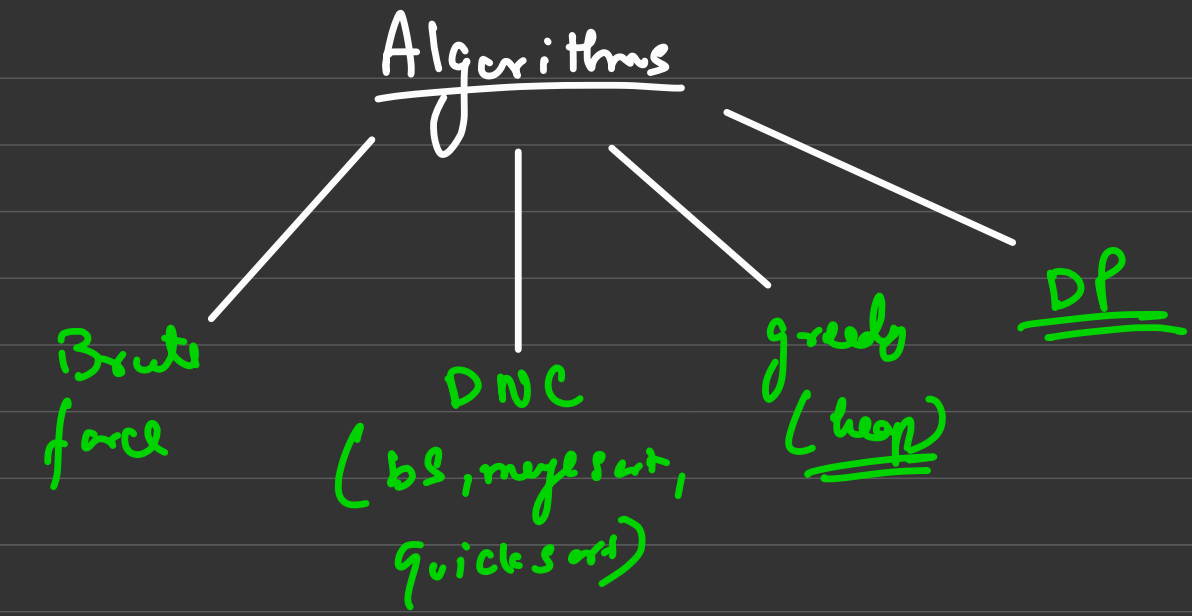
Dynamic Programming

- Understand the basic Crux
- Patterns of problems
- Problem Solving


Algorithms
Brute force
DNC
(bs, mergesort, quicksort)
greedy
(heap)
DP

DP → (Dynamic Programming) → Not a new topic

optimisation over brute force.

