Pre-requisite →

Patterns

1) Fib

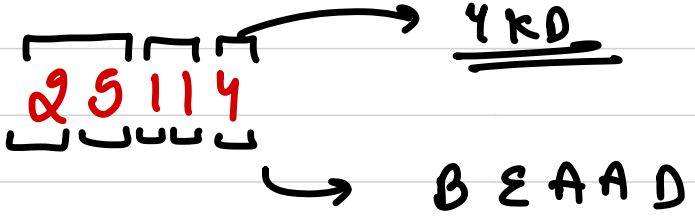
2) recurrence, state

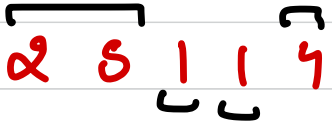
3) LIS, LCS, Rod Cutty, Min Coin change

4) Knapsack - 0-1, Subset Sum

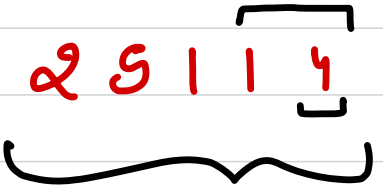
5) Catalan No

Alphacode (Spoj) → fib





↳ Brute force → We can try all possible comb.



for every digit we have 2 choices either consider it alone or consider it with a digit adjacent to it.

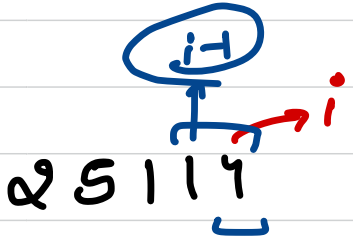
$f(s, i)$ =

= the no. of ways in which we can decode the string s from index (0-i)

