Dynamic Programming

⇒ Recursion (overlapping subproblems)

Memo (1 min)

tab (2 min) → 2-3 tricks

Fibonacci series

0	1	2	3	4	5	6	7	
0,	1,	1,	2,	3,	5,	8,	13,	21 - - - - -

$n=2$ → side

ans = 1

$n = 3$

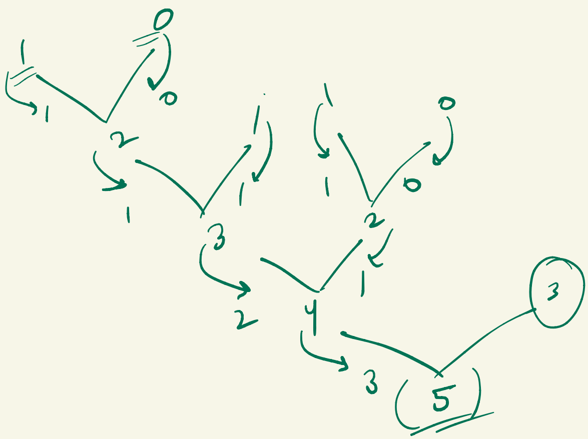
ans = 2

$n \to 0$

$n = n$

$n+1$

$n+1$



⇒ 2

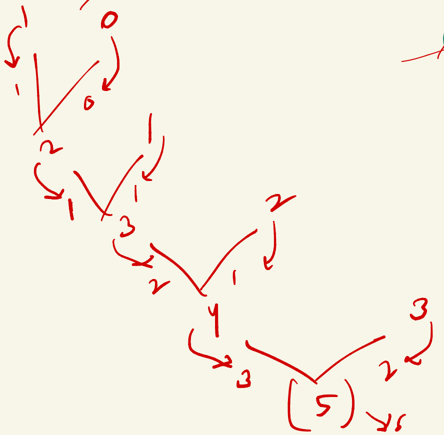
(number of calls) height of tree → $(2)^n$

space ⇒ recursive space → height of tree

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

memo == tab

smallest problem



→ problem

~~sub problems~~

sp

sp

O(n)

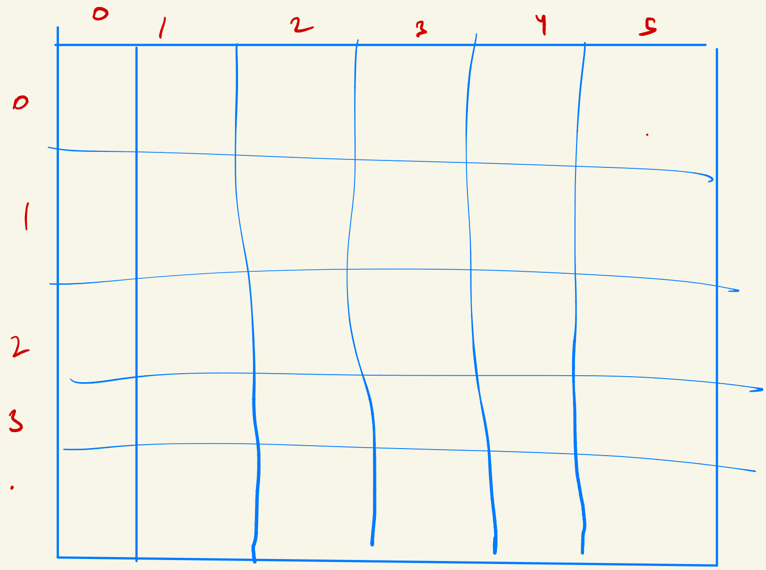
space → recursive

```java
public static int memo_fib(int n,int[] memo){
    if(n==0 || n==1){
        return n;
    }

    if(memo[n]!=0) return memo[n];

    int ans=0;
    ans=memo_fib(n-1,memo)+memo_fib(n-2,memo);

    return memo[n]=ans;
}
```

return → continue;

