Date: 19/04/21

Black Board

Design and Analysis of Algorithms

Topics:

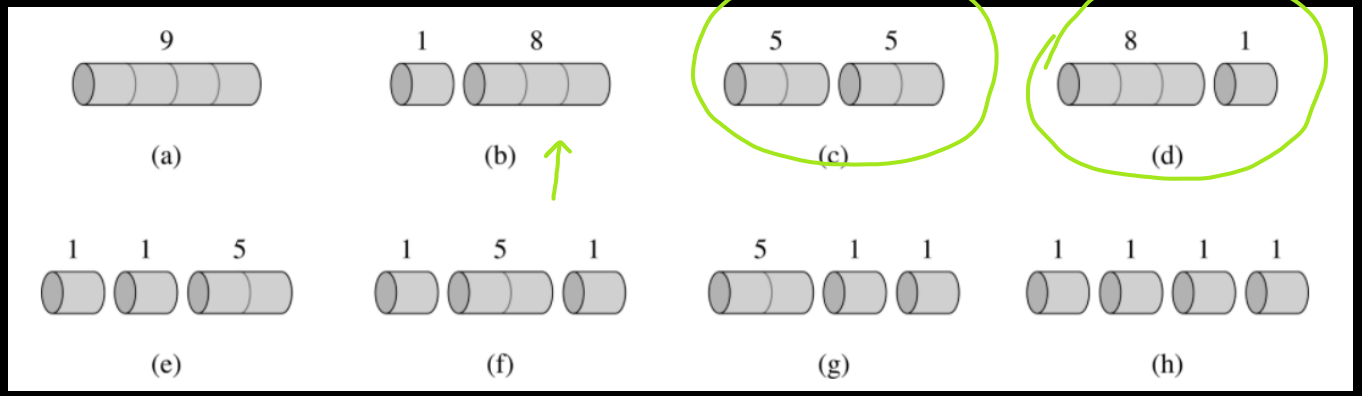
- Dynamic Programming II
  - THE ROD CUTTING PROBLEM

The Rod Cutting Problem

LENGTH	PRICE
1	1
2	5
3	8
4	9
5	10
6	17
7	17
8	20
9	24
10	30

integral length pieces

10



Input: • length of rod, n
• price list p[1,...,n]

Output: • optimal revenue (maximum possible)
• the cutting corresponding to opt-rev.

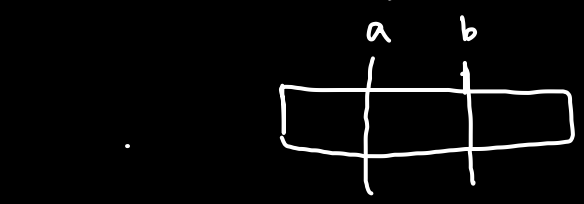
The Exhaustive Solution

→ Searches through all possible candidate Sol.

→ For this problem: all possible cutting.

→ Too many cuttings to check!!

