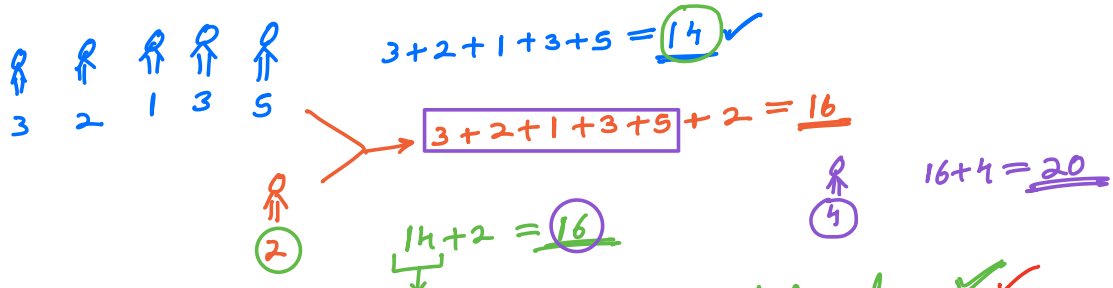
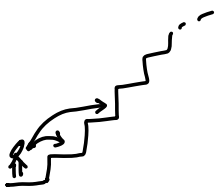
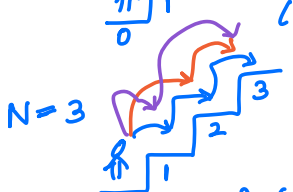


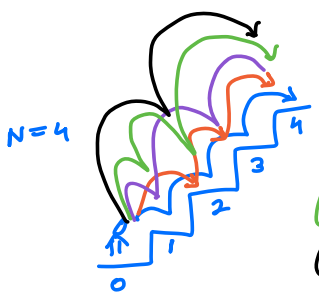
Q →

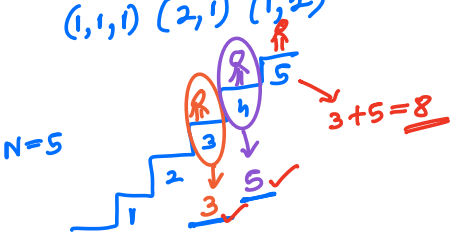


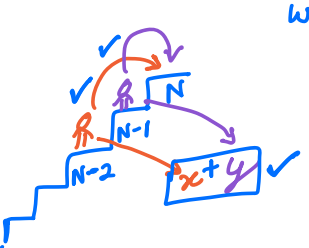
One step → climb 1 or 2 stairs.

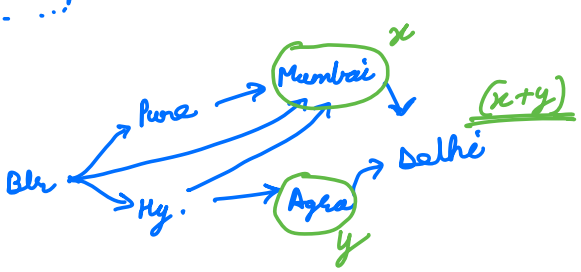
In how many ways can we climb N stairs.











look 1 state back from final position. ✓

ways (N) = ways (N-1) + ways (N-2)

ways (1) = 1

ways (0) = 1 (person do not move)

Fibonacci Numbers → $F[N] = F[N-1] + F[N-2]$ for $N > 1$ ✓

$F[N] = 1$ for $N <= 1$ ✓

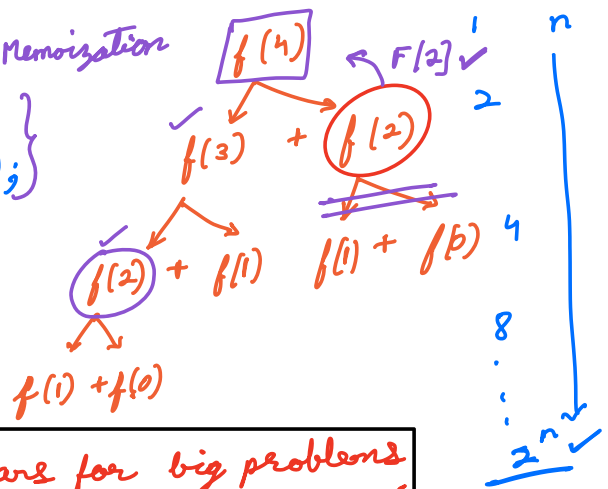
1	1	2	3	5	8	13 - -
0	1	2	3	4	5	6

```
int fib(n) {
    if (n <= 1) return 1;
    if (F[N] > 0) return F[N];
    F[N] = fib(n-1) + fib(n-2);
    return F[N];
}
```

TC $\leq O(2^N)$

SC $= O(N)$ → Height of recursion tree.

Memoization



Optimal substructure → Calculate ans for big problems using smaller subproblems. ✓

Overlapping subproblems → Same subproblem is computed multiple times. ✓ ⇒ Use DP ✓

↳ Store subproblem to avoide recalculation. ✓✓

TC $= O(N)$ → every value is calculated only once.

SC $= O(N)$ → recursion + $F[]$ ✓

✓ **Top Down DP / Recursive DP** → start from big problem → smaller problem for which we know the answer.

(Recursion + Memoization)

✓ **Bottom Up DP / Iterative DP** → start with smallest problem for which we know the answer → big problem.

