Mock Interviews -

→ Last class of DSA

→ 30 days Expire.

Fibonacci Series

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

fib(n) = fib(n-1) + fib(n-2).

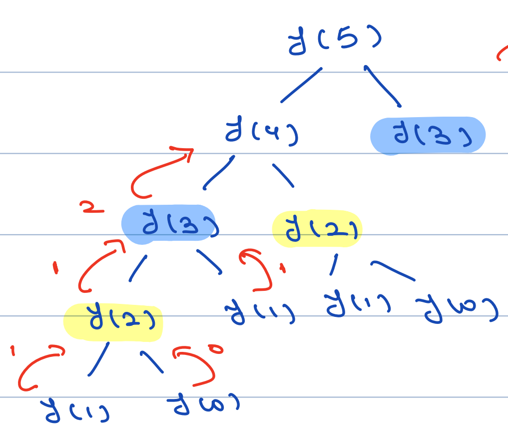
```
int fib(n) {
        if (n <= 1) { return n }

        return fib(n-1) + fib(n-2)
}
```

T.C → $O(2^n)$

S.C → $O(n)$

$f(10) = 2^{10} = 1024$.

$f(20) = 2^{20} = 10^6$



→ Too many repetitive calls.

fact(5)
↓
fact(4)
↓
fact(3)
⋮

Conditions for dp

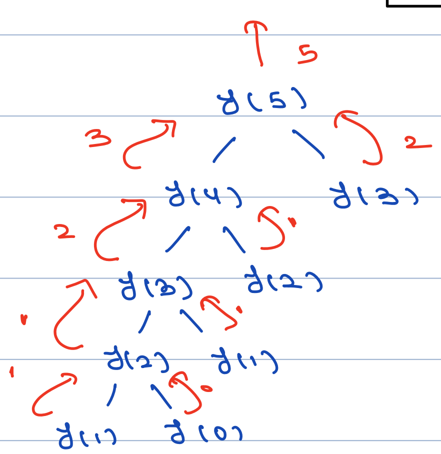
1) Overlapping subproblems.
2) optimal substructure.
   ↓
   solving a problem by breaking into subproblems.

DP → Calculate all unique results once.

```
int dp[n+1] = {-1}

int fib(n, dp)
{
    if (n<=1) { return n }
    if (dp[n] != -1) { return dp[n] }
    dp[n] = fib(n-1,dp) + fib(n-2,dp)
    return dp[n];
}
```

for N=5,

0	1	2	3	4	5
-1	-1	~~-1~~ 1	~~-1~~ 2	~~-1~~ 3	~~-1~~ 5



T.C → O(n)

S.C → O(n)

Types of DP

(Recursion)	(iteratively)
Memoization	Tabulation
Top down	Bottom up.

```
int fibTab(n) {
    int dp[n+1] = {-1}
    dp[0]=0;
    dp[1]=1;
    for(i=2; i<=n; i++) {
        dp[i]= dp[i-1]+dp[i-2];
    }
    return dp[n];
}
```

T.C → O(n)
S.C → O(n)

0	1	2	3	4	5
~~-1~~ 0	~~-1~~ 1	1	2	3	5

→ dp table.

→ dp state.

→ dp Expression.

dp[i]= dp[i-1]+dp[i-2];

T.C → O(n)
S.C → O(1)

```
int fib(n)
{
    a = 0, b = 1;
    for (i = 2; i <= n; i++) {
        c = a + b;
        a = b;
        b = c;
    }
    return c;
}
```

a	b	c
~~0~~	~~1~~	~~1~~
~~1~~	~~1~~	~~2~~
~~1~~	~~2~~	~~3~~
~~2~~	~~3~~	5
3	5	

We can optimise space After tabulation if your current state is dependent upon some fixed previous

States

pf[5] → 0 to 5

pf[5] → pf[4] + arr[5];

Break 8:04 Am - 8:14 Am

Stairs

Given N stairs, how many ways we can go from 0^{th} step to n^{th} step.

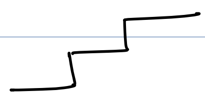
You can take a step of len 1 or len 2 at a time.

eg1 $n=1$

$\{1\} \Rightarrow 1$

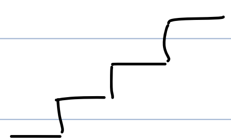
$n=2$

$\{1\ 1,\ 2\} \Rightarrow 2$

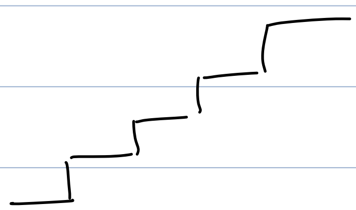


$n=3$

$\{1\ 1\ 1,\ 1\ 2,\ 2\ 1\} \Rightarrow 3$



$n=4$



$$\left\{\begin{matrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & \\ 1 & 2 & 1 & \\ 2 & 1 & 1 & \\ 2 & 2 & & \end{matrix}\right\} \Rightarrow 5$$

4^{th} step $\Rightarrow$ 5.

3rd step

$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & \\ 2 & 1 & \end{pmatrix}$ 1

2nd step

$\begin{pmatrix} 1 & 1 \\ 2 & \end{pmatrix}$ 2

n^{th} step

($n-1^{th}$ step) 1

($n-2^{th}$ step) 2

5^{th} step $\Rightarrow$ 8 ways.