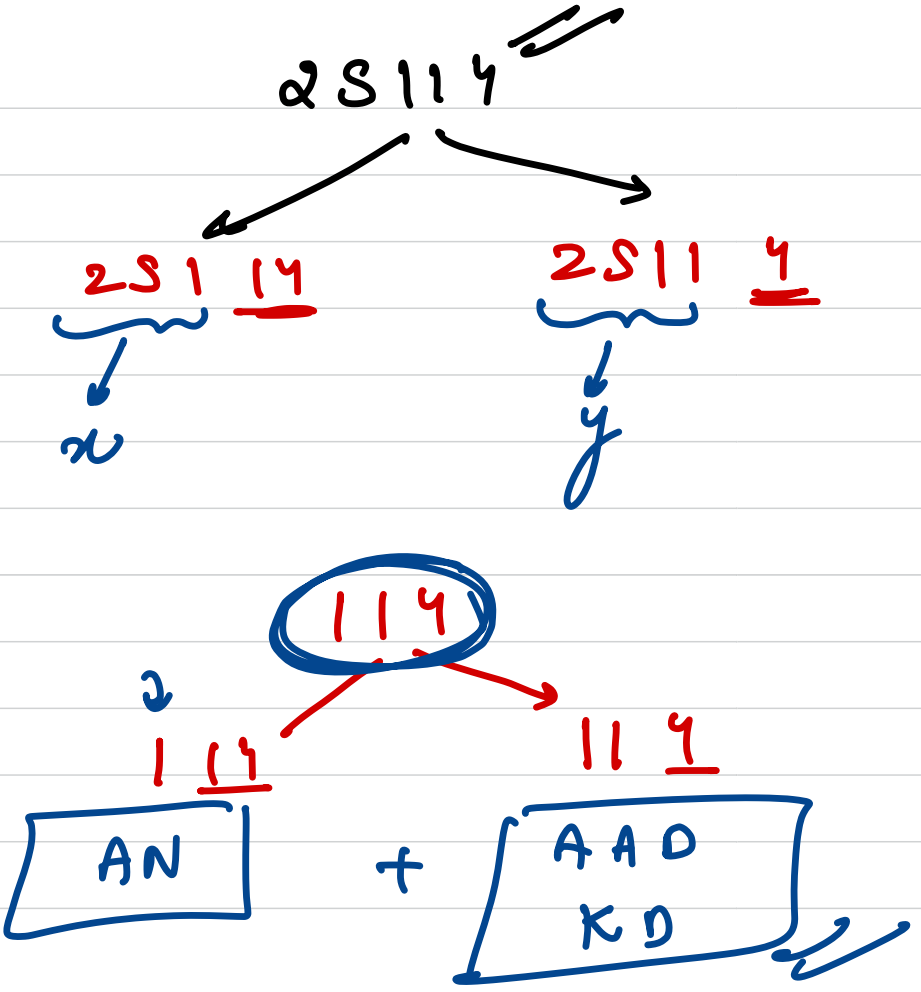
$s[0-i]$

$$f(s, i) = f(s, i-1) + f(s, i-2)$$

if $s[i-1]*10 + s[i] \leq 26$



State depends on 1 parameter. → 1D dp

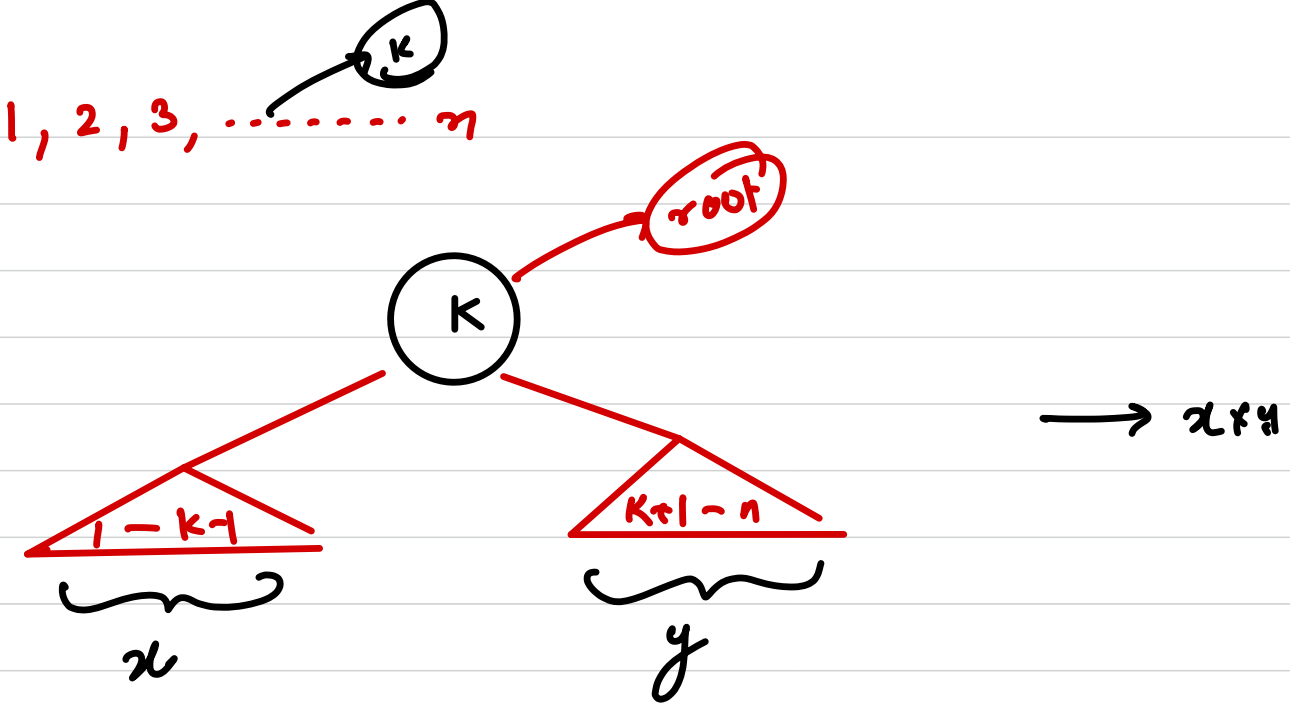

2S114
2S1 14
2S11 4
x
y
114
2
1 14
11 4
AN
+
AAD
KD

Unique Binary Search Trees

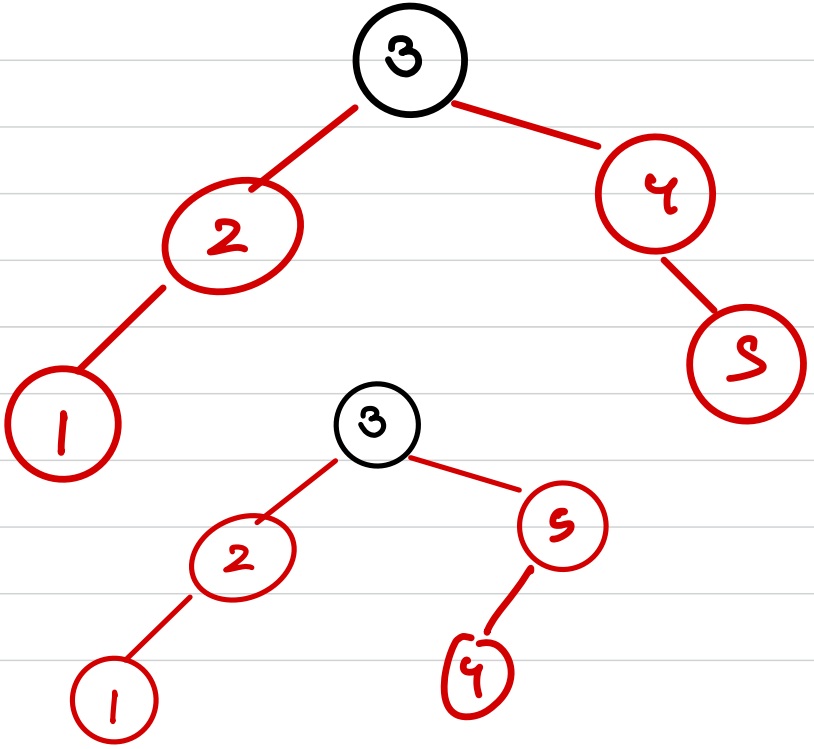
$n \rightarrow$ no. of nodes

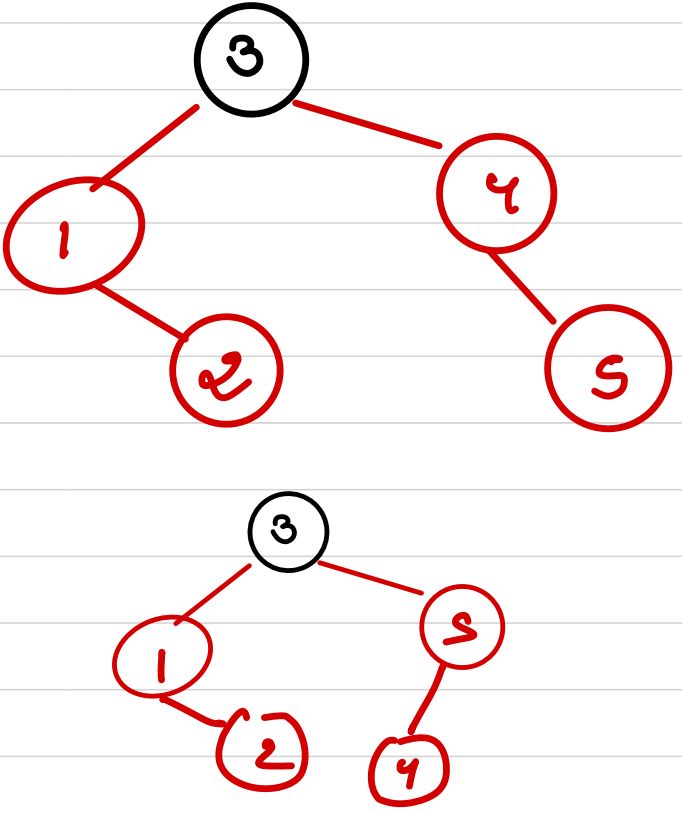
$f(n) \rightarrow$ this function will calculate the no. of structurally unique BST's

every node from 1 - n can become root once.


k
1, 2, 3, ... n
root
K
1 - k-1
K+1 - n
x
y
→ x+y

1, 2, 3, 4, 5





if LST can be formed in x ways and RST can be formed in y ways then for a given root no. of trees possible is

$$x \times y$$

$g(i) = g(i-1) \times g(n-i)$

$g(i)$: no. of tree we can create keeping i as the root node

$g(i-1)$: no. of ways of forming LST

$g(n-i)$: no. of ways of forming RST

$$f(n) = G(1) + G(2) + G(3) \dots\dots G(n)$$

$$f(n) = \sum_{K=1}^{n} G(K)$$

$G[0] = 1$

$G[1] = 1$

→ for calculate any value i we need already precomputed value of (1 - i-1)

