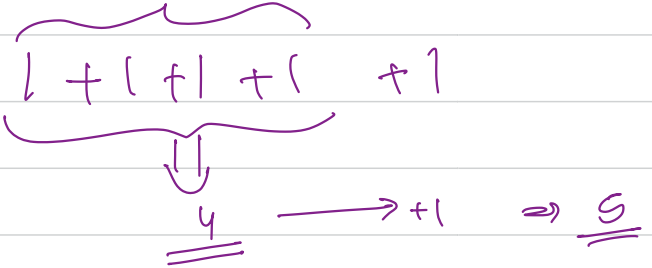
Agenda → Introduction to DP

2:30 PM → Problem solving DP

Pre-requisite → Basic Knowledge of loops and Recursion

Count Something




⇒ "If you don't remember your past"

"You're condemned to repeat it"

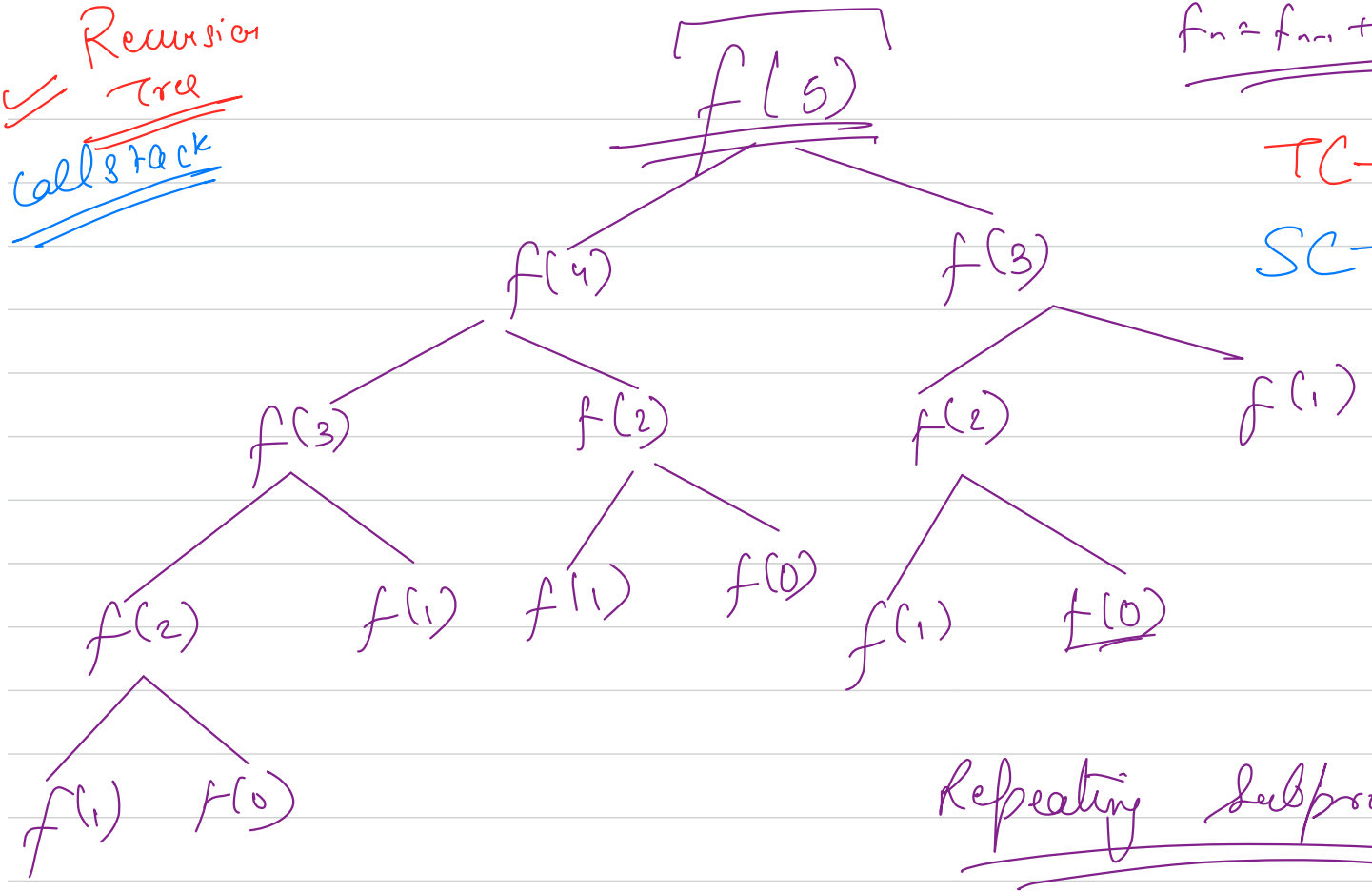
0, 1, 1, 2, 3, 5, 8, 13 .....

