13. Dynamic Programming - 1

21 July 2024 14:09

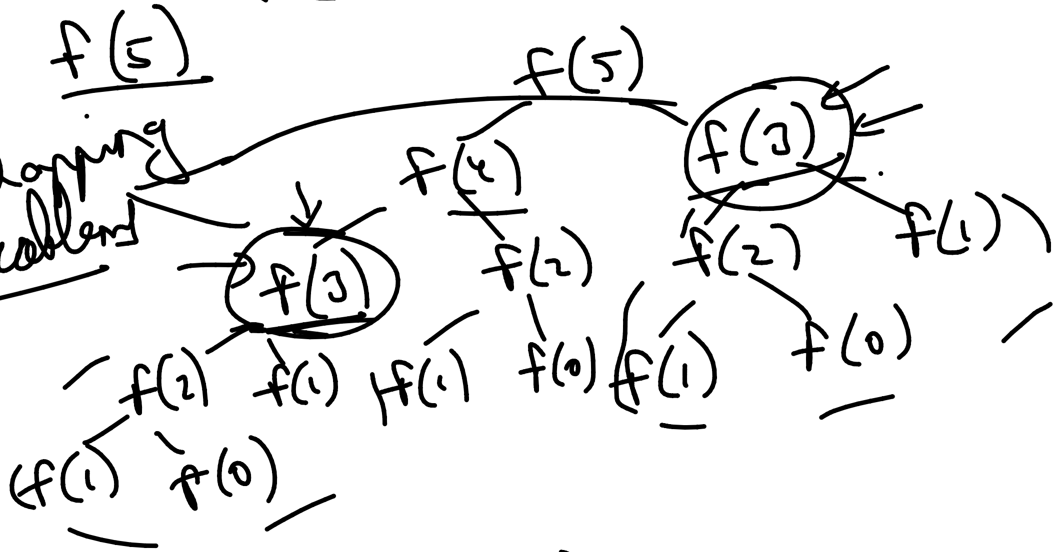
⇒ Q1. Find the n^{th} Fibonacci number.

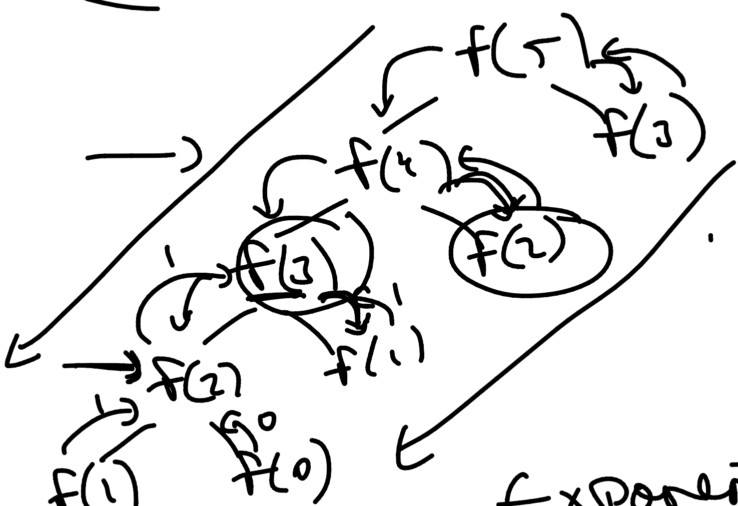
```
int fun(int n){
    if(n==0 || n==1)
        return n;
    return fun(n-1)+fun(n-2);
}
```

$TC = O(2^n)$ → expensive.

f(5)

overlapping subproblems





$O(n+n-1)$

$O(2n-1)$

$O(n)$

Exponential → Polynomial.

DP is optimization over plain recursion.

Here we save the results of subproblems

Here, we save the results of subproblems to avoid calling same recursion again and again.

store result in array ← DP code

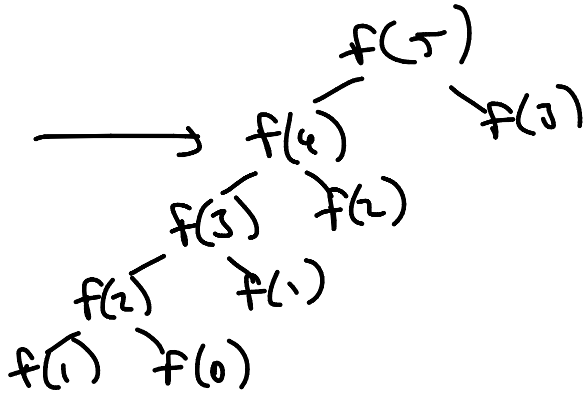
Top-Down	Bottom-Up
→ Recursive	→ Non-recursive.

