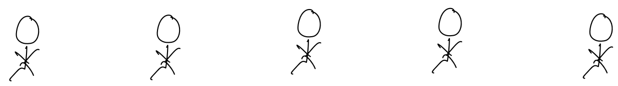
Lecture :- Dynamic programming - 1

Agenda

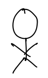
- fibonacci
- Stairs problem
- Min count of perfect squares.

Introduction

Students: 

Marks 5 3 4 7 9.

Total marks ⇒ 28

New student  Goutam

5

Total marks

- $5 + 3 + 4 + 7 + 9 + 5 = 33$
- $\underbrace{\text{prev-total-marks}}_{28} + 5 = 33$

Dynamic programming :- Use previous results to calculate current results.

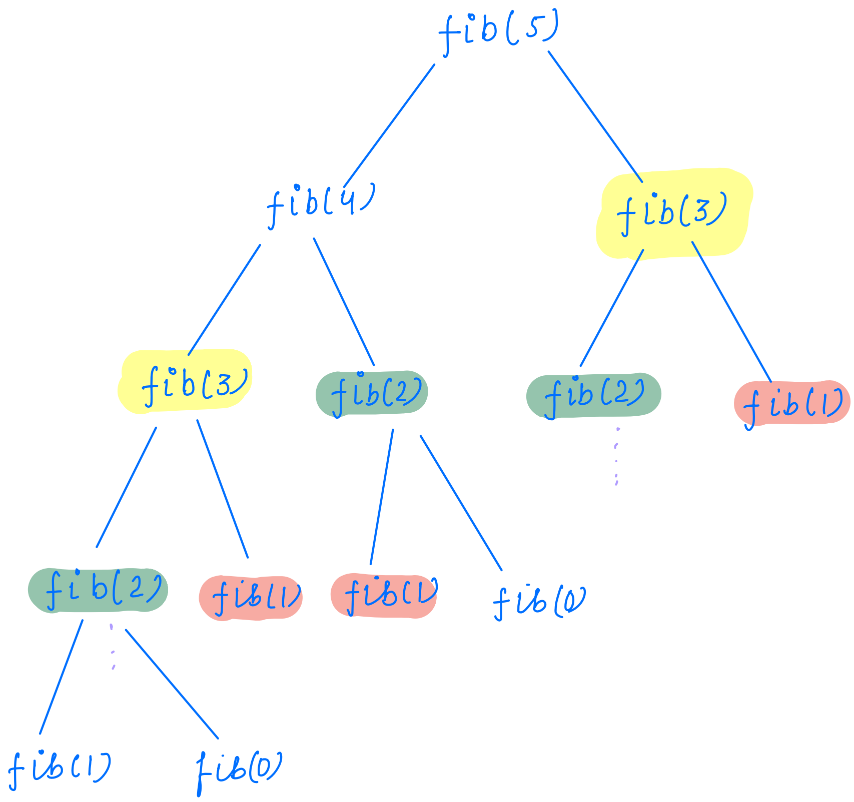
Example prefix sum array.

$pf[i] = pf[i-1] + arr[i]$

Fibonacci Series

0	1	2	3	4	5	6	7	8	....			
0	1	1	2	3	5	8	13	21	34	55	89	----

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




```
int fib(int n) {
   if(n==0 || n==1){
      return n;
   }
   fir = fib(n-1);
   sec = fib(n-2);
   return fir + sec;
}
```

T.C $\Rightarrow O(2^n)$

S.C $\Rightarrow O(n)$

Bad time complexity ?

$n = 10 \quad 2^{10} = 1024$ units

$n = 20 \quad 2^{20} = 10^6$ unit

$n = 50 \quad > 10^9$ TLE.

Conditions of DP

- Overlapping subproblems
- Optimal substructure | Later |

Dynamic programming :

