DP - 1D

Problem 1

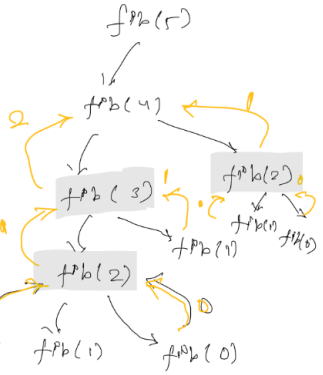
Problem 1 is

Fibonacci Series

0	1	2	3	4	5	6	7
0	1	1	2	3	5	8	13

$fib(N) = fib(N-1) + fib(N-2)$;

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



T.C : $O(2^N)$

S.C : $O(N)$ → for N calls.

→ There are repeated computations which would have been done once and stored. (Overlapping Subproblems)

→ We can store these values in fib[N+1] array.

Soln is

```
int dp[N+1] = {-1}

int fib(N) {
    if (N <= 1)
        return N;
    if (dp[N] != -1)
        return dp[N];
    dp[N] = fib[N-1] + fib(N-2);
    return dp[N];
}
```

fib[i] = ith fibonacci.

dp

0	1	2	3	4	5	6	7	8	9	10
-1	-1	-1	-1	-1	-1	-1	-1	-1	-1	-1

dp.

Recursive Code + Memorization = Top down DP.

5-4-3-2-1

T.C : O(N)

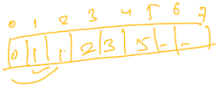
S.C : O(N)

Types of DP and Steps

Types of DP

1) Recursive DP

2) Iterative DP.




```
int dp[N+1];
dp[0] = 0;
dp[1] = 1;        } Base Condition.
for(int i = 0; i <= N; i++)
{   dp[i] = dp[i-1] + dp[i-2]
}
```

T.C : O(N)
S.C : O(N)

Bottom-up DP ⟶ 0 → 1 → 2 - - - N.

DP without extra Space :

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

return N
```

SC : O(1)
T.C : O(N)

Steps for DP :

1) Optimal Substructure :-
   → Are we able to solve the problem using subproblem?

2) Overlapping Subproblems :-
   → Subproblems should repeat.

Above two checks needed to use DP approach.

3) DP State (Assumption)

4) DP expression (Main Logic)

5) Base Case (Base Condition)

Problem 2

Problem 2

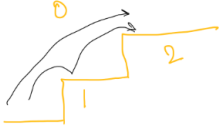
Given N stairs, how many ways we can go from 0 to N^{th} step.

Note :- from i^{th} step we can go directly to $(i+1)^{th}$ or $(i+2^{nd})$ step.

for Ex: N = 1

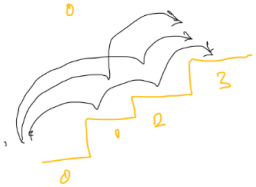
} 1 way

N = 2



} 2 ways.

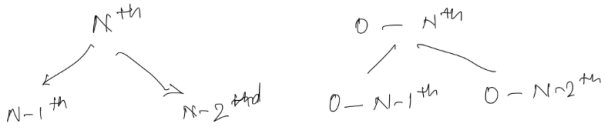
Ex N = 3



} 3 ways

N = 4

} 5 ways.

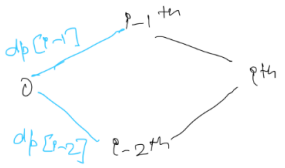


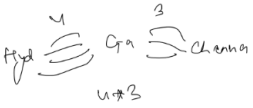
dp State

dp [i] = No of ways to reach i^{th} step from 0th step.

dp expression

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



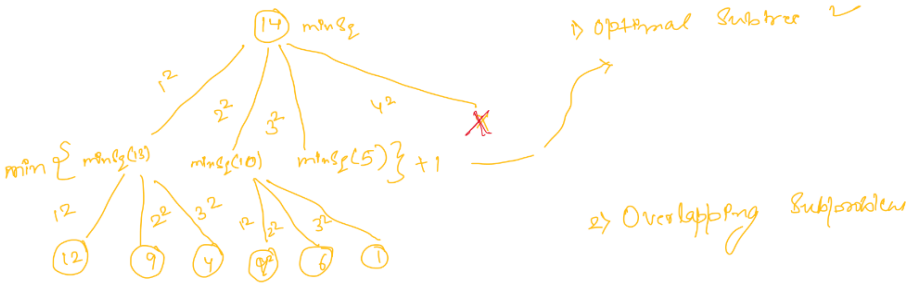


Problem 3

Problem 3 :-

Find min^m no of perfect squares to get sum = N ?

N = 6	{ $1^2 + 1^2 + 2^2$ }	(3)
N = 10	{ $3^2 + 1^2$ }	(2)
N = 12	{ $3^2 + 1^2 + 1^2 + 1^2$ }	(4) ✗
	{ $2^2 + 2^2 + 2^2$ }	(3)



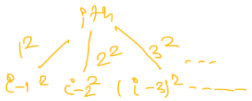
1) Optimal Substructure

2) Overlapping Subproblems

3) dp State

dp[i] = Min^m no of perfect squares to get i

4) dp expression :-



dp[i] = min { dp[i-1²] + 1
dp[i-2²] + 1
:
dp[i-j²] + 1 } ; i-j² >= 0

i >= j²

j <= √i

Pseudo Code — Bottom up

```
int dp[N+1]
dp[0] = 0;
for (i = 1; i <= N; i++) {
    dp[i] = INT_MAX;   // min^m no of perfect Sq to get i
    for (j = 0; j*j <= i; j++)
        dp[i] = min(dp[i], dp[i-j*j] + 1);
}
return dp[N];
```

T.C : O(N √N) ;

S.C : O(N)