$f(n) = f(n-1) + f(n-2)$ → recurrence for fib

f → function which calculates $n^{th}$ fibonacci

Recursion Tree

call stack

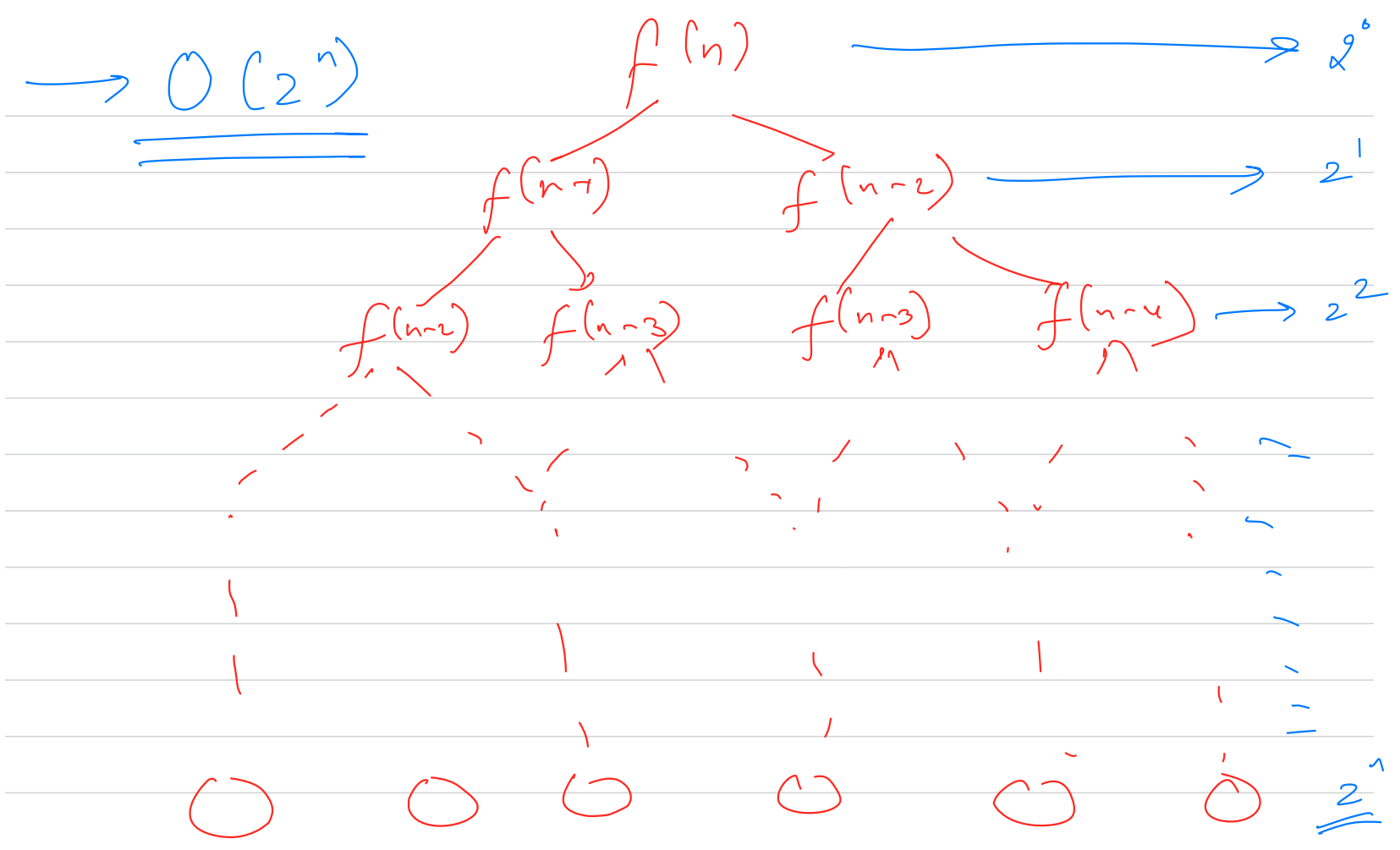


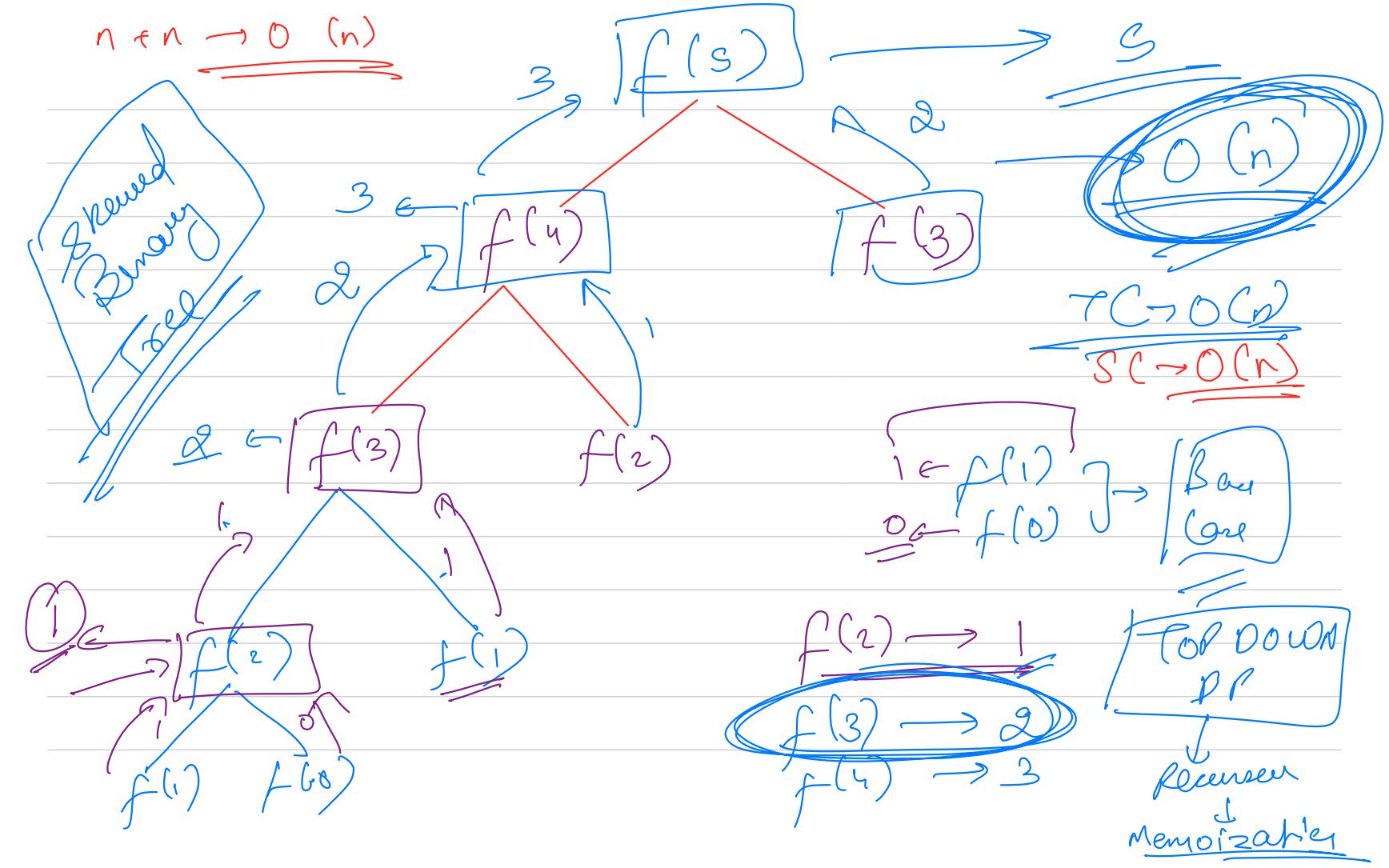

$f_n = f_{n-1} + f_{n-2}$

TC → $O(2^n)$

SC → $O(n)$

Repeating subproblems


O(2^n)
f(n)
2^0
f(n-1)
f(n-2)
2^1
f(n-2)
f(n-3)
f(n-3)
f(n-4)
2^2
2^n


n + n → O (n)
Skewed Binary Tree
f(5)
3
2
f(4)
f(3)
f(3)
f(2)
f(2)
f(1)
f(1)
f(0)
O (n)
TC → O(n)
SC → O(n)
Base Case
TOP DOWN DP
Recursion
Memoization
f(2) → 1
f(3) → 2
f(4) → 3

f(2) f(3) f(4) f(5)

