Dynamic Programming

⇒ Recursion (overlapping subproblems)

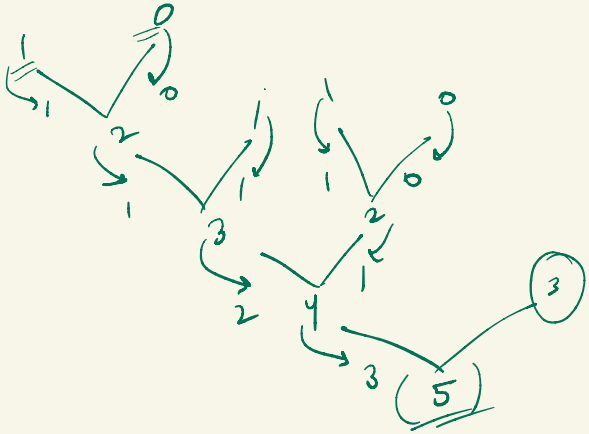
↓

- Memo (1 min)
  - ↓ tab (2 min) → 2 → tricks

Fibonacci series

0 1 2 3 4 5 6 7

0, 1, 1, 2, 3, 5, 8, 13, 21 - - - - -



n=2 → 1 (ans)

ans = 1

n = 3

ans = 2

n → 0

n = n

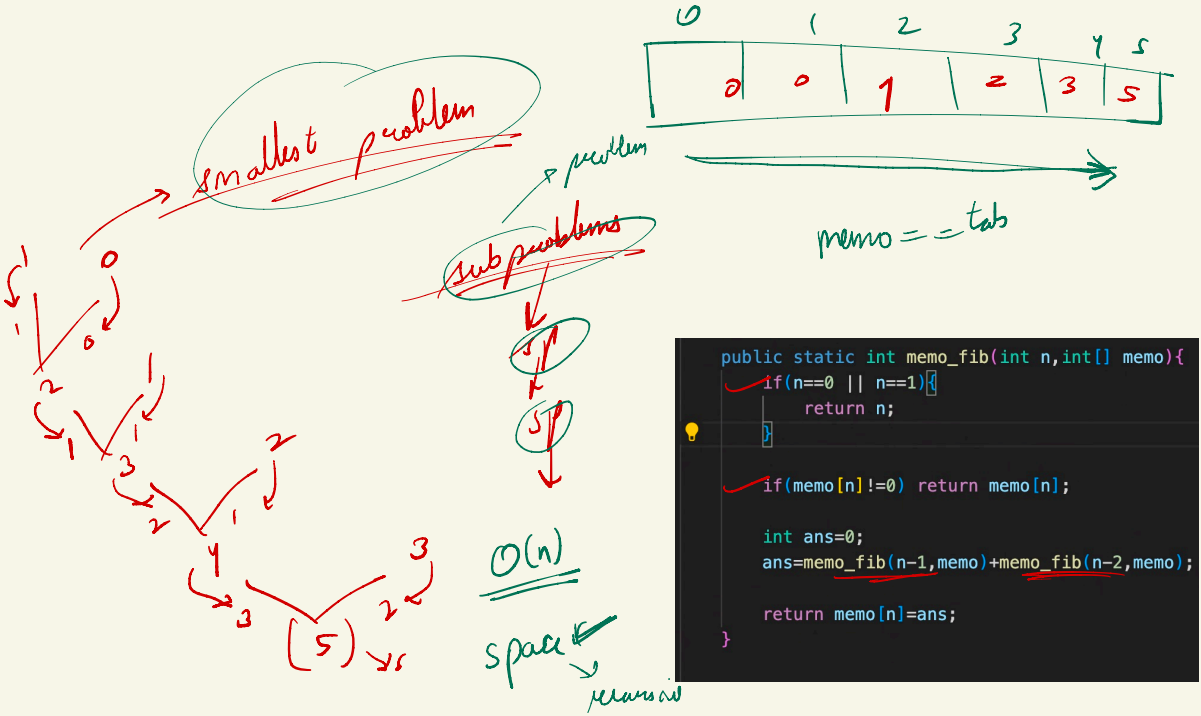
n+1

n+1

⇒ 2

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

space ⇒ recursive space (height of tree)

