Dynamic Programming

$\{2 \quad 4 \quad 1 \quad 3 \quad 4\} = 14$

$+5$

$19.$

5

$$\left[\, pSum[i] = pSum[i-1] + arr[i] \,\right]$$

- **N^{th} Fibonacci Number**

N = 10.

0 1 1 2 3 5 8 13 21 . 34 55

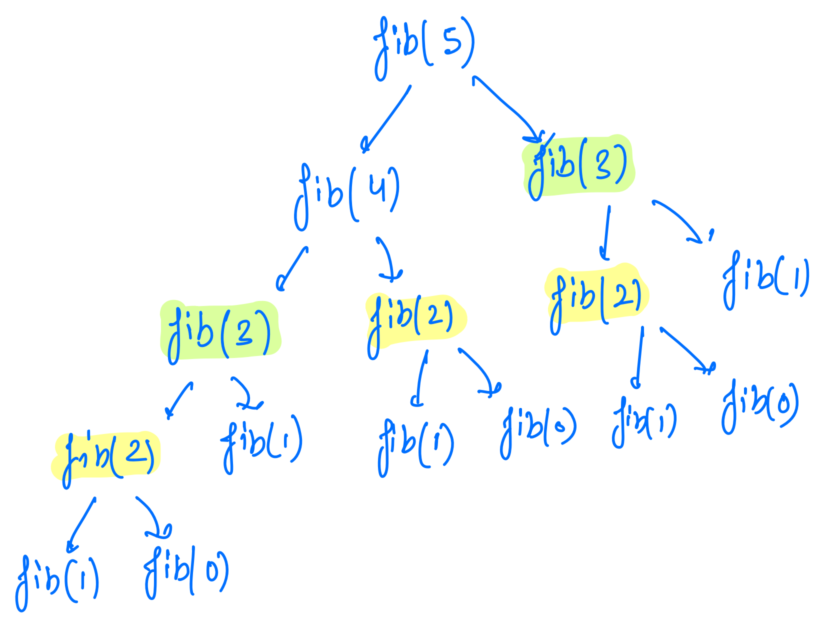
$$\boxed{fib(N) = fib(N-1) + fib(N-2)}$$ def.

$fib(0) = 0, \; fib(1) = 1$

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

$\left[\begin{array}{l} T.C \to O(2^N) \\ S.C \to O(N) \end{array}\right]$

\# optimal sub-structure. solving problem by solving smaller sub-problems.



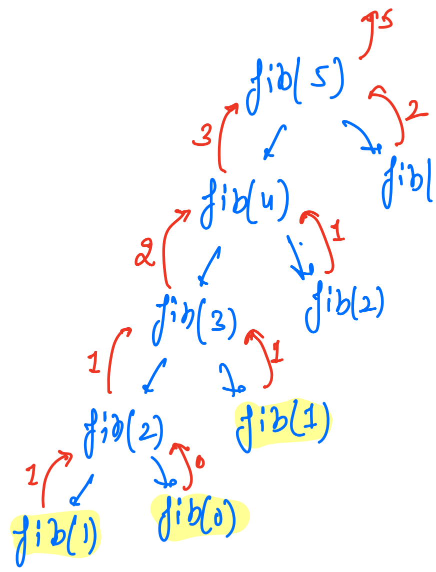
Overlapping sub-problems → solving same problem again and again.

[Solution ⇒ Store the result about already solved problem and use it]

N^{th} fibonacci:

```
int f[N+1]; → initialise with -1

int fib(int N){
    if(N <= 1) { return N }
    //if already solved, don't solve it again
    if( f[N] != -1) return f[N];
      ans = fib(N-1) + fib(N-2)
       f[N] = ans
      return ans;
}
```



f →

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

[T.C → O(N)
S.C → O(N)]

Top-down → starting from biggest problem
(Recursion)
Memoization

Bottom-Up → start from the smallest problem.
(iterative)
Tabulation

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

```
int dp[N+1];
    dp[0] = 0 , dp[1] = 1
  for( i = 2 ; i <= N ; i++){
        dp[i] = dp[i-1] + dp[i-2]
  }
```

$\left[\begin{array}{l} T.C \to O(N) \\ S.C \to O(N) \end{array}\right]$

No recursive stack

Can we further optimize S.C ?

```
int a = 0 , b = 1
  int c ;

for( i = 2 ; i <= N ; i++){
        c = a + b ;
        a = b
        b = c
}
```

2 3 5
a b c

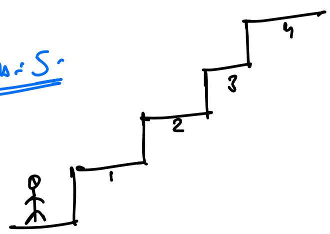
$\left[\begin{array}{l} T.C \to O(N) \\ S.C \to O(1) \end{array}\right]$

- **Climbing Stairs** (N stairs)

  1 step ↙ ↘ 2 step

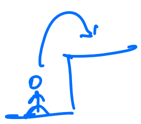
N = 4.

