Todays Content

a) Introduction to Dp

b) Fibanacci Series

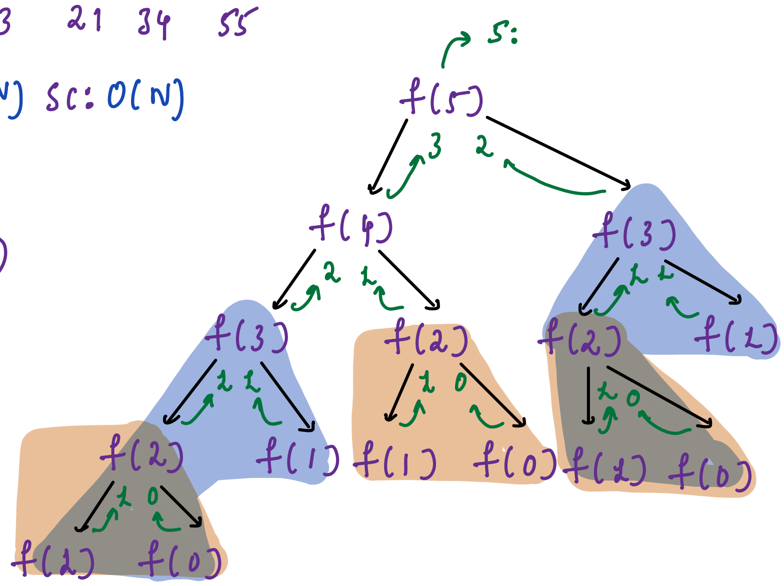
c) N Stairs

d) Min No: of Squares

Q: Find N^{th} fib number?

	0	1	2	3	4	5	6	7	8	9	10
Fib :	0	1	1	2	3	5	8	13	21	34	55

```
int fib(int N){        TC = O(2^N)  SC: O(N)
    if(N<=1){return N}
    return fib(N-1) + fib(N-2)
}
```



Obs:

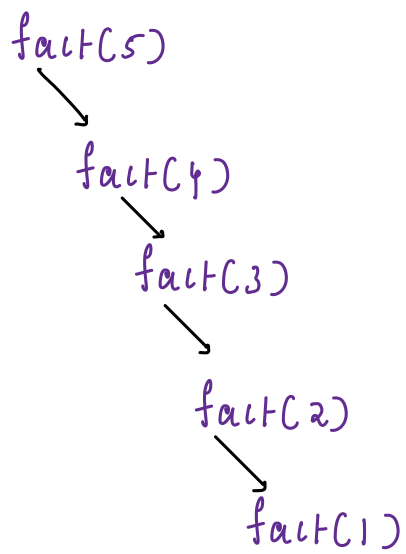
1. While solving Problems with the help of subproblems
2. If same subProblems is solved again & again, apply Dp?

Dynamic Programming:

Solve a SubProblem for 1^{st} time, store it & re-use it, if it's called again.

fact(5): $5 * 4 * 3 * 2 * 1 = 120$

```
int fact(n){
    if(n==1){return 1}
}   return n* fact(N-1)
```



Optimize fibanacci with Dp

// invalid values

Full arr[]: fib can never be negative, ==-1; solving i^{st} time.

int dp[N+1] = -1;   dp[i] = i^{th} fibanacci number → dp[N] = N^{th} fib number

ans = dp[N] : N^{th} fibanacci number

```
int fib(N){
    if(N<=1){return N}
    if(dp[N]==-1){ // solving 1st time.
        dp[N] = fib(N-1) + fib(N-2)
    }
    return dp[N]
}
```

Note: solve base condition : O(1)
store & re-use it : O(1)
} won't matter

TC: O(N) SC: O(N)

TC of Dp codes: # No: of : distinct Sub Problems * # TC for each Sub Problems

Trace:

fib(5) : 5 = 3 + 2

dp[5] = fib(4) + fib(3)   (3 + 2)

dp[4] = fib(3) + fib(2)   (2 + 1)

dp[3] = fib(2) + fib(1)   (1 + 1)

