Dynamic Programming

⇒ Recursion (overlapping subproblems)

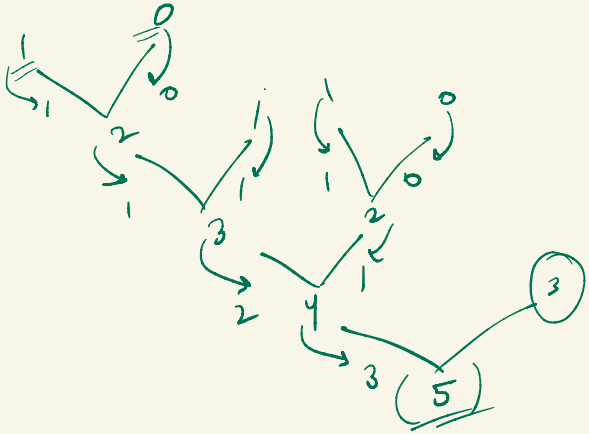
↓

- 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

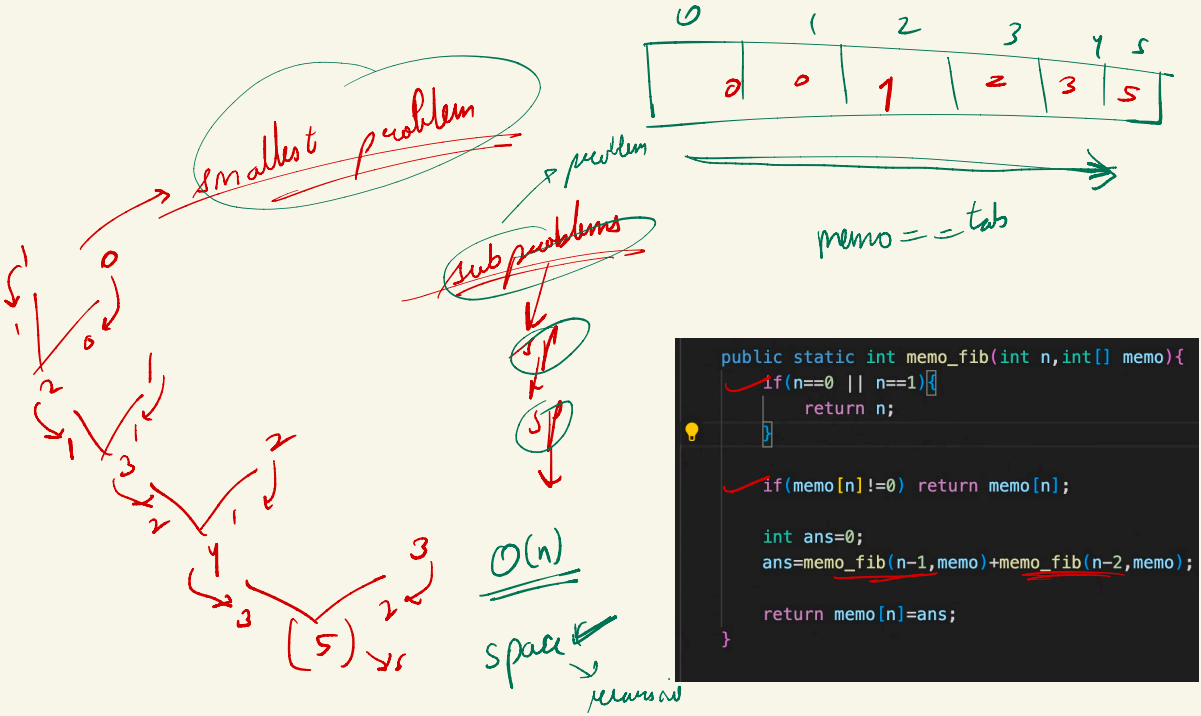
n+1

n+1

⇒ 2

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

space ⇒ recursive space (height of tree)

