Homework 7: Recovering Solutions from Dynamic Programming Algorithms

You may work in groups, but you must write solutions yourself. List your collaborators at the top of your submission.

Instructions for submission: Submit your homework as a single PDF file to the course Gradescope page. Either scan your *legible* handwritten solutions or type them up in LaTeX.

In lecture, we've seen how to compute the optimal value of a solution using dynamic programming. In this homework, we will see how to modify a dynamic programming algorithm to also recover the optimal solution itself, not just its value. For example, in the knapsack problem, we want to find the maximum value of items we can fit in a knapsack, but we also may want to know which items to put in the knapsack to achieve that maximum value. We can recover the optimal solution through two steps:

1. Keep track of the choices made at each step of the dynamic programming algorithm.

2. Trace back through these choices to construct the optimal solution.

Recall that the rod cutting problem is defined as follows: Given a rod of length n and a list of prices p_i for $i = 1, 2, \ldots, n$, where p_i is the price of a rod of length i, determine the maximum revenue that can be obtained by cutting up the rod and selling the pieces. The dynamic programming solution to the rod cutting problem involves creating an array M where $M[j]$ represents the maximum revenue that can be obtained from a rod of length j. The algorithm iteratively fills in this array based on the prices and the previously computed values. To recover the optimal solution, we can maintain an additional array B where $B[j]$ stores the length of the first piece cut from a rod of length j that leads to the optimal revenue. After filling in the M and B arrays, we can trace back through the B array starting from $B[n]$ to determine which pieces were cut to achieve the maximum revenue.

Problem 1. Below is the pseudocode for the iterative algorithm to compute the maximum revenue for the rod cutting problem. Modify this pseudocode to keep track of the choices made at each step.

Algorithm 1 Rod Cutting Iterative Algorithm

```
1:  function RodCuttingItr(n, p)
2:      M ← array of size n + 1
3:      M[0] ← 0
4:      for j = 1 to n do
5:          ν ← 0
6:          for i = 1 to j do              ▷ Find the length of the first piece to cut
7:              if p[i] + M[j − i] > ν then
8:                  ν ← p[i] + M[j − i]
9:          M[j] ← ν
10:     return M
```

Hint: this can be done by adding three lines of code to the existing pseudocode.

Problem 2. Consider the following prices for the rod cutting problem: $p[1] = 2$, $p[2] = 5$, $p[3] = 7$, $p[4] = 8$, $p[5] = 10$. Use your modified algorithm to compute the maximum revenue for a rod of length 5 and determine which pieces to cut to achieve that maximum revenue.