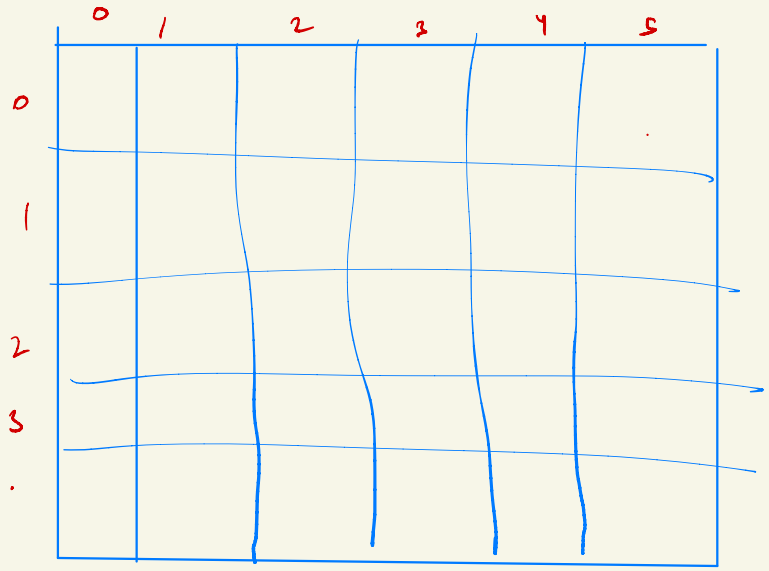



```java
public static int memo_fib(int n,int[] memo){
    if(n==0 || n==1){
        return n;
    }

    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;
}
```

recursion → continues

caller → dp array

recursion → memo → tab → optimization.



(2,2) → (2,2)

(i,j) → (n-1, m-1)

3x3

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

row

(2,2)

(1,2) (2,1) (2,2)

(1,2) (1,1) (0,2)

(0,2)

(0,1) (1,1) (1,0)

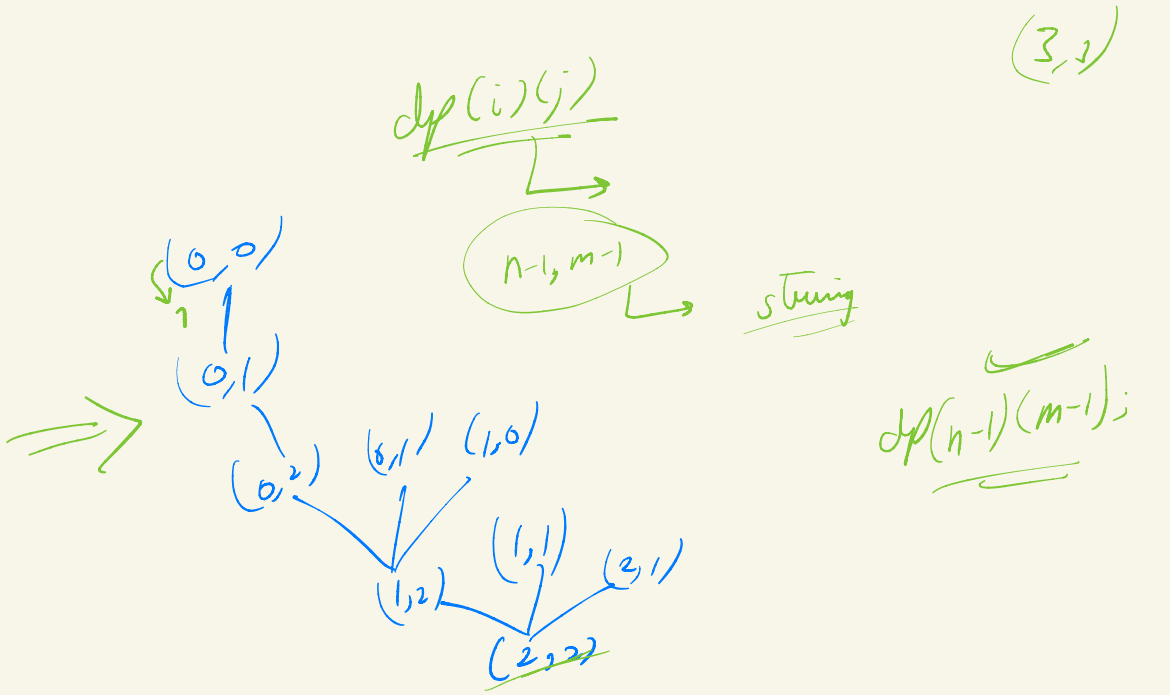
(0,0)