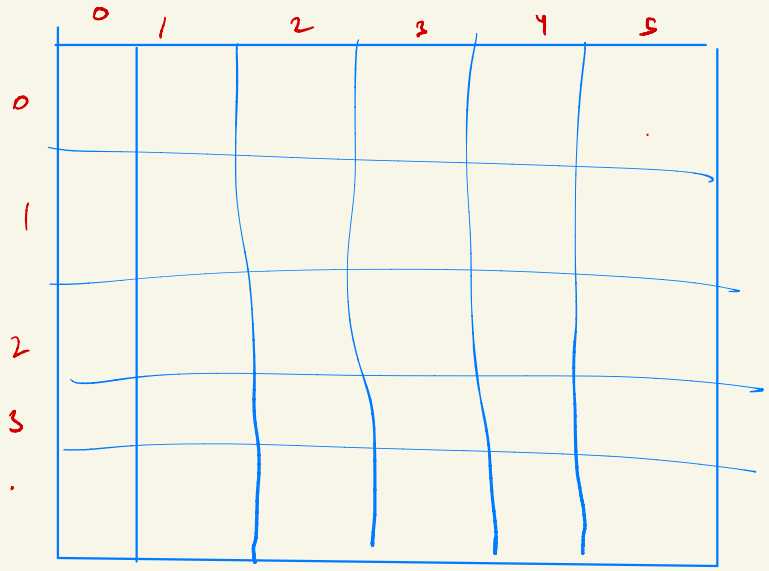



```java
public static int memo_fib(int n,int[] memo){
    if(n==0 || n==1){
        return n;
    }

    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)

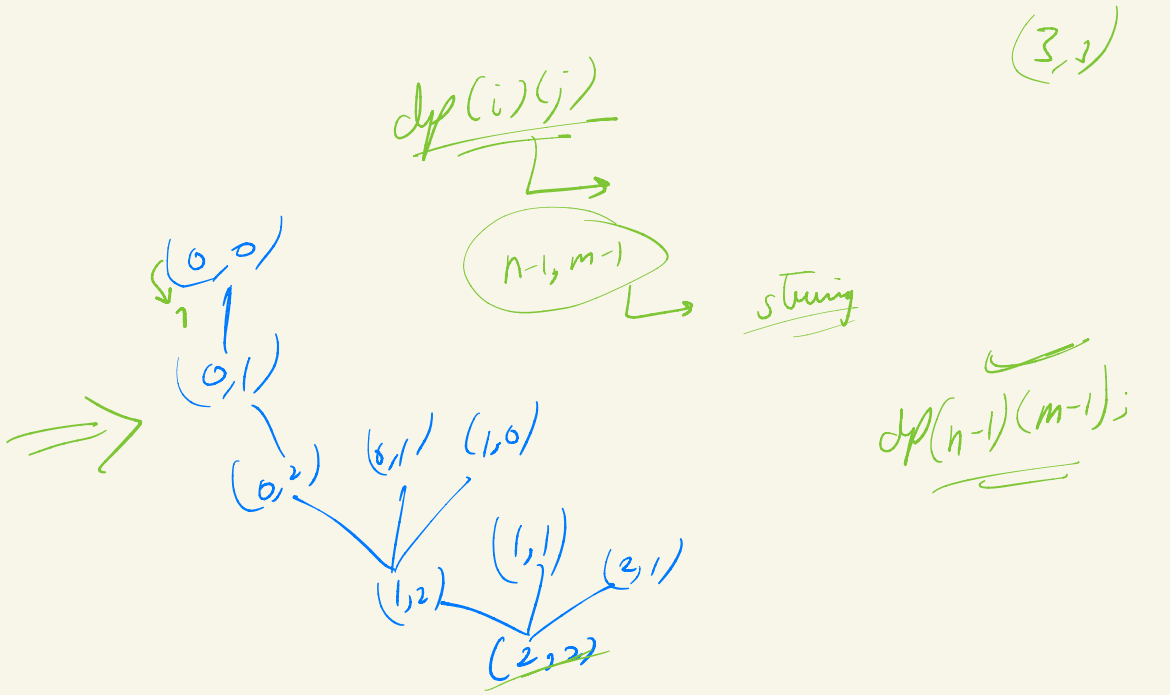
(0,0)

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

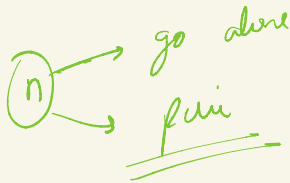
Tricks

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

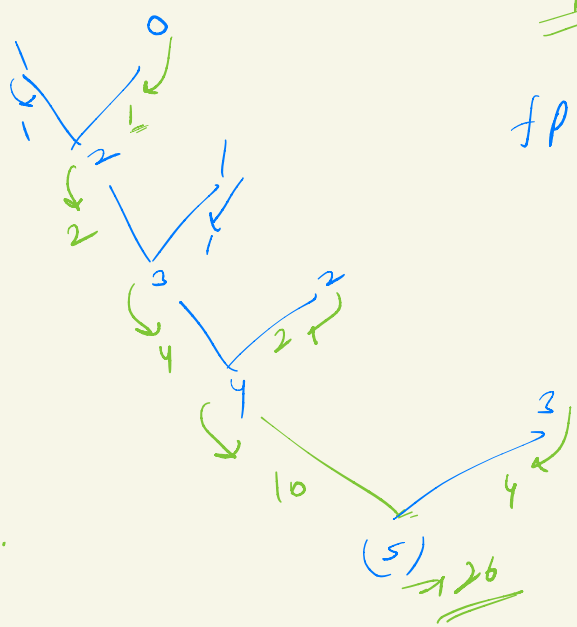
---



$fp(2) = 2$

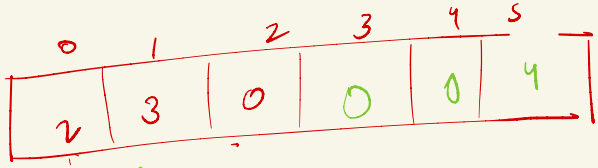


$$fp(n) = fp(n-1) + fp(n-2) \text{ if } (n-1)$$

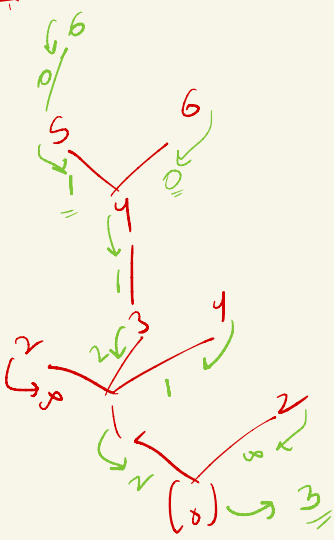


$10 + 4 + 4$

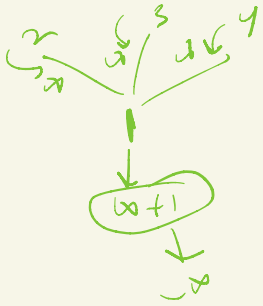
$\Rightarrow 26$



$=$



∞



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.

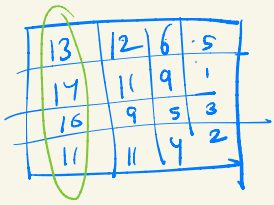
↓

char At(i) == char At(j) ✓

