

# Algorithms Lab 7

csci2200, Laura Toma, Bowdoin College

## In lab: (COLLABORATION LEVEL: PAIR PROGRAMMING)

Write code, in a language of your choice, to implement the rod cutting problem. You need to implement TWO functions:

- One that corresponds to solving the problem recursively without DP. You could name it `maxrev_straightforward`.
- One that corresponds to solving the problem with DP (it can be recursive or not, your choice). You could name it `maxrev_DP`.
- Each function should take  $i$  (the length of the rod) as variable and should return the optimal revenue of cutting a rod of length  $i$ .
- The array that holds the prices can be a global.
- The length of the rod  $n$  has to be read from the command line. If you use C/C++, use `argc`, `argv`. Use something similar if you use python, but make sure that the user does not need to edit your code to change  $n$ .
- Initialize the price array with random values and (assume that prices are integers), but come up with a reasonable scheme where prices are generally not decreasing (e.g.: a rod of length 10 should get a price that's the same or better than a rod of length 9).
- Your code should time the two functions and show the timings. For example when I run Ryan's code, I expect to see something like this:

```
g++ ryan.cpp -o ryan
./ryan 100
you entered n=100
running rev_dynprogr with n=100: maxrev= 128, total time = ...
running rev_straightforward with n=100: maxrev= 128, total time =
```

- Test both functions on increasing values of  $n$ , until you notice a significant difference in running time between the two methods.
- Run your code on the price array in the handout, and print how long the two functions take, and what maximum revenue you get.

# Homework problems ( COLLABORATION<sup>1</sup>LEVEL:1)

## 1. Rod cutting:

- (a) Assume you are given a price table  $p[1..n]$  where  $p[i]$  represents the revenue of selling a rod of length  $i$ . Describe a non-recursive, dynamic programming algorithm for finding the maximum revenue obtainable from a rod of given length  $n$ , and analyse its running time.
  - (b) Describe how to augment your solution for the rod cutting problem so that it computes the actual cuts corresponding to the optimal revenue, instead of just the revenue.
2. Someone suggests the following strategy as an optimization to our dynamic programming solution: when you determine the next cut, instead of trying all possibilities, go with a piece that maximizes the revenue per length. This is called a *greedy* strategy, because it greedily chooses the option that looks locally best, without exploring the choices “deeply”.

Naturally, greedy strategies always work in the sense that they always compute something — the question for a particular greedy strategy is whether it always computes the correct (optimal) solution. In this case we say that the greedy strategy is correct.

If this particular greedy strategy for rod cutting is correct, then we could solve the problem in  $O(n)$  time plus an initial sort (instead of  $O(n^2)$  with dynamic programming).

Prove that this strategy is not correct by showing a counterexample.

3. You have been hired to design algorithms for optimizing system performance. Your input is an array  $J[1..n]$  where  $J[i]$  or  $J_i$  represents the running time of job  $i$ ; jobs do not have specific start and end times, but they can be started at any time (this is a different scenario than in interval scheduling). The running times are integers.

Generally speaking, your task is to find an optimal load balance of these tasks over two processors.

Design an algorithm for determining whether there is a subset  $S$  in  $J$  such that the running time of the elements in  $S$  sum up precisely to the same amount as the sum of the elements not in  $S$ ; more formally,  $\sum_{J_i \in S} J_i = \sum_{J_i \in J-S} J_i$ . The algorithm should run in time  $O(n \cdot N)$ , where  $N$  is the sum of the running times of the  $n$  jobs.

---

<sup>1</sup>Collaboration level 1: verbal collaboration without solution sharing. You are allowed and encouraged to discuss ideas with other class members, but the communication should be verbal and additionally it can include diagrams on board. No one is allowed to take notes during the discussion (being able to recreate the solution later from memory is proof that you actually understood it). Communication cannot include sharing pseudocode for the problem. Check complete guidelines at: <https://turing.bowdoin.edu/dept/collab.php>