Rod Cutting

Manuel Serna-Aguilera

Introduction

Given a rod of length n units and a table of prices p_i for $i = 1, 2, \ldots, n$, determine the maximum revenue r_n obtainable by cutting up the rod and selling the pieces.

Note that a rod of length n can be cut in $2^{(n-1)}$ different ways.

Next, an example from the book (with the last couple of columns removed for brevity's sake) is used.

Example

We are given a table of lengths and associated prices.

length i	1	2	3	4	5	6	7	8
price p_i	1	5	8	9	10	17	17	20

Now, with the following formula we can find the optimal cost r_i of the cutting a rod of length i, where p_i is the price for a rod of length i.

$$r_i = \max_{1 \le i < j} \{p_i + r_{j-i}\}$$

Now, start the procedure, starting from

$$r_0 = 0$$

$$r_1 = 1$$

$$\begin{aligned} r_2 = &\{1 + r_{2-1} = 1 + r_1 = 1 + 1 = 2, \\ &5 + r_{2-2} = 5 + r_0 = 5 + 0 = 5\} \\ = &5 \end{aligned}$$

$$\begin{aligned} r_3 = &\{1 + r_{3-1} = 1 + r_2 = 1 + 5 = 6, \\ &5 + r_{3-2} = 5 + r_1 = 5 + 1 = 6, \\ &8 + r_{3-3} = 8 + r_0 = 8 + 0 = 8\} \\ = &8 \end{aligned}$$

$$
\begin{aligned}
r_4 =& \{1 + r_3 = 1 + 8 = 9 \\
& 5 + r_2 = 5 + 5 = 10 \\
& 8 + r_1 = 8 + 1 = 9 \\
& 9 + r_0 = 9 + 0 = 9\} \\
=& 10
\end{aligned}
$$

Now, simplifying the process to only show the max values.

$$r_5 = 13$$

$$r_6 = 17$$

$$r_7 = 18$$

$$
\begin{aligned}
r_8 =& \{1 + r_{8-1} = 1 + r_7 = 1 + 18 = 19, \\
=& 5 + r_{8-2} = 5 + r_6 = 5 + 17 = 22, \\
=& 8 + r_{8-3} = 8 + r_5 = 8 + 13 = 21, \\
=& 9 + r_{8-4} = 9 + r_4 = 9 + 10 = 19, \\
=& 10 + r_{8-5} = 10 + r_3 = 10 + 8 = 18, \\
=& 17 + r_{8-6} = 17 + r_2 = 17 + 5 = 22, \\
=& 17 + r_{8-7} = 17 + r_1 = 17 + 1 = 18, \\
=& 20 + r_{8-8} = 20 + r_0 = 20 + 0 = 20\} \\
=& 22
\end{aligned}
$$

So lengths 6 and 2 combined give the highest profit. The max value of r_2 and r_6 are themselves, so no more cutting. The ordering of the rods of length 2 or 6 does not matter.

Solving with code

A regular recursive solution, even for small inputs, may have exponential running time due to repeating many calculations. Instead, use a little more memory to store calculation results. Remember that the bottom-down approach to dynamic programming is often better.