0 1 1 2 3 5

TC O(n)

variable

a b c → ans

variables

SC → O(1)

DP

Bottom UP dP

→ iterative

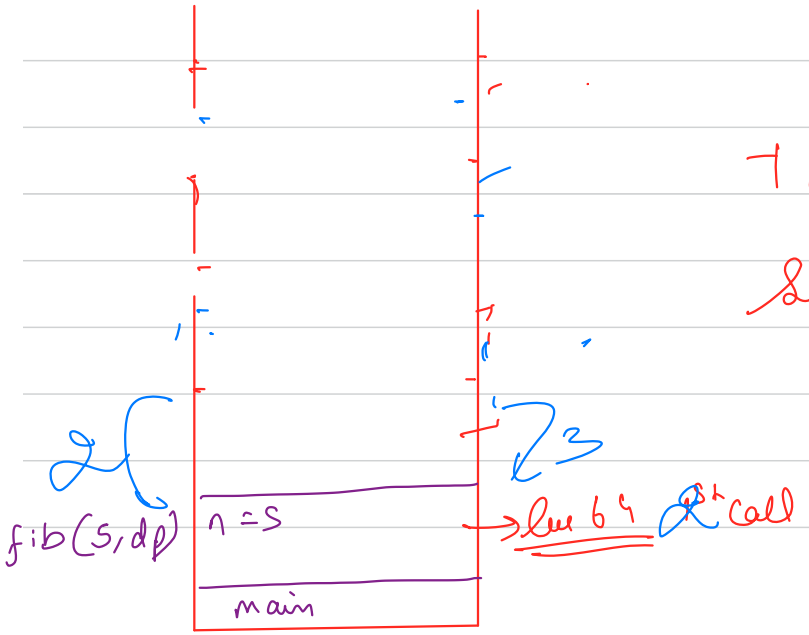
→ Tabulation

```
57 int fib(int n, int *dp) {
58     if(n == 0 or n == 1) return n;
59     if(dp[n] != -1) {
60         // if dp[n] was -1 that means we are calculat
61         // if dp[n] is not -1 then we already have a
62         return dp[n];
63     }
64     return dp[n] = fib(n-1, dp) + fib(n-2, dp);
65 }
```

TD (n=5)

Size = 6

| -1 | -1 | 1 | 2 | 3 | 5 |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 |