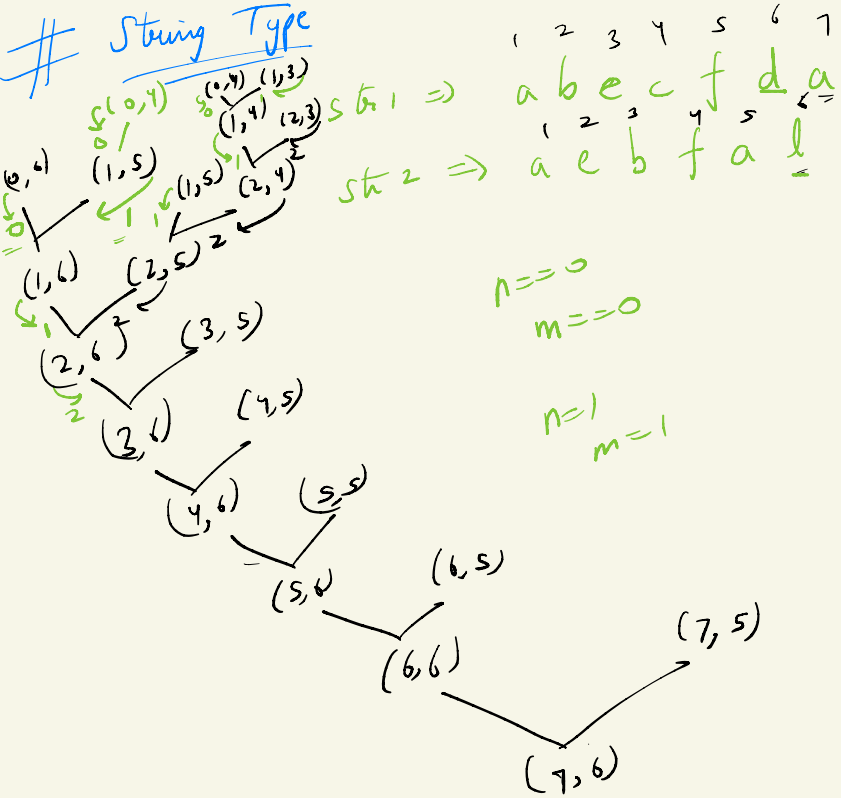
+

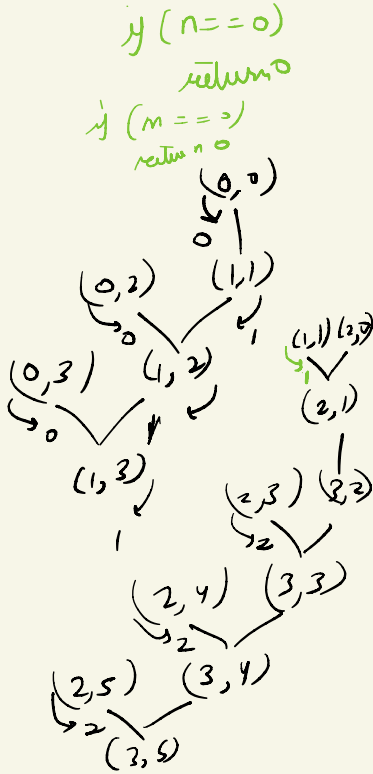
dp(i+1)(j-1) == true ✓

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



String Type

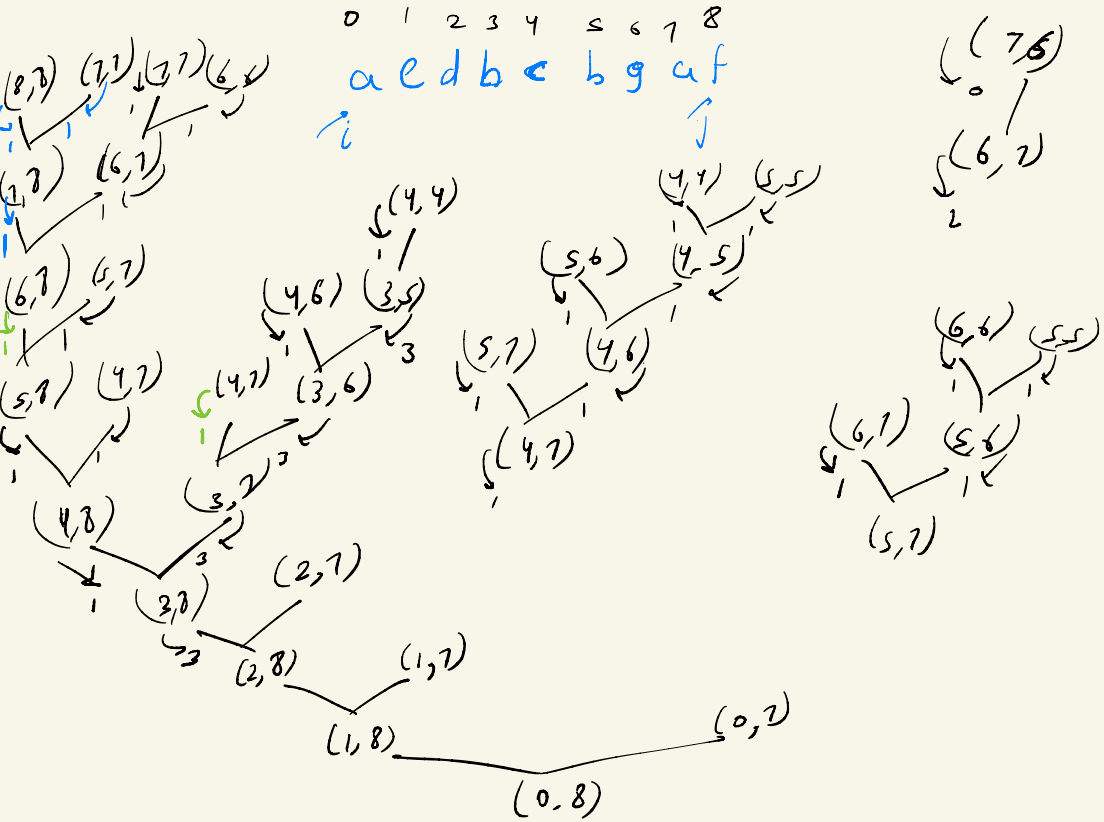




dp(n)(m) ⇒ lcs from first n characters of string 1 and first m characters of string 2.

---

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

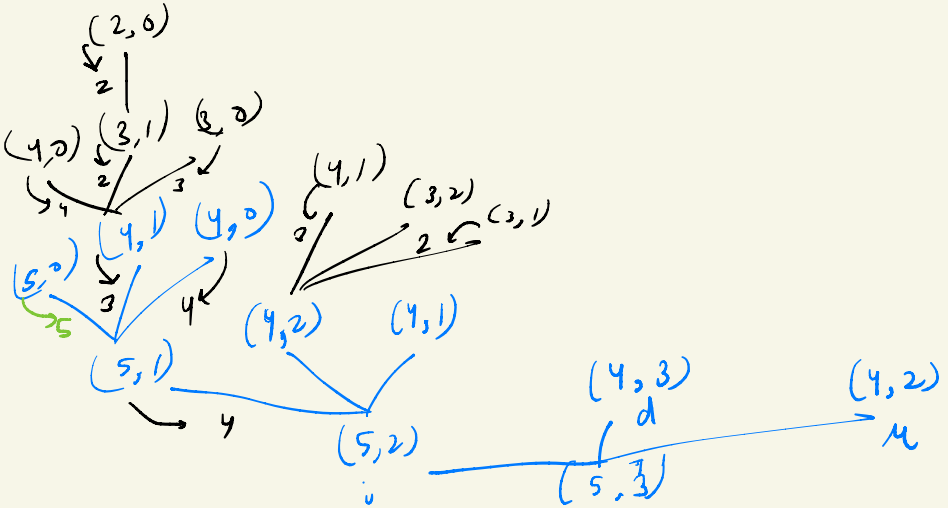
if (m == 0)
return n;

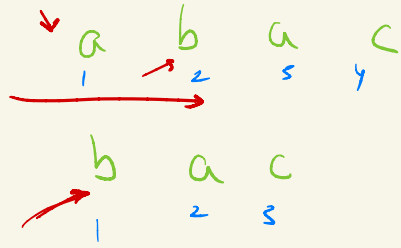
r o s

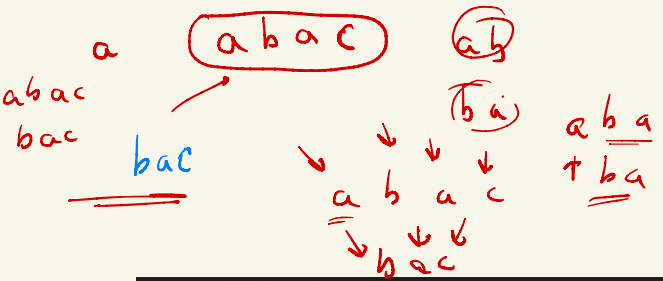
1 2 3

if (n == 0)
return m;

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







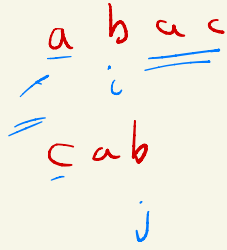
		o	b	a	c
		0	0	0	0
a		0	0, ""	1, a	1, a
b		0	1, b	1, b	1, a
a		0	1, b	2, ba	2, ba
c		0	1, b	2, ba	3, bac

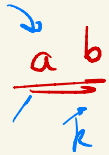
lcs

