SCALER

Dynamic Programming 1

TABLE OF CONTENTS

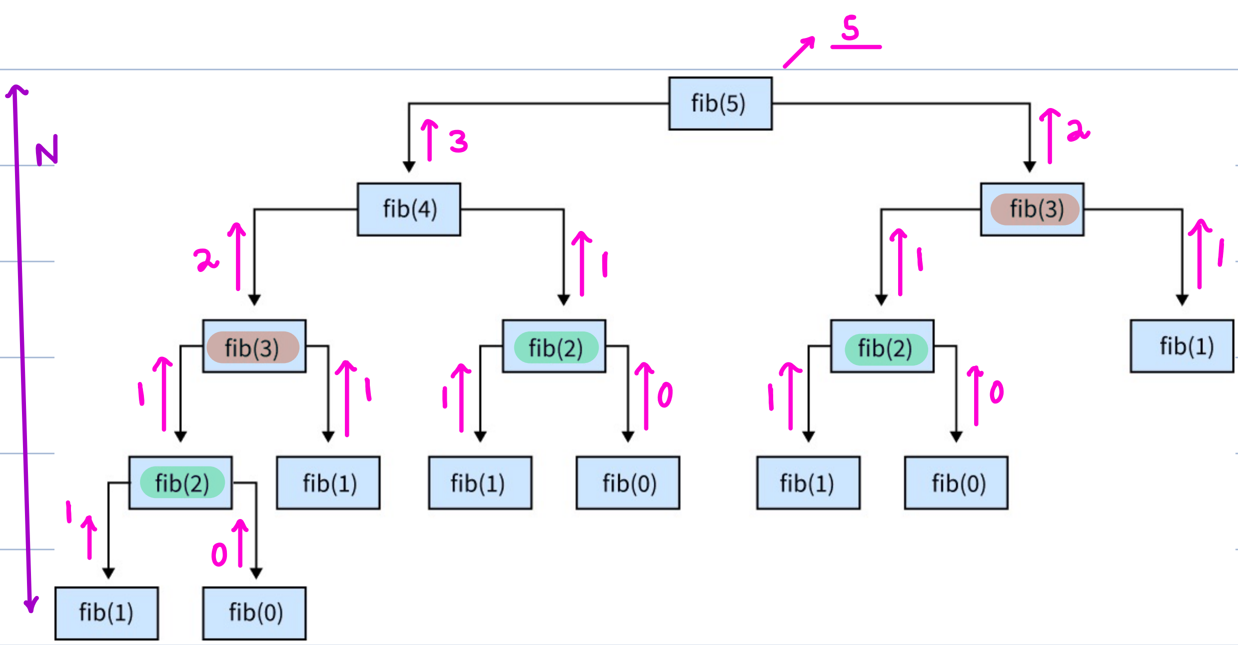
Notes

Nth Fibonacci Number

N →	0	1	2	3	4	5	6	7	8	9	10
fib(N) →	0	1	1	2	3	5	8	13	21	34	55

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

$TC = \underline{O(2^N)}$ $SC = \underline{O(N)}$



1. Optimal Structure → A problem can be solved by dividing it into smaller subproblems.

2. Overlapping Subproblems → Same subproblems are calculated multiple times.

$\boxed{\text{store \& reuse}}$ → Dynamic Programming

```
// ∀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]
}
```

$TC = \underline{O(N)}$ $SC = O(N + N) = \underline{O(N)}$

$2^N \longrightarrow N$ (using DP)

Top - down Approach → Memoization in recursive solutions.

Start with big problem, break it to smaller subproblems & recursively calculate answer.

→ Easy to understand & implement.

Bottom - up Approach → Iterative approach.

Start with base case & iteratively calculate ans for bigger problem.

→ No recursion space ⇒ potential to optimize SC in some cases.

```
F[0] = 0        F[1] = 1

for i → 2 to N {
    F[i] = F[i-1] + F[i-2]
}

return F[N]
```

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

Further S.C Optimisation

a b c

a b c

```
a = 0      b = 1
for i → 2 to N {
    c = a + b
    a = b
    b = c
}
return c
```

TC = O(N)   SC = O(1)

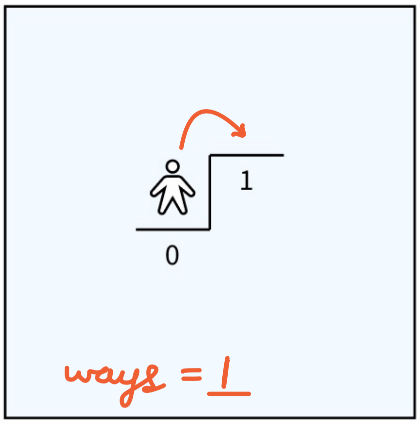
Climbing Stairs

$1 \leq N \leq 10^5$

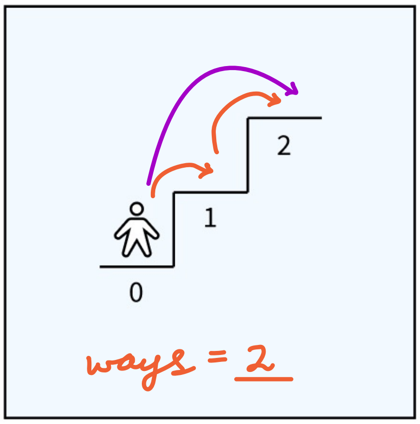
Calculate the number of ways to reach Nth stair.