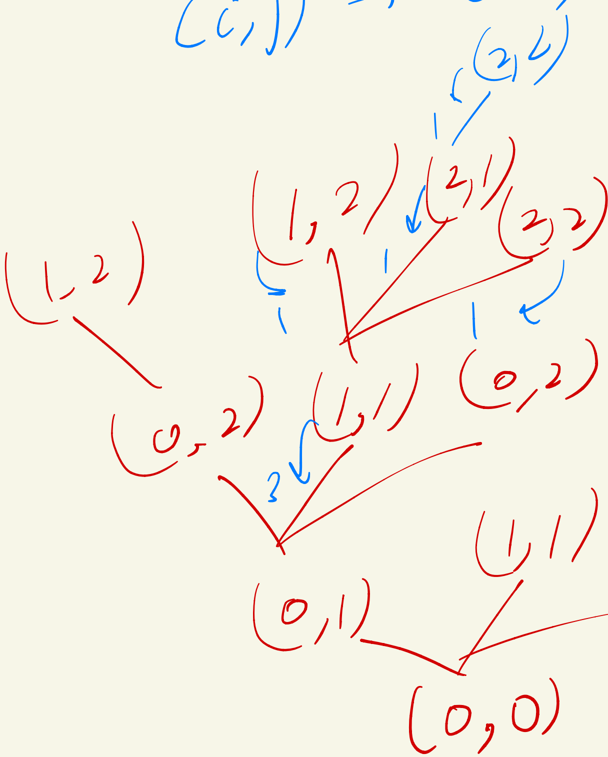
calls → dp array

recursion → memo → tab → optimization.



$(2,1) \to (2,2)$

$(i,j) \to (n-1, m-1)$



$\boxed{3 \times 3}$

	0	1	2
0	13	5	1
1	5	3	1
2	1	1	1

memo

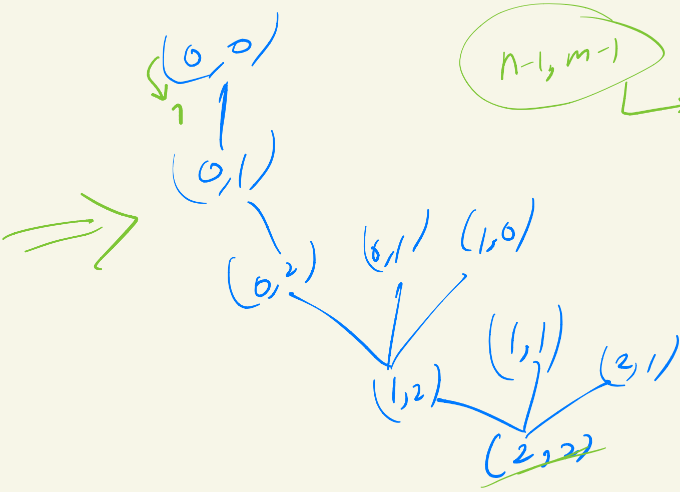
dp(i)(j) → numbers of paths from (i, j) to (n-1, m-1);

Tricks

1) for loop starts from base cases.
2) your final answer lies at from where you called your main function

(3, 3)

dp(i)(j) → (n-1, m-1) → string

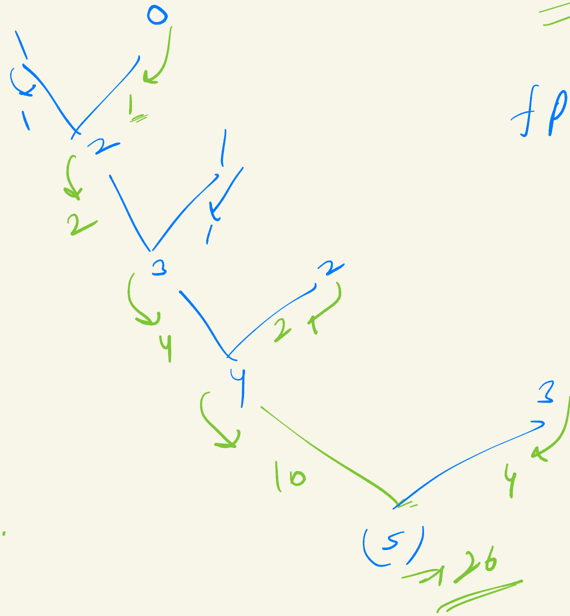


dp(n-1)(m-1);

