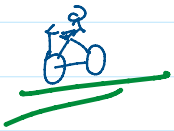
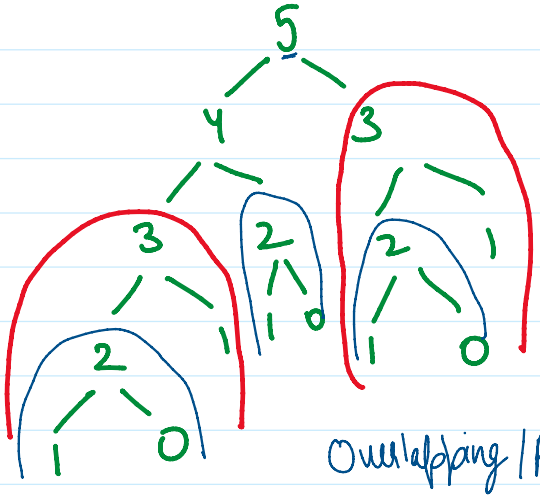
DP

09 April 2023 09:11

↳ Recursion



Fibonacci



2

2^n

$1 + 2 + 2^2 + 2^3 + \ldots = 2^n$

Overlapping / Repeating Subproblems

Input args kitne → 1 (n)

↳ Range : 0 to n

total recursive states ? → 2^n

Unique subproblems ? → $n+1$

Agar kisi D.S mein kisi subproblem ka ans store karvalu and usse use karlu toh baat bnjayegi?

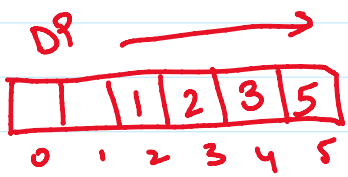
↳ Array Hashmap

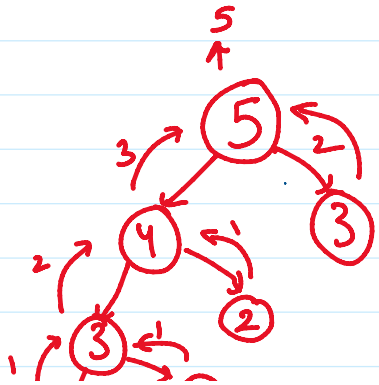
K : V.

↳ 0 to n

$n+1$

O(1)



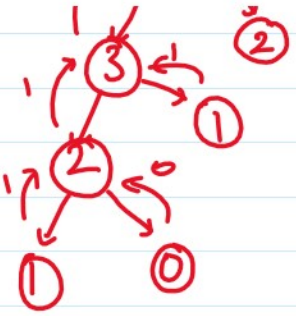



```
public static int FiboTD(int n, int[] dp) {
        if (n <= 1) {
            return n;
        }
```

Stack Memory ??

↳

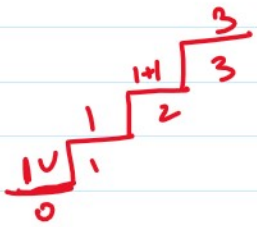
```
    if (n <= 1) {
        return n;
    }
    if(dp[n]!=0) {
        return dp[n];
    }
    int sp1 = FiboTD(n - 1, dp);
    int sp2 = FiboTD(n - 2, dp);
    dp[n]=sp1+sp2;
    return sp1 + sp2;
}
```



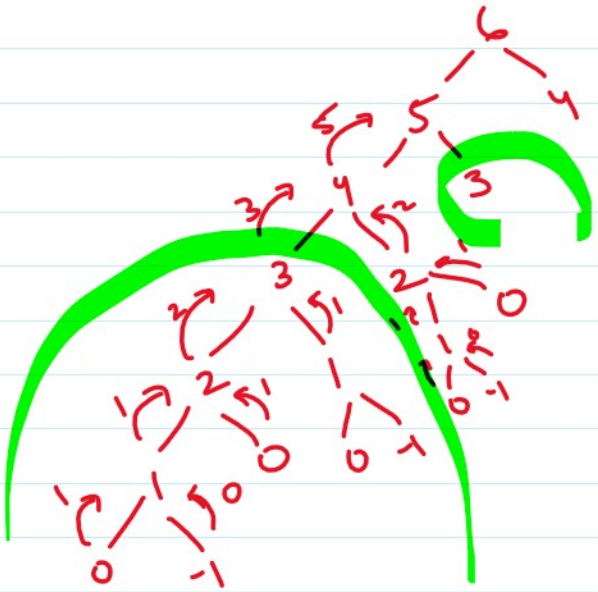
Stack memory :