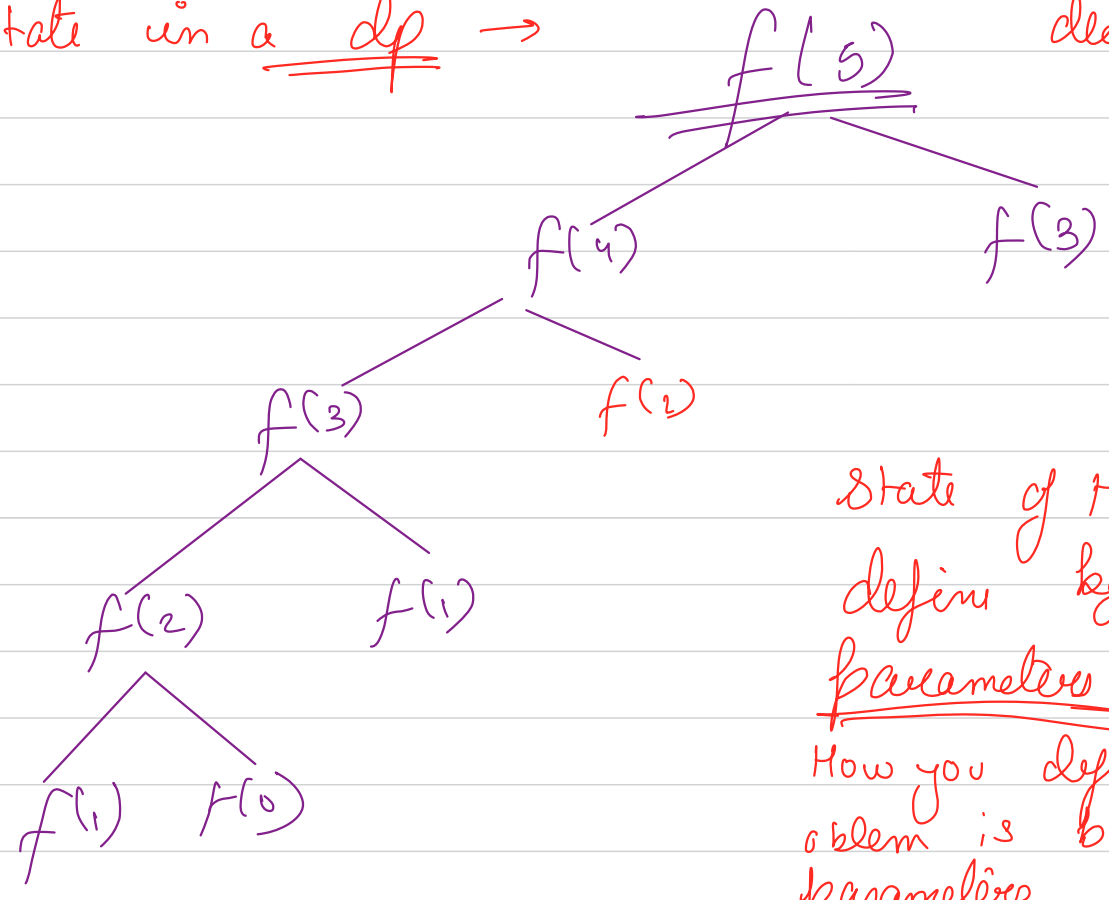



-1 denotes that this value is still not computed

n + n → 2n

∴ O(n) space

# State in a dp →




and no. of such changing parameters defines the dimension of DP storage

State of the DP is defined by some parameters

How you define a unique subproblem is based on some parameters

Q ⇒ There is a robber, who is going to rob some houses in a street. The main constraint is he cannot rob two consecutive adjacent houses. There is gold in every house defined by a value a[i].
Calculate the max loot robber can do.


Ans → 4

1, 2, 1, 1, 3

2 1 1 2

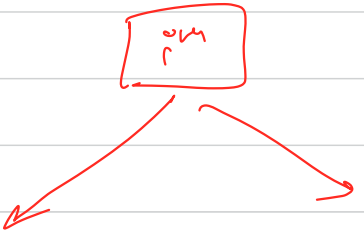
2 2 1 1

4 paradigms

Algorithms

- Greedy
- Divide & Conquer
- Brute force
- DP

→ Brute force → for every house you have 2 choice → either rob it or do not rob it



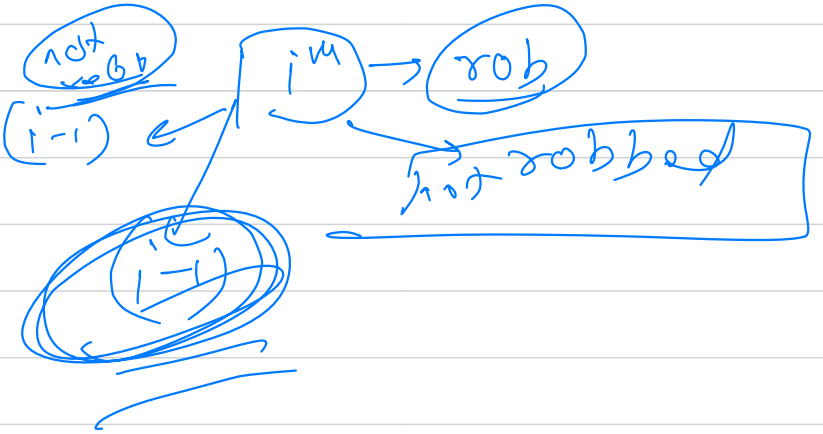
$f(i) = \max(f(i-2) + gold[i],\ f(i-1))$