→ ① **Top-Down** :-

```
int fib( int n, int dp[])
{     if(dp[n] == -1) {
          if(n == 0  || n == 1)
                dp[n] = n;
          else
                dp[n] = fib(n-1, dp) + fib(n-2, dp);
      }
      return dp[n];
}
```

TC = O(n)

```
main() {
    fib(5, dp);
}
```



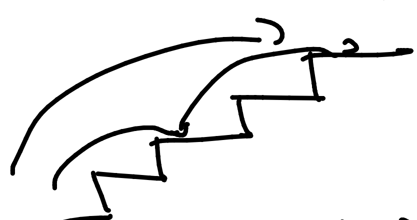
⇒ Bottom-Up Approach :-

```
int dp[n+1];
    dp[0] = 0;    dp[1] = 1;
T C = O(n)   for (i=2 ; i<=n ; i++)
        dp[i] = dp[i-1] + dp[i-2];

    return dp[n];
```

Top-Down	Bottom-Up
① Recursive	① Non-recursive
② Slower	② Faster.
③ Easier to write	③ A bit tough.

Q/. $n = 4$ (1, 2)

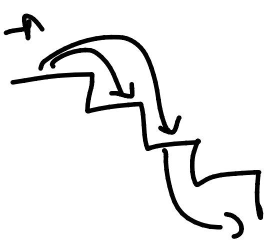


2, 2
2, 1, 1
1, 2, 1
1, 1, 2
1, 1, 1, 1

} 5

0 → n
n → 0

n → 1, 2



f(n) → overlapping subproblems.

f(n−1) + f(n−2)

f(n−2) → f(n−4)

f(n-1) + f(n-2)

f(n-2) f(n-3) f(n-3) f(n-4)

n = 1 → (1) ①

n = 2 → (1, 1) (2) ②

i → a, b, c

dp(i-2) dp(i-1) dp(i)

i+1 → a = b

b = c

c = (a+b)

int → [−p, +q]

dp(i) = dp(i-1) + dp(i-2)

% mod

p % 100 => [0, 99]

→ 100 % 30

10

LCS :-

str1 str2 → l ans = max(ans, l);

i i

A B C D S

D P Q D j

A B C D E S

K E C P E

① ②

D DPQD j D

CE 1 CE 2
EC CE
EC X

D

ABC
PD 1

ABCDQ
PDQ

ABCD ABC
PD PDQ

PDQ
ABCD

fun(n, m)
if(s1(n-1) == s2(m-1))

1 + fun(n-1, m-1) → s1(n-2) != s2(m-2)

max

fun(n-2, m-1) fun(n-1, m-2);

fun(n, m)

fun(n-1, m-1) + 1

max

fun(n-2, m-1) fun(n-1, m-2)

fun(n-3, m-1) fun(n-2, m-2) fun(n-2, m-2) fun(n-1, m-3)

int dp(n+1)(m+1);

dp(i)(j);

dp(n)(m);

