① # Two pointer

Lec-1 23/10/2020 (Fri)

① Fibonacci
② Total path in matrix using H,V,d motion.
  ① only 1 jump allowed.
  ② ∞ jump allowed.
③ 62.
④ 63.
⑤ 64.

Lec-2 25/10/2020 (Sun)

① goldmine.
② Friend's pairing.
③ min cost climbing stairs (746).
④ 70 / climbing stairs.

Lec-3 26/10/2020 (Mon).

① board path.
② Board path optimization.
③ 90. decode ways.
④ 639. decode way - II.

② # String.

Lec-4 28/10/2020 (Wed)

① Pallindromic substring.
② 647. (Pallindromic substring).
③ 5 (longest pallindromic substring)
④ 516 (longest pallindromic subsequence).
⑤ Print path using back engineering.
  ① Min coin sum path - 64.
  ② Max path shortest path - HVD (1 jump)
  ③ Max path sho " - HVD (∞ jump)

lect-5 30/10/2020 (Fri)

1. 115 distinct subsequence.
2. Count Palindromic subsequence (1143) (gfg).
3. longest common subsequence (1143).
4. 1035 (uncrossed lines).
5. Edit distance.
6. wild card Matching (44).
7. Regular Expression Matching (10).

3 # lect-6 (targetset) 1/11/2020 (Sun).

1. Coin change permutation.
2. Coin change combination.
3. 377
4. 322 (Min coin).
5. 416. (Partition equal subset sum).
6. subset sum problem dp-25.
7. knapsack 0/1
8. knapsack Unbounded (based on ∞ supply of coin).

limited supply of coin combination.

lect-7 2/11/2020 (Mon).

1. 494.
2. find no. of solution of linear eqn of n variable.

4. # ~~Lis~~.

1. Lis. (longest Ping subsequence).
2. lds.
3. lbs. (~~Jump~~). dp-15 gfg. always asked.
4. map sum Ping subseq. dp-14. & lis ~~sum~~ have to check
5. Max sum bitonic Subseq. "
6. min no. of deletion.
7. 673 / No. of longest Ping subseq.

Lect - 7

Stock buy and sell. 3/11/2020 (Tue)

1.
2.
3.
4.
5.
6.

Lect - 8. 4/11/2020 (Wed)

1. Russian doll envelop. (gfg). } bis based
2. Building bridges - gfg

5.# Cut Type

1. MCM rec. (gfg).
2. MCM (dp).
3. print which order of MCM multiplied (string).

Lect - 9. 6/11/2020 (Fri).

1. Max and min. calculation. - gfg
2. Burst balloon. - leetcod (315).
3. Optimal BST - gfg.
4. Min score triangulation (leet).

lect - 10

① boolean parenthesization. (gfg)
② Pallindromic cut (132).
③ 1278| Pallindromic cut-3.

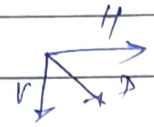
	~~done~~ total	~~not~~ done	~~Total~~ on · not done	H.W.
lec-1	6	✓	✓	① ~~Too points~~
lec-2	4			②
lec-3	4			③
lec-4	5	✓	1 (H.W)	④ 3 on · & string sets
lec-5	7	✓	✓ (	⑤ 1 H.W & target sum ⑤
lec-6	9	7	2	⑥
lec-7	9	6	3 (173, 4, 5 gfg)	⑦ 2 H.W] & stock buy sell
lec-8	5	4	1	⑧] cut type
lec-9	4			⑨

APCO
Date: / /

Lec-1 dp (23/10/2020)

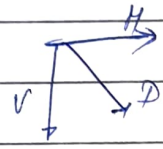
① Fibbonacci series → memoization.
→ tabulation.

②

matrix A: ① 

jump 1

total path to destination.

② 

jump = ∞

③ 62,

④ 63. //

⑤ 64, // Min path Sum.

dynamic programming.

ei ⟶ [State.] ⟶ yi

processing time – 4s. (Any time any where).
if called 'k' times.
total time = 4k s. — ①.

dp dfn. :– If any states is resolved multiple times & उसको answer हर बार same आता है for same arguement तो उसे store करा दो so, that there will be fast retrival, if next time any query occur. Then instead of solving again we retrive from stored value.

different ds for storage.