$fp(2) = 2$

go alone

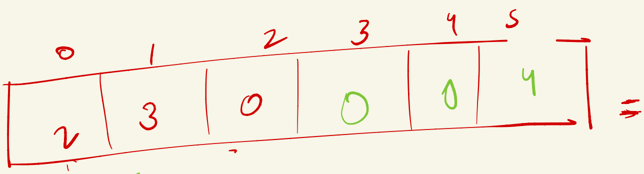
pair

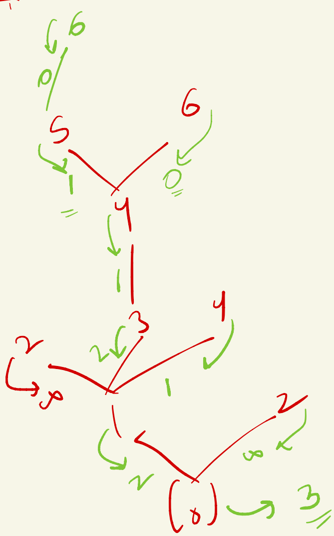


$$fp(n) = fp(n-1) + fp(n-2) * (n-1)$$

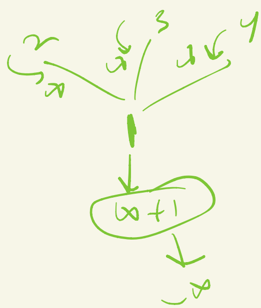
$10 + 4 * 4$

$\Rightarrow 26$





∞



string str = b a a b a c c a b

0 1 2 3 4 5 6 7 8

query → out

si ei

	diag 0 0	diag 1 1	diag2 2	d3 3	d4 4	d5 5	d6 6	d7 7	d8 8
0	✓	✗	✗	✓	✗	✗	✗	✗	✗
1		✓	✓	✗	✗	✗	✗	✗	✗
2			✓	✗	✓	✗	✗	✗	✗
3				✓	✗	✗	✗	✗	✓
4					✓	✗	✗	✓	✗
5						✓	✓	✗	✗
6							✓	✗	✗
7								✓	✗
8									✓

We can tell

→ longest pall (length)

→ number of pall.

→ string pallindrome (longest)

dp(i)(j) → substring from (i, j) is pallindrome or not.

↓

charAt(i) == charAt(j) ✓

+

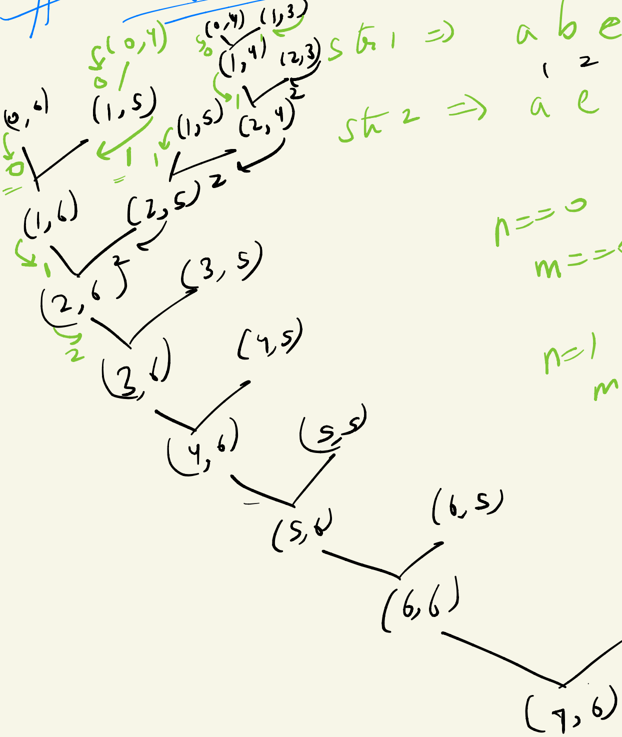
dp(i+1)(j−1) == true ✓

```
M = {{1, 3, 1, 5},
     {2, 2, 4, 1},
     {5, 0, 2, 3},
     {0, 6, 1, 2}};
```