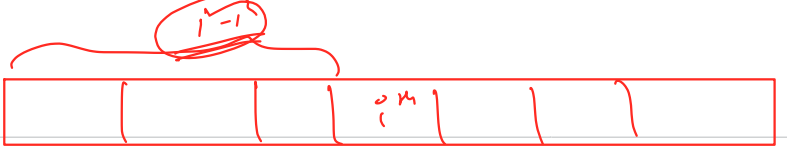
max profit till the $i^{th}$ house

TC $\rightarrow O(2^n)$

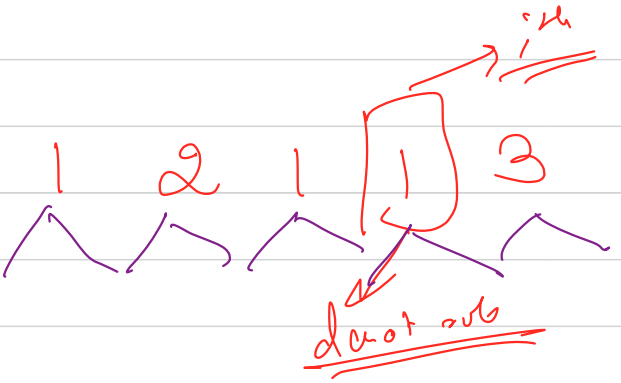
Base case $\rightarrow$ $i=0 \rightarrow a[0]$

$i=1 \rightarrow \max(a[0], a[1])$






$2^n$



Q. You have a value n, on this value you can perform 3 steps, if n is divisible by 2 divide n by 2, if it is divisible by 3 divide by 3 or Subtract by 1

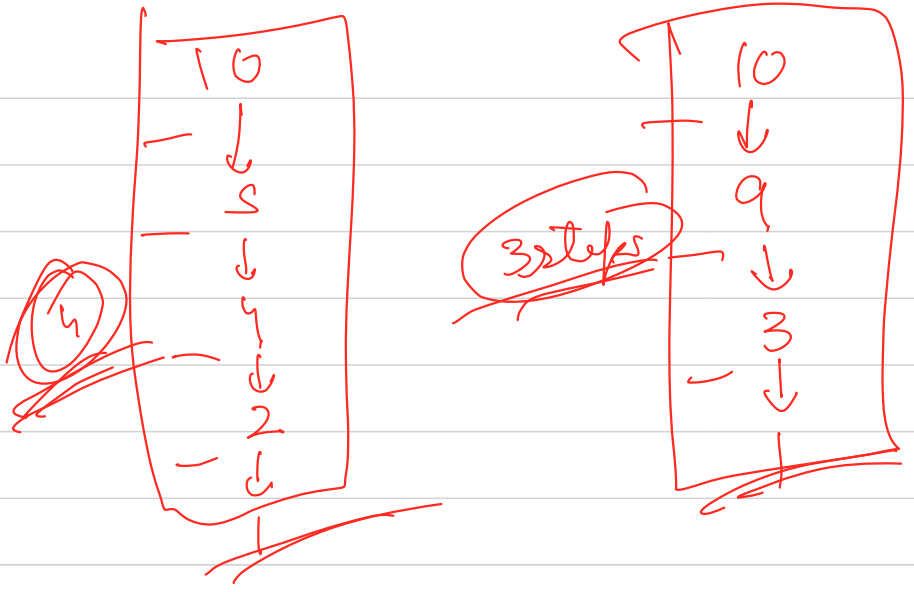
Calc Min steps required to reduce $n \longrightarrow 1$