Calculate unique results only once

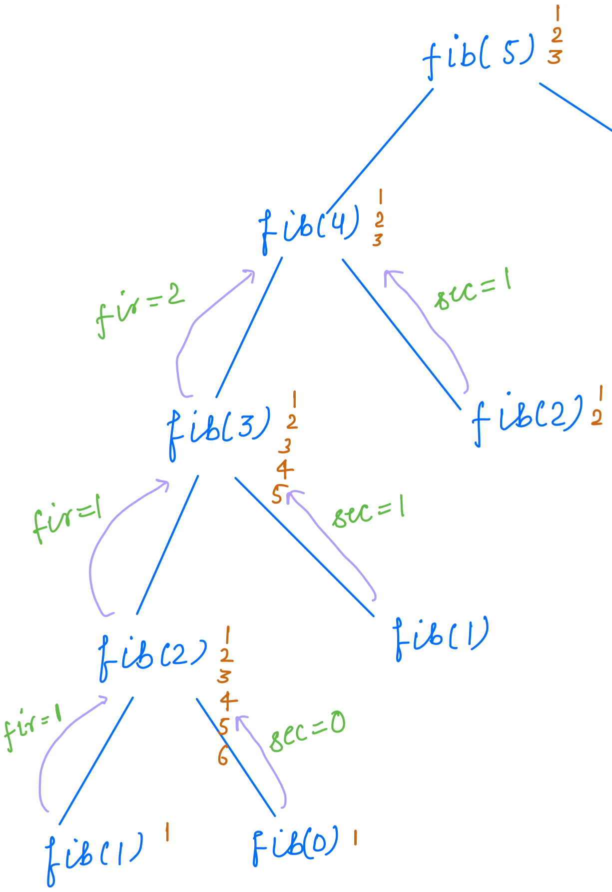
Dynamic programming code for fibonacci [Memoization]

```
int fib ( int n , int[] dp) {

    if(n==0 || n==1){
        dp[n] =n;
        return n;
    }
    if ( dp[n] ! = -1) {
        return dp[n];
    }
    fir = fib(n-1);

    sec = fib(n-2);
    dp[n] = fir + sec;
    return fir + sec;
}
```

Dry run

fib(5)

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




```
int fib(int n, int[] dp) {
  1 if(n==0 || n==1){
        dp[n]=n;
        return n;
    }
  2 if( dp[n] != -1) {
        return dp[n];
    }
  3 fir = fib(n-1, dp);
  4 sec = fib(n-2, dp);
  5 dp[n] = fir + sec;
  6 return fir + sec;
}
```

TC: O(n)

SC: O(n) + O(n)

↑ stack space

Tabulative approach of DP fibonacci

fib(5)

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

dp[2] = dp[0] + dp[1]

dp[3] = dp[1] + dp[2]

dp[4] = dp[2] + dp[3]

⋮

dp[i] = dp[i-1] + dp[i-2]

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

TC: O(n)

SC: O(n)

Qu Solve fibonacci using O(1) space.

0	1	2	3	4	5	6	7	8	....			
0	1	1	2	3	5	8	13	21	34	55	89	.....
a	b											

$c = fib(2) = a + b = 1$

```
int fib(int n) {
    if(n==0 || n==1) {
        return n;
    }
    a = 0
    b = 1;
    for(i=2; i<=n; i++) {
        c = a+b;
        a = b;
        b = c;
    }
    return c;
}
```

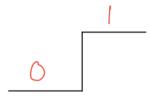
TC: $O(n)$

SC: $O(1)$

Qu Given n stairs, how many ways can we go from 0th step to nth step.

Note: You can take only 1/2 steps.