→ f(n) → no. of ways to form unique bst

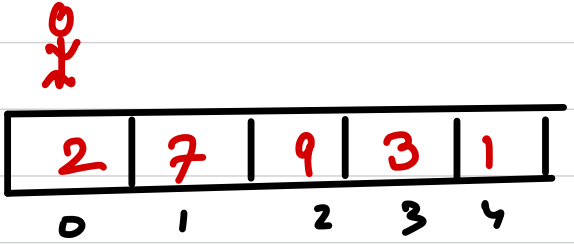
$$f(i) = f(i-1) \times f(n-i)$$

ans → $\sum f(i)$

House Robber

Brute force

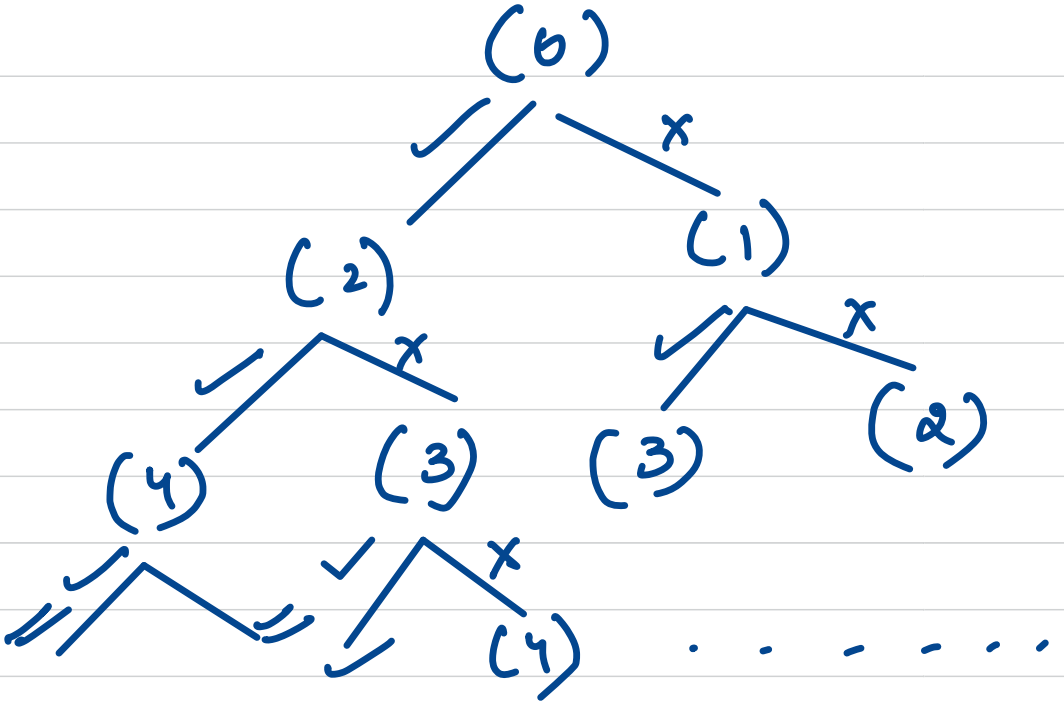
↳ all possible subset



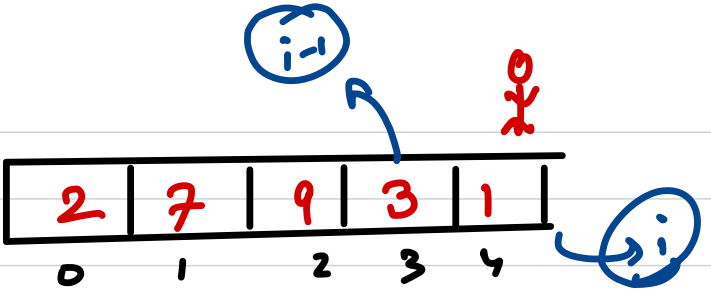
for every house we have 2 choice

→ either we can rob that house ↗ condition: we can't rob adjacent houses

→ or we can skip that house


(6)
✓
x
(2)
(1)
✓
x
✓
x
(4)
(3)
(3)
(2)
✓
x
(4)
. . . . . .

