Lab Week 6: Dynamic Programming

Design a dynamic programming algorithm for the following problem. Find the maximum total sale price, MTSP, that can be obtained by cutting a rod of n units long into integer-length pieces if the sale price of a piece i units long is $\mathbf{p_i}$ for i = 1, 2, …, n .

Examples

- Rod Length 4, $p_1 = 2$, $p_2 = 4$, $p_3 = 7$, $p_4 = 8$ then MTSP = 9 sell lengths (1, 3)
- Rod Length 4, $p_1 = 3$, $p_2 = 7$, $p_3 = 9$, $p_4 = 12$ then MTSP = 14 sell lengths (2, 2)
- Rod Length 4, $p_1 = 2$, $p_2 = 4$, $p_3 = 7$, $p_4 = 11$ then MTSP = 11 sell lengths (4)
- Rod Length 4, $p_1 = 3$, $p_2 = 5$, $p_3 = 8$, $p_4 = 11$ then MTSP = 12 sell lengths (1, 1, 1, 1)

The following is a rough guide to the thought process you should follow:

1. Write an English description of what you are trying to optimize along with what the parameter represents: MTSP (k)
2. Write a recurrence relation for the objective function: MTSP (k) = some function involving smaller versions of the problem. Specify the base case. This specifies the optimal substructure of the problem.
3. Determine the table you will use to keep track of the MTSP(k)? The parameter(s) in the recurrence relation most likely will determine the dimensions.
4. Fill in the table and trace back for a simple version of the problem.
5. Write pseudo code snippets to fill in the table and then to trace back using the table to find the rod sizes that give you the optimal solution.
6. Implement your solution in java.

Submit your program to PolyLearn: A Java class *RodCutter.java*. You will demo this program in lab. Your program should read in the name of a text file, either from the command line or as an argument. The text file will contain a set of problems for your program to solve, see below. The output must be as shown!

Note: The out pus shows the contents of the table that is used in your dynamic programming solution to the given problem, then shows an optimal set of rod lengths.

INPUT FILE does not contain the comments that are shown!!

```
2                                    // 2 test cases
10                                   // first case rod of length 10
2 4 4 5 12 13 14 15 40 41            // p1 = 2, p2=4 etc
16                                   // second case rod of length 16
1 4 6 25 28 31 80 81 82 83 84 85 86 88 90 92
```

```
Output:
Case 1:
total for length 1        = 2                 total for length 8        = 18
total for length 2        = 4                 total for length 9        = 40
total for length 3        = 6                 total for length 10       = 42
total for length 4        = 8                 Optimal rod cutting
total for length 5        = 12                Number of rods of length 1      = 1
total for length 6        = 14                Number of rods of length 9      = 1
total for length 7        = 16



Case2:                                        total for length 10       = 86
total for length 1        = 1                 total for length 11       = 105
total for length 2        = 4                 total for length 12       = 108
total for length 3        = 6                 total for length 13       = 111
total for length 4        = 25                total for length 14       = 160
total for length 5        = 28                total for length 15       = 161
total for length 6        = 31                total for length 16       = 164
total for length 7        = 80
total for length 8        = 81                Number of rods of length 2      = 1
total for length 9        = 84                Number of rods of length 7      = 2
```