$n = 7$

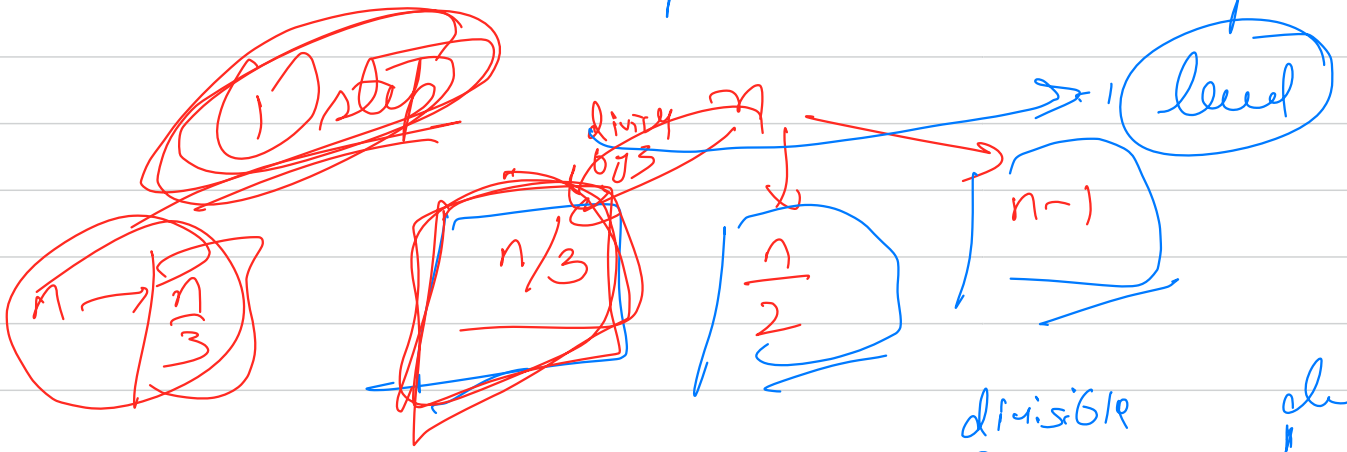
7 → 6 → 2 → 1

→ 3

$n \leq 10^6$


10
5
4
2
1
4
10
9
3
1
3 steps

we have to explore all the possibilities




$f(n) \Rightarrow 1 + \min\left[ f(n/3), f(n/2), f(n-1) \right]$

divisible ? else

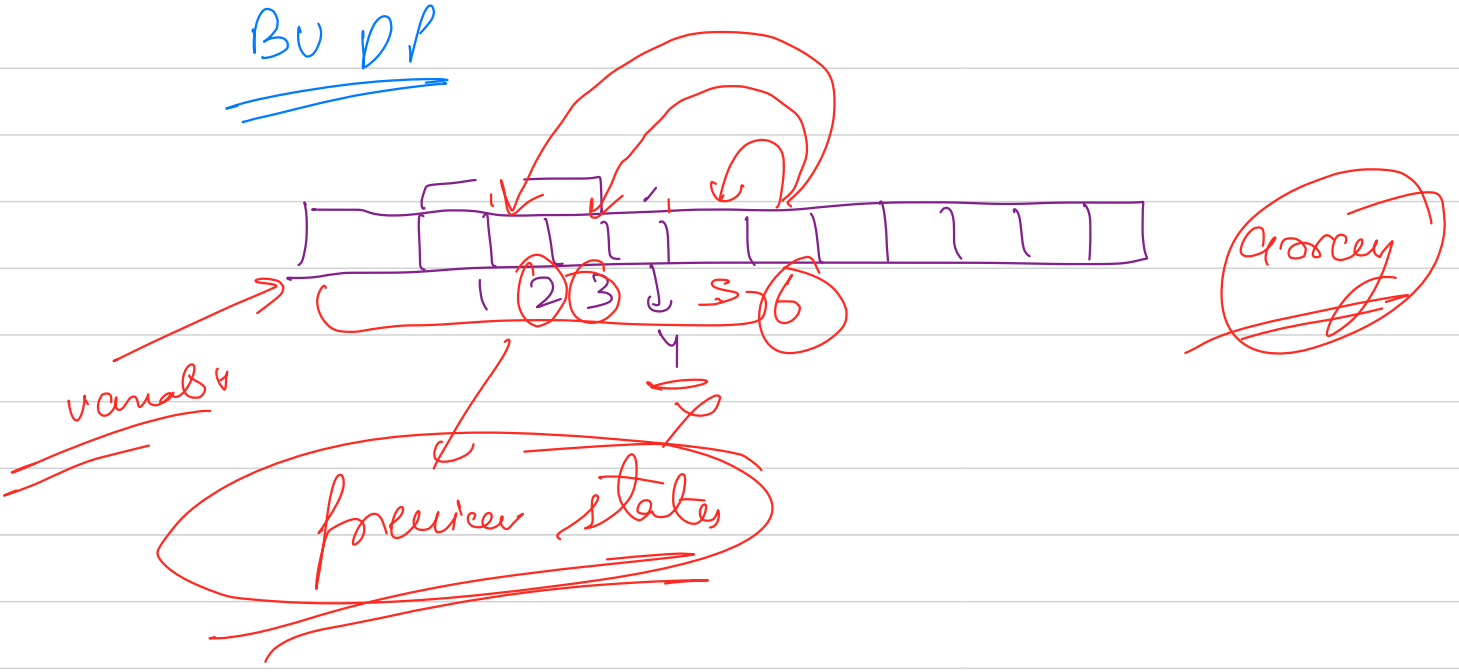
$f(n)$ → min steps to reduce n to 1

Base

2 → 1 step

3 → 1 step

1 → 0 step


BU DP
1 2 3 5 6
variable
previous states
greedy