1. Array
2. HashMap.
3. Tree Map.
4. ll
5. AL

} use any — but array has fast retrival and I/P.

Now have stored any where.
now time taken.

(4 + k) s. time.
↑ preprocess ↑ query.

APCO
Date: / /

Same argument, same answer. use dp.

steps to dp.

① faith
② Tree diagram / Recursion tree } 80% (20min)
③ code.
④ Memoization.
⑤ o/p dp analysis. } 10 + 5 = 15% } 5min.
⑥ Tabulation.
⑦ Further optimization if possible. } 5%

#1 Qu Fibonnacci series.

→ return करने से पहले store into dp at given argument कहीं भी).

Two Pointer.

what :- write programme to calculate nth Fibonacci number.

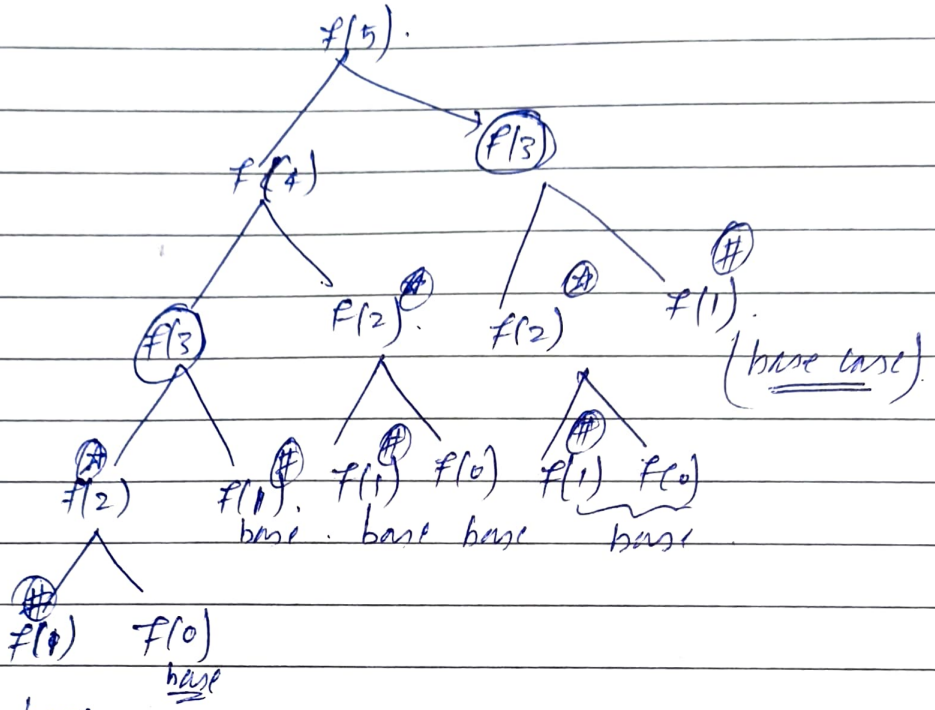
No :- 0 1 1 2 3 5 8
Index :- 0 1 2 3 4 5 6

How :- we know equation.

① think faith → $f(n) = f(n) + f(n-1)$.
② make tree diagram using this eq.
③ Think base case in tree and write code.
④ where ever we get answer store that into dp for given argument.

APCO
Date: / /

(8) Tree digram: for f(5).



→ there is a lot of overlaping subproblem for which we do computation again and again so store that in data structure array

→ Any time when we get requirement / argument retrive from there only

Code Recursion.

```
int fib(n) {
    if (n<=1) {
        return 1;
    }

    int a = fib(n-1);
    int b = fib(n-2);

    return a+b;
}
```

APCO
Date: / /

Memoization

```
fibo1 (int n, vector<int> dp){
    if(n<=1){
        return dp[n] = n;
    }
    if (dp[n] != 0){
        return dp[n];
    }
    int a = fibo1(n-1, dp);
    int b = fibo1(n-2, dp);

    return dp[n] = a+b;
}
```

