Recitation Week 10- Rod Cutting

Michael Levet

October 27, 2020

1 Introduction to Dynamic Programming

Dynamic programming is a powerful technique to solve computational problems, which have a recursive substructure and recurring subproblems. The idea is to solve these recursive subcases and store these solutions in a lookup table. When a solved recursive subcase is encountered, the existing solution is accessed using only a constant number of steps. A solution to the initial instance is constructed from the solutions to the subcases, typically in a bottom-up manner. Frequently, the computational problems of interest are optimization problems. We begin by introducing some examples amenable to the dynamic programming technique.

1.1 Rod-Cutting Problem

In this section, we examine the Rod-Cutting Problem. Let us consider a motivating example. Suppose we have a rod of length 5, which can be cut into smaller pieces of integer lengths $1, 2, 3$, or 4. These smaller rods can then be further cut into smaller pieces. We stress that the rod pieces can only be cut into pieces of integer length; so pieces of size $1/2$ or π are not considered in this problem. Now suppose that we can sell rods of length 1 for \$1, which we denote $p_1 = 1$. Similarly, suppose that the prices for rods of length $2, \ldots, 5$ are given by $p_2 = 4, p_3 = 7, p_4 = 8$, and $p_5 = 9$ respectively. We make two key assumptions: we will sell all the smaller rods, regardless of the cuts; and that each cut is free. Under these assumptions, how should the rod be cut to maximize the profit? We note the following cuts and the corresponding profits.

- If the rod is cut into five pieces of length 1, we stand to make $5 \cdot p_1 = \$5$.
- If the rod is cut into one piece of length 2 and one piece of length 3, we stand to make $p_2 + p_3 = 4 + 7 = \$11$.
- If the rod is cut into two pieces of length 2 and one piece of length 1, we stand to make $2 \cdot p_2 + p_1 = 8 + 1 = \$9$.

Out of the above options, cutting the rod into one piece of length 2 and one rod of length 3 is the most profitable. Of course, there are other possible cuts not listed above, such as cutting the rod into one piece of length 1 and one piece of length 4. The goal is to determine the most profitable cut. The Rod-Cutting Problem is formalized as follows.

Definition 1 (Rod-Cutting Problem)**.**

- `Instance:` Let $n \in \mathbb{N}$ be the length of the rod, and let $p_1, p_2, \ldots, p_n$ be non-negative real numbers. Here, p_i is the price of a length i rod.
- `Solution:` The maximum revenue, which we denote r_n, obtained by cutting the rod into smaller pieces of integer lengths and selling the smaller rods.

Intuitively, the maximum revenue is determined by examining the revenues for the subdivisions and taking the largest. Mathematically, this amounts to the following expression:

$$r_n = \max(p_n, r_1 + r_{n-1}, r_2 + r_{n-2}, \ldots, r_{n-1} + r_1). \tag{1}$$

We begin by working through an example of utilizing dynamic programming to determine the maximum profit.

Example 1. Suppose we have a rod of length 5, with prices $p_1 = 1, p_2 = 4, p_3 = 7, p_4 = 8, p_5 = 9$. We proceed as follows.

(a) Initialize a lookup table $T[1, \ldots, 5]$. Now for a rod of length 1, there is only one price: p_1. So we set $T[1] := p_1$.

(b) Now consider a rod of length 2. There are two options: either don't cut the rod, or cut the rod into two smaller pieces of length 1. Here, $p_2 = 4$ represents the case in which no cuts to a rod of length 2. Now suppose instead we cut the rod up into two smaller pieces each of length 1. We know that the maximum profit for a rod of length 1 is $r_1 = T[1] = 1$. So the profit of cutting the rod into two smaller pieces each of length 1 is $2r_1 = 2$. Now $r_2 = \max(p_2, 2r_1) = 4$, so we set $T[2] = 4$.

(c) Now consider a rod of length 3. Here, we have more options: we can leave the rod untouched, we can divide the rod into smaller pieces of length 1 or length 2; or we can divide the rod into three pieces of length 1. If we do not divide the rod into smaller pieces, the profit is $p_3 = 7$. Now suppose we divide the rod up into smaller pieces of length 2 and length 1. We may now keep this configuration, or further divide the rod of length 2 into two rods each of length 1, as discussed in the previous bullet point. Rather than re-solving this problem, we can simply look up the maximum profit for a rod of length 2 in the lookup table. Recall that $r_2 = T[2] = 4$ and $r_1 = 1$. So the profit from cutting the rod into smaller pieces of length 2 and length 1 is $r_2 + r_1 = 5$. Now $r_3 = \max(p_3, r_2 + r_1) = 7$, so we set $T[3] = 7$.

(d) Now consider a rod of length 4. We have the following options for the first cut: leave the rod uncut, in which we stand to make profit $p_4 = 8$; cut the rod into smaller pieces of length 1 and length 3; or cut the rod into two smaller rods, each of length 2. Consider the case in which we cut the rod up into smaller pieces of length 1 and length 3. The natural, though inefficient, approach here is to consider all the ways in which we could cut up the rod of legnth 3. It turns out that we don't need to do this, as the maximum revenue attainable from a rod of length 3 was found in the previous bullet point. This is the power of dynamic programming: once a solution to a smaller problem is found, we simply look it up rather than re-solving the smaller problem. Similarly, we can look up the maximum profit for a rod of length 2. So given our cases, we have the following possible profits:

- The uncut rod of length 4 will result in profit $p_4 = 8$.
- The rod cut into pieces of length 3 and length 1 will result in profit $r_3 + r_1 = T[3] + T[1] = 7 + 1 = 8$.
- The rod cut into two pieces, each of length 2, will result in profit $2r_2 = 2 \cdot T[2] = 2 \cdot 4 = 8$.