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

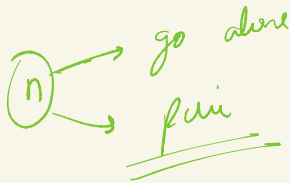
Tricks

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

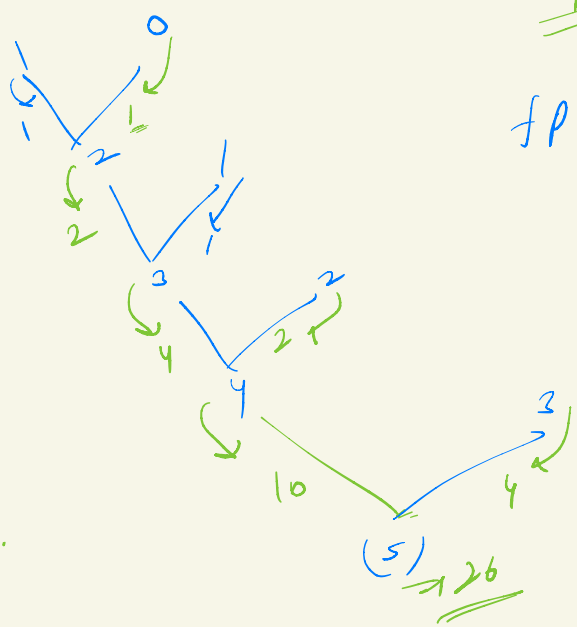
---



$fp(2) = 2$

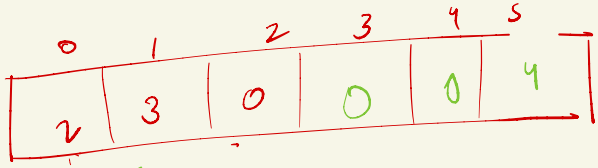


$$fp(n) = fp(n-1) + fp(n-2) \text{ if } (n-1)$$

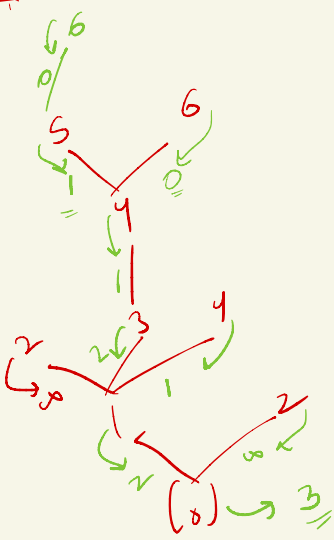


$10 + 4 + 4$

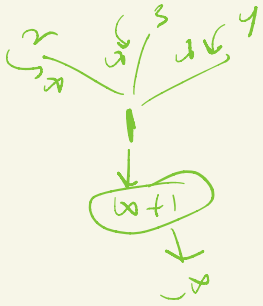
$\Rightarrow 26$



$=$



∞



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

0 1 2 3 4 5 6 7 8

query → output

si ei

We can tell
→ longest palindrome (length)
→ number of palindrome.
→ string palindrome. (longest)

