$$\text{5 students} \longrightarrow \begin{array}{ccccc} 1 & 2 & 3 & 4 & 5 \\ \end{array}$$
$$\text{5 students} \longrightarrow [\,3 \quad 2 \quad 3 \quad 1 \quad 5\,]$$

Total sum $= 3+2+3+1+5 = \underline{14}$

$$\text{1 more student} \longrightarrow \begin{array}{c} 6 \\ [\,2\,] \end{array}$$

Total sum → $3+2+3+1+5+2 = \underline{16}$ ✓

→ $14+2 = \underline{16}$ ✓ ✓

less time & efforts

∵ we remember previous calculated value. ✓

$A = [\,3 \quad 2 \quad 3 \quad 1 \quad 5\,]$

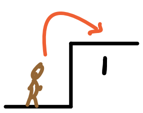
$P = [\,3 \quad 5 \quad 8 \quad 9 \quad 14\,]$

$P[i] = P[i-1] + A[i]$ ✓

Q → In how many ways we can climb N stairs s.t

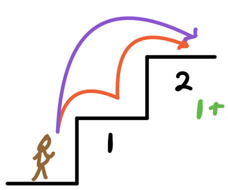
Informatica

in one step we can climb 1 or 2 stairs only.



N = 1

Ans = $\underline{1}$



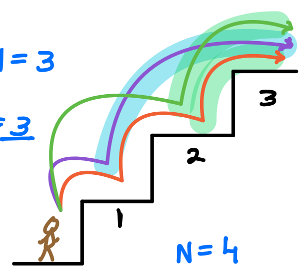
N = 2

Ans = $\underline{2}$

$1+1 = \underline{2}$

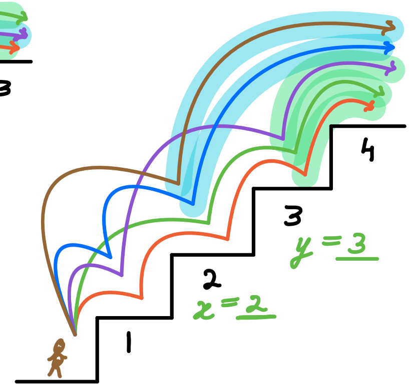
N = 3

Ans = $\underline{3}$

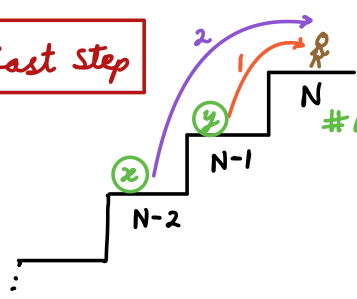


N = 4

Ans = $\underline{5}$



→ $\boxed{\text{Last Step}}$

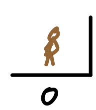


#ways $= \underline{x+y}$ ✓

{ #ways to do a task ✓

\# steps taken to do a task ✗

$$\boxed{\#ways(N) = \#ways(N-1) + \#ways(N-2)}$$



#step = 0 ✗

#ways = $\underline{1}$ ✓

$\boxed{\#\text{ways to not do anything} = 1}$

ways $(0) = 1$

ways $(1) = 1$

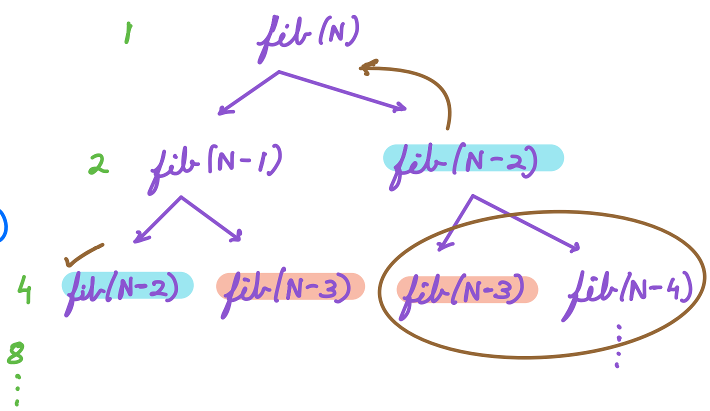
Fibonacci Numbers

$\boxed{\#\text{ways}(N) = fib(N+1)}$ ✓

#ways →	0	1	2	3	4	5	6	
	0	1	1	2	3	5	8	13 . . .
N =	0	1	2	3	4	5	6	7

$$\boxed{F(i) = \begin{cases} F(i-1) + F(i-2), & i > 1 \\ i, & i \le 1 \end{cases}}$$

```
int fib(N) {
    if (N <= 1) return N
    return fib(N-1) + fib(N-2)
}
```



$TC = O(2^N)$ $SC = O(N)$

Optimal Substructure → Ans for current problem can be found using answer of subproblems. ✓