Number of ways to reach N^{th} stair?

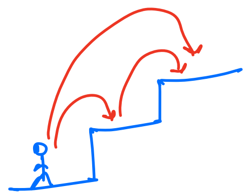


ans = 5.

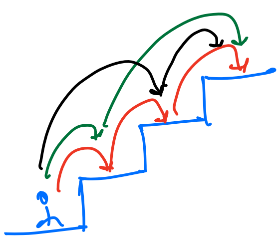
1111
112
121
211
22



N = 1
ans = 1



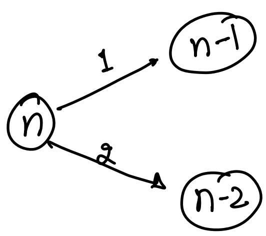
N = 2
ans = 2

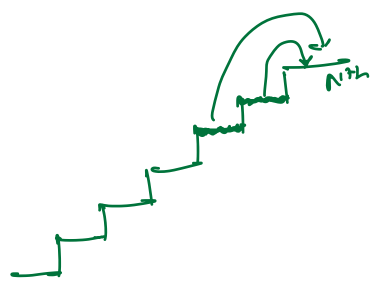


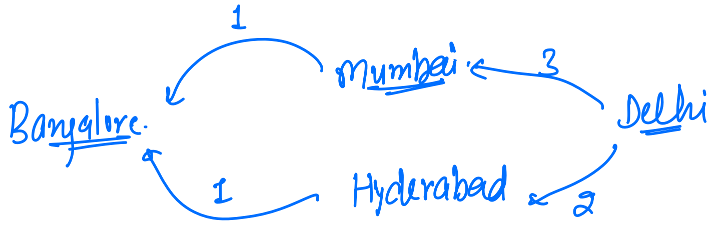
111
12
21

N = 3
ans = 3

Idea 1 → Try all possible ways. → Backtracking







$(3*1) + (2*1)$
$= 5.$

$$\left[ways(N) = ways(N-1) * 1 + ways(N-2) * 1 \right]$$

$ways(1) = 1$, $ways(2) = 2$

Similar to fibonacci.

Q) Find minimum number of perfect squares required to get sum = N.

$N = 6$.
$1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 = 6$
$2^2 + 1^2 + 1^2 = 6$
[ans = 3]

$N = 10$
$1^2 + 1^2 + 1^2 + \text{———} 1^2$
$2^2 + 1^2 + 1^2 + 1^2 +$ 6 times
$2^2 + 2^2 + 1^2 + 1^2$
$3^2 + 1^2$
[ans = 2]

$N = 9$
$1^2 + 1^2 + 1^2$ ——— 9 times
$2^2 + 1^2 + 1^2 +$ — — 5 times
$2^2 + 2^2 + 1^2$
3^2
[ans = 1]

