Dynamic Programming

Best part → You already have seen an example

Prefix sum →

$$pf(i) = pf(i-1) + arr(i)$$

where $pf(i)$ ⇒ Sum of $[0:i]$

Q1 Fibonacci Series

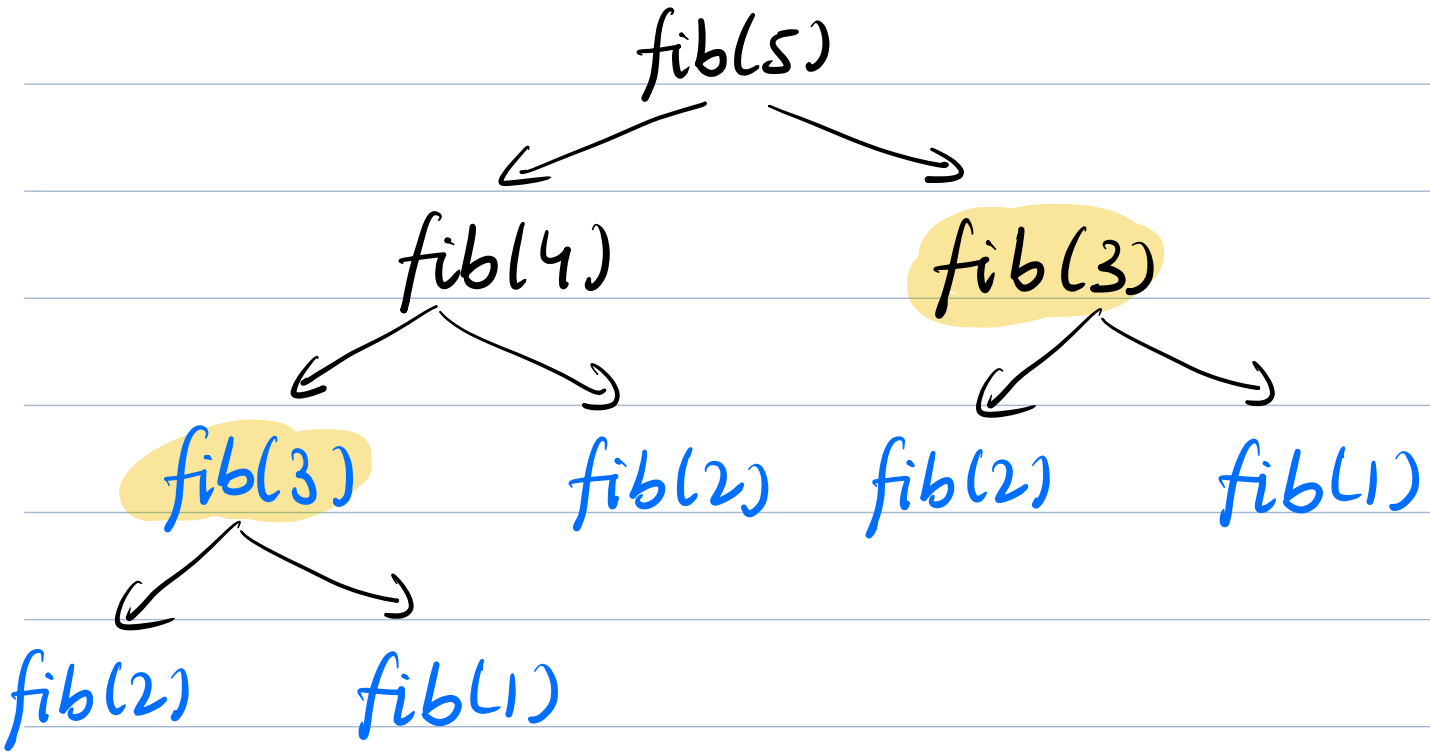
1 1 2 3 5 8 13 21 34 ....

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

$$fib(1) = 1 \qquad fib(2) = 1$$

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

TC: $O(2^n)$



Same problem solved multiple times.

