Dynamic Programming

Used when:

- **Optimal substructure - the optimal solution to your problem is composed of optimal solutions to subproblems (each of which is a smaller instance of the original problem)**
- **Overlapping subproblems**

Methodology

- **Characterize structure of optimal solution**
- **Recursively define value of optimal solution**
- **Compute in a bottom-up manner**

Example: Rod Cutting

Problem: **Given a rod of length n inches 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.**

length i	1	2	3	4	5	6	7	8	9	10
price p_i	1	5	8	9	10	17	17	20	24	30

How can we cut a rod of length 4?

Optimal Substructure

Suppose that we know that optimal solution makes the first cut to be length k, then the optimal solution consists of an optimal solution to the remaining piece of length $n-k$, plus the first piece of length k

Suppose not. Then we are saying that the optimal solution consists of some way to cut the piece of length $n-k$ that is not optimal, plus the piece of length k. Let p_k be the profit from the piece of length k, and let y be profit from the non-optimal solution to the piece of length $n-k$. Then we are receiving a total profit of $y+p_k$. Now suppose that instead of the proposed solution to the piece of length k, we used an optimal solution to the piece of length k instead. Let y' be the profit associated with the optimal solution to the piece of length $n-k$, and since it is optimal $y'>y$. We could then put this together with the piece of length k and obtain a solution of profit $y'+k > y+k$, contradicting the claim that the original solution was optimal.

Recursive Implementation

Recurrence

$$r_n = \max_{1 \le i \le n} (p_i + r_{n-i}) \quad . \tag{1}$$

Code

$Cut - Rod(p, n)$

```
if n == 0
    then return 0
q ← −∞
for i ← 1 to n
      do q ← max(q, p[i] + Cut-Rod(p, n − i))
return q
```

What is the running time?

DP solution

$Bottom - Up - Cut - Rod(p, n)$

let $r[0\,..\,n]$ **be a new array**
$r[0] \leftarrow 0$
for $j \leftarrow 1$ **to** n
    do $q \leftarrow -\infty$
        for $i \leftarrow 1$ **to** j
            do $q \leftarrow \max(q, p[i] + r[j-i])$
        $r[j] \leftarrow q$
return $r[n]$

What is the running time?