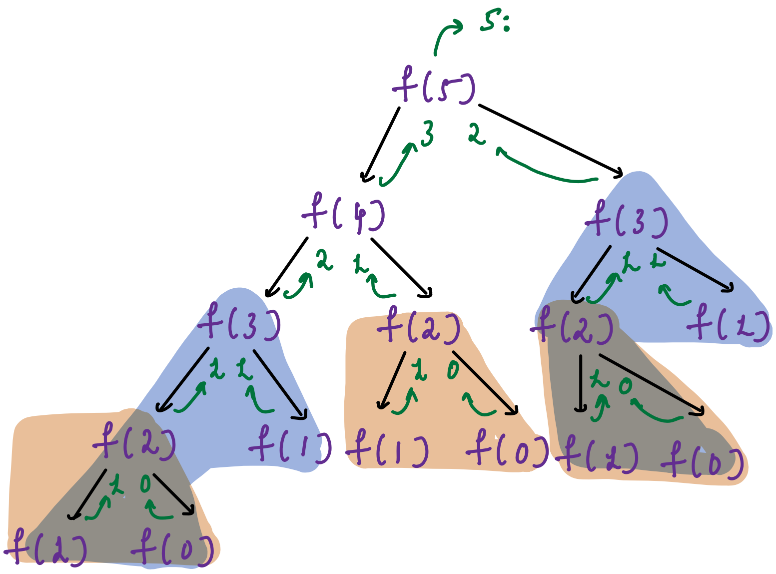
dp[2] = fib(1) + fib(0)   (1 + 0)

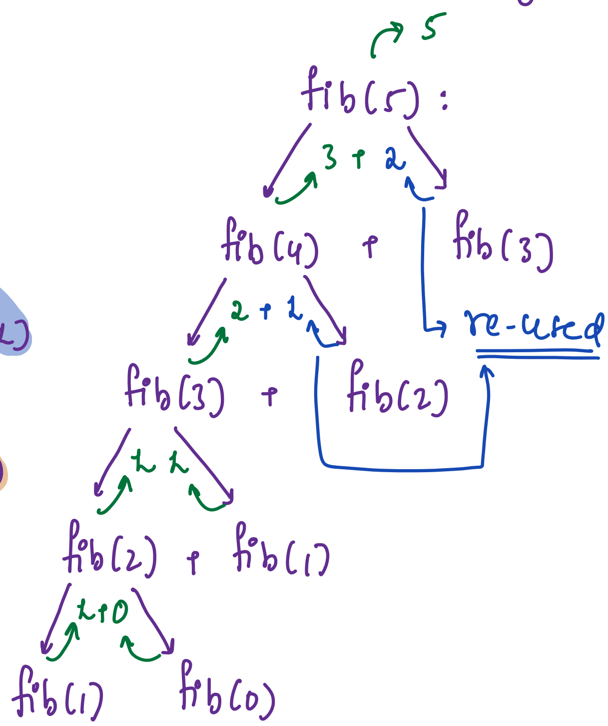
dp[6] =

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

10:15pm break

Comparison: Recursion vs Dynamic Programming





Steps for Dp:

1. Solve it with Recursion
2. Overlapping SubProblems : Sub Problems repeating.

Apply Dp:

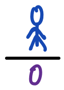
1. Create dp table & initialize
2. If Solving subproblems 1st time
    a. Solve subproblems & store in dp table
    b. Return Subproblem

   If solving subproblem 2nd time
    a. Re-use it, return subproblem from dp table
3. Calculate TC & SC

Q) Given N, find total no: of ways to go from $0^{th} \to N^{th}$ step

Note: From i^{th} step we can go to $i+1^{th}$ step or $i+2^{nd}$ step

N=0



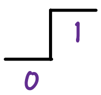
ans: 1 way

N=0: ways to reach 0^{th} step from 0^{th} step:

if ans = 0: To reach 0^{th} step from 0^{th} step: 0 ways
It's not possible to reach 0^{th} step: *

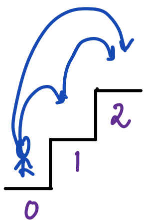
if ans = 1: To reach 0^{th} step from 0^{th} step: 1 way
person staying at 0^{th} step.