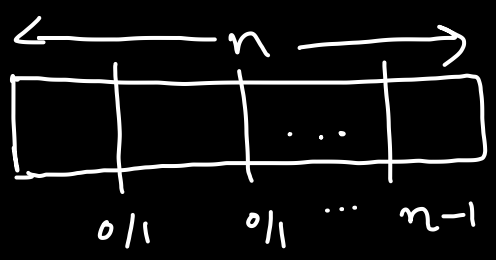
→ For a rod of length 3



a	b	cutting type
0	0	3
0	1	2, 1
1	0	1, 2
1	1	1, 1, 1
		⋮

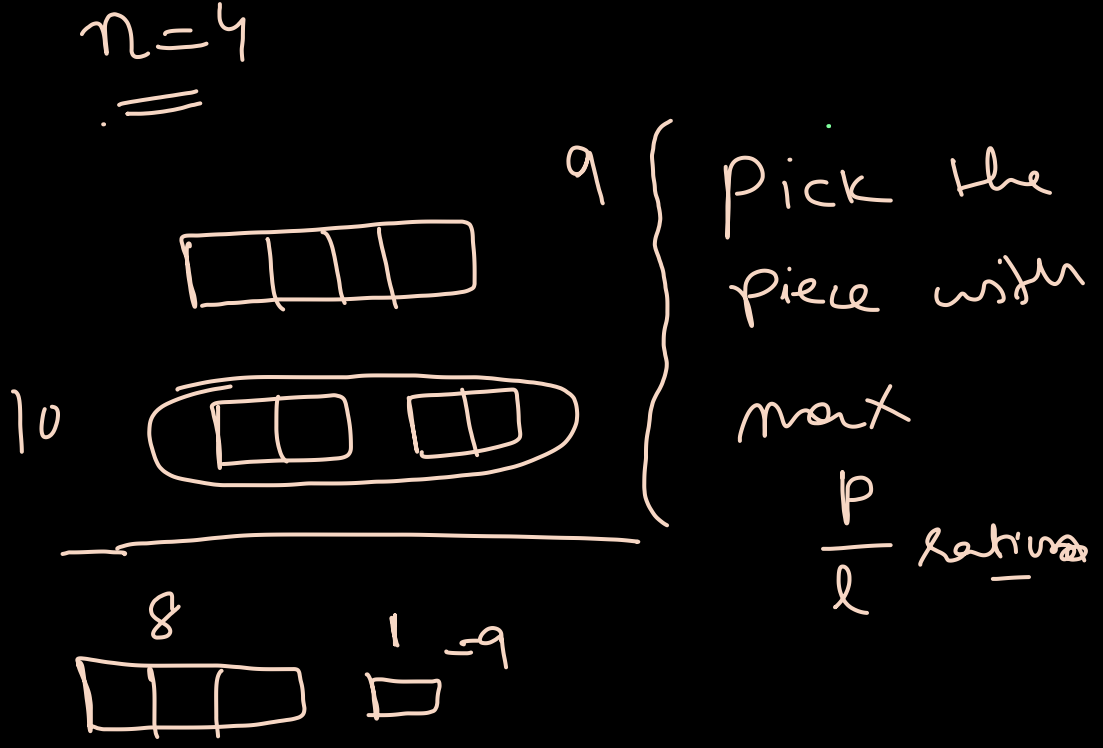
2^{n-1}

= immense



Does greed work?

LENGTH	PRICE
1	1
2	5
3	8
4	9



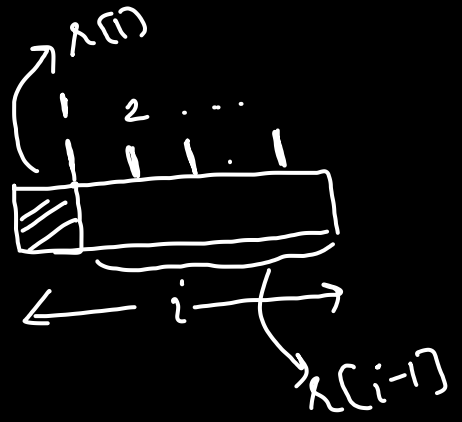
Formulating the DP Solution (The first way)

① Define the Subproblem

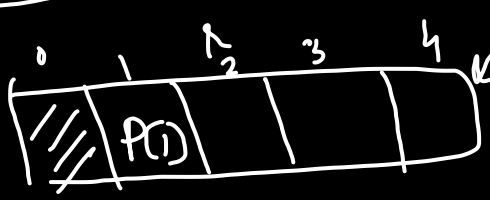
$r[i]$:= max. rev. for a rod of length i

② Recurrence of $r[i]$

$$r[i] = \max\left\{\max_{1 \le j < i} \{r[j] + r[i-j]\},\ p[i]\right\}$$



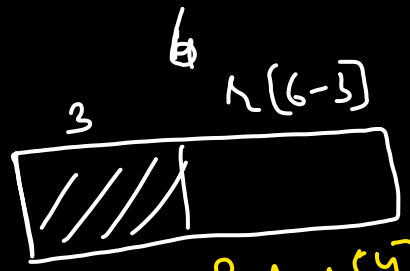
$r[1] = p[1]$



$r[2] = \max\{r[1]+r[1],\ p[2]\}$

max
- $p[4]$
- $r[1] + r[3]$
- $r[2] + r[2]$
- $r[3] + r[1]$
- $r[4] + 0$

Formulating the DP Solution (the Second Way)



② Recurrence.