```
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];
}
```

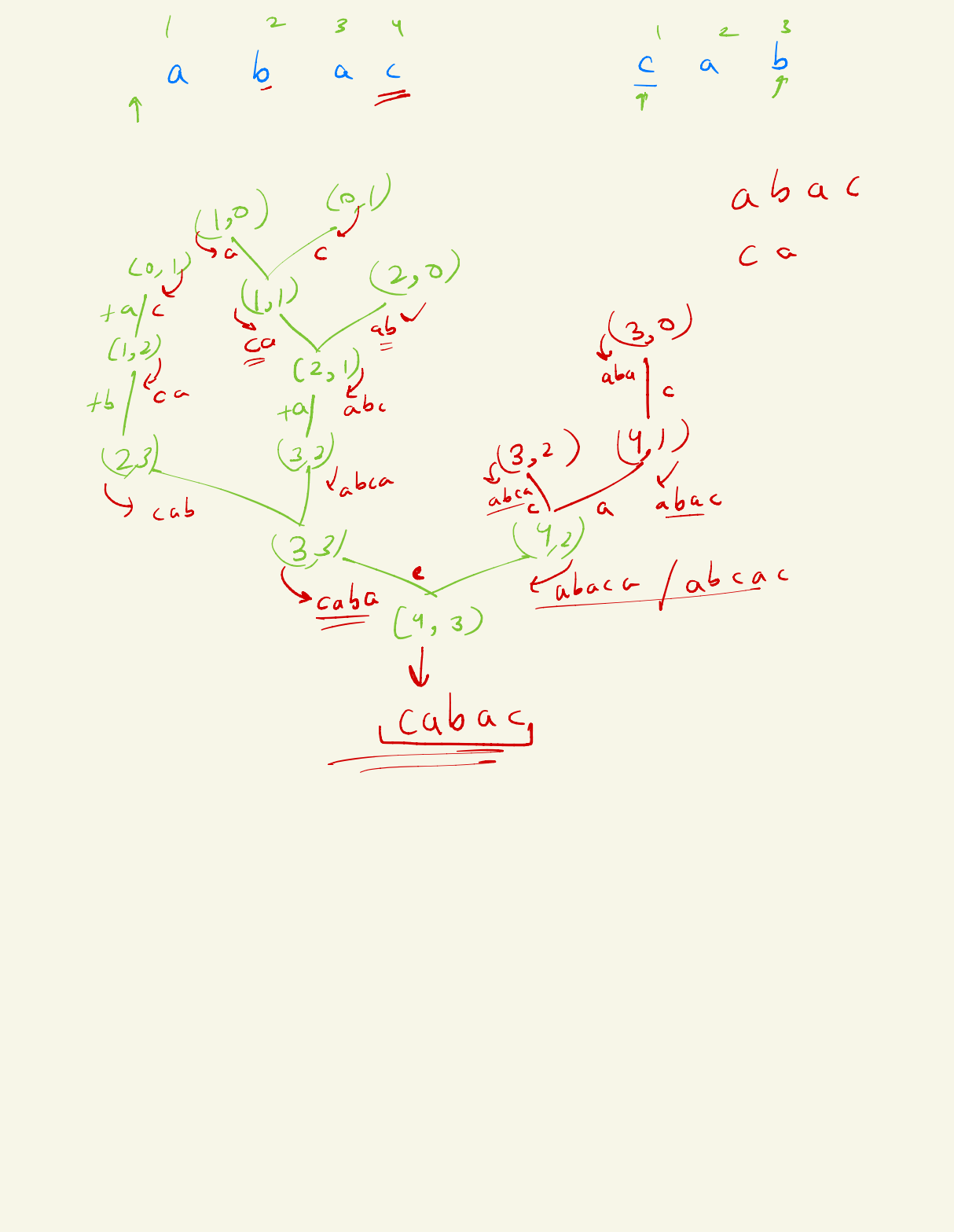
str1, str2 → lcs

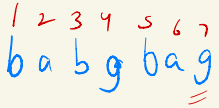
i

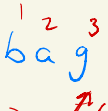


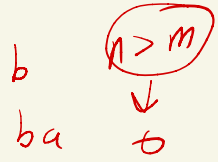


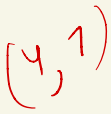
Ans = c a b ac

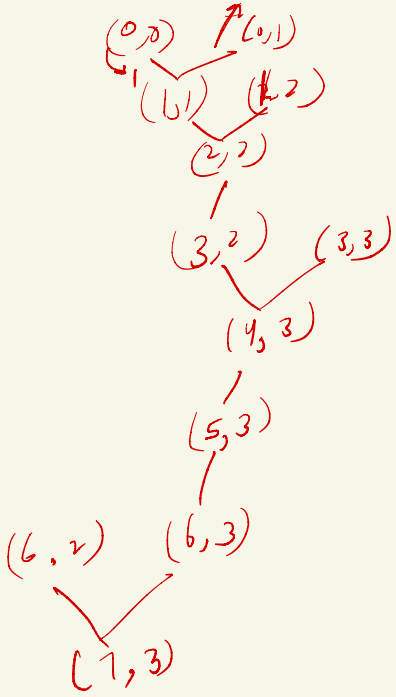

1 2 3 4
a b a c
1 2 3
c a b
abac
ca
(1,0)
(0,1)
a
c
(0,1)
(1,1)
(2,0)
+a/c
ca
ab
(1,2)
(2,1)
+b
ca
abc
+a
(2,3)
(3,2)
abca
cab
(3,3)
caba
e
(4,3)
cabac
(3,0)
aba
c
(3,2)
(4,1)
abca
c
a
abac
(4,2)
abaca / abcac

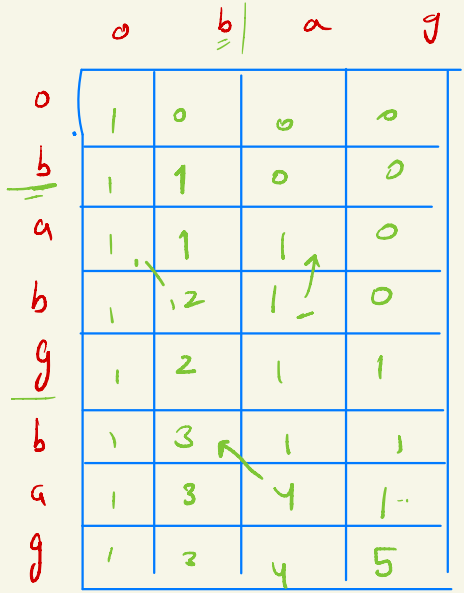

1 2 3 4 5 6 7
b a b g b a g


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

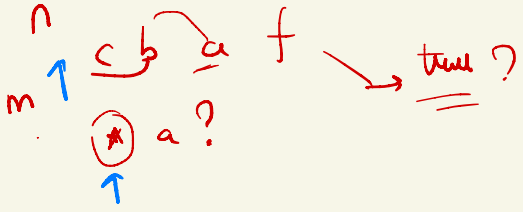
ending with abc

abbbc
abbc
abbc
abbc
abc
abc
abc

7

21 + 2*7

14



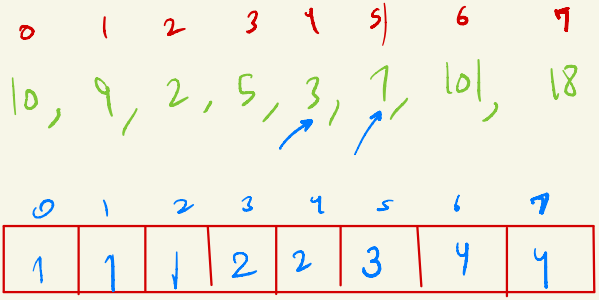
? → one char

* → 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

4 9 5 1 2 3 7

(3 5 1 2 4 7 9)

(3 2 0) (4) (5) (7) (9)

dp → smaller subproblems → bottom up

bag capacity → 10

coins → | 3 | 1 | 4 | 2 | 1 | 5 |

weight → | 5 | 7 | 1 | 3 | 4 | 10 |

→ 1 Logn

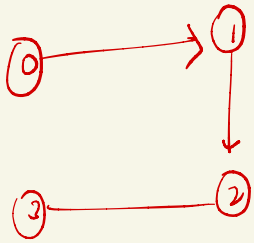
n^2

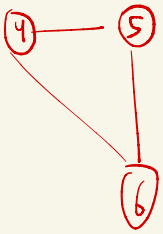
Graph

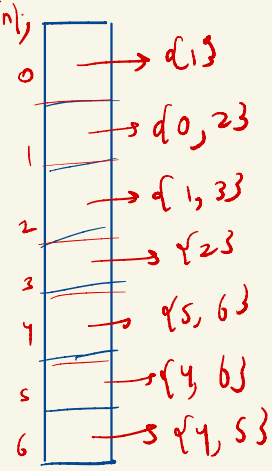
ArrayList<Integer>[] = new ArrayList(n);

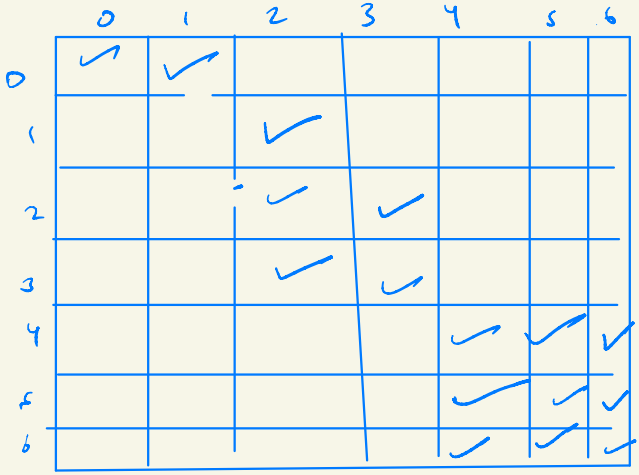
1) Vertices

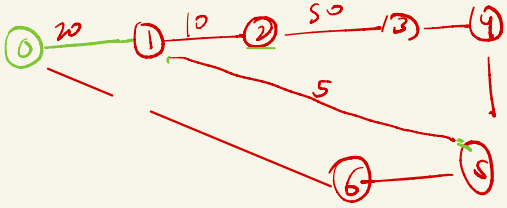
2) Edges





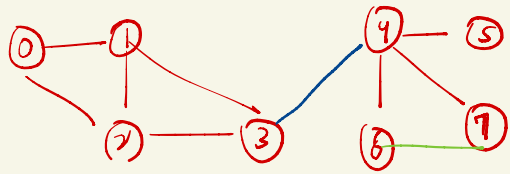






Hamiltonian path →

Hamiltonian circuit → src == des



ArrayList<Integer> al;

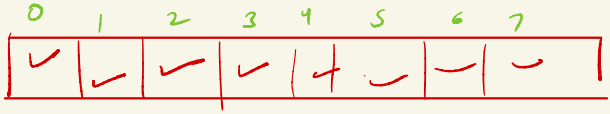
dfs

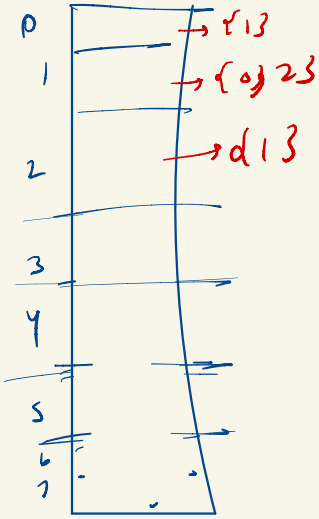
0 1 3

1

Stack, visited

O(V+E)