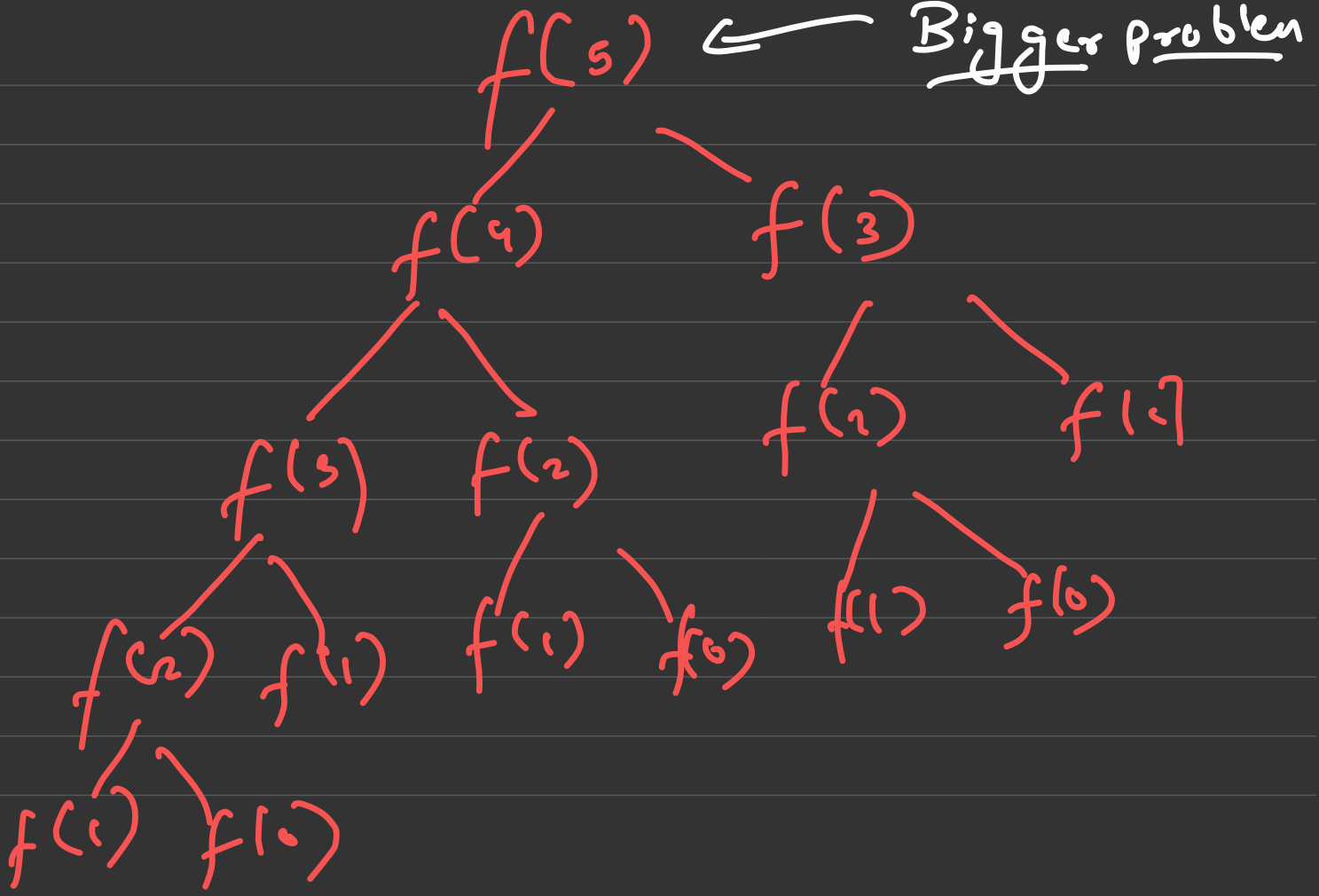
1- Leetcode

1→ CP3

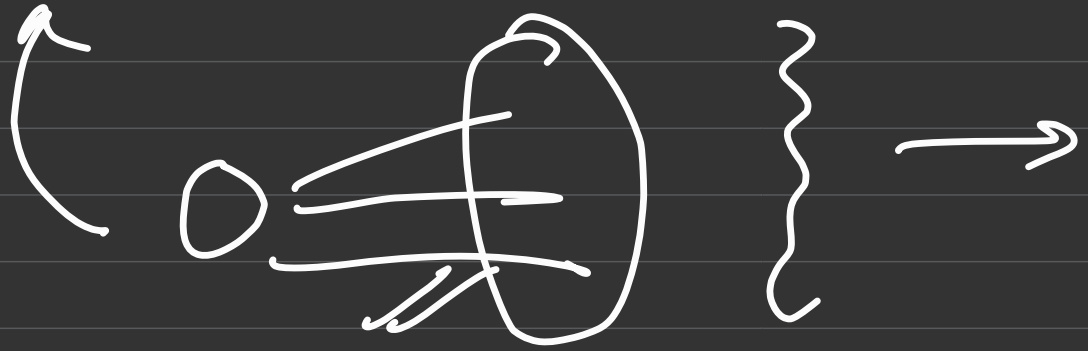
Criteria → 1) Overlapping Subproblem

2) Optimal Substructure

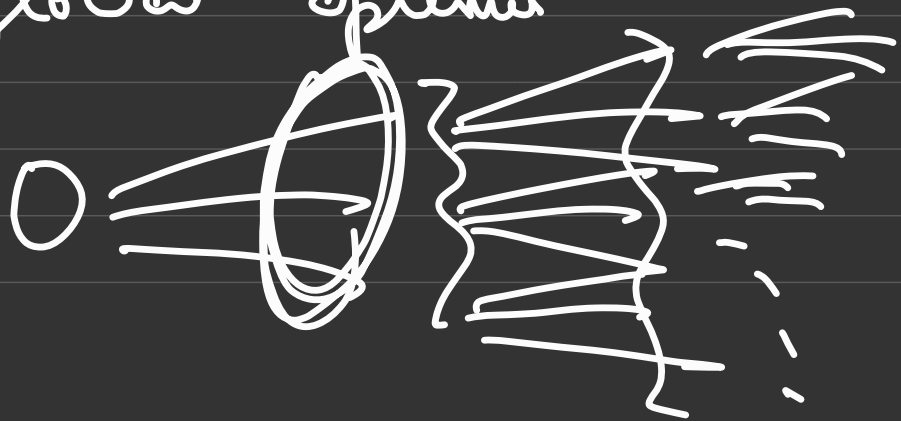
→ To find the optimal soln of a bigger problem, then if you're able to get optimal soln of smaller subproblem & they contribute optimally to the bigger problem.


f(5)
Bigger problem
f(4)
f(3)
f(3)