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

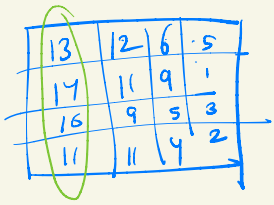
↓

char At(i) == char At(j) ✓

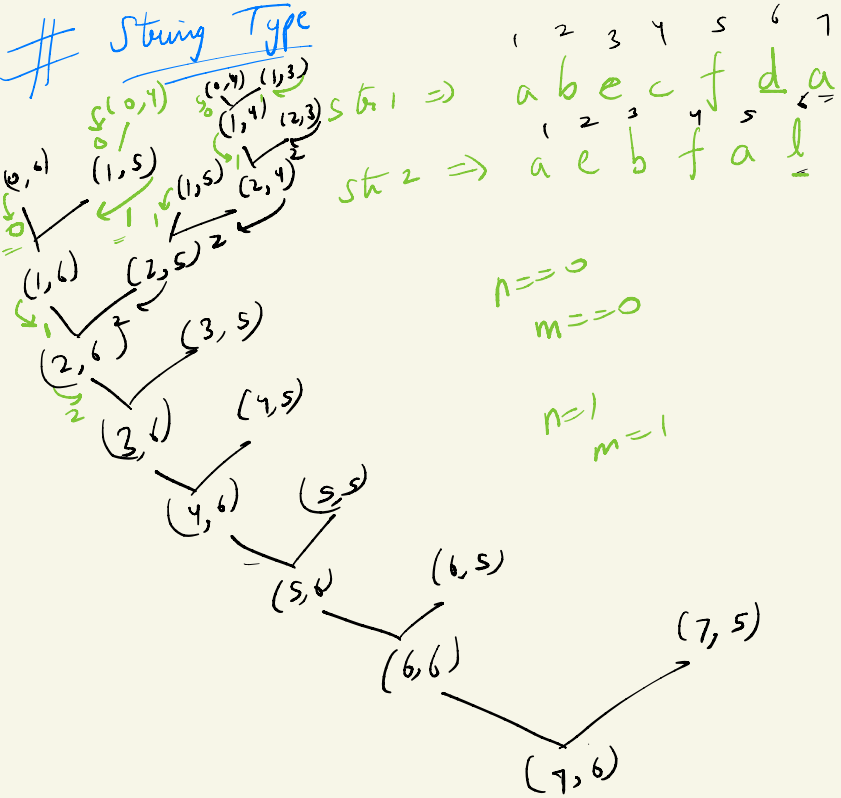
+

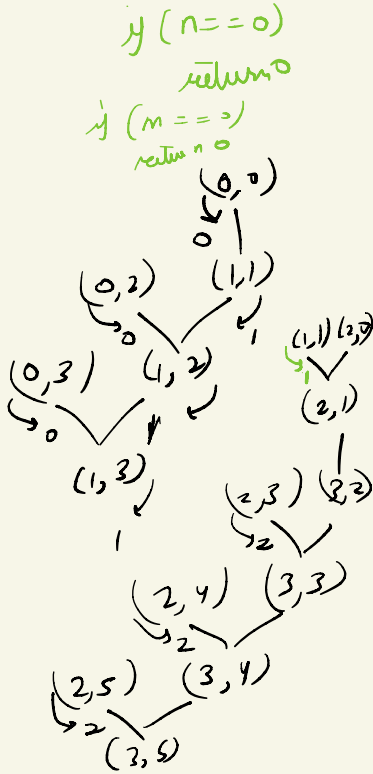
dp(i+1)(j-1) == true ✓

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



String Type

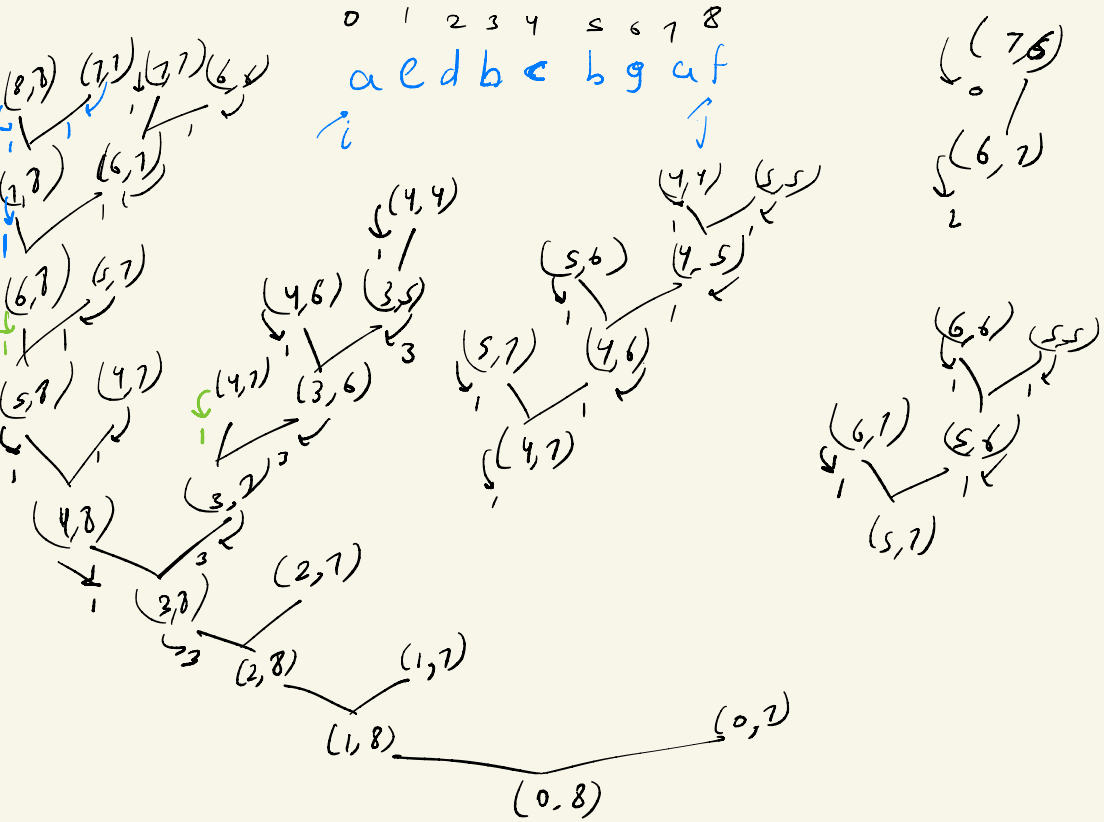




dp(n)(m) ⇒ lcs from first n characters of string 1 and first m characters of string 2.