DP ⇒ Smart Recursion

- Optimal Substructure : Solve problem using smaller subproblem
- Overlapping subproblems: Solve the same subproblem multiple times.

DP soln

Top-down (recursive)

Break larger prob into smaller

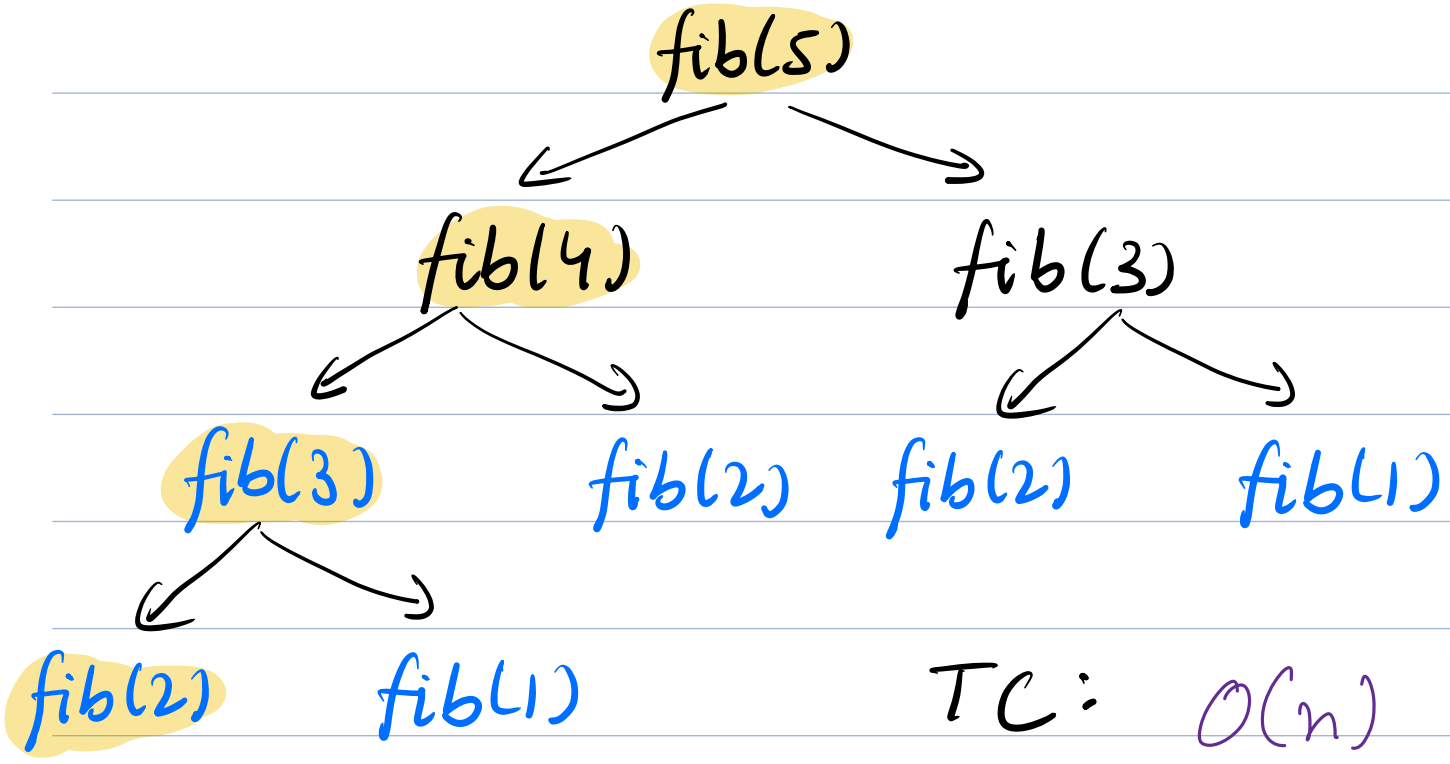
Bottom up (iterative)