Tabulation

Replace or get fn call from dp (index).

```
int fibo1_dp(int N, vector<int> & dp){

    for(int n=0; n<=N; n++){
        if (n<=1){
            dp[n] = n;
            continue;
        }
        int a = dp[n-1]; // fibo1(n-1, dp);
        int b = dp[n-2]; // fibo1(n-2, dp);

        dp[n] = a+b;
    }
    return dp[N];
}
```

APCO
Date: / /

Further optimization if possible.

```
int fibo_optim(int N){
    int a=0, b=1;
    for(int n=2; n<=N; n++){
        int sum = a+b;
        a=b;
        b=sum;
    }
    return b;
}
```

Note to dp size is varies [0, n] so. (n+1) size.

0 5

6 size dp will be made.

Complexity

Mem	Tab	optimized
t: O(n)	t: O(n)	t: O(n)
s: O(n)	s: O(n)	s: O(1).

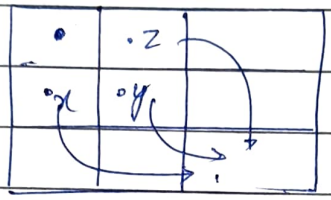
→ Memorization and tabulation have same complexity in [1-D]

→ In 2-d memorization run better than tabulation.

APCO
Date: / /

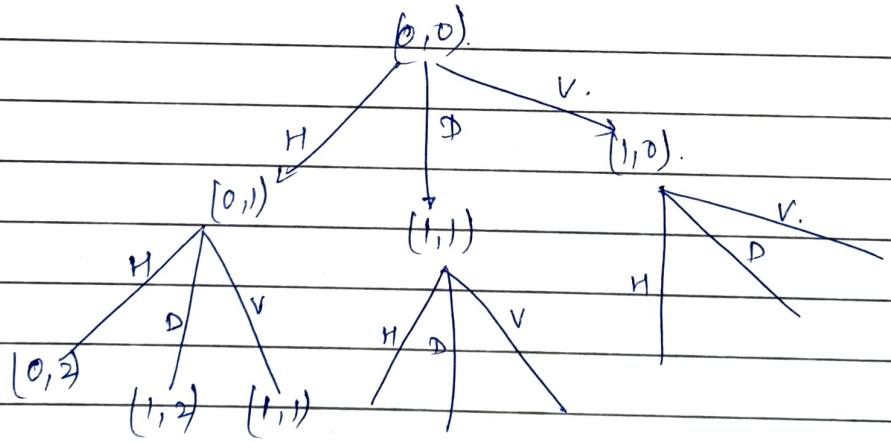
Que:- ① Total path in maze using H, V, d motion.
only 1 jump allowed.

How:- faith:- faith that (0,1), (1,1) (1,0) call give me the count.



Count = x + y + z.

→ Recursive tree



APCO
Date: / /

memoized code:-

```
int mazepath(int sr, int sc, int er, int ec, vector<
    vector<int>>&dp){
    if(sr==er && sc==ec){
        return dp[sr][sc]=1;
    }
    if(dp[sr][sc] !=0)
        return dp[sr][sc];

int count=0;
if(sr+1<=er)
    count += fn(sr+1, sc, er, ec, dp);

if(sc+1 <= ec)
    count += ~~mazepat~~ fn(sr, sc+1, er, ec, dp);

if(sr+1<=er && sc+1<=ec){
    count += fn(sr+1, sc+1, er, ec, dp);
return dp[sr][sc] = count;
}
```

~~dp~~

20	10	4	1
10	6	3	1
4	3	2	1
1	1	1	1

APCO
Date: / /

(2) Maze path with Multiple jump

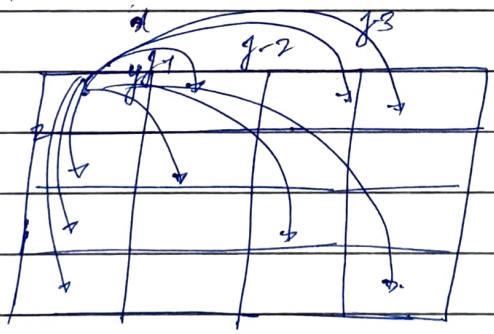
How :- same as maze path with one jump just run here loop of jump till sr + jump < er, ec. respectively.

