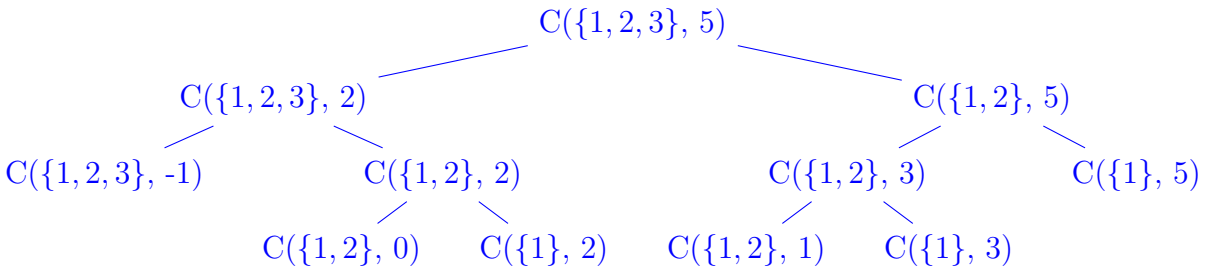

Figure 9.1:

5. Suppose you are managing the construction of billboards on a highway of length M miles. The possible sites for billboards are given by increasing mile numbers $x_1, x_2, \ldots, x_n$ each in the interval $[0, M]$. If you place a billboard at location x_i, you receive a revenue of $r_i > 0$.

   Regulations require that no two billboards be within less than or equal to 5 miles to each other. You'd like to place billboards at a subset of the sites so as to maximize your revenue, subject to this restriction.

   For example, suppose $M = 20$, $n = 4$, $\{x_1, x_2, x_3, x_4\} = \{6, 7, 12, 14\}$, $\{r_1, r_2, r_3, r_4\} = \{5, 6, 5, 1\}$. Then the optimal solution would be to place billboards at x_1 and x_3 for a total revenue of 10.

   Give an efficient algorithm that takes an instance of this problem as input and returns:

   a) the maximum total revenue that can be obtained from any valid subset of sites, and

   b) a subset of sites that achieves this maximum total revenue.

6. Suppose that we have a set of n activities, each with start time s_i and finish time f_i, to schedule among a large number of lecture halls. We wish to schedule all the activities using as few lecture halls as possible. Give an efficient algorithm to determine which activity should use which lecture hall. Prove that your algorithm is optimal and discuss the asymptotic runtime of your algorithm. DO NOT JUST USE THE GREEDY-ALGORITHM ONCE PER ROOM; there is a more efficient algorithm.