⇒ DP ✓

Overlapping Subproblems → Same subproblem is calculated multiple times. ✓

↓

store & reuse the calculation ✓

(Memoization)

```
// ∀i, F[i] = -1
int fib(N) {
    if (N <= 1) return N
    if (F[N] != -1) return F[N]
    F[N] = fib(N-1) + fib(N-2)
    return F[N]
}
```

$TC = O(N)$ $SC = O(N+N) = O(N)$

$\boxed{O(2^N) \xrightarrow{DP} O(N)}$ ✓

Types of DP → 1) Top-Down / Recursive DP
(bruteforce + memoization)

2) Bottom-Up / Iterative DP
(start with base cases & use them to calculate ans of big problems)

$F[0] = 0 \qquad F[1] = 1$

for $i \to 2$ to N {
  $F[i] = F[i-1] + F[i-2]$
} return $F[N]$

$TC = O(N) \qquad SC = O(N)$

a b c
a b c

$a = 0 \qquad b = 1$

for $i \to 2$ to N {
  $c = a + b$
  $a = b$
  $b = c$
} return c $TC = O(N) \qquad SC = O(1)$

Recursive DP → Easy to understand & write the code.
Bruteforce → store calculated value.

Iterative DP → No recursion space.
Sometimes it is possible to optimize SC.

Q → Find min count of perfect sq. to add to get sum = N.

1, 4, 9, 16 ...

$N = 2 \leftarrow 1^2 + 1^2$ Ans = 2

$N = 4 \leftarrow 2^2$ Ans = 1

$N = 6 \leftarrow 2^2 + 1^2 + 1^2$ Ans = 3

$N = 50 \leftarrow 7^2 + 1^2$ Ans = 2
$\leftarrow 5^2 + 5^2$

min count ⇒ large perfect sq.
Greedy

$50 - 7^2 = 1 - 1^2 = 0$

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

Greedy → $\underline{12} - 3^2 = 3 - 1^2 = 2 - 1^2 = 1 - 1^2 = 0$ ✗

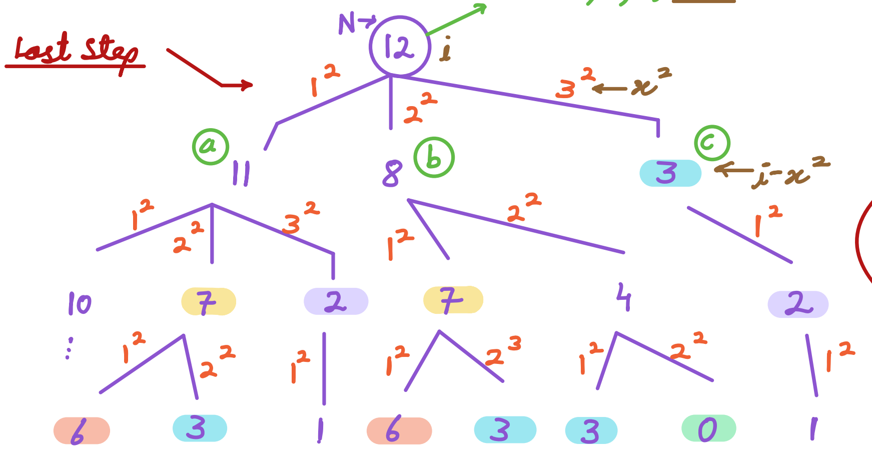
Solution → $12 = 2^2 + 2^2 + 2^2$

$Ans(12) = \underline{3}$

Last Step

$\min(a,b,c) + 1$ ✓

$\boxed{ans(0) = 0}$ ✓



optimal substructure ✓

overlapping subproblems ✓

DP ✓

$\forall i,\ Ans[i] = INT\_MAX$ ✓

$Ans[0] = 0$

$TC = \underline{O(N\sqrt{N})}$

$SC = \underline{O(N)} \leftarrow \underline{H.W}$ ✓

```
for i → 1 to N {                              ← N
    for (x=1 ; x*x <= i ; x++) {              ← √N
        Ans[i] = min (Ans[i], Ans[i - x*x] + 1)
                                8 - 2² = 4
    }
}   return Ans[N]
```

$x^2 <= N$

$x <= \sqrt{N}$

$x \rightarrow \underline{1 \text{ to } \sqrt{N}}$ ✓

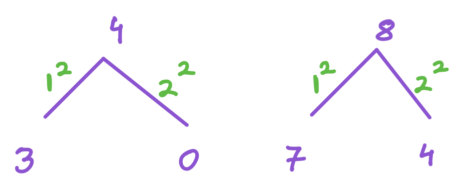
$N = 7$

0	1	2	3	4	5	6	7	8
0	1	2	3	~~4~~ 1	2	3	4	~~8~~ 2

✓

$i \rightarrow \cancel{1}\ \cancel{2}\ \cancel{3}\ \cancel{4}\ \cancel{5}\ \cancel{6}\ \cancel{7}\ 8$

$x \rightarrow \cancel{1}\ 2$



$\max x = \underline{floor(sqrt(N))}$