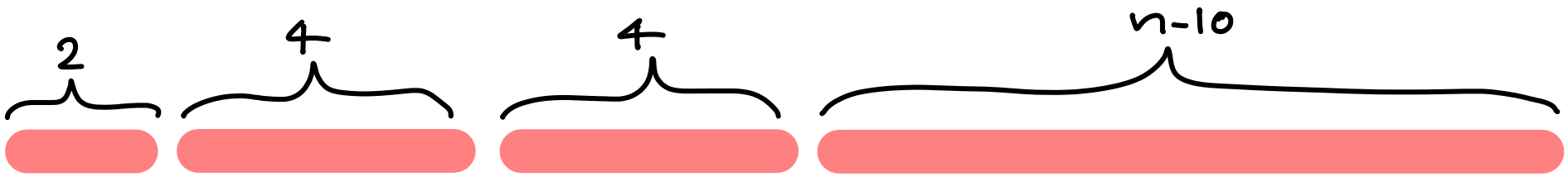
DYNAMIC PROGRAMMING – ROD CUTTING

You can cut at any integer position, for free.



price: p_2 p_4 p_4 p_{n-10}

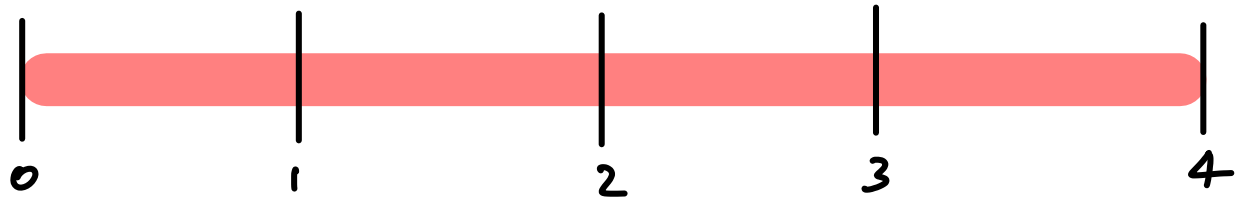
Every resulting piece will be sold, at a predefined price.

– Maximize profit –

We have the option to cut at every position $\{1 \ldots n-1\}$

"Naive" counting: $\{0,1\}^{n-1} \rightarrow 2^{n-1}$ solutions : pick best.

example



prices 1 5 8 9

option 1) Don't cut : profit = $1 \times p_4 = 9$

2) Cut once :
- (2.1) cut at 1 : profit = $p_1 + p_3 = 1 + 8 = 9$
- (2.2) cut at 2 : profit = $5 + 5 = \boxed{10}$
- (2.3) cut at 3 : profit = $8 + 1 = 9$

3) Cut twice :
- (3.1) cut at 1&2 : profit = $1+1+5 = 7$
- (3.2) cut at 1&3 : profit = $1+5+1 = 7$
- (3.1) cut at 2&3 : profit = $5+1+1 = 7$

4) Cut 3 times : profit = $4 \times p_1 = 4$

$$\$(n) = \max \begin{cases} \text{no cut} & \longrightarrow p_n \\ \text{cut at position } 1 & \longrightarrow p_1 + \$(n-1) \\ \text{cut at position } 2 & \longrightarrow \$(2) + \$(n-2) \\ \quad \text{,,} \quad \text{,,} \quad \text{,,} \quad 3 & \longrightarrow \$(3) + \$(n-3) \\ \vdots & \vdots \\ \text{cut at position } n-2 & \longrightarrow \$(n-2) + \$(2) \\ \text{cut at position } n-1 & \longrightarrow \$(n-1) + p_1 \end{cases}$$

$$\$(n-1) = \max \begin{cases} p_{n-1} \\ p_1 + \$(n-2) \\ \$(2) + \$(n-3) \\ \$(3) + \$(n-4) \\ \vdots \\ \$(n-3) + \$(2) \\ \$(n-2) + p_1 \end{cases}$$

AWFUL if just using recursion

Change description of solution:

$\$(n) = \max$

- no cut ⟶ $p_n + \$(0)$
- cut at position 1 ⟶ $p_1 + \$(n-1)$
- cut at position 2 ⟶ $p_2 + \$(n-2)$
- " " " 3 ⟶ $p_3 + \$(n-3)$
- ⋮
- cut at position $n-2$ ⟶ $p_{n-2} + \$(2)$
- cut at position $n-1$ ⟶ $p_{n-1} + \$(1)$

Cut at i
↳ keep size(i)
+ recurse on $n-i$

$$\$(n) = \max_{1 \le i \le n} \{ p_i + \$(n-i) \}$$

$$\$(0) = 0$$

Compute & store all $\$(n-i)$
↓
$\rightarrow T(n) = \Theta(n)$

TOTAL = $\Theta(n^2)$

Example

size :	1	2	3	4	5	6	7	8
price :	1	5	8	9	10	17	17	20
$:	1	5	8	10	13	17	18	22

$$\$(n) = \max_{1 \le i \le n} \{ p_i + \$(n-i) \}$$

DONE

$\$(1) = 1$

$\$(2) = 1 + \$(1)$ OR $5 + \emptyset$ → 5

$\$(3) = 1 + \$(2)$ OR $5 + \$(1)$ OR 8 → 6 OR 6 OR 8 → 8

$\$(4) = 1 + \$(3)$ OR $5 + \$(2)$ OR $8 + \$(1)$ OR 9 → $\max\{9, 10, 9, 9\}$ → 10

$\$(5) = 1 + \$(4)$ OR $5 + \$(3)$ OR $8 + \$(2)$ OR $9 + \$(1)$ OR 10 → $\max\{11, 13, 13, 10, 10\}$

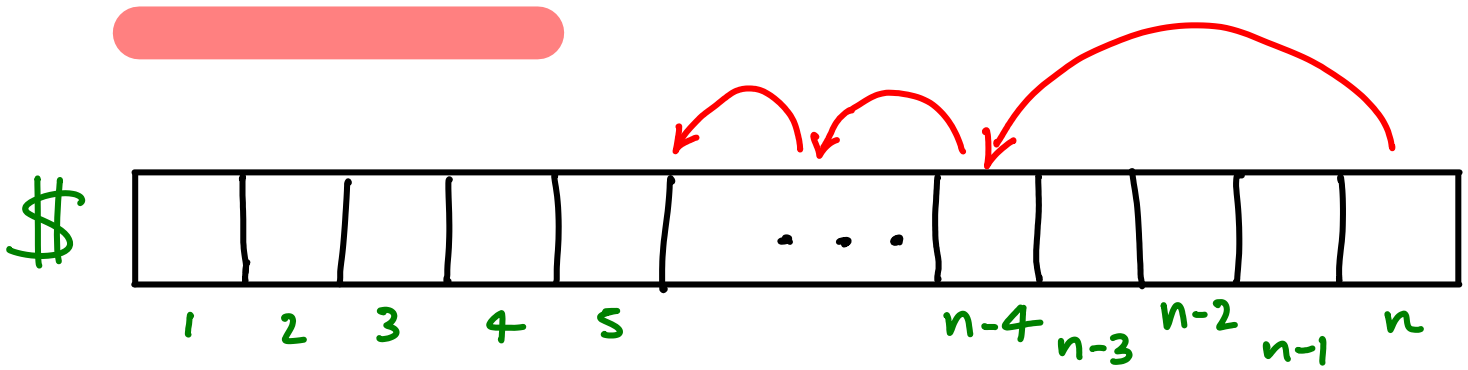
$\$(6) = \{1+13, 5+10, 8+8, 9+5, 10+1, 17\}$ → 17

$\$(7) = \{1+17, 5+13, 8+10, 9+8, 10+5, 17+1, 17\}$ → 18

$\$(8) = \{1+18, 5+17, 8+13, 9+10, 10+8, 17+5, 17+1, 20\}$ → 22

cut 2 & 6 = 6 & 2

Retrieving cuts (not just profit)



ex:

$\$(n) = \max\{\cdot\} =$

$p_4 + \$(n-4)$

When computing $\$(n)$ we also see where we should cut

e.g. cut at p_4, leftovers: $\$(n-4)$, so follow pointer & report 4

↳ e.g. $\$(n-17)$, report $17-4 = 13$

↳ e.g. $\$(n-50)$, report $50-17 = 33$ etc