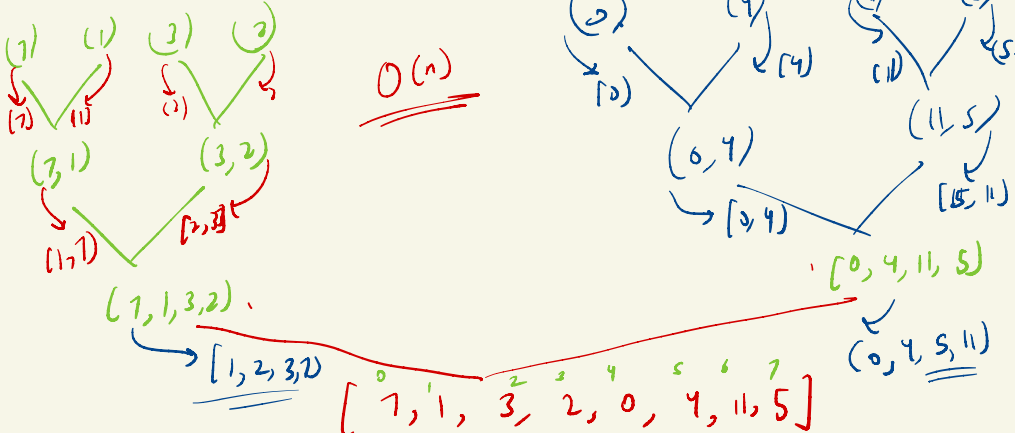




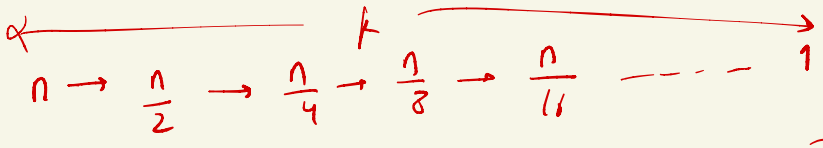
size=1

level 1 → 1 → 2 → 3 → 4 → 5 → 6 → 7

O(n)



[0, 1, 2, 3, 4, 5, 7, 11]



$\rightarrow \frac{n}{2^0}, \frac{n}{2^1}, \frac{n}{2^2}, \frac{n}{2^3}, \frac{n}{2^4}$ ---- $\frac{n}{2^k}$

$\frac{n}{2^k} = 1$

$n = 2^k$

$k = \log_2 n$

$TC = n \log n$

---

a → 1000

b → 10

100

	Frequency (in thousands)	Fixed-length codeword
'a'	45	000
'b'	13	001
'c'	12	010
'd'	16	011
'e'	9	100
'f'	5	101

a → 45

b → 13

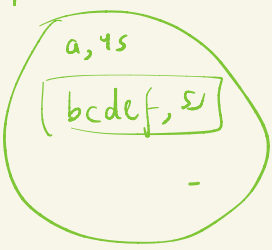
c → 12

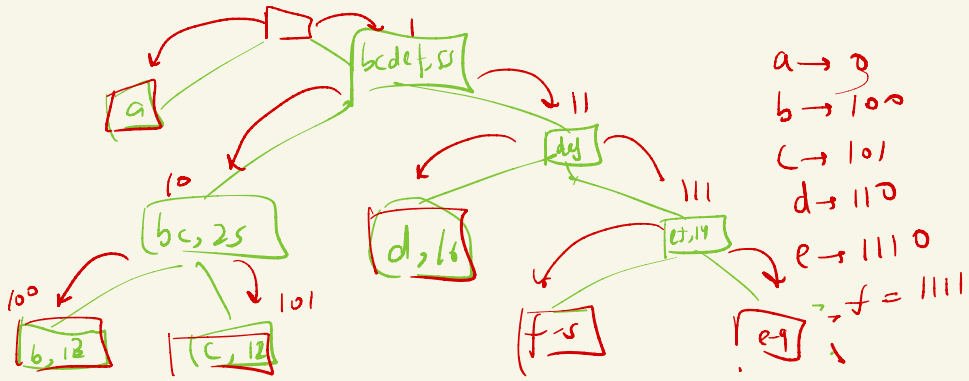
d → 16

e → 9

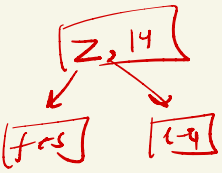
f → 5

priority queue

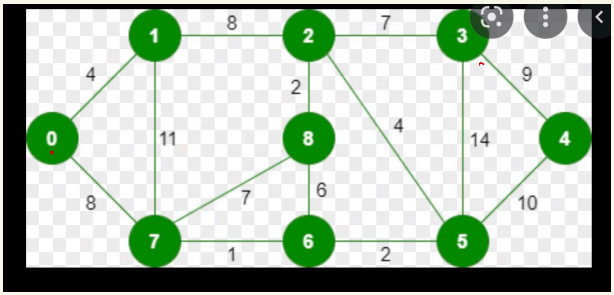




class Node {
char ch
int fq
}



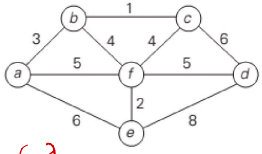
MST



1) connected
n , n-1 edges

spanning tree → connected graph

minimum → sum of weight of edge should be minimum.



$k = \cancel{0}\ 1$

bc	ef	ab	bf	cf	af	df	ae	cd	de
1	2	3	4	4	5	5	6	6	8

ecounter = $\cancel{0}$ 1

while (ecounter < v-1){

E_T = {(bc 1)

3

3

↓ new graph MST

ALGORITHM *Kruskal(G)*

//Kruskal's algorithm for constructing a minimum spanning tree
//Input: A weighted connected graph $G = \langle V, E \rangle$
//Output: E_T, the set of edges composing a minimum spanning tree of G
sort E in nondecreasing order of the edge weights $w(e_{i_1}) \le \cdots \le w(e_{i_{|E|}})$
$E_T \leftarrow \varnothing$; $ecounter \leftarrow 0$ //initialize the set of tree edges and its size
$k \leftarrow 0$ //initialize the number of processed edges
while $ecounter < |V| - 1$ **do**
  $k \leftarrow k + 1$
  if $E_T \cup \{e_{i_k}\}$ is acyclic
