

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

$$\begin{array}{r}
 3 \quad 1 \quad 2 \\
 + \quad + \quad + \\
 \hline
 2 \quad + \quad 5 \quad + \quad 1 \\
 \hline
 \quad \quad \quad \downarrow \quad 8
 \end{array}$$

Example: Rod Cutting

$$\begin{array}{ccccccccc}
 1 & 3 & 1+8 & 1+10 & \\
 2 & 2 & 5+5 & 2+3 & 5+8 \\
 3 & 1 & 8+9 & 3+2 & 8+5 \\
 \downarrow & 0 & & 4+1 & 9+1 \\
 \end{array}$$

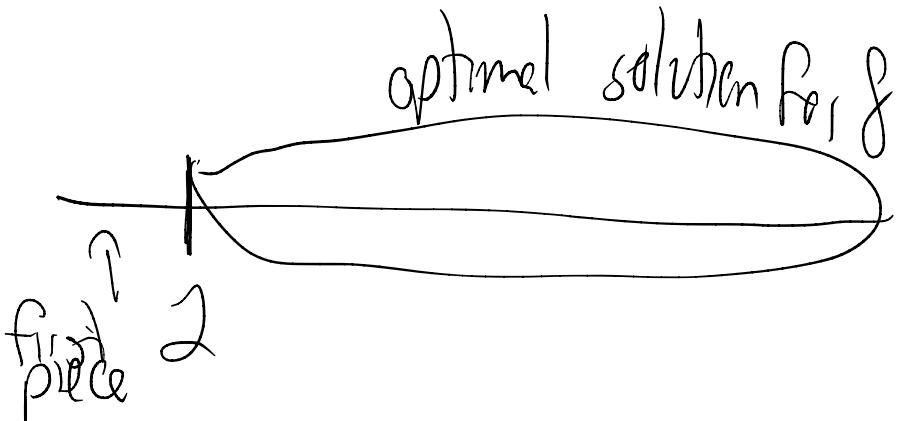
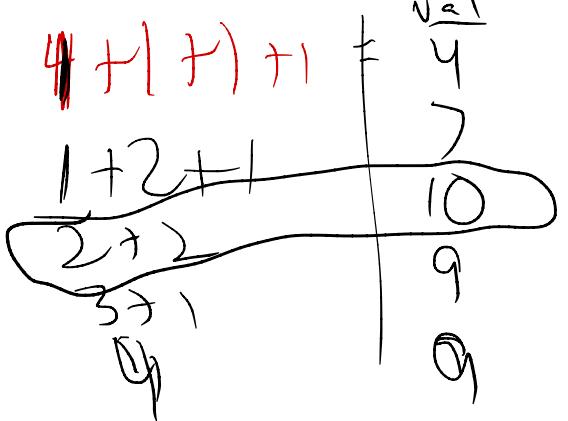
1	2	3	4	5	6	7	8	9	10
15	8	10	13						

Problem: Given a rod of length n inches and a table of prices p_i for $i = 1, 2, \dots, n$, 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

10

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

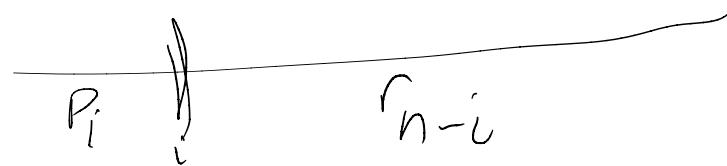
r_n optimal solution (Value) for
a rod of size n

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

Code

$Cut - Rod(p, n)$

```
1 if  $n == 0$ 
2   then return 0
3  $q \leftarrow -\infty$ 
4 for  $i \leftarrow 1$  to  $n$ 
5   do  $q \leftarrow \max(q, p[i] + \text{CUT-ROD}(p, n - i))$ 
6 return  $q$ 
```



What is the running time?

DP solution

Biggest piece is of
size k

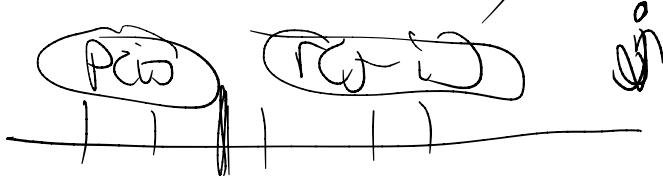
Bottom - Up - Cut - Rod(p, n)

```

1 let  $r[0..n]$  be a new array
2  $r[0] \leftarrow 0$ 
3 for  $j \leftarrow 1$  to  $n$ 
4   do  $q \leftarrow -\infty$ 
5   for  $i \leftarrow 1$  to  $\min(j, k)$ 
6     do  $q \leftarrow \max(q, p[i] + r[j-i])$ 
7    $r[j] \leftarrow q$ 
8 return  $r[n]$ 

```

$O(nk)$



remember which i was the max

$O(n^2)$

What is the running time?

DP solution

Bottom – Up – Cut – Rod(p, n)

```
1 let  $r[0..n]$  be a new array
2  $r[0] \leftarrow 0$ 
3 for  $j \leftarrow 1$  to  $n$ 
4     do  $q \leftarrow -\infty$ 
5         for  $i \leftarrow 1$  to  $j$ 
6             do  $q \leftarrow \max(q, p[i] + r[j - i])$ 
7          $r[j] \leftarrow q$ 
8 return  $r[n]$ 
```

What is the running time?