Cell का meaning = faith of dp. (A)

Memoization का 1 cell का meaning = faith of Recursion होता है

faith :-

I will get path of all jumps from sr. in all direction.



count = x + y + z.

APCO
Date: / /

Code:-

```
int count = 0.
for (int jump = 1; sr + jump <= er; jump++){
        count += fn(sr+jump, sc, er, ec, dp)
}

for( int jump=1; sc+jump <= ec}; jump++)
        count += fn( sr, sc+jump, er, ec, dp);

for (int jump=1; sc+jump<ec && sr+1<= er ; jump++){
        count += fn( sr+jump, sc+jump, er, ec}, dp);

return count;
```

APCO
Date: / /

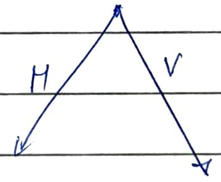
Que 64 LeetCode:

what :- given m×n grid filled with non-negative numbers.
Find path from left to bottom right which minimizes sum of all number along its path

1	3	1
1	5	1
4	2	1

V, H → motion allowed

Faith :- मेरा answer जो है (h and v) का min होगा.



min(H, V), self)

APCO
Date: / /

But dp ~~of~~ map value = (1e9).

// Code

```
int minPathSum(sr, sc, er, ec, grid, dp){   // value of cell
    if(sr == er && sc == ec)
        return grid[sr][sc] = grid[sr][sc];

    if(dp[sr][sc] != 1e8){
        return dp[sr][sc];

    if(sr+1 <= er)
        dp[sr][sc] = min(dp[sr][sc], fn(sr+1, sc, er
                ec, grid, dp);
    if(sc+1 <= ec)
        dp[sr][sc] = min(dp[sr][sc], fn(sr, sc+1,
                er, ec, grid, dp);

    return dp[sr][sc] = grid[sr][sc];

}
```

modulus Property

① $(A+B)\%C = (a\%C + b\%C)\%C$.

② $(A-B)\%C = (a\%C - b\%C + C)\%C$.

③ $(A*B)\%C = (a\%C * b\%C)\%C$.

used ans

AP

Date:

lecture-2 (dp). (25-10-2020).

1. goldmine.
2. goldmine-II //1219
3. ~~746~~ // Min cost climbing stairs.
4. Friends pairing.

Date:

lecture - 3 (dap) 26/10/2020.

Agenda:

1. Board path.
2. Board path optimization.
3. 91 decode ways.
4. 639. decode ways. 2.

Date:

Lecture - 4. (dp). (28/10/2022)

Agenda:

1. Palindromic substring.
2. 647. palindromic substring Count
3. 005. (longest palindromic substring).
4. 516. longest palindromic subsequence.
5. Print path. using back engineering.
   1. ~~53~~. Min ~~cost~~ Coin Sum path (64) leetcode.
   2. maze path shortest path → HVD. (one jump).
   3. maze path shortest path → multi jump.

APCO
Date: / /

① Pallindromic Substring :-

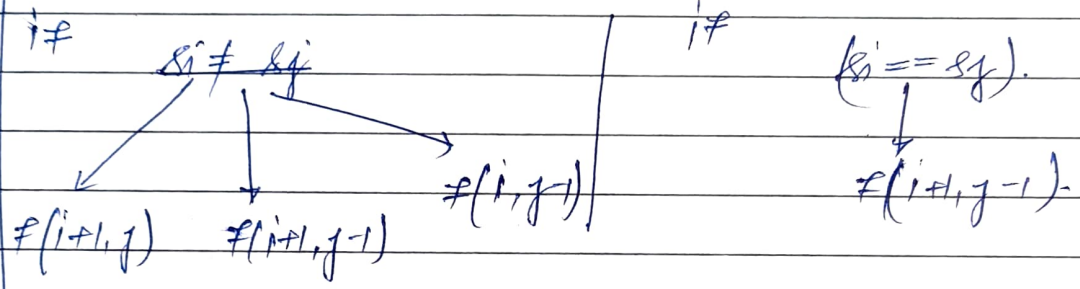
what :- I/P =

a	b	c	b	d	d	b	e	f	g
0	1	2	3	4	5	6	7	8	9

We'll have index range and have to tell whether string is pallindrome / not.