    $E_T \leftarrow E_T \cup \{e_{i_k}\}$; $ecounter \leftarrow ecounter + 1$
return E_T

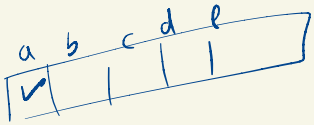
$O(|E| \log |$

O (E + E log E + V²)

O (V² + V² log V² + V²)

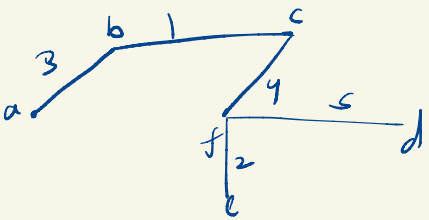
O (2V² log V²)

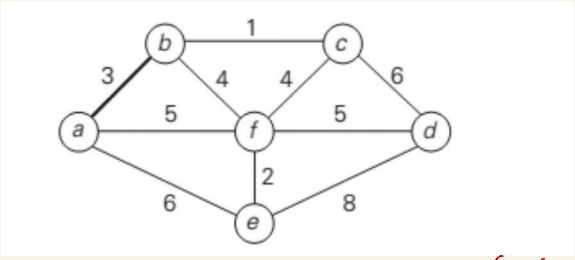
Prims



↓
V, U, W

E_T



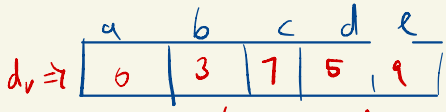


{ , (af, 5), (ae, 6), (ac, ∞), (ad, ∞)}

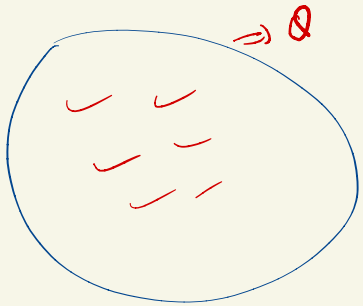
, (cd, 6), (ed, 8) (fd, 5)}

V_T = {a, b}

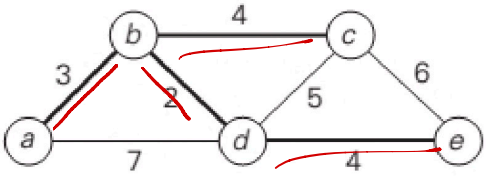
V = {a, b, c, d, e}



E log V



p_v = parent



class Path {
int par
int v
int wsf;
}

copying

400
380
310
280
200

10

110

contending

6

20 → 6
50 → 2
50 → 5
80 → 7
90 → 3
100 → 0
150 → 4
200 → 1

100 → 0
200 → 1
50 → 2
90 → 3
150 → 4
50 → 5
20 → 6
80 → 7

- Given that ,
  - n = 8,
  - i = [1,2,3,4,5,6,7,8],
  - w_i = [100,200, 50, 90, 150, 50, 20, 80]
  - C= 400

cap = 15

p → 10, 5, 15, 7, 6, 18, 3

w → 2, 3, 5, 7, 1, 4, 1

p	w	p/w
10	2	5
5	3	1.67
15	5	3
7	7	1
6	1	6
18	4	4.5
3	1	3

sort acc. to profit

p	w
18	4
15	5
10	2
7	7
6	1
5	3
3	1

47

p	w
3	1
6	1
10	2
5	3
18	4
15	5
7	7

54

6	1
10	2
18	4
3	1
15	5
5	3
7	7

55.33

- Consider a knapsack problem of finding the optimal solution where, M=15, (p1,p2,p3...p7) = (10, 5, 15, 7, 6, 18, 3) and (w1, w2, ..., w7) = (2, 3, 5, 7, 1, 4, 1).
- In order to find the solution, one can follow three different strategies.

3 → 5

1 → 5/3

10/3 × 3.33 5/3 × 2

cap = 3

1	2	3
10	5	13

7, 3, 5 → 3, 5, 7

7 + (7+3) + (7+3+5)

3 3+5 3+(7)

7+10+15 ⇒ 32/3 ⇒

a < b < c a b c

a a+b a+b+c

3a + 2b + c

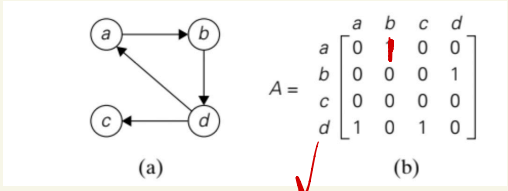
Warshall Algorithm