N=1



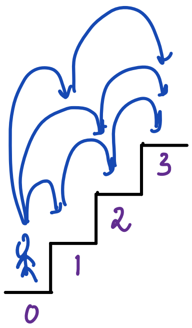
ans: $0 \to 1$

N=2



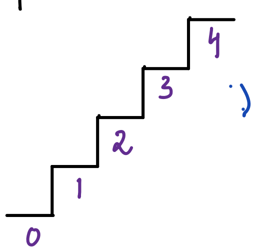
ans: $0 \to 1 \to 2$
$0 \to 2$

N=3



ans: $0 \to 1 \to 2 \to 3$
$0 \to 1 \longrightarrow 3$
$0 \longrightarrow 2 \to 3$

N=4



ans: 4^{th} step = can reach it via 3^{rd} step & 2^{nd} step: 5 paths

Reaching 4^{th} step via 3^{rd} step from 0^{th} step

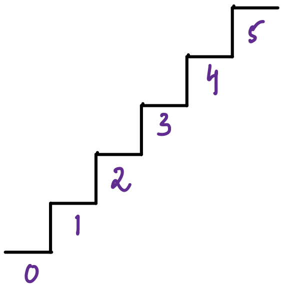
$0 \to 1 \to 2 \to 3 \to 4$
$0 \to 1 \longrightarrow 3 \to 4$
$0 \longrightarrow 2 \to 3 \to 4$

Reaching 4^{th} step via 2^{nd} step from 0^{th} step

$0 \to 1 \to 2 \to$ $0 \to 1 \to 2 \to 3 \to 4$
$0 \to 1 \to 2 \longrightarrow 4$

$0 \longrightarrow 2 \to$ $0 \to 2 \to 3 \to 4$
$0 \to 2 \longrightarrow 4$

N=5



5^{th} Step: can reach it via 4^{th} Step or 3^{rd} step: 8 ways

Reaching 5^{th} Step via 4^{th} Step or 3^{rd} Step from 0^{th} Step

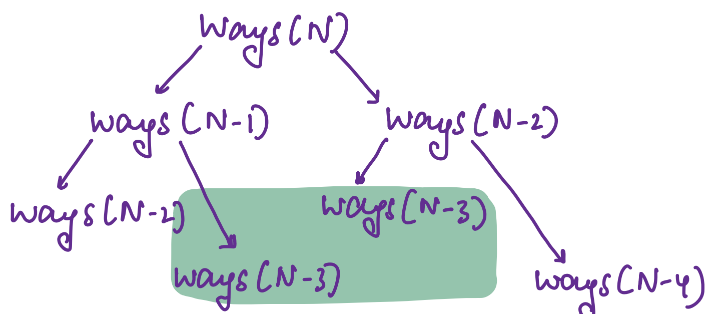
0 → 1 → 2 → 3 → 4 → 5	0 → 1 → 2 → 3 → 5
0 → 1 → 3 → 4 → 5	0 → 1 → 3 → 5
0 → 2 → 3 → 4 → 5	0 → 2 → 3 → 5
0 → 1 → 2 → 4 → 5	
0 → 2 → 4 → 5	

Idea: Calculate using recursion?

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

Recursion:

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



overlapping sub

Dp:

int dp[N+1] = -1 or 0   dp[i] = ways to reach i^{th} Step from 0^{th} Step:

ans = dp[N], ways to reach N^{th} Step

```
int ways(int N){
    if(N<=1){return 1}
    if( dp[N] == 0){ // solve 1st time.
        dp[N] = ways(N-1)+ways(N-2);
    }
    return dp[N]
}
```

TC: O(N) * (1) = O(N)   SC: O(N)