---

longest palindromic subsequence



Edit distance

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

1) insert

$dp[n][m]$

horse

ros

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

replace

horse

ros

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

2) delete

horse

ros

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

h o r s e

1 2 3 4 5

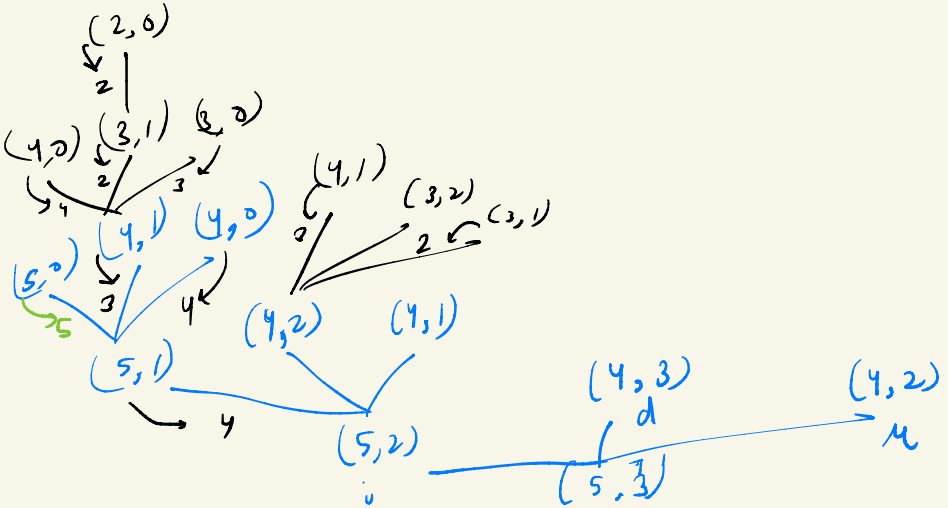
if (m == 0)
return n;