Small बनाते जाओ

Large बनाते जाओ

Top down

```
int dp[n+1]   // initialize everything = -1
int fib(int n) {
  if (n <= 2)
      return 1
  if (dp[n] != -1)
      return dp[n]      } if already calculated, return from memory.
  ans = fib(n-1) + fib(n-2)
  dp[n] = ans           } memoization
  return ans
}
```



TC: $O(n)$

SC: $O(n)$

Bottom up (Iterative)

```
int dp(n+1)
dp[1] = 1      dp[2] = 1
for(i=3; i<=n; i++) {
    dp[i] = dp[i-1] + dp[i-2]
}
```

TC: } $O(n)$

SC: }

1 2 3 4 5

1 1 2 3 5

Q2 Find minimum no of perfect squares to get the sum = N

Eg 6 → $1^2+1^2+1^2+1^2+1^2+1^2 = 6$

→ $2^2+1^2+1^2 = 4+1+1 = 6$

ans = 3

10 3^2+1^2 ans = 2

9 3^2 ans = 1

Idea Greedy ?

N − last perfect square

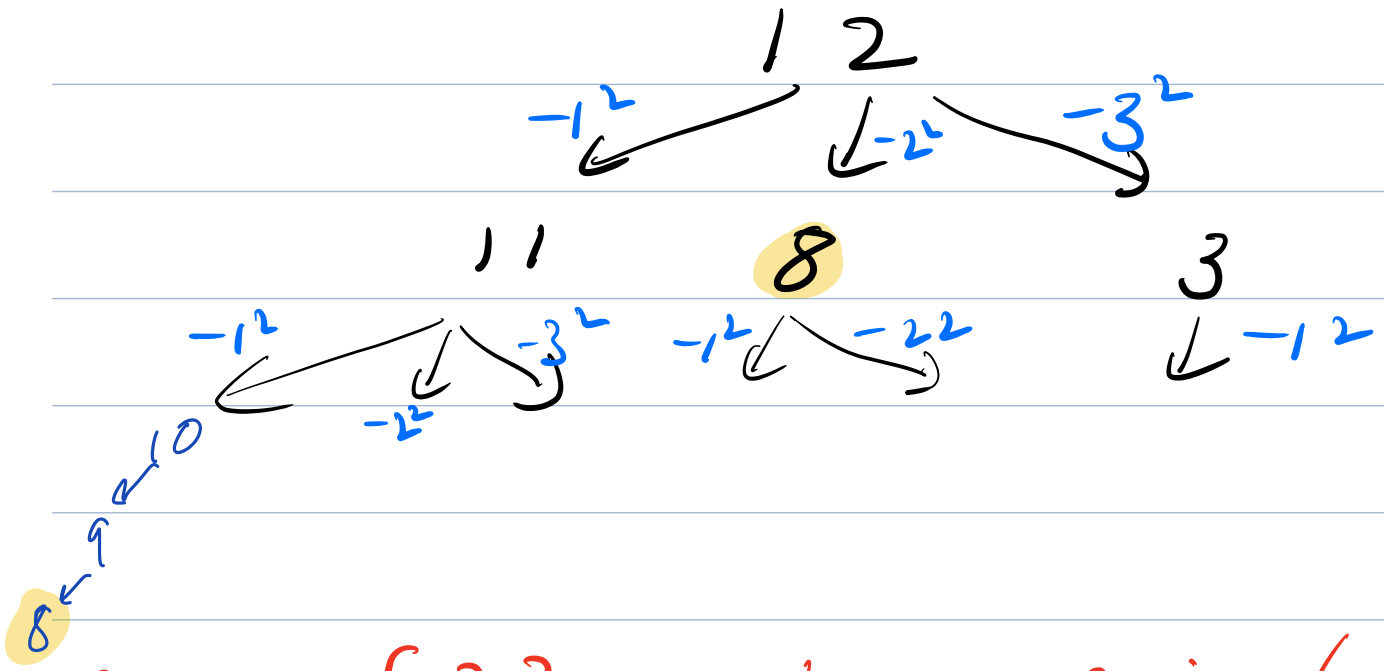
12 → $12-3^2 = 3$

$3^2+1^2+1^2+1^2$ $3-1^2 = 2$

$2-1^2 = 1$

$2^2+2^2+2^2$ $1-1^2 = 1$

but better is $\underbrace{2^2 + 2^2 + 2^2}_{ans = 3}$



squares [12] = 1 + min (
squares [12 − 1^2],
squares [12 − 2^2],
squares [12 − 3^2])

squares [i] = 1 + min (
squares [i − 1^2],
squares [i − 2^2],
—
squares [i − x^2])