You can take 1 step at a time or 2 steps at a time.

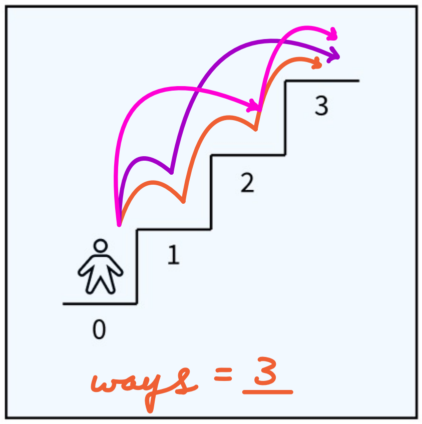
N = 1



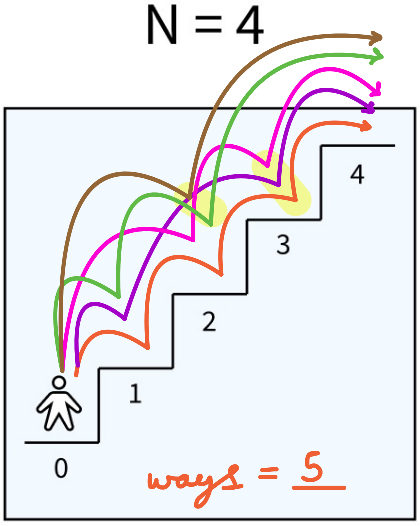
N = 2



N = 3

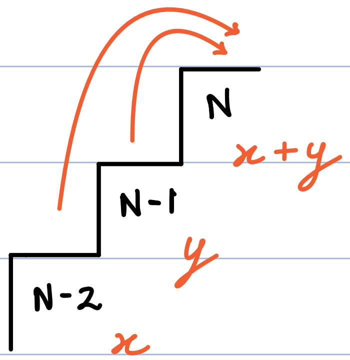


N = 4



ways(0) = 1

way to do a task = 0
→ impossible



$$\boxed{\begin{array}{l} ways(N) = ways(N-1) + ways(N-2) \\ ways(0) = ways(1) = \underline{1} \end{array}}$$

code $\rightarrow$ same as fibonacci number.

< Question > : Find the minimum number of perfect squares required to get sum = N?

[numbers can repeat]

N = 6 $1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2$

$2^2 + 1^2 + 1^2$ Ans = 3

N = 10 $1^2 + 1^2 + \dots$ 10 times | $2^2 + 2^2 + 1^2 + 1^2$

$2^2 + 1^2 + 1^2 + \dots$ 6 times | $3^2 + 1^2$ Ans = 2

N = 9 3^2 Ans = 1

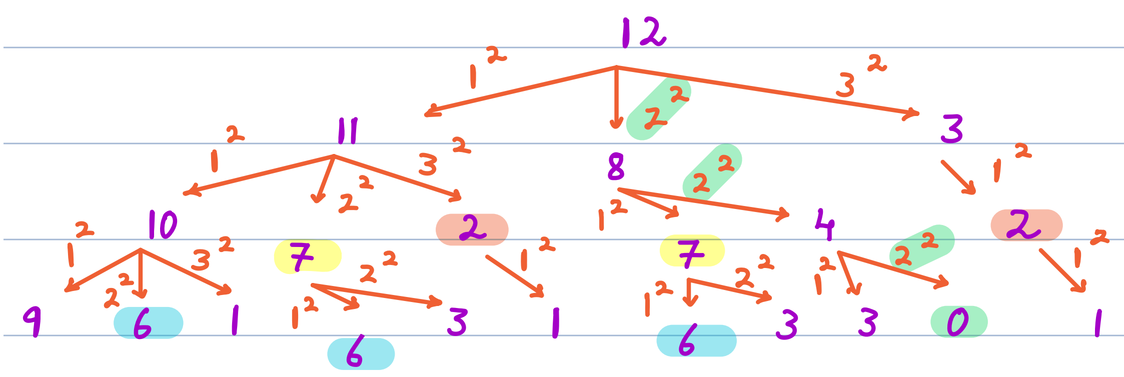
N = 5 $2^2 + 1^2$ Ans = 2

→ use large perfect sq. to reduce count. (Greedy) ✗

N = 50 $7^2 + 1^2$ Ans = 2

N = 12 $3^2 + 1^2 + 1^2 + 1^2$ ✗

$2^2 + 2^2 + 2^2$ Ans = 3



optimal substructure ✓
overlapping subproblems ✓
} use DP

$cnt(12) = \min(cnt(12-1^2), cnt(12-2^2), cnt(12-3^2)) + 1$

$cnt(N) = \forall x \min(cnt(N-x^2)) + 1$

$1 <= x^2 <= N$

$\forall i$, $cnt[i] = i$ // $1^2 + 1^2 + \dots$ i times   ($cnt[0] = 0$)

```
for i → 1 to N {
    for x → 1 to √i {        // 1<=x²<=i     x→1, 2
        cnt[i] = min(cnt[i], cnt[i-x²] + 1)
                                5 - 2²   + 1
    }
} return cnt[N]
```

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

N = 5

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