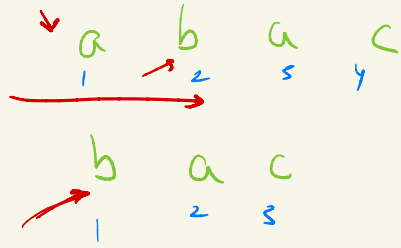
r o s

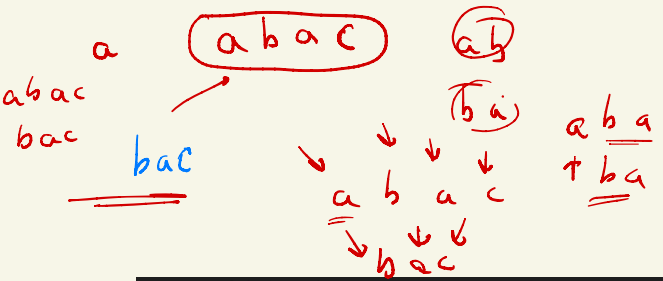
1 2 3

if (n == 0)
return m;

if (n, m-1)
d (n-1, m)
r (n-1, m-1)







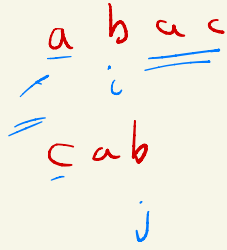
		o	b	a	c
		0	0	0	0
a		0	0, ""	1, a	1, a
b		0	1, b	1, b	1, a
a		0	1, b	2, ba	2, ba
c		0	1, b	2, ba	3, bac

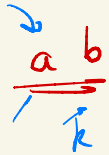
lcs

```
public static int tab_lcs(String text1, String text2, int N, int M){
    int[][] dp=new int[N+1][M+1];
    String[][] sdp=new String[N+1][M+1];

    for(int n=0; n<=N; n++){
        for(int m=0; m<=M; m++){
            if(n==0 || m==0){
                dp[n][m]=0;
                sdp[n][m]="";
                continue;
            }

            //if(dp[n][m]!=-1) return dp[n][m];

            if(text1.charAt(n-1)==text2.charAt(m-1)){
                dp[n][m]=dp[n-1][m-1]+1;//memo_lcs(text1, text2, n-1, m-1, dp) +
                sdp[n][m]=sdp[n-1][m-1]+text1.charAt(n-1);
            } else {
                if(dp[n-1][m]>dp[n][m-1]){
                    dp[n][m]=dp[n-1][m];
                    sdp[n][m]=sdp[n-1][m];
                } else {
                    dp[n][m]=dp[n][m-1];
                    sdp[n][m]=sdp[n][m-1];
                }
                // dp[n][m]=Math.max(dp[n-1][m],dp[n][m-1]);//Math.max(memo_lcs(
            }
        }
    }
    print(sdp);
    return dp[N][M];
}
```