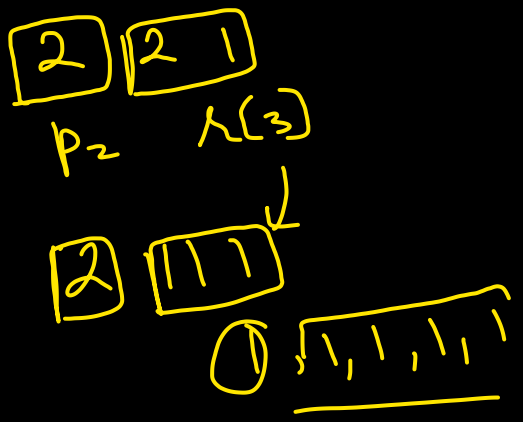
$$r[i] = \max_{1 \le j \le i} \{ p_j + r[i-j] \}$$

$r[0] = 0$

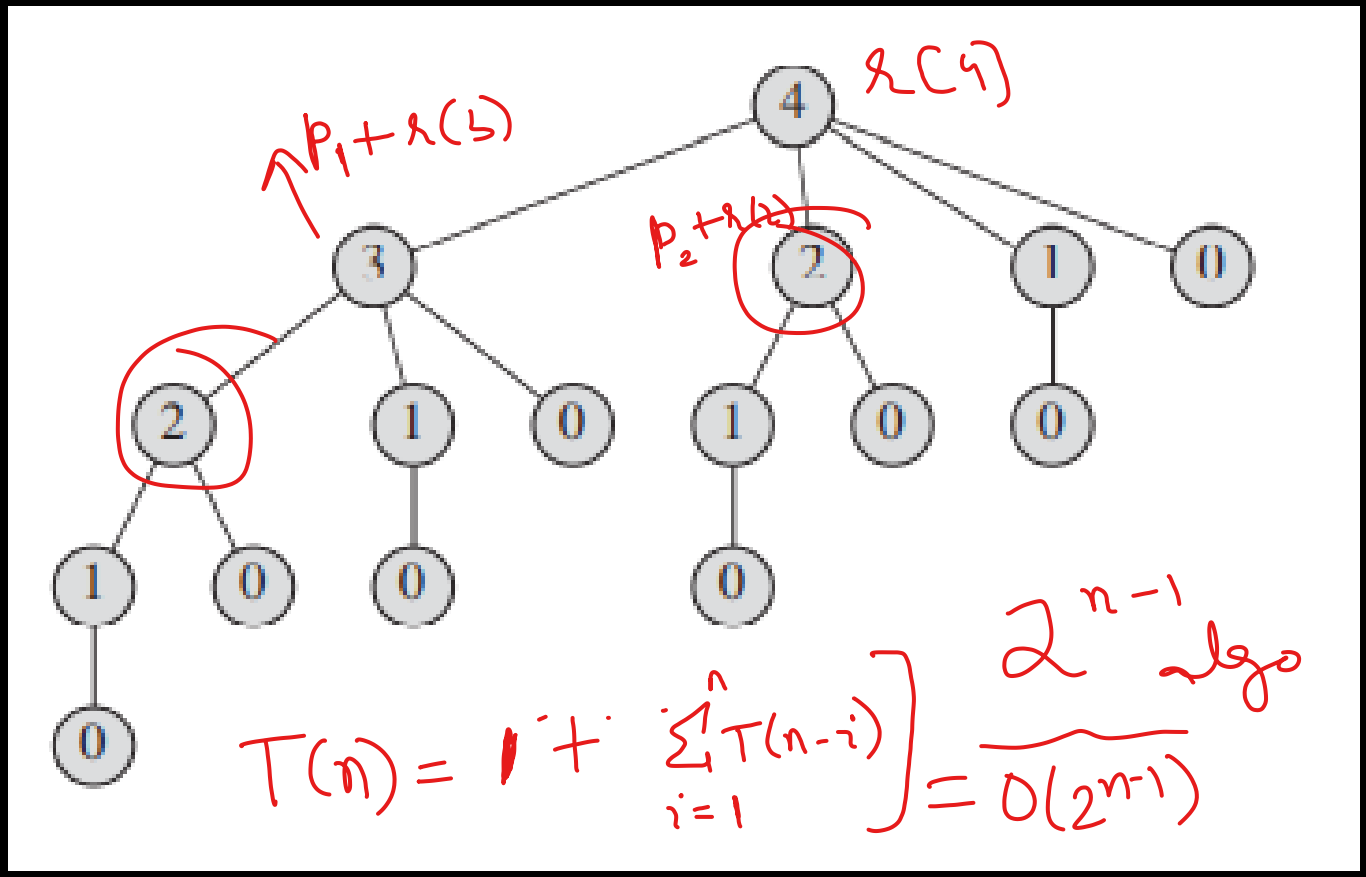
$r[1] = p[1]$

$$r[5] = \max \begin{cases} p_1 + r[4] \\ p_2 + r[3] \\ p_3 + r[2] \\ p_4 + r[1] \\ p_5 + r[0] \end{cases}$$

③ Bottom up Sol.



Regular top-down recursion?



Bottom up implementation

```
Rodcut(P, n)
    Create r[n+1]
    Create prev[n+1]
    r[0] = 0 , prev[0] = -1, prev[1] = 1
    For i = 1 to n
        maxsofar ← -∞
        For j = 1 to i
            IF (p[j] + r[i-j] > maxsofar)
                maxsofar ← p[j] + r[i-j]
                prev[i] = j
    return (r[n], cutting(prev, n))
```