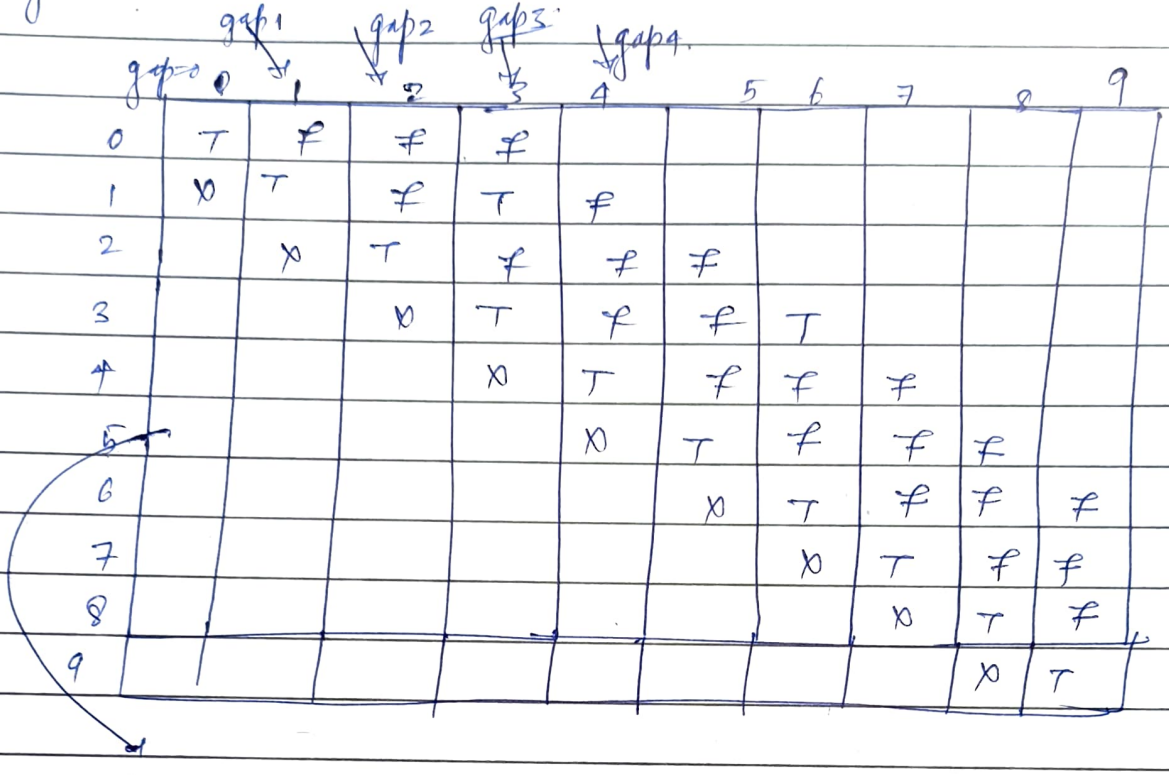
How :-

Recursive Calls.



APCO
Date: / /

By Tabulation -



Invalid :- because (3,0) does not exist in string (0,9)

→ gap=0 or length=1 will always be pallindrome.

eg1 s(i,j). will be pallindrome.
if (si == sj).

s(i+1, j-1) is also pallindrome.

eg2 bcb (1,3) (will be pallindrome.
c (2,3) ✓

Similarly take any range and check for that

APCO
Date: / /

How gap loop work.

gap=0	1	2
0,0	0,1	0,2
1,1	1,2	1,3
2,2	2,3	2,4
3,3	3,4	3,5
4,4		

→ outer loop for gap and
→ inner loop to change (i,j).