```
for (int k = a; k <= d; k++) {
    for (int i = a; i <= d; a++) {
        for (int j = a; j <= d; j++) {
```

} R^1, $k = a$ — $k = a$ to b, $i = a$, $j = a$ to d

} R^2, $k = b$

}



R^0

	a	b	c	d
a	0	1	0	0
b	0	0	0	1
c	0	0	0	0
d	1	0	1	0

$R^2 \Rightarrow R^1 = 1$

$R^1(i,k) + R^1(k,j)$

$R^2(i,j) = 1$

$ab = 1 \rightarrow R^1$

$bd = 1 \rightarrow R^1$

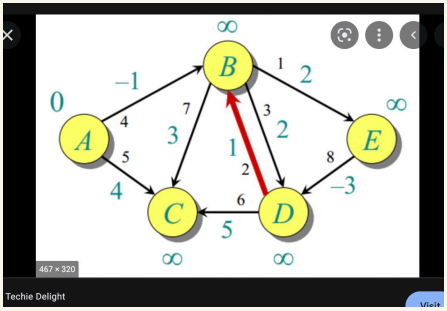
$R^2 \rightarrow R^1(ab)$ and $R^1(bd)$

R^2 ($k = b$)

	a	b	c	d
a	0	1	0	1
b	0	0	0	1
c	0	0	0	0
d	1	0	1	0

$R^0 \rightarrow R^1 \rightarrow R^2 \rightarrow R^3 \rightarrow R^4$

$k = a$ $\quad k = b$ $\quad k = c$ $\quad k = d$



→ A B −1
→ A C 4
→ B C 3
→ B D 2
→ D C 5
→ B E 2
→ E D −3
→ D B 1

A → B
↓
C

0 → (n−1)

(u, v, w)

{ dis(u) + w < dis(v)
dis(v) = dis(u) + w

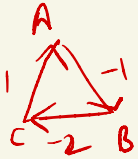
A B −1
dis(A) + w < dis(B)

Src = A

	A	B	C	D	E
wt	A	B	E	B	

E D = −3

	0	1	2	3	4	5
A	0	0	0	0	0	
B	∞	−1	−1	−1	−1	
C	∞	4	2	2	2	
D	∞	∞	1	−2	−2	
E	∞	∞	1	1	1	



	0	1	2	3
A	0	0	0	−2
B	∞	−1	−1	−1
C	∞	∞	−3	−3

→ A B −1
→ B C −2
→ C A 1

−2−1 = −3

−3+1 = −2

Suppose we have started at city ***1*** and after visiting some cities now we are in city ***j***. Hence, this is a partial tour. We certainly need to know ***j***, since this will determine which cities are most convenient to visit next. We also need to know all the cities visited so far, so that we don't repeat any of them. Hence, this is an appropriate sub-problem.

For a subset of cities ***S ∈ {1, 2, 3, ... , n}*** that includes ***1***, and ***j ∈ S***, let ***C(S, j)*** be the length of the shortest path visiting each node in **S** exactly once, starting at ***1*** and ending at ***j***.

When ***|S|*** > 1, we define ***C(S, 1)*** = ∞ since the path cannot start and end at **1**.

Now, let express **C(S, j)** in terms of smaller sub-problems. We need to start at ***1*** and end at **j**. We should select the next city in such a way that

$$C(S,j) = min\ C(S-\{j\},i) + d(i,j)\ where\ i \in S\ and\ i \neq jc(S,j) = minC(s-\{j\},i) + d(i,j)\ where\ i \in S\ and\ i \neq j$$

1 → 3

1 → j

S ∈ {1, 2 ... n}

C(S, j) 1 → j

C(S, j)

→ min { d(i, j) + cost{ S − {j}, i }

1 → 1

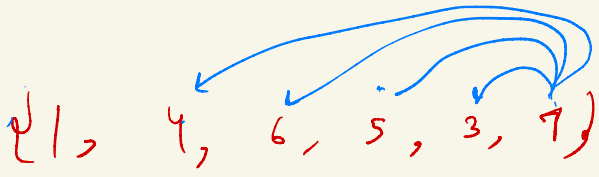
cost

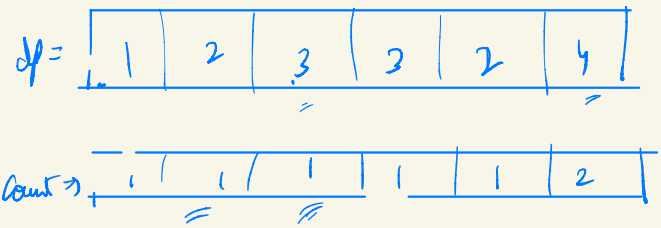
1 {2, 3, 4} 1

d(1, 2) + {3, 4} 1

d(1, 3) + cost(3 {2, 4} 1)

d(1, 4) + cost{ 4 {2, 3} 1





dp(i) = 1

dp(j) + 1 > dp(i) {

dp(i) = dp(j) + 1

count(i) = count(j)

}

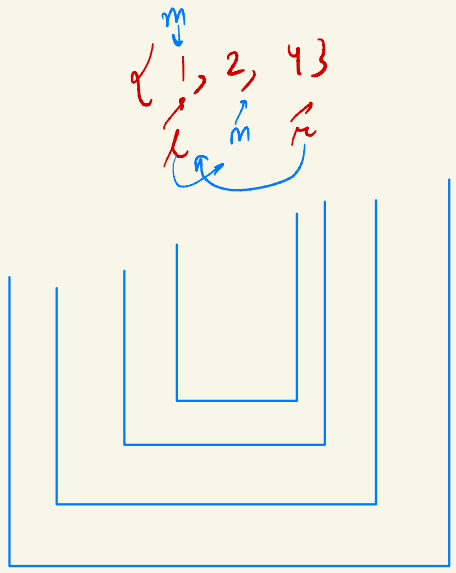
dp(j) + 1 == dp(i)

count(i) += count(j)

(1, 3) (1, 2) (2, 1) (4, 4) (4, 2)

2

2, 3



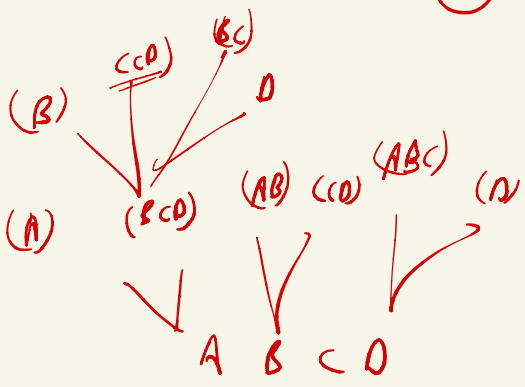
l = m+1

2×3 3×4

2×3×4 →

2, 4, 12, 13, 19

14



(ABCD)

$$C = \begin{array}{c} \begin{matrix} \text{job 1} & \text{job 2} & \text{job 3} & \text{job 4} \end{matrix} \\ \begin{bmatrix} 9 & 2 & 7 & 8 \\ 6 & 4 & 3 & 7 \\ 5 & 8 & 1 & 8 \\ 7 & 6 & 9 & 4 \end{bmatrix} \end{array} \begin{matrix} \text{person } a \\ \text{person } b \\ \text{person } c \\ \text{person } d \end{matrix}$$

1 → $9+3+1+4 = 17$

2 → $2+3+1+4 = 10$

3 → $7+4+5+4 = 20$

4 → $8+3+1+6 = 18$

$a=2$, $b=3$, $b=4$

1 → $6+1+4 = 11$

$3+5+4 = 12$

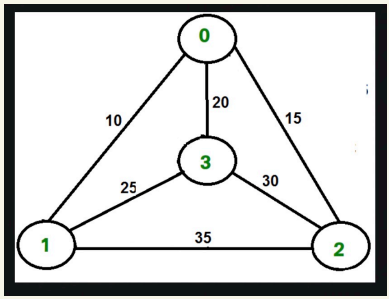
$7+3+7 = 17$

$b=1$

3 → $5+6+2 = 13$

4 → $8+9$

1 → $d=4$ → 5



Node	Least cost edges	Total cost
0	(0, 1), (0, 2)	25
1	(0, 1), (1, 3)	35
2	(0, 2), (2, 3)	45
3	(0, 3), (1, 3)	45

Thus a lower bound on the cost of any tou
1/2(25 + 35 + 45 + 45)
= 75

$10+2+2+3+4 = 21 \rightarrow 25$

$1+0+3+0+0 = 4$

	N_0	N_1	N_2	N_3	N_4
N_0	INF	20	30	**10**	11
N_1	15	INF	16	4	**2**
N_2	3	5	INF	**2**	4
N_3	19	6	18	INF	**3**
N_4	16	**4**	7	16	INF

	N_0	N_1	N_2	N_3	N_4
N_0	∞	10	17	0	1
N_1	12	∞	11	2	0
N_2	0	3	∞	0	2
N_3	15	3	12	∞	0
N_4	11	0	0	12	∞

(0, 1) → $25+10 = 35$

(0, 3) → $25+0+0$

b b b b b

$N_1 \Rightarrow 35$

$N_2 \Rightarrow 53$

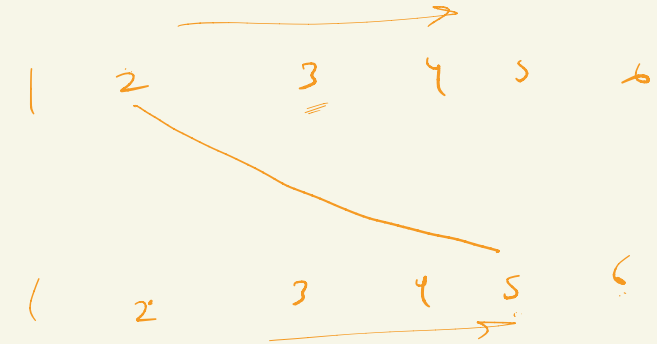
$N_3 \Rightarrow 25$

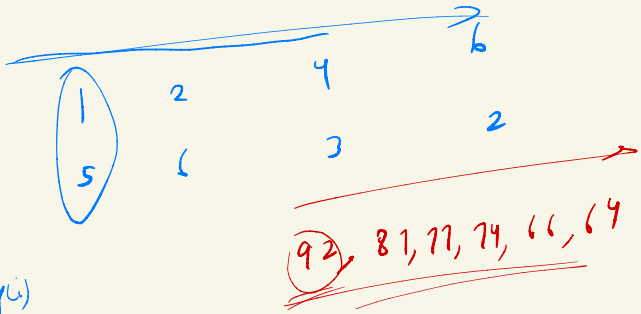
12 ✗ 11 ✗ 0

0 3 ✗ ✗ 2

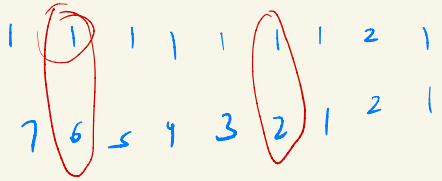
✗ 3 12 ✗ 3

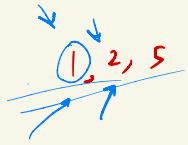
11 0 0 ✗ ✗



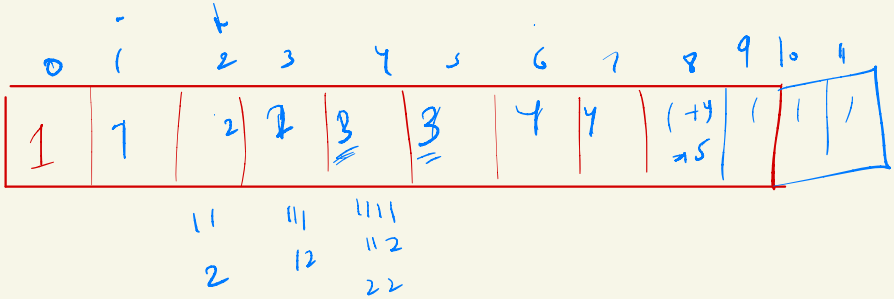


[100,92,89,77,74,66,64,66,64]





answer +1



---

n=5

{4, 2, 3, 1, 3}

A → [4, 2]
B → [2, 3]
C → [3, 1]
D → [1, 3]

(4x3) (3x3)
(AB) (CD)
24 9

24 + 9 + 36

4 x 3 x 3

left + + right

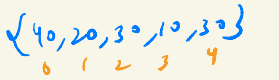
arr(ci) x arr(left) x arr(right)

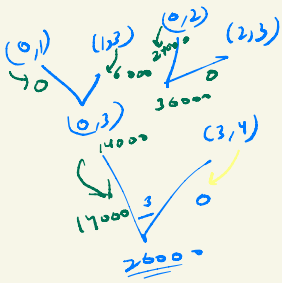
(0,1) (1,2) (2,3) (3,4)
(0,2) (2,4) (0,3)
24 9
(0,1) (1,4) (3,4)
(0,4)
69

A → (0, 1)
B → (1, 2)

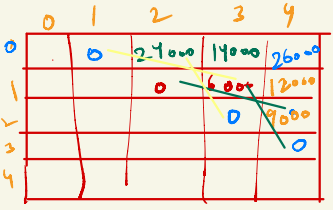
B C
AC D B CD
ABC BCD AB CD ABC
A D
AB CD
24 36

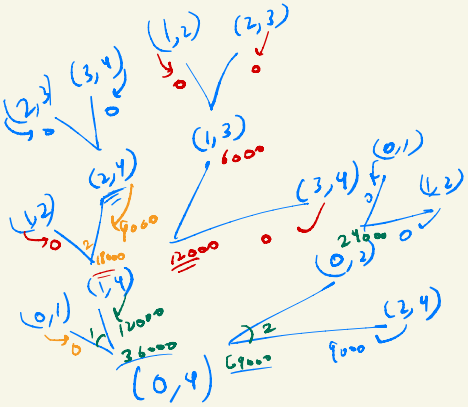
```java
public static int mcm_memo(int[] arr, int si, int ei, int[][] dp){
    if(si+1==ei){
        return dp[si][ei]=0;
    }