10000

Climbing Stairs



1 1 1
2 1
1 2

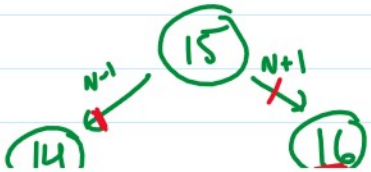


2^n

n+1

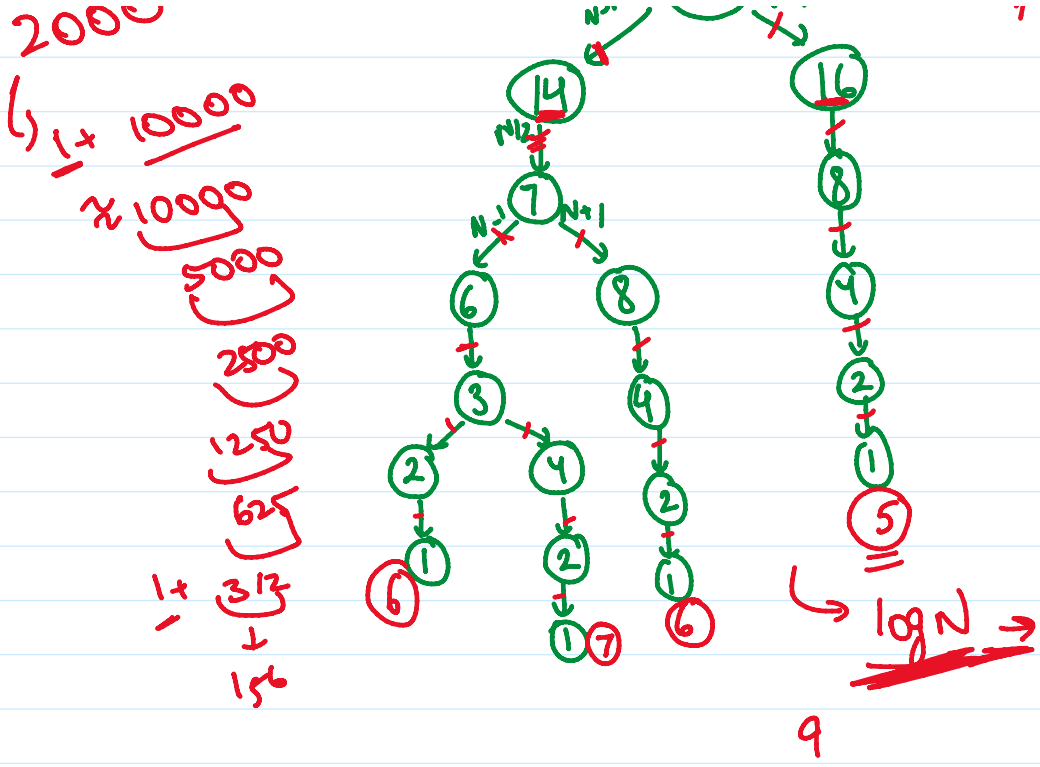
Go to n

20001



Odd
N-1

$2^x \rightarrow N2$



$2 \approx x \rightarrow N_2$

Odd → 2

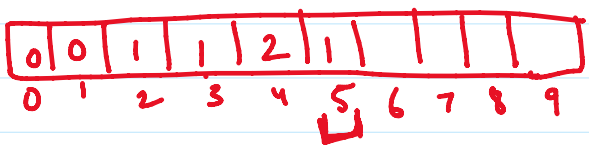
Even → 1

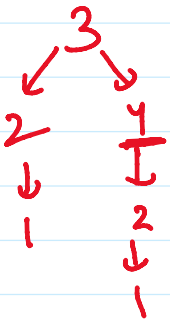
$n/2$

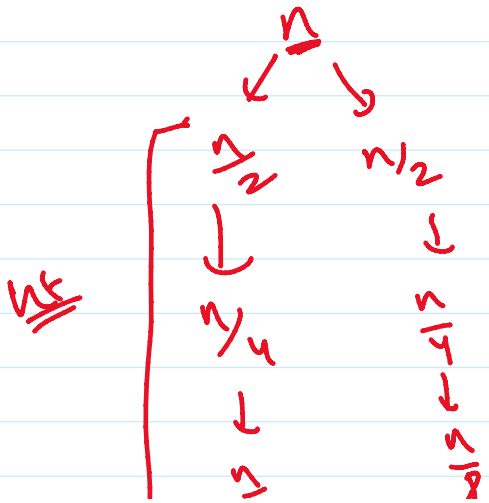
$1 + n/2 \approx n/2$

9

```
public static int opsBU(int n) {
        int[] dp = new int[n+1];
        for(int i=2;i<=n;i++) {
            if(i%2==0) {
                dp[i]=1+dp[i/2];
            }else {
                int sp1 = 1+dp[i-1]; 3
                int sp2 = 1+dp[i+1]; 1
                int res= Math.min(sp1, sp2);
                dp[i]=res;
            }
        }
        return dp[n];
    }
}
```





