Homework #7

Grading policy: Only P1-P5 will be graded. The others are for your practice.

P1: Write a program in *bottom-up* dynamic programming to solve cutting rod problem. It should return both the optimal revenue and the corresponding decomposition. What is the output of your program with the following inputs?

1. $p_i = [1, 4, 6, 7, 12, 14, 15, 18, 20, 22, 25, 100], n = 8$
2. $p_i = [1, 4, 6, 7, 12, 14, 15, 18, 20, 22, 25, 100], n = 30$, you can assume the price for any longer rod above $len(p_i)$ is the last value, 100 in this case.

Instructor's comment: we won't ask you to write a program in the exam. But having a hand-on experience writing DP is quite important. You don't need to hand in the source codes. You can compute the solution by hand if really don't want to code.

P2: Show, by means of a counterexample, that the following "greedy" strategy does not always determine an optimal way to cut rods. Define the *density* of a rod of length i to be p_i/i, that is, its value per inch. The greedy strategy for a rod of length n cuts off a first piece of length i, where $1 \leq i \leq n$, having the maximum density. It then continues by applying the greedy strategy to the remaining piece of length $n - i$.

P3: The mergesort algorithm also uses a recursive divide-and-conquer scheme. Can we use dynamic programming and memorization to accelerate the algorithm? Why?

P4: If we add an additional constraint to the rod-cutting problem, so that we have limit l_i on the number of pieces of length i that we are allowed to produce $i = 1, 2, ..., n$. Is the optimal substructure property still hold? If yes, provide a proof. If no, provide a counterexample.

P5 Weighted interval scheduling problem: Describe the dynamic programming optimal substructure of the following problem. To be specific:

1. List what things you need to memorize in the DP table.
2. The formula/function to calculate a optimal solution.

3. Any other additional things you need to do in addition to the regular DP flow.

If you are not sure what to submit, write the pseudocode also work.

Given a set of job J. Each $j \in J$ has three attributes: s_j denotes the start time of j, f_j denotes the finish time of j, and v_j denotes the value of j.

Find the maximum value subset of jobs such that no two jobs in the subset overlap.

There is actually a Leetcode for this problem 1235. Maximum Profit in Job Scheduling

P6: 53. Maximum Subarray

P7: 1143. Longest Common Subsequence You can read CLRS Chapter 14.4 which covers this algorithm.

P8: 322. Coin Change

P9: 1626. Best Team With No Conflicts

P10: Leetcode 174. Dungeon Game