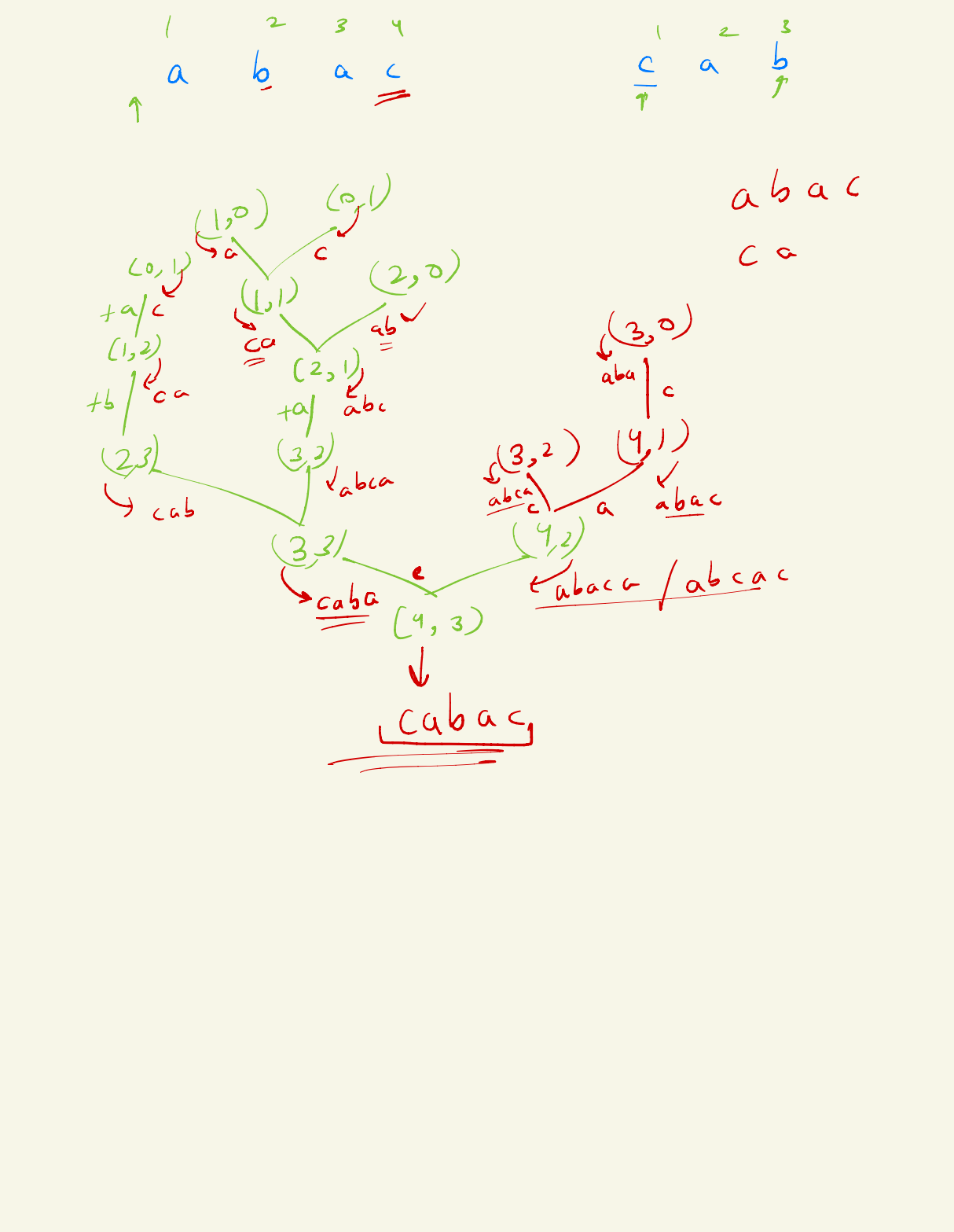
str1, str2 → lcs

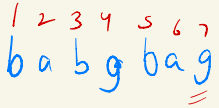
i

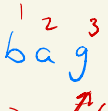


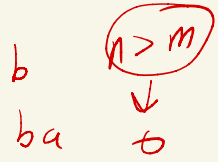


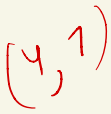
Ans = c a b ac

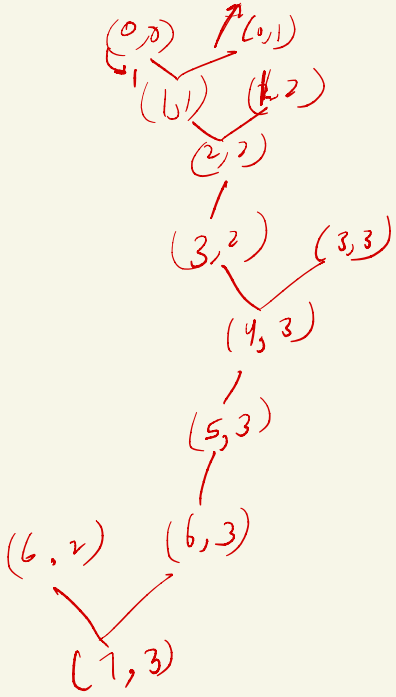

1 2 3 4
a b a c
1 2 3
c a b
abac
ca
(1,0)
(0,1)
a
c
(0,1)
(1,1)
(2,0)
+a/c
ca
ab
(1,2)
(2,1)
+b
ca
abc
+a
(2,3)
(3,2)
abca
cab
(3,3)
caba
e
(4,3)
cabac
(3,0)
aba
c
(3,2)
(4,1)
abca
c
a
abac
(4,2)
abaca / abcac