    if(dp[si][ei]!=0) return dp[si][ei];

    int ans=(int)(1e9);
    for(int cut=si+1; cut<ei; cut++){
        int left = mcm_memo(arr, si, cut,dp);
        int right= mcm_memo(arr, cut, ei,dp);

        int curr_ans= left + arr[si]*arr[cut]*arr[ei] +right;
        ans=Math.min(ans,curr_ans);
    }

    return dp[si][ei]=ans;
}
```

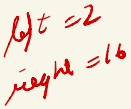


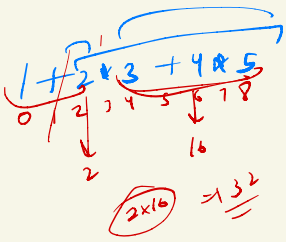


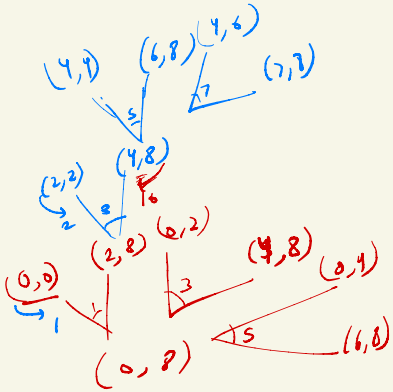

```
0 0 24000 14000 26000
0 0 0 6000 12000
0 0 0 0 9000
0 0 0 0 0
0 0 0 0 0
```







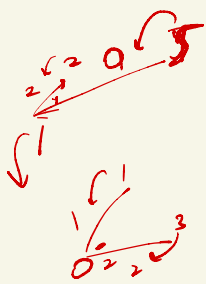


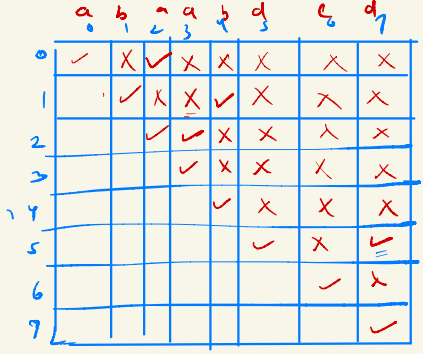


a b a a b d e d

$n^3 \rightarrow n^2$

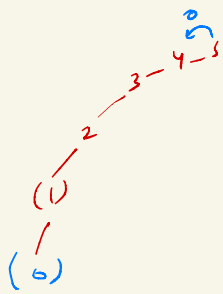
```
public int minCut_memo(String str,int si, boolean[][] isPallindrome, int[] dp){
    if(isPallindrome[si][str.length()-1]){
        return dp[si]=0;
    }

    if(dp[si]!=-1){
        return dp[si];
    }

    int ans=(int)(1e9);
    for(int cut=si; cut<str.length(); cut++){
        if(isPallindrome[si][cut]){
            ans=Math.min(ans,minCut_memo(str,cut+1,isPallindrome,dp)+1);
        }
    }

    return dp[si]=ans;
}
```

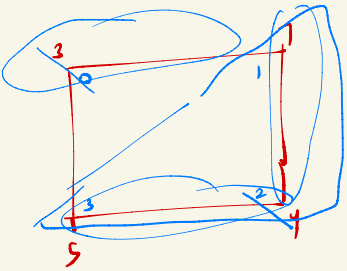




0	1	2	3	4	5	6	7
2	1	2	2	1	0	-1	-1

a b a a b d e d
0 1 2 3 4 5 6 7





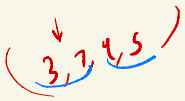
(1)

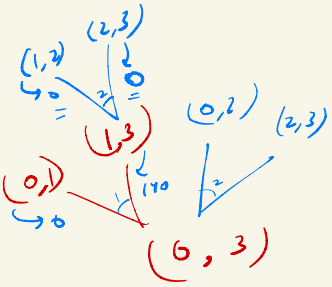
2 points → 1 side

3 points → 2 side

3 X 7 X 5

7 X 5 X 4





0, 1, 1, 1

1, 1, 1, 1

0, 1, 1, 1

1 side → 10

2 side → 4

3 side → 1

→ 15

	0	1	2	3
0	0	1	1	1
1	1	1	2	2
2	0	1	2	3