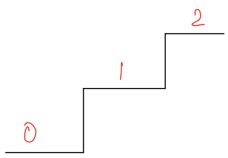
n=1



1

1 way

n=2

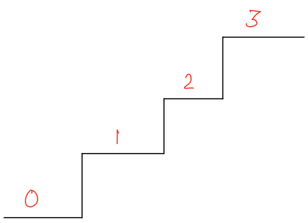


1 1

2

2 ways

n=3



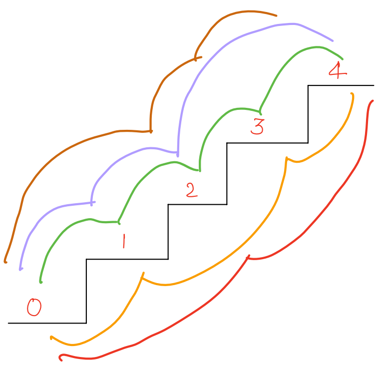
1 1 1

1 2

2 1

3 ways

n=4



1 1 1 1

1 1 2

2 1 1

1 2 1

2 2

5 ways

Idea foundation

$n = 4 \longrightarrow \overset{3}{ways(n-1)} + \overset{2}{ways(n-2)}$

$n = 3$ 1 1 1 1

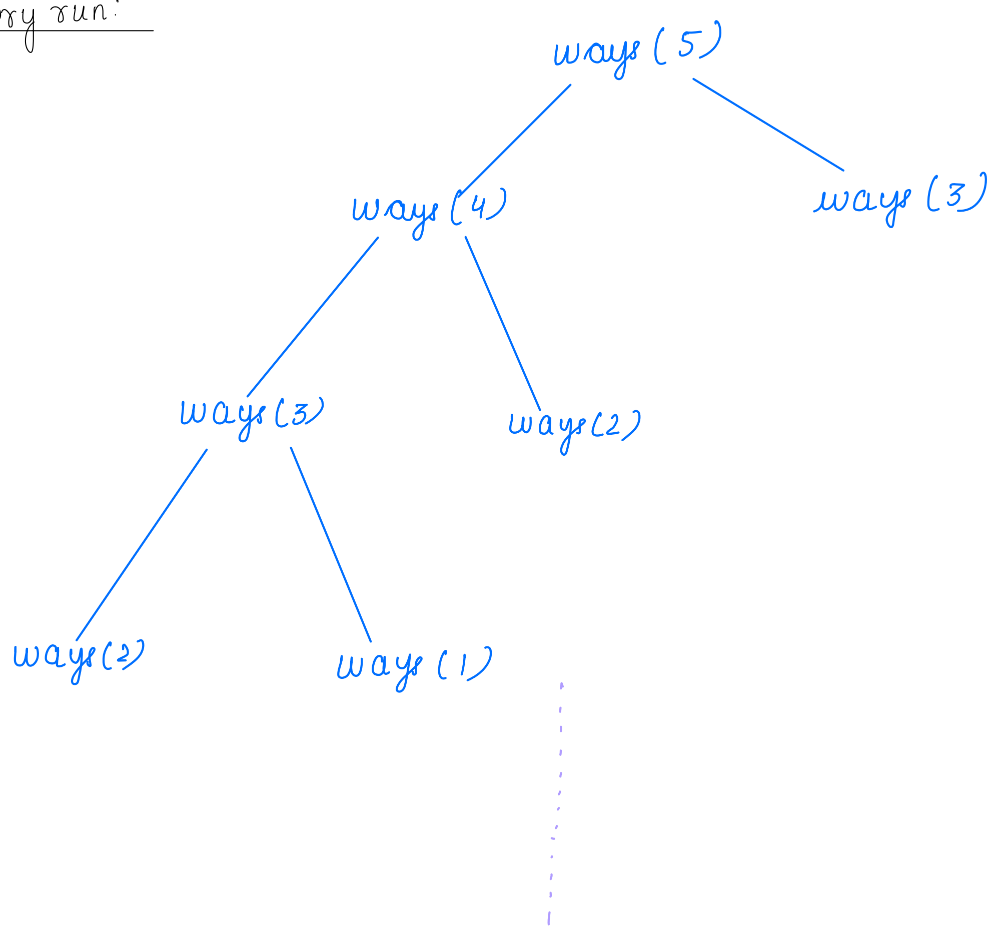
3 ways   1 2 1

2 1 1

$n = 2$ 1 1 2

2 ways   2 2

Dry run:



Same as fibonacci

Recursive code

```
int ways(n) {
    if(n==0 || n==1) {
        return n;
    }
    int fir = ways(n-1);
    int sec = ways(n-2);
    return fir + sec;
}
```