Sometimes can be solved in SC $= O(1)$ ✓

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

TC $= O(N)$
SC $= O(N)$ ✓

⇒

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

TC $= O(N)$ ✓
SC $= O(1)$

Q→ Find min no. of perfect sq. to add s.t sum = N.

N = 6 → $1^2+1^2+1^2+1^2+1^2+1^2$
→ $2^2+1^2+1^2$ ✓ Ans = 3

N = 9 → $1^2+1^2+\dots 1^2$ (9 times)
→ $2^2+1^2+1^2+1^2+1^2+1^2$
→ $2^2+2^2+1^2$
→ 3^2 ✓ Ans = 1

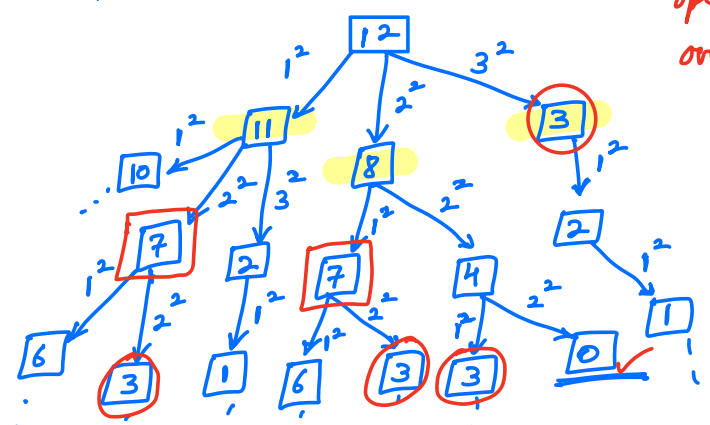
N = 12 $-3^2 = 3-1^2 = 2-1^2 = 1-1^2 = 0$
$12 = 2^2+2^2+2^2$ ⇒ Ans = 3

Greedy → [N − (greatest perfect sq. ≤ N)
↳ do this till the no. = 0

Eg → $N=6-2^2=2-1^2=1-1^2=0$ ✓

$N=10-3^2=1-1^2=0$ ✓

optimal substructure ✓
overlapping subproblems ✓
⇓
DP



$dp[12] = 1 + \min(dp[11], dp[8], dp[3])$ → x = 1, 2, 3 $x^2 <= 12$

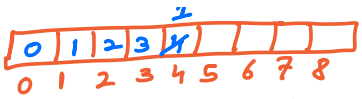
$dp[n] = 1 + \min(dp[n-x^2])$ ∀ x s.t $x^2 <= n$

$dp[0] = 0$
↓
No perfect sq. is req to make sum = 0.

```
dp[0] = 0;   // dp[n+1];
for (i = 1; i <= n; i++) {
    dp[i] = i;   // adding 1^2, i times
    for (x = 1; x*x <= i; x++) {
        dp[i] = min(dp[i], 1 + dp[i - x^2]);
    }
}
return dp[n];
```

TC = O(N√N) ✓ SC = O(N) ✓

N = 8 i = 4
x = x2



$dp[4] = \min(dp[4], 1 + dp[4 - 2^2]) = 1$

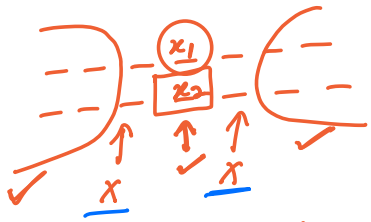
$\min(4, 1 + 0)$

Q → Given a 2*N matrix of numbers.
Find max sum by selecting same elements s.t elements are not adjacent horizontally, vertically or diagonally.

M → $\begin{bmatrix} 2 & -3 & (8) & 5 & 1 & 6 \\ (4) & 1 & 5 & 2 & 3 & (8) \end{bmatrix}_{2*N}$

Ans = 20 ✓

observation ⇒ (greedy)



∀ column only one element is selected & it selection will not effect other selections. ✓

A[i] = max(M[0][i], M[1][i]) ∀i ✓

A → [4 1 8 5 3 8]

Ans = 20

i → (i-1) ✗
(i+1) ✗

∀ elements → take it / leave it

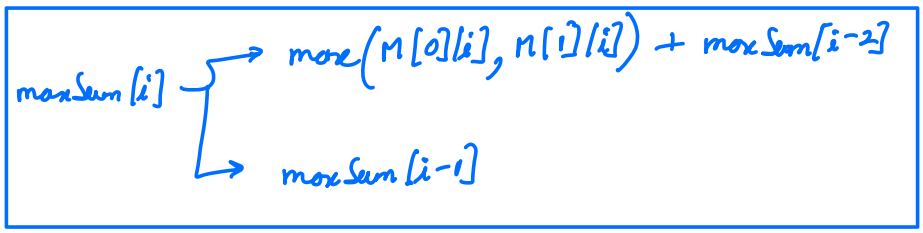
maxSum[i] → max sum by selecting first i elements.

maxSum[i] → A[i] + maxSum[i-2] ✓
maxSum[i-1] ✗

	1	2	3	4	5	6
A →	4	1	8	5	3	8
maxSum →	0	4	4	12	12	15	20 → Ans

TC = O(N)
SC = O(N) → O(1)

only previous 2 values are req to get current answer ✓


maxSum[i]
max(M[0][i], M[1][i]) + maxSum[i-2]
maxSum[i-1]