unique sub problm



rob

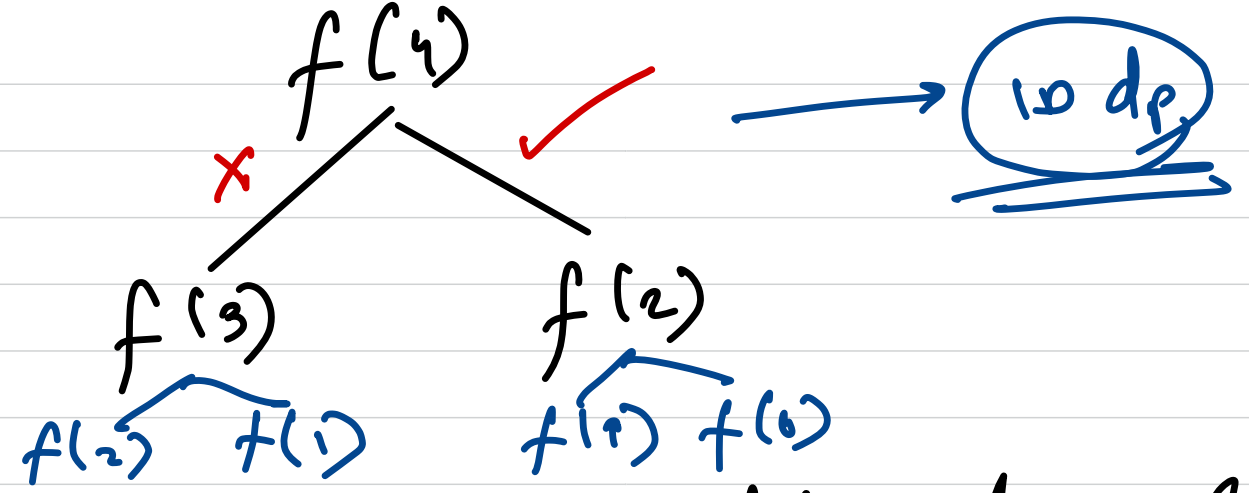
not rob

$$f(i) = \max(c[i] + f(i-2),\ f(i-1))$$

$f(i)$ returns the max profit you can by robbing houses from $0-i$

$c[i] + f(i-2)$: if we decide to rob the i^{th} house

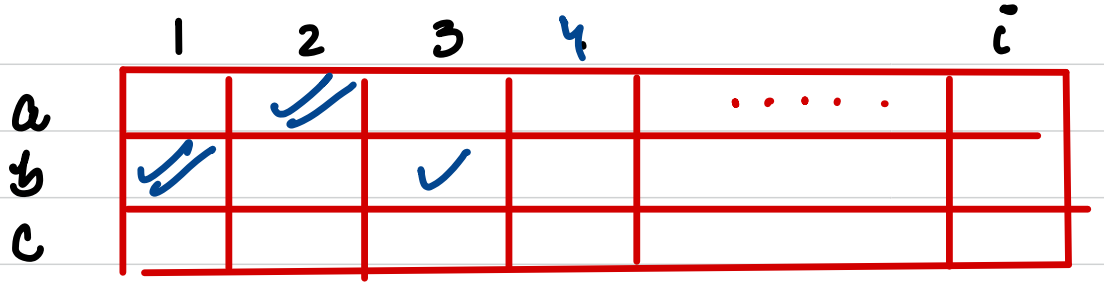
$f(i-1)$: if we decide not to rob



A unique subproblem will be determined by only a single parameter :→ house no.

n houses → n unique subproblems

Vacation- atcoder



if we have picked an activity we cannot pick the same one on the next day.

$$f(act, i) = \begin{cases} p[a_i] + \max(f(b, i-1), f(c, i-1)) \\ p[b_i] + \max(f(a, i-1), f(c, i-1)) \\ p[c_i] + \max(f(a, i-1), f(b, i-1)) \end{cases}$$

↓

the max profit you can gain by doing the 'act' actually on the first i days

$f(a, i)$ → a^{th} cube on i^{th} day

$$f(a,i) = p[a_i] + \max(f(b,i-1), f(c,i-1))$$

$$f(b,i) = p[b_i] + \max(f(a,i-1), f(c,i-1))$$

$$f(c,i) = p[c_i] + \max(f(a,i-1), f(b,i-1))$$

ans → $\max(f(a,n-1), f(b,n-1), f(c,n-1))$

Ex

	①	②	③
A →	10	40	70
B →	20	80	80
C →	30	60	90

$f(a, n-1) =$

a	10	70	150
b	20	80	160
c	30	80	120

120

	①	②	③
a	10	20	30
b	40	30	60
c	70	80	90

10	90	150
40	120	180
70	120	210

210 Ans