$x^2 \leq i$

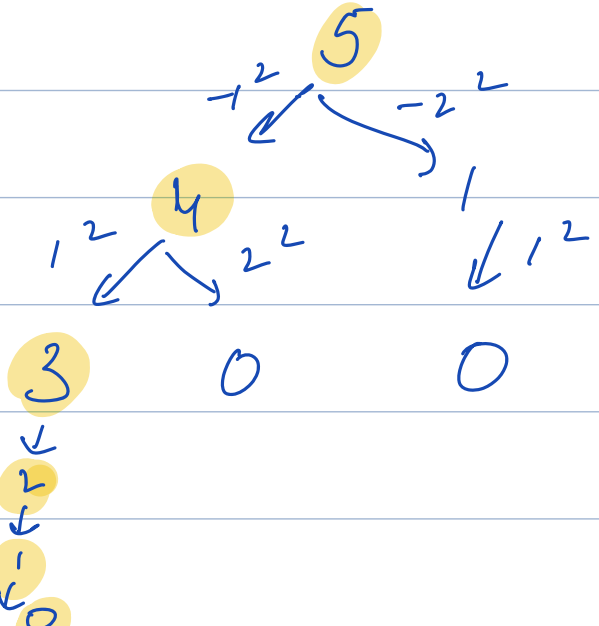
Code

```
int dp[n+1]  // initialize = -1
int calc(int n){
  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, calc(n-x^2))
  }
  dp[n] = 1+ans
  return dp[n]
}
```

[done]

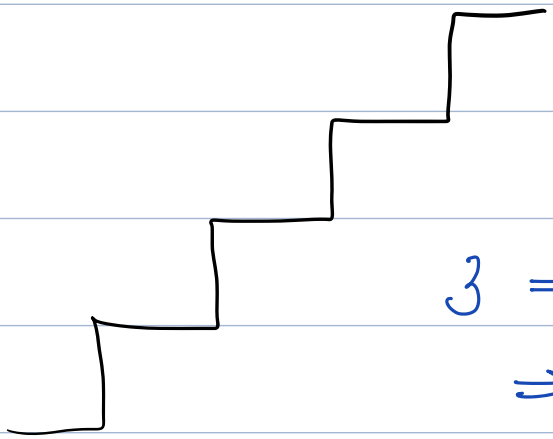
TC: O(N*sqrt(N))

SC: O(n)



Q3 No of ways to reach N^{th} stair

Can take one step or 2 steps



step size = 1 or 2

4 ⇒ 5 ways

3 ⇒ 1 + 1 + 1

⇒ 1 + 2

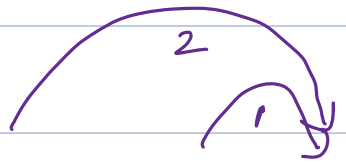
⇒ 2 + 1

ans = 3

2 ⇒ 1+1 or 2

ans = 2

1 ⇒ 1



1 2 3 4 ..... n-3 n-2 n-1 n

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

1 ⇒ 1 ①

2 ⇒ 2 1+1 2

Top down

```
int dp[n+1]    // initialize everything =-1
int ways(int n){
  if(n<=2)
     return n
  if(dp[n]!=-1)
     return dp[n]
  ans = ways(n-1) + ways(n-2)
  dp[n] = ans
  return ans
}
```

} if already calculated, return from memory.

} memoization

TC: } O(n)
SC: }

TC of DP ⇒

no of states × TC of 1 state