$\boxed{N - \text{nearest perfect square}}$

~~Greedy Approach~~

$N = 10$. $\xrightarrow{-9}$ 1 $\xrightarrow{-1}$ 0

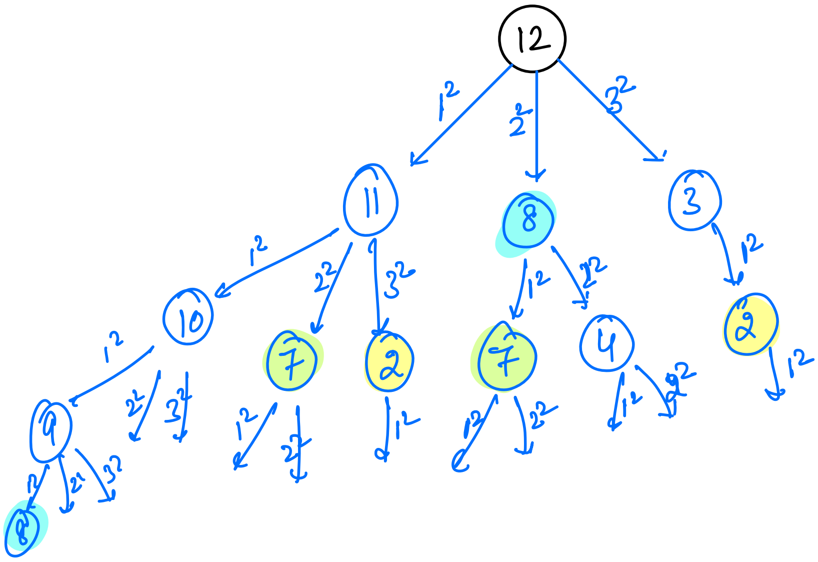
$N = 9$. $\xrightarrow{-9}$ 0

$N = 6$ $\xrightarrow{4}$ 2 $\xrightarrow{1}$ 1 $\xrightarrow{1}$ 0

$N = 12$. $\xrightarrow{9}$ 3 $\xrightarrow{1}$ 2 $\xrightarrow{1}$ 1 $\xrightarrow{1}$ 0

$N = 12$ $2^2 + 2^2 + 2^2$

B.F. → try every possible way to form the sum.



[optimal substructure
+
overlapping subproblems] ✓

$$minsq(12) = Min(minsq(11), minsq(8), minsq(3)) + 1.$$

$$minsq(12) = 1 + Min\{ minsq(12-1^2), minsq(12-2^2), minsq(12-3^2) \}$$

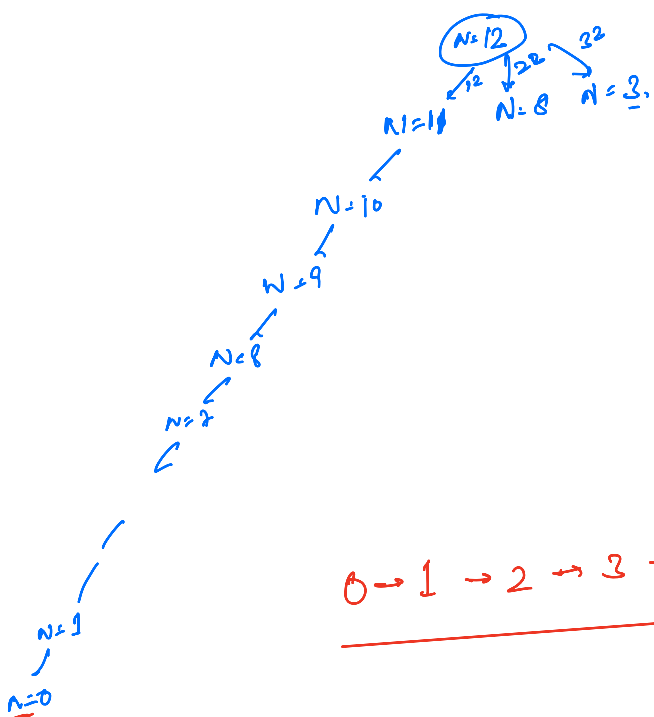
$$\left[squares(N) = 1 + Min\{ squares(N-x^2) \ \forall\ x^2 \le N \} \right]$$

$squares(0) = 0$.

pseudo-code.

```
int dp[N+1]; // initialise dp[i] → -1

int   pfsquares( int N, int[] dp){
    if (N == 0) {return 0}
    if ( dp[N] != -1) { return dp[N] };
    ans =   INT_MAX;
    for( x = 1 ; x * x <= N ; x++){
          ans = min( ans, pfsquares( N - x², dp));
    }
    dp[N] = ans+1;
    return dp[N];
}
```



$$\boxed{\begin{aligned} T.C &\to O(N\sqrt{N}) \\ S.C &\to O(N) \end{aligned}}$$

0 → 1 → 2 → 3 → 4 → ——— 12.

```
int dp[N+1];
dp[0] = 0

for( i=1 ;  i <= N;  i++){
        ans = INT_MAX;
        for( x = 1 ;  x * x <= i ;  x++){
                ans = min(ans , dp[ i - x^2] );
        }
        dp[i] = ans + 1
}
return dp[N];
```

$T.C \to O(N\sqrt{N})$

$S.C \to O(N)$

ans $\to \{\infty, 3, 2, 3\}$, $x = \cancel{1}\ \cancel{2}\ \cancel{3}\ 4$

N=12

dp →

0	1	2	3	4	5	6	7	8	9	10	11	12
0	1	2	3	1	2	3	4	2	1	2	3	3

i ↑ (at index 4)

x p