4^{th} step

$\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & 2 & \\ 1 & 2 & 1 & \\ 2 & 1 & 1 & \\ 2 & 2 & & \end{pmatrix}$ 1

3rd step

$\begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & \\ 2 & 1 & \end{pmatrix}$ 2

dp[i] $\longrightarrow$ no. of ways to reach ith step.

0	1	2	3	4	5
1	1	2	3	5	8
-	1	1 1	1 1 1	1 1 1 1	1 1 1 1 1
		2	2 1	2 1 1	2 1 1 1
			1 2	1 2 1	1 2 1 1
				1 1 2	1 1 2 1
				2 2	2 2 1
					1 1 1 2
					2 1 2
					1 2 2

$dp[i] = dp[i-1] + dp[i-2]$

$dp[0] = 1, \quad dp[1] = 1$

return dp[N]

Space Optimization.

Ques

Find minimum number of perfect sqaures required to get sum = N.

↳ 1, 4, 9, 16, 25, 36 . . .

$N = 2 = 1^2 + 1^2 \Rightarrow 2$

$N = 3 = 1^2 + 1^2 + 1^2 \Rightarrow 3$

$N = 4 = 2^2 \Rightarrow 1$

$N = 5 = 2^2 + 1^2 \Rightarrow 2$

$N = 6 = 2^2 + 1^2 + 1^2 \Rightarrow 3$

$N = 50 \Rightarrow 7^2 + 1^2$ or $5^2 + 5^2$ $\Rightarrow 2$

Greedy Idea

$N = 50 - 7^2 = 1 - 1^2 = 0 \Rightarrow 2$

$N = 70 - 8^2 = 6 - 2^2 = 2 - 1^2 = 1 - 1^2 = 0$

$\Rightarrow 4$

↳ $6^2 + 5^2 + 3^2$

↳ 3 perfect squares needed.

$N = 12$,

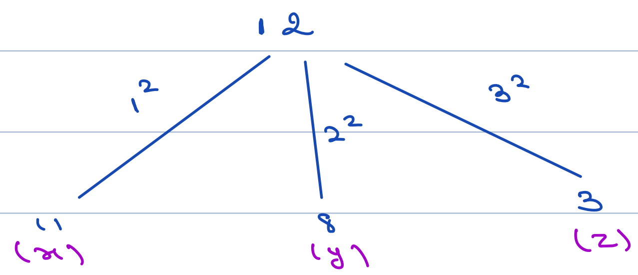
$12 - 8^2 = 3 - 1^2 = 2 - 1^2 = 1 - 1^2 = 0 \Rightarrow 4$

$(2^2 + 2^2 + 2^2)$

↳ 3 perfect squares.

Greedy failed.

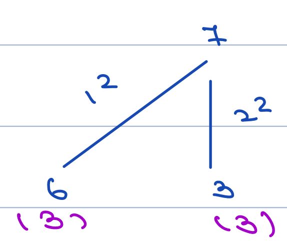
Soln 2:-

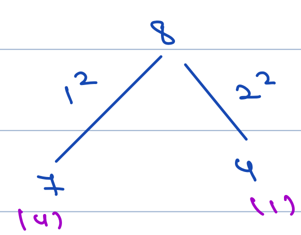


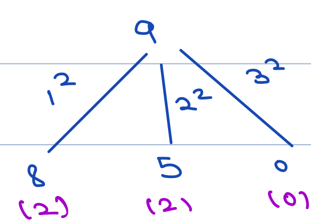
min(x, y, z) + 1

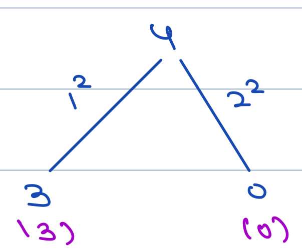
dp[i] = Min no. of perfect squares to form i.

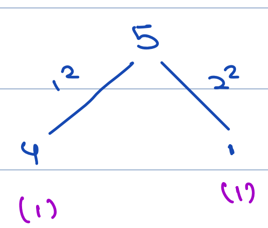
0	1	2	3	4	5	6	7	8	9
0	1	2	3	1	2	3	4	2	1
	1^2	1^2+1^2	$1^2+1^2+1^2$	2^2	2^2+1^2	$2^2+1^2+1^2$	$2^2+1^2+1^2+1^2$	2^2+2^2	3^2

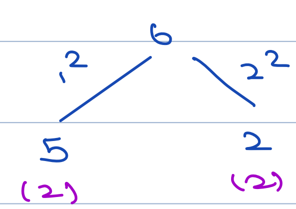






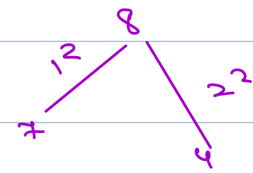






for all

$\forall i$, dp[i] = $+\infty$




```
dp[0] = 0;
for i → 1 to N   (→ 8)
    for (x=1; x*x<=i; x++) {
        dp[i] = min(dp[i], dp[i-x2] +1);
                           dp[7]
                           dp[4]
    }
}
```

T.C → $O(n\sqrt{n})$

S.C → $O(n)$

Space optimisation is not possible since we can't determine a constant number of previous data for every index

```
solve (N) {      (→ 9)
    if (N==0) { return 0 }
    ans = +∞
    for (x=1; x*x <= N; x++) {
        ans = min(ans, solve(N-x2)) +1;
    }
    return ans;
}
```