13	12	6	5
14	11	9	1
16	9	5	3
11	11	4	2

String Type

	1	2	3	4	5	6	7
str1 =>	a	b	e	c	f	d	a
str2 =>	a	e	b	f	a	l	



$n==0$

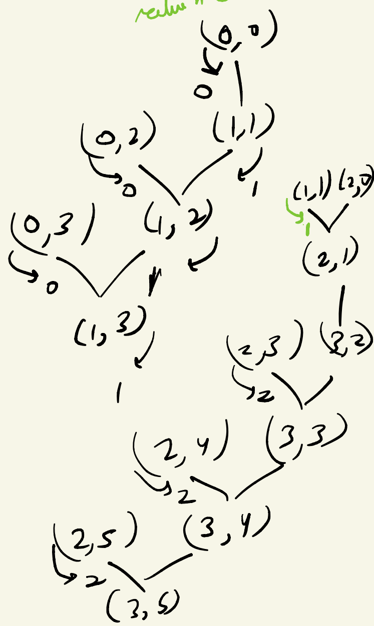
$m==0$

$n=1$

$m=1$

if ($n==0$)
return 0

if ($m==0$)
return 0



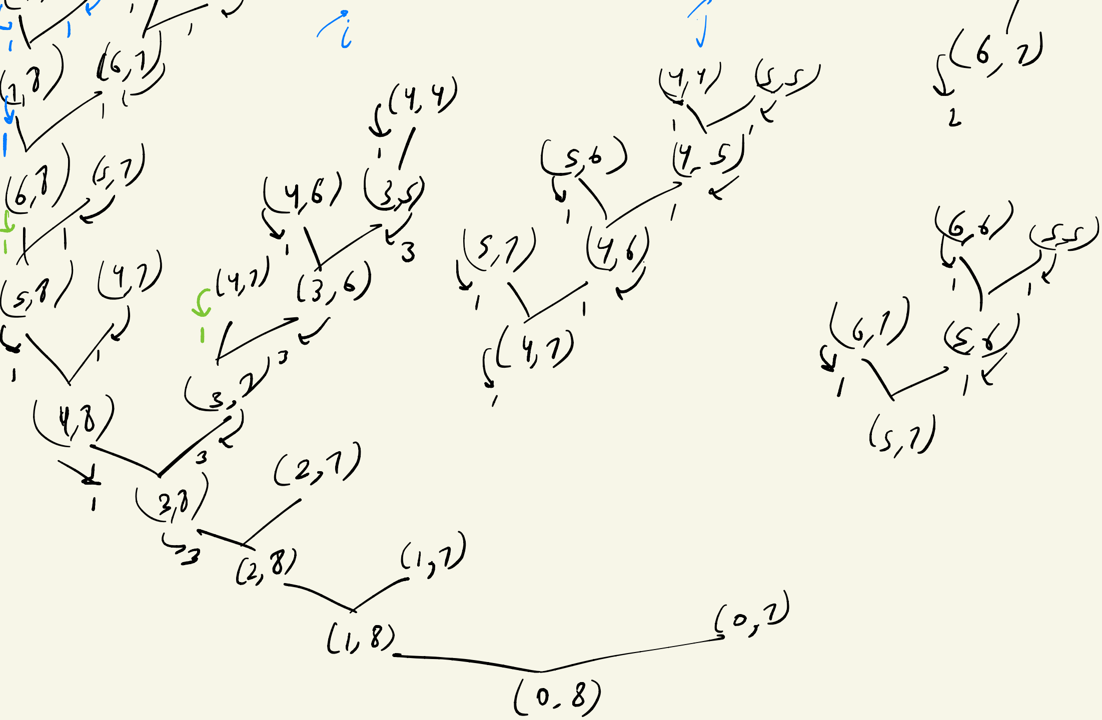
dp[n][m] => lcs from first n characters of string1 and first m characters of string 2.

---

longest pallindromic subsequence

0	1	2	3	4	5	6	7	8
a	e	d	b	c	b	g	a	f

i, j



Edit distance

$(n, m) \to (n-1, m-1)$

1) Insert

dp[n][m]

ho s rse

ros

$(n, m) \to (n, m-1)$

replace

ho s se

ros

$(n, m) \to (n-1, m-1)$

2) delete

horse

ros

$(n, m) \to (n-1, m)$

1	2	3	4	5
h	o	r	s	e

if (m == 0)
return n;

1	2	3
r	o	s

if (n == 0)
return m;

i → $(n, m-1)$
d → $(n-1, m)$
r → $(n-1, m-1)$