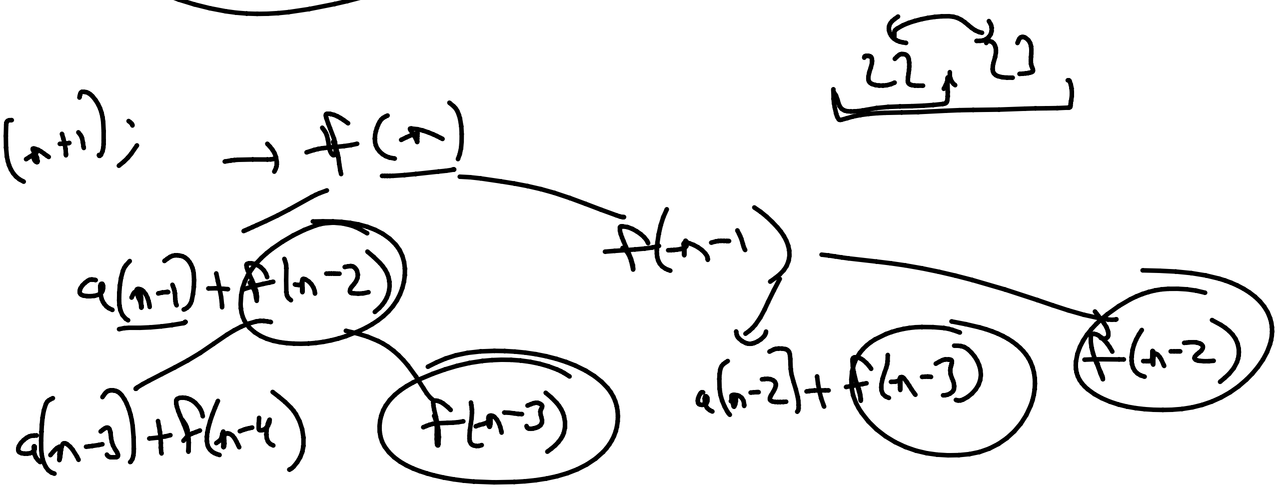
Q/. 0, 1, 2, 3, 4, 5, 6

1 1, 1, 1, 1 1

22

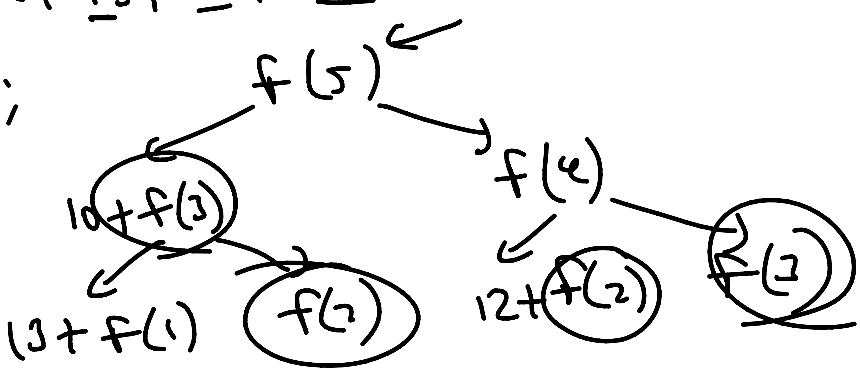
(22)

int dp(n+1);



10, 12, 13, $\overset{\times}{12}$, 10

int dp(n+1);



Q/. Partition Equal Subset Sum

s1 s2

↓ sum of array → even
↓ subset sum problem.

$s1 + s2 = \Sigma arr(i)$

$2x = \Sigma arr(i)$ → even

$x = \Sigma arr(i)$

$$x = \frac{\sum arr[i]}{2}$$

Find a subset of array with sum $= x$

$\boxed{arr \rightarrow x}$ $\quad x = \frac{\sum arr[i]}{2}$

int dp(n+1)(x+1) $\quad dp[i][j] \rightarrow 1$, $\rightarrow 0$

Top-Down:-

$TC = O(n \times sum)$

```
int subset(int i, int n, int s)
{ if (s < 0)
      return F;
  if (dp[i][s] == -1)
  { if (s == 0) dp[i][s] = 1;
    else if (i == n) dp[i][s] = 0;
    else dp[i][s] = subset(i+1, n, s-a[i-1]) ||
                    subset(i+1, n, s);
  } return dp[i][s];
}
```

1, 5, 11, 5 $\rightarrow n = 4$

$x = 22 \rightarrow 11$

Q/. $\quad W = 10$

$n = 3$ wt [7, 5, 5]

val (7, 4, 4)

val [1, 0.8, 0.8]

$\frac{val}{wt} = [1, 0.8, 0.8]$

fun(n, w)

val(n-1) + fun(n-1, w - wt(n-1))

fun(n-1, w)

[_ _ _ _ _ _ 7, 5, 12]

fun(n, 100)

+ fun(n-1, 88)

fun(n-1, 100)

fun(n-2, 88)

fun(n-2, 95)

fun(n-3, 88)

fun(n-3, 88)

int dp[n+1][w+1];

dp[i][j];

dp[n][w];

${}^nC_r$:-

$i = j = 4$

$${}^nC_r = {}^{n-1}C_r + {}^{n-1}C_{r-1}.$$

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

0

${}^3C_4$

fun(n, r)

fun(n-1, r)

fun(n-1, r-1)

fun(n-2, r-1)

fun(n-2, r)

fun(n-2, r-1)

fun(n-2, r-2)

$dp[i][j] = {}^iC_j$

int dp[n+1][r+1]

vint $\underline{dp[n+1][r+1]}$ $\quad dp[i][j] = C_j$

$r = 0 \quad {}^nC_0 = 1$

$r = 1 \quad {}^nC_1 = n$

$$\frac{\underline{|n}}{\underline{|r}\,\underline{|n-r}}$$

% mod

$\left(\frac{a}{b}\right) \% \underline{mod} \rightarrow$ prime

$\Rightarrow \underline{\left(a \times b^{mod-2}\right) \% mod}$