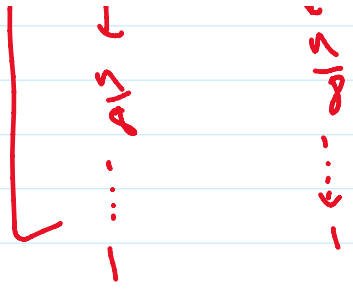


$$2\left(\frac{n}{2} + \frac{n}{4} + \frac{n}{8} + \dots n\right)$$

$0 < a < 1$

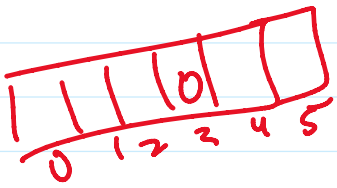
$$2\left(\frac{a}{1-a}\right)$$

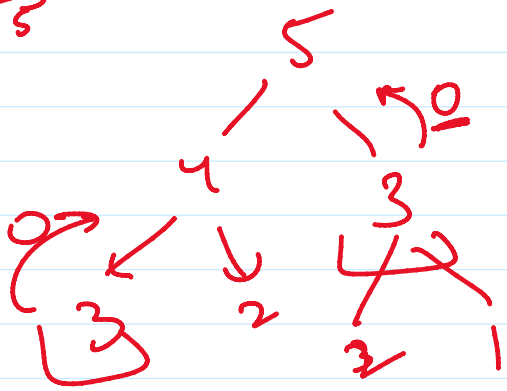
$$2\left(\frac{a}{1-\frac{1}{2}}\right)$$