Break: 8:09 – 8:19

Qu find min count of perfect squares to add to get sum = n

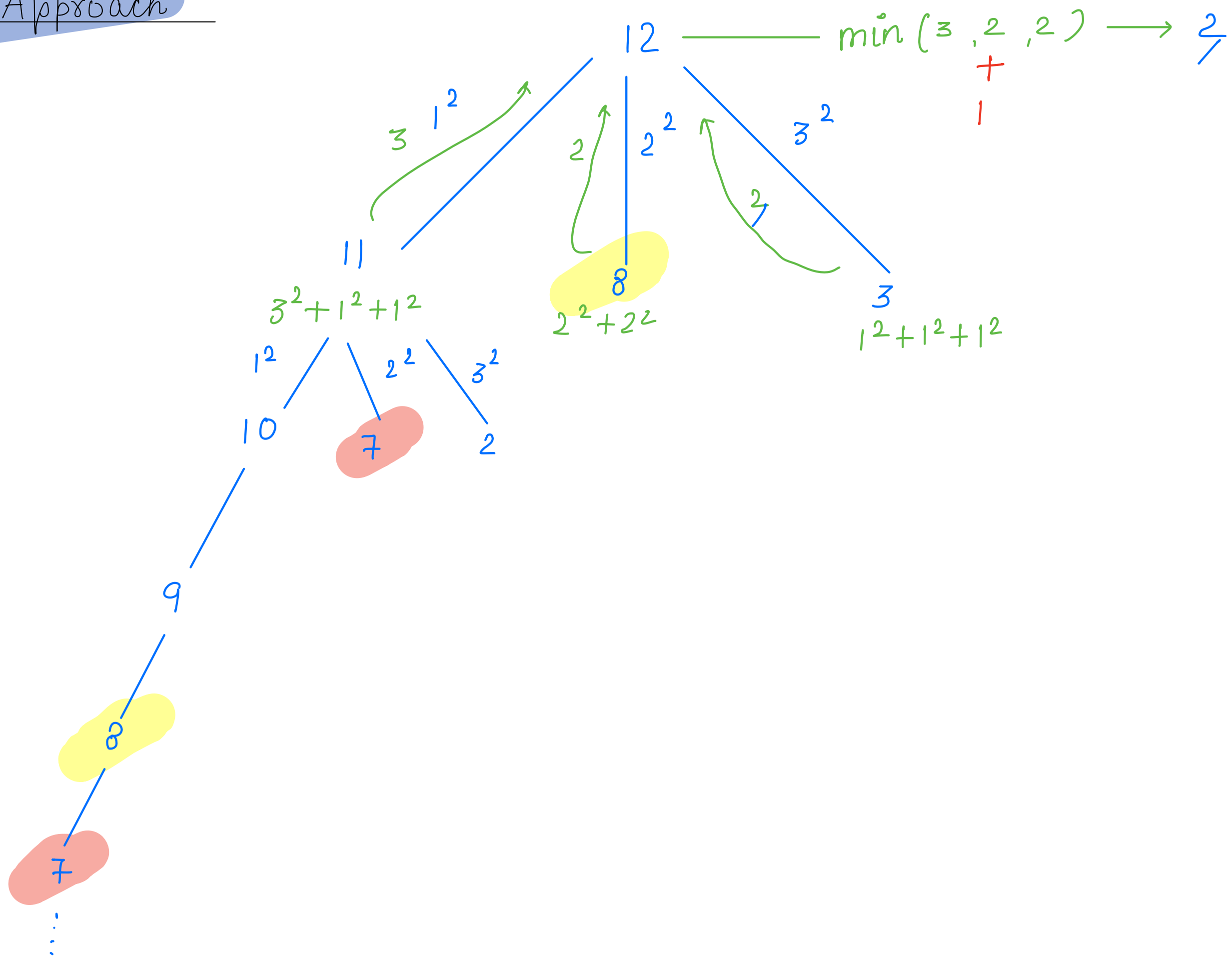
n		count
2	$1^2 + 1^2$	2
3	$1^2 + 1^2 + 1^2$	3
4	$1^2 + 1^2 + 1^2 + 1^2$ 2^2	1
5	$2^2 + 1^2$	2
6	$2^2 + 1^2 + 1^2$	3
7	$2^2 + 1^2 + 1^2 + 1^2$	4
50	$5^2 + 5^2$ $7^2 + 1^2$	2

Greedy idea Subtract greatest perfect square $<= n$ from n

$n = 50$	$50 - 7^2 = 1 - 1^2 = 0$
$n = 70$	$70 - 8^2 = 6 - 2^2 = 2 - 1^2 = 1 - 1^2 = 0$
$n = 12$	$12 - 3^2 = 3 - 1^2 = 2 - 1^2 = 1 - 1^2 = 0$ $2^2 + 2^2 + 2^2 \Rightarrow 3$

Greedy idea won't work

Approach



Brute force code

```
int count(int n) {
    if(n==0 || n==1) {
        return n;
    }
    if(n<0) {
        return 0;
    }
    ans = ∞;
    for(i=1; i*i<=n; i++) {
        ans = min(ans, count(n-i*i));
    }
    return ans +1;
}
```

TC:

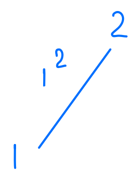
SC: $O(n)$

Dynamic programming

$n \Rightarrow 12$

0	1	2	3	4	5	6	7	8	9	10	11	12
$\cancel{\infty}$ 0	$\cancel{\infty}$ 1	$\cancel{\infty}$ 2	$\cancel{\infty}$ 3	$\cancel{\infty}$ 1	∞	∞	∞	∞	∞	∞	∞	∞

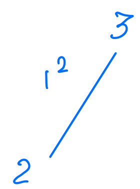
dp[2]



min (dp[2], dp[1]) + 1;

1 + 1 = 2

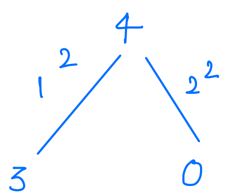
dp[3]



dp[2] = 2

ans = 2 + 1 = 3

dp[4]



dp[3] = 3

dp[0] = 0

ans = min(3, 0) + 1 ⇒ 1

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

TC: $O(n\sqrt{n})$

SC: $O(n)$

Thankyou 🙂