$$r[i] = \max_{1 \le j \le i} \{p_j + r[i-j]\}$$

$$r[0] = 0$$

$T(n) = O(n^2)$

Bottom-up Solution, dry run

r[0] = 0

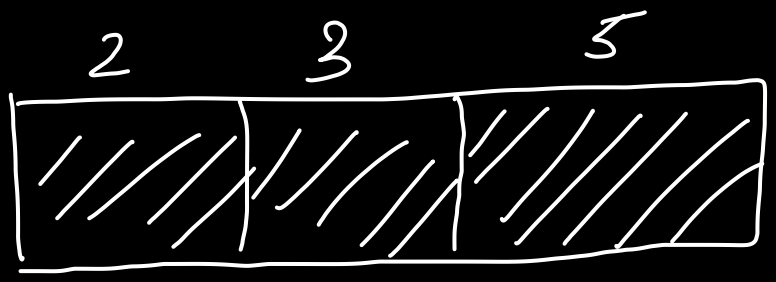
prev[0] = -1

i	1	2	3	4	5	6	7	8	9	10
p	10	50	80	100	135	155	180	190	220	235
r										
prev[i]										

Can we remove the problem of overlapping sub-problems, while still working top-down?

Array P

$n = 10$



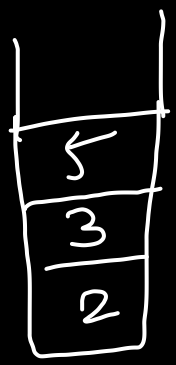
P[10] = 2

P[8] = 3

P[5] = 5

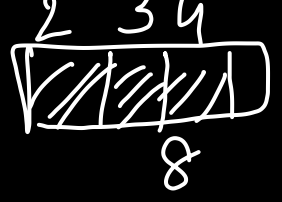
P[0] = -1

2, 3, 5



Cutting (prev, n)

≡



n = 10

prev[10] = 2

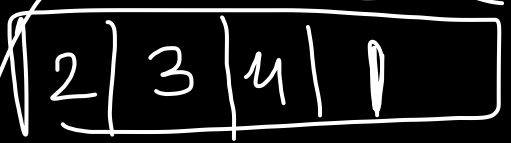
prev[8] = 3

prev[5] = 4

prev[1] = 1

-1

what does this mean?



Queue