f(2)
f(2)
f(1)
f(1)
f(0)
f(1)
f(0)
f(1)
f(0)

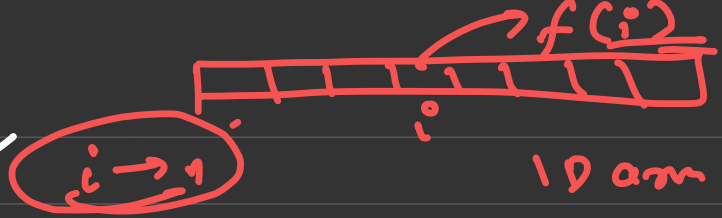
greedy → local optimum



DP → global optimum



→ State of a problem

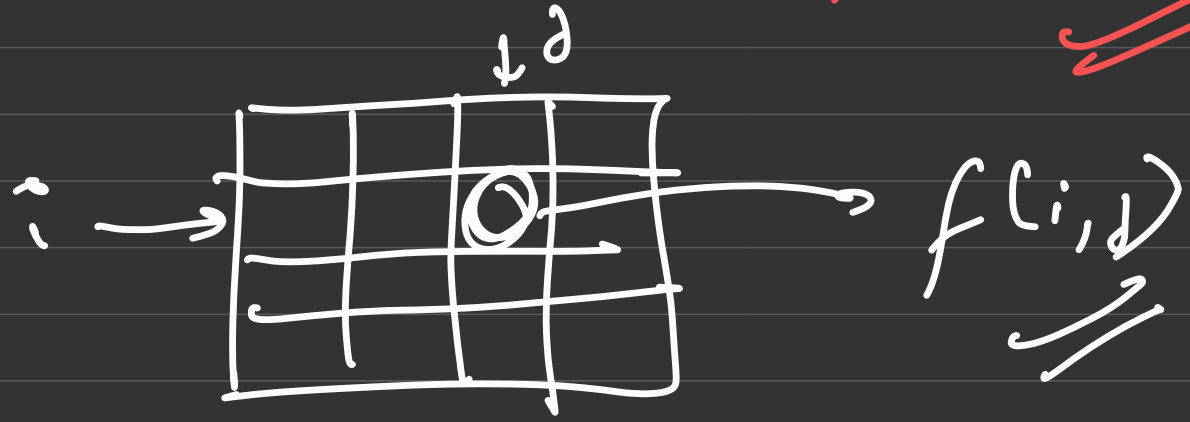


$$f(n) = f(n-1) + f(n-2)$$

↳ unique subproblem] unique state

x args → max → x dimensional storage

$$f(x,y) = f(x-1,y) + f(x,y-1)$$



Types of DP Soln

1. Top Down
2. Bottom UP

→ [h_1 h_2 h_3 h_4 h_5]