TC of Dp code: # No: of States * # TC for each State

38) Find minimum number of perfect squares sum required to get N

$N = 6$: $2^2 + 1^2 + 1^2 = 6$: 3 sq
$1^2 + 1^2 + 1^2 + 1^2 + 1^2 + 1^2 = 6$: 6 sq
} mpq = 3

$N = 10$: $2^2 + 2^2 + 1^2 + 1^2 = 10$: 4 sq
$3^2 + 1^2 = 10$: 2 sq
} mpq = 2

$N = 9$: $3^2 = 9$ } mpq = 1

$N = 12$: $3^2 + 1^2 + 1^2 + 1^2 = 12$
$2^2 + 2^2 + 2^2 = 12$
} mpq = 3

obs: Greedy not working

$N = 0$:

Idea: Solving using Recursion : Sub Problems

mps(N) = min perfect square sums required to get N.

$11 + 1^2 = 12 \rightarrow$ 4 sq
$8 + 2^2 = 12 \rightarrow$ 3 sq
$3 + 3^2 = 12 \rightarrow$ 4 sq
} min = 3

mps(12)

$+1^2$ → mps(11) ans = 3
$+2^2$ → mps(8) ans = 2
$+3^2$ → mps(3) ans = 3
$+4^2$ → mps(-4) *

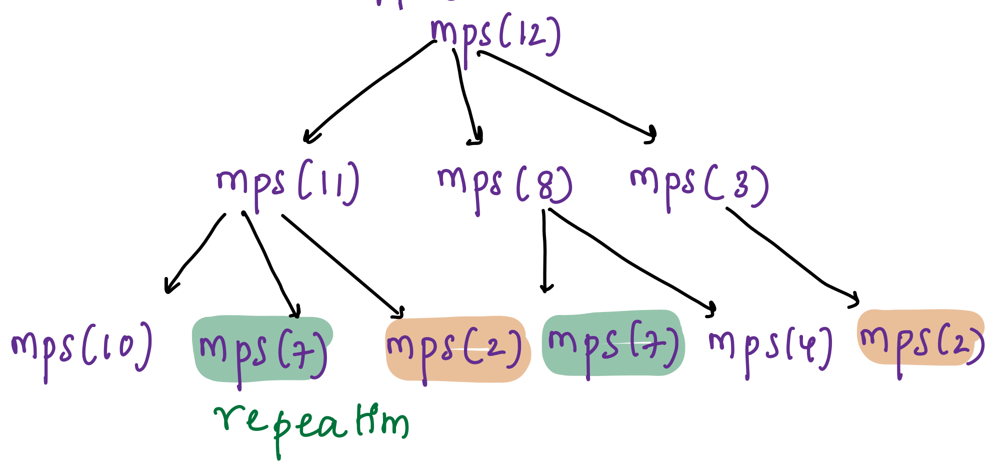
```
int mps(int N){
    if(N==0){return 0}
    int i=1;
    int ans = INT_MAX
    while(N >= i^2){
        ans = min(ans, mps(N-i^2))
        i=i+1
    }
    return ans+1;
}
```

mps(40) →

$mps(40-1^2) + 1^2$
$mps(40-2^2) + 2^2$
$mps(40-3^2) + 3^2$
$mps(40-4^2) + 4^2$
$mps(40-5^2) + 5^2$
$mps(40-6^2) + 6^2$
} min + 1 → square we are adding

$mps(40-7^2)$ = -ve

Idea2: Check for overlapping SubProblems.




```
int dp[N+1] = -1;
int minSquare(int N){
    if(N==0){return 0}
    if( dp[N]==-1){
        int i=1;
        int ans= INT_MAX
        while(i^2 <= N){  -> i:1 i<=√N
            ans = min(ans, mps(N-i^2))
            i=i+1
        }
        dp[N] = ans+1;
    }
    return dp[N];
}
```

dp[i] = min perfect square req to get sum = i

ans = dp[N]

TC: #subproblems * Time for each

: $O(N) * \sqrt{N} = O(N\sqrt{N})$

SC: $O(N)$

TC of Dp codes: # No: of States * # TC for each State