$2\left(\frac{a}{1-\frac{1}{2}}\right)$

$= \boxed{\frac{\log n}{2}}$





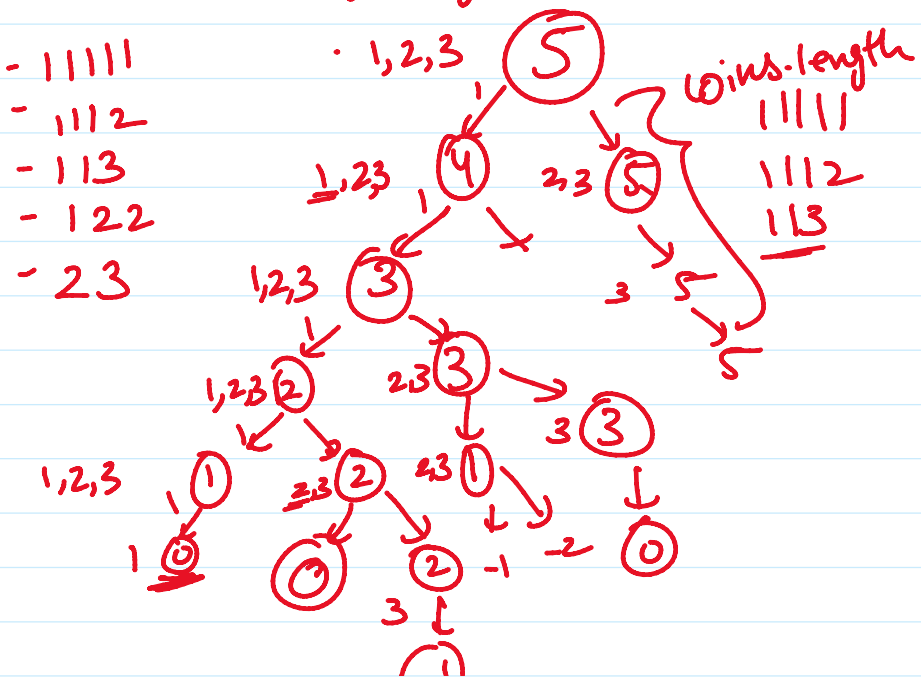
Coin change Combinations

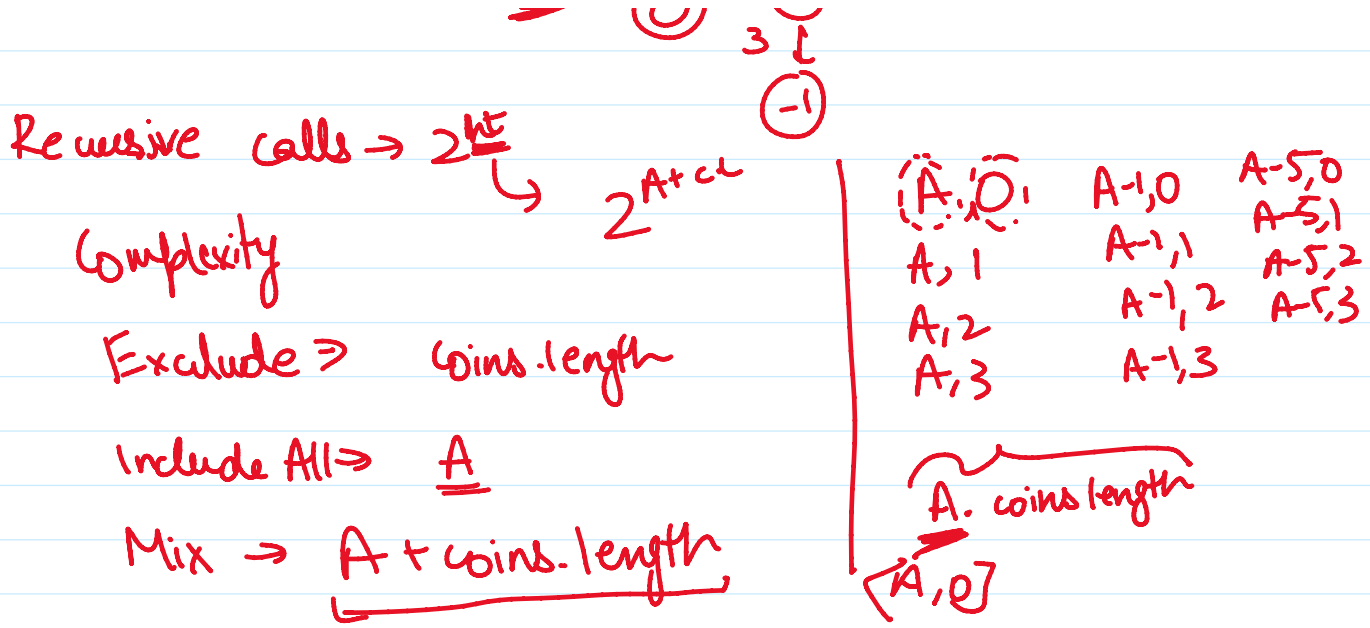
Amount = 5

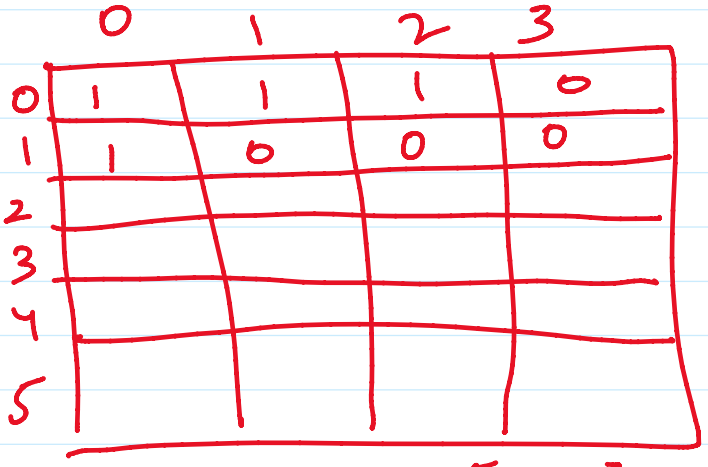
coins = {1, 2, 3}

How many ways to change??

- 11111
- 1112
- 113
- 122
- 23








```
private static int changeTD(int[] coins, int A, int idx, Integer[][] dp) {
        if (A == 0) {
            return 1;
        }
        if (A < 0 || idx == coins.length) {
            return 0;
        }
        if (dp[A][idx] != null) {
            return dp[A][idx];
        }
        // include
        int sp1 = changeTD(coins, A - coins[idx], idx, dp);

        // exclude
        int sp2 = changeTD(coins, A, idx + 1, dp);
        dp[A][idx] = sp1 + sp2;
        return sp1 + sp2;
    }
```