1) Opti
2) Count

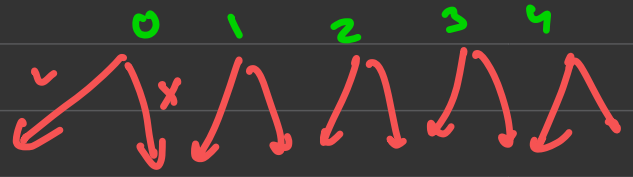
① No 2 adjacent houses can be robbed

[9, 2, 3, 6, 1]

(1) → Pick Not Pick → 10 / 20
→ fibonacci

element of choice

[9, 2, 3, 6, 1]



① Everywhere I have a choice
↳ Loot OR NotLoot

additional constraint

i^{th} → last loot can be at i-2 only

Recurrence

$$f(i) = \max\left(f(i-1), \; a[i] + f(i-2) \right)$$

$f(i)$ = max loot till the i^{th} house starting from 0^{th} house.

$f(i-1)$ → 1 child Not loot

$a[i] + f(i-2)$ → loot

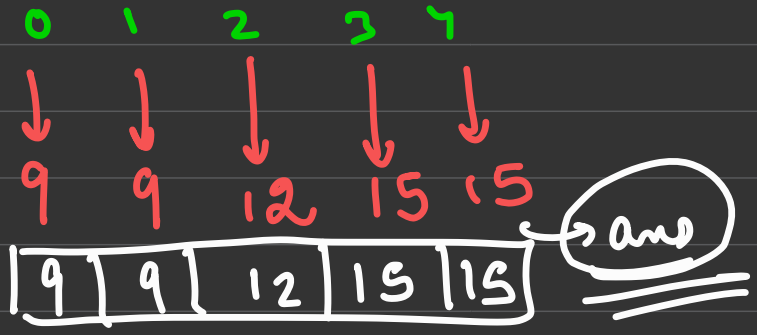
Bottom Up Tabulation

[9, 2, 3, 6, 1]