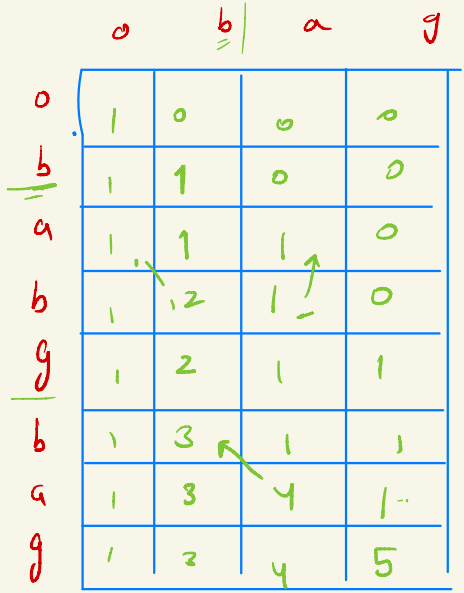

1 2 3 4 5 6 7
b a b g b a g


1 2 3
b a g


b
ba
n > m


(4,1)


(0,0)
(0,1)
(1,1)
(1,2)
(2,2)
(3,2)
(3,3)
(4,3)
(5,3)
(6,2)
(6,3)
(7,3)



babgba g

bag

babg

bag

ba

```
public static int tab(int n, int m, String s,String t, int[][]qb){
    int N=n;
    int M=m;
    for(n=0; n<N; n++){
        for(m=0; m<M; m++){
                if(n<m){
            qb[n][m]=0;
                continue;
        }
            if(m==0){
                qb[n][m]=1;
            continue;
            }

            int ans=0;
            if(s.charAt(n-1)!=t.charAt(m-1)){
                ans+=qb[n-1][m];//some(n-1,m,s,t,qb);
            } else {
                ans+=qb[n-1][m-1]+qb[n-1][m];//some(n-1,m-1,s,t,qb)+some(
            }
         qb[n][m]=ans;
        }
    }
    return qb[N][M];
}
```

a b b b a c a b c

$a^i b^j c^k$

ending with a

aaa
aa
aa
aa
a
a
a

ending with ab

not adding

abbb
abb
abb
abb
ab
ab
ab

behind even old

abbbb
abbb
abbb
abbb
abb
abb
abb

21

aaab
aab aab
aab
ab
ab
ab

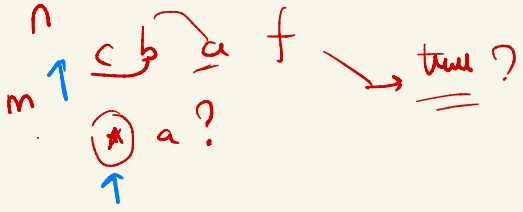
ending with abc

abbbc
abbc
abbc
abbc
abc
abc
abc

7

21 + 2*7

14



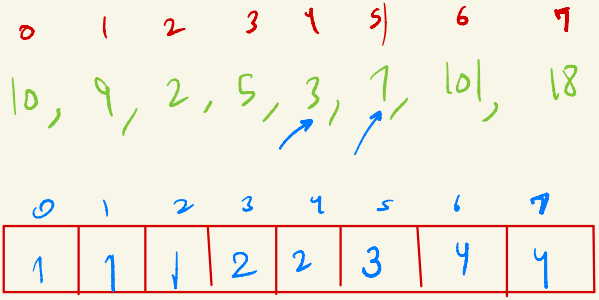
? → one char

* → empty / some number of char.

n==0 && m==0

return 1

n==0 m=1



dp(i) ⇒ length of LIS ending with arr(i);

5, 10, 9, 2, 3, 5, 3, 7, 50, 101, 18, 110

inserting-position = ~~X X 0 1 2 3 4 5~~ ?

```
public int BS(ArrayList<Integer> dp, int ele){
    int si=0;
    int ei=dp.length()-1;

    while(si<=ei){
        int mid=(si+ei)/2;

        if(dp.get(mid)<ele){
            si=mid+1;
        } else {
            ei=mid-1;
        }
    }

    return si;
}
```

dp → { 2, 3, 5, 7, 18, 101, 110 }

5, 9 50

si = ~~0 2~~ 4 mid = 2 4 3

ei = ~~5~~ 3

dp.add

→ Three diagrams

→ memo, tab (rec, memo, tab)

table

after

→ class

→ a week after

→ 14-15

lists

code

code, dry runs

notes

longest decreasing subsequence

bitonic subsequence → 9 10

code → comments