So $T[4] = \max(8, 8, 8) = 8$.

(e) Finally, consider our original rod of length 5. We have the following possible initial cuts:

- We can leave the rod uncut, in which case we will make profit $p_5 = 9$.
- We can cut the rod into one piece of length 4 and one piece of length 1. The maximum revenue attainable by cutting up a rod of length 4 was determined already. So we can simply look up this solution in $T[4]$. Thus, the profit in this case is $r_4 + r_1 = T[4] + T[1] = 8 + 1 = 9$.
- We can cut up the rod into one piece of length 3 and one piece of length 2. By similar argument as above, we may simply look up the maximum revenues attainable from a rod of length 3 and a rod of length 2. So our profit is $r_3 + r_2 = T[3] + T[2] = 7 + 4 = 11$.

So $r_5 = \max(9, 9, 11) = 11$. Thus, we set $T[5] = 11$.

We conclude that we stand to make \$11 from a rod of length 5.

While the expresion (1) may not seem insightful, it in fact provides an algorithm to compute r_n. Example 1 provides a tangible example of this algorithm. The goal now is to generalize the algorithm from Example 1 to work for any rod of positive integer length any list of prices. We proceed as follows.

(a) Initialize the lookup table $T[1, \ldots, n]$, and set $T[1] := p_1$.

(b) We set $T[2] := \max(p_2, 2r_1)$. Here, p_2 represents the case in which no cuts to a rod of length 2, and $2r_1$ represents the case in which a rod of length 2 is cut into two rods each of length 1. We note that $r_1 = T[1] = p_1$.

(c) We set $T[3] := \max(p_3, r_1 + r_2)$. Now $r_1 = T[1]$, and $r_2 = T[2]$. We have already solved the rod cutting problem for a length 2 rod, so we simply look up r_2 in the table T rather than re-solving the problem.

(d) $T[4] := \max(p_4, r_1 + r_3, 2r_2)$. As we have already computed r_1, r_2, r_3, we may look up their respective values in T rather than re-computing these values.

Continuing in this manner, we compute r_n, which is the value in $T[n]$ after the algorithm terminates.

Remark 1. This algorithm only provides the maximum revenue. It does not tell us how to achieve that result. As an exercise, modify the algorithm to produce an optimal set of rod cuts.

2 Exercises

(Recommended) Problem 1. Modify the lookup table from Example 1 to allow us to determine the optimal sub-division. That is, at index i, your lookup table should provide the maximum revenue for a rod of length i, as well as an optimal cut for the rod of length i. Work through Example 1 to determine an optimal cut for the original rod of length 5.

(Recommended) Problem 2. Suppose we have a rod of length 6, with prices given as follows: $p_1 = 1, p_2 = 5, p_3 = 8, p_4, p_5 = 9, p_6 = 15$. Determine the maximum revenue r_6 obtained by cutting up this rod. Clearly specify the optimal cut you obtained from the dynamic programming procedure.

(Recommended) Problem 3. Suppose we modify the Rod Cutting problem, so that each cut costs 1 unit of money. Do the following

(a) Modify the recurrence from (1) to compute the maximum revenue r_n for cutting a rod of length n.

(b) As in Problem 2, suppose we have a rod of length 6, with prices given as follows: $p_1 = 1, p_2 = 5, p_3 = 8, p_4 = 8, p_5 = 9, p_6 = 15$. Using the recurrence in part (a), determine the maximum revenue r_6 obtained by cutting up this rod. Clearly specify the optimal cut you obtained from the dynamic programming procedure.

(Recommended) Problem 4. The purpose of this problem is to highlight the power of the dynamic programming approach for the Rod Cutting problem.

(a) How many possible ways are there to cut a rod of length n? [**Hint:** Start by determining the number of places to make a cut.]

(b) The power of the dynamic programming approach is that the order in which we compute sub-cases contributes to the efficiency. Suppose we wish to find an optimal cut for a rod of length n. To do this, we first consider a rod of length 1, then a rod of length 2, and so on.

(i) How many calculations are needed to compute r_1?

(ii) Once r_1 is computed, how many calculations are needed to compute r_2?

(iii) Once r_1 and r_2 have been computed, how many calculations are needed to compute r_3?

(iv) Once $r_1, \ldots, r_{n-1}$ have been computed, how many calculations are needed to compute r_n?

(v) As the dynamic programming approach computes $r_1, \ldots, r_n$, what is the runtime complexity $T(n)$ of this algorithm? Give a suitable function $f(n)$ such that $T(n) = \Theta(f(n))$.

(vi) Compare your answer in (v) to your answer in (a). Notice that the dynamic programming algorithm is quite efficient in parsing through a search space of exponential size.