0-0

0-1

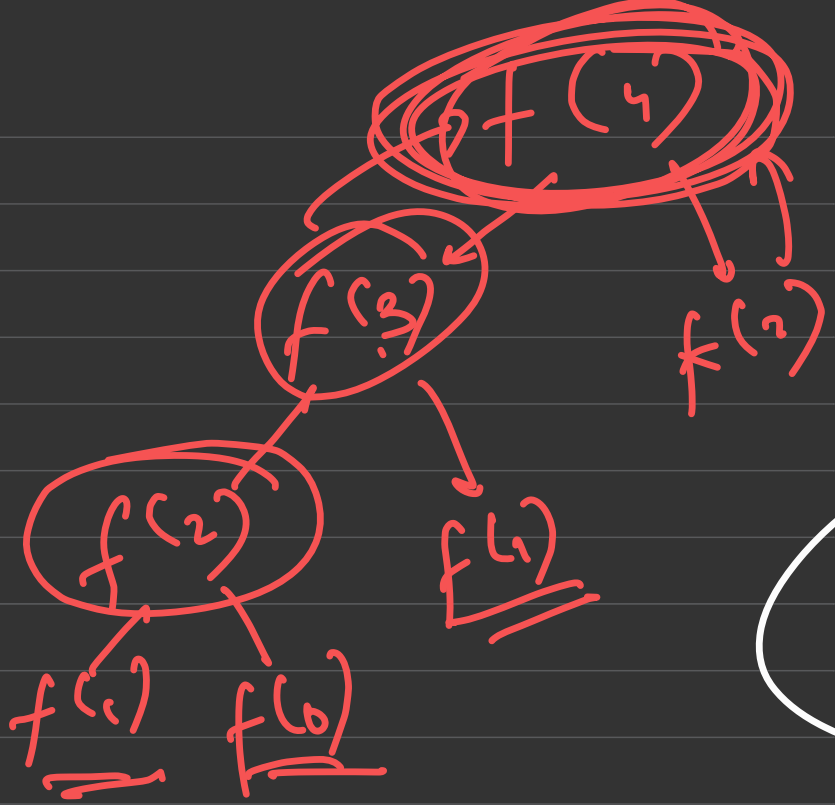
0-2



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

$dp[i] = \max(dp(i-1), a[i] + dp(i-2))$

1D space



1D → 0 1 2 3 4

-1	-1	x	y	z

dp(2)

↓ f(2)

Top Down / Memoization

3

10 30 40 30 20

0 1 2 3 4

final ans $f(n-1)$

$$f(i) = \min\left(f(i-j) + |a_i - a_{i-j}|\right) \quad \forall\, j \in [1, k]$$

min cost to reach from 0 to i

$$= \min\left(f(i-1) + |a_i - a_{i-1}|,\ f(i-2) + |a_i - a_{i-2}|,\ f(i-3) + |a_i - a_{i-3}| \text{ ---- --}\right)$$

Q » You've a notepad software with only one character 'P' displayed on it. You can perform 2 operation,

Op1 → Copy all → copy all chars on screen

Op2 → Paste → Paste whatever you copied last time

Given, a value n (integer) return min ops to get exactly n 'P's on the screen.

n=3 → Ans → 3

'P' Op-copy Paste Paste

PPP


9
ppp ppppppp
ppp
cpp
cpp
6

$$f(i) = \min\left(f(j) + C(i/j)\right)$$

$f(i)$: min ops to print i p's

if $i \% j == 0$

$\forall j \in [1, i-1]$

i = 24

12 ..... 18 19 20 21 22 23 24

$f(24) = f(12) + (24/12)$

$f(12) = f(6) + (12/6)$

$f(6) = f(3) + (6/3)$

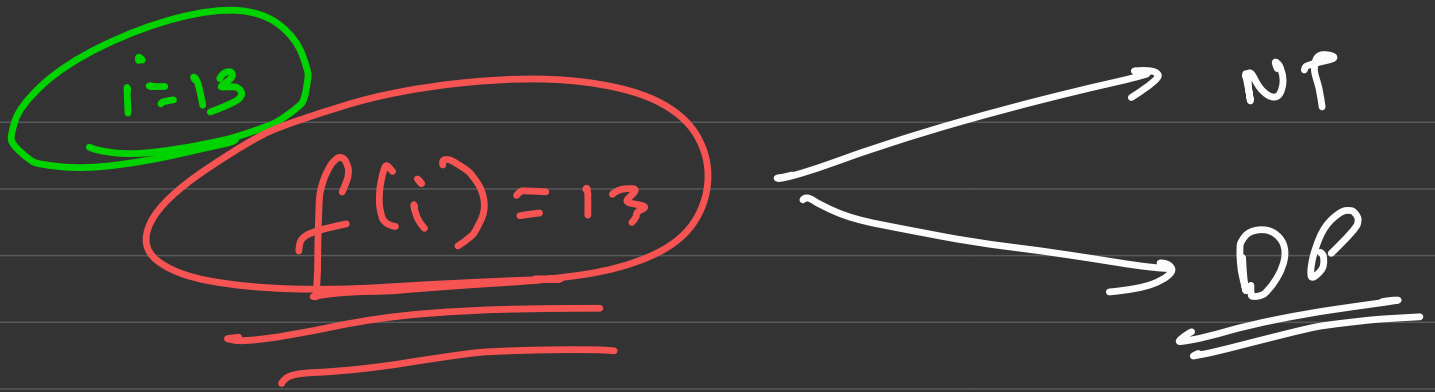
PPPPPP

3 + 2 + 2 + 2 → 9

CP

PPPPPPPPPPPP

12


i=13
f(i)=13
NT
DP