// Code :-

```
Public static void pallindromic substring(String str, boolean[][]
    dp){
    int n = str.length();
    for(int gap=0; gap<n; gap++){
        for(int i=0; j=gap; i<n; gap++, j++){

            if(gap==0) dp[i][j] = true;
            else if(gap==1) dp[i][j] = str.charAt(i)
                                == str.charAt(j);
            else dp[i][j] = str.charAt(i) ==
                str.charAt(j) && dp[i+1][j-1];

        }
    }
}
```

APCO
Date: / /

Que 647 Pallindromic substring count

What? given a string
"abc".

o/p = 3.

count all possible pallindromic substring.
'a', 'b', 'c'
= 3.

How :-

Code is same a previous just maintain a count and ↑ment when we found dp[i][j] = true.

// Code :-

```
Public int countSubstrings(String s){
    int n = str.length();
    boolean[][] dp = new boolean[n][n];
    int count=0;
    for(int gap=0; gap<n; gap++){
        for(int i=0, j=gap; j<n; i++, j++){
            if(gap==0) dp[i][j] = true;
            else if(gap==1)
                dp[i][j] = str.charAt(i)==str.charAt(j)

            else
            {dp[i][j] = str.charAt(i)==str.charAt(j)
                && dp[i+1][j-1]; }
            if(dp[i][j]) count++;
        }
    }
    return count;
}
```

APCO
Date: / /

(0, 2) is not a palindrome so,
we can't expect answer to that

'abc'
0 1 2

	0	1	2
0	T	F	F
1	X	T	F
2	X	X	T

→ ~~Index~~ cell represent whether (0, 2) is palindrome / not.

So, we can't expect answer from the given cell.

APCO
Date: / /

Que 005 (Longest pallindromic Substring).

what? given a string 's'
return longest pallindromic substring in s.

How:- s = "babad"
o/p = "bab"
or (aba.)

app:- we'll do here the same thing diff.
① make dp of int and store dp[i][j]=1 at gap=0.
② 2 for gap=1 and s(i)==s(j).
③ if (s(i) = s(j) at any other point the.
dp[i][j] = 2 + dp[i+1][j-1].
④ We'll maintain a count/length to make longest and update it.

Code-

APCO
Date: / /

Avoid(it's recursive solution).

Code:

```
Public string longestPalindrome(String str){
    int n = str.length();
    int[][] dp = new int[n][n];

    int si=0, ei=0, length=0; // starting index,
// end index of longest palindromic substring.

For(int gap=0; gap<n; gap++){
    for(int i=0; j=gap; j<n; i++, j++){

    if(gap==0) dp[i][j] = 1;
    else if(gap==1 && s(i)==s(j)) dp[i][j] = 2;
    else if( s(i)==s(j) && dp[i+1][j-1]>0){
      dp[i][j] = dp[i+1][j-1]+2;

    if( dp[i][j] > length){
        length = dp[i][j];
        si = i;
        ei = j;

    }

}

}
return str.substring(si, ei+1);

}
```

v. v Imp.
always do tabul(in substring). it is used always so remember.

Strings :- ① Recursive.
② Tabulation

APCO
Date: / /

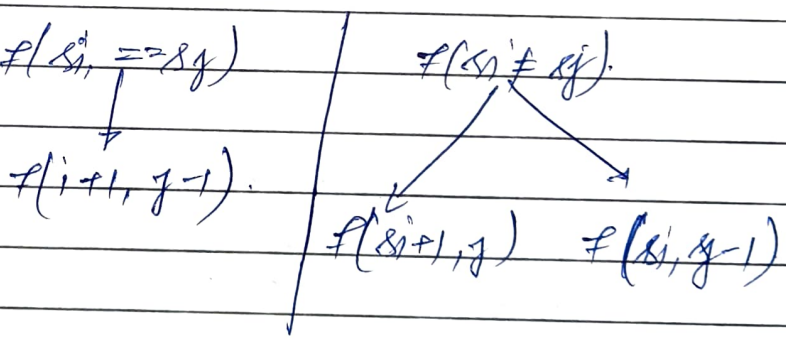
Note :-
whenever there is comprision of index (i, j).
i → ← j
then they go in a in for (for) loop
→ if we write recursive app in for loop the that work guaranteed.

Que 516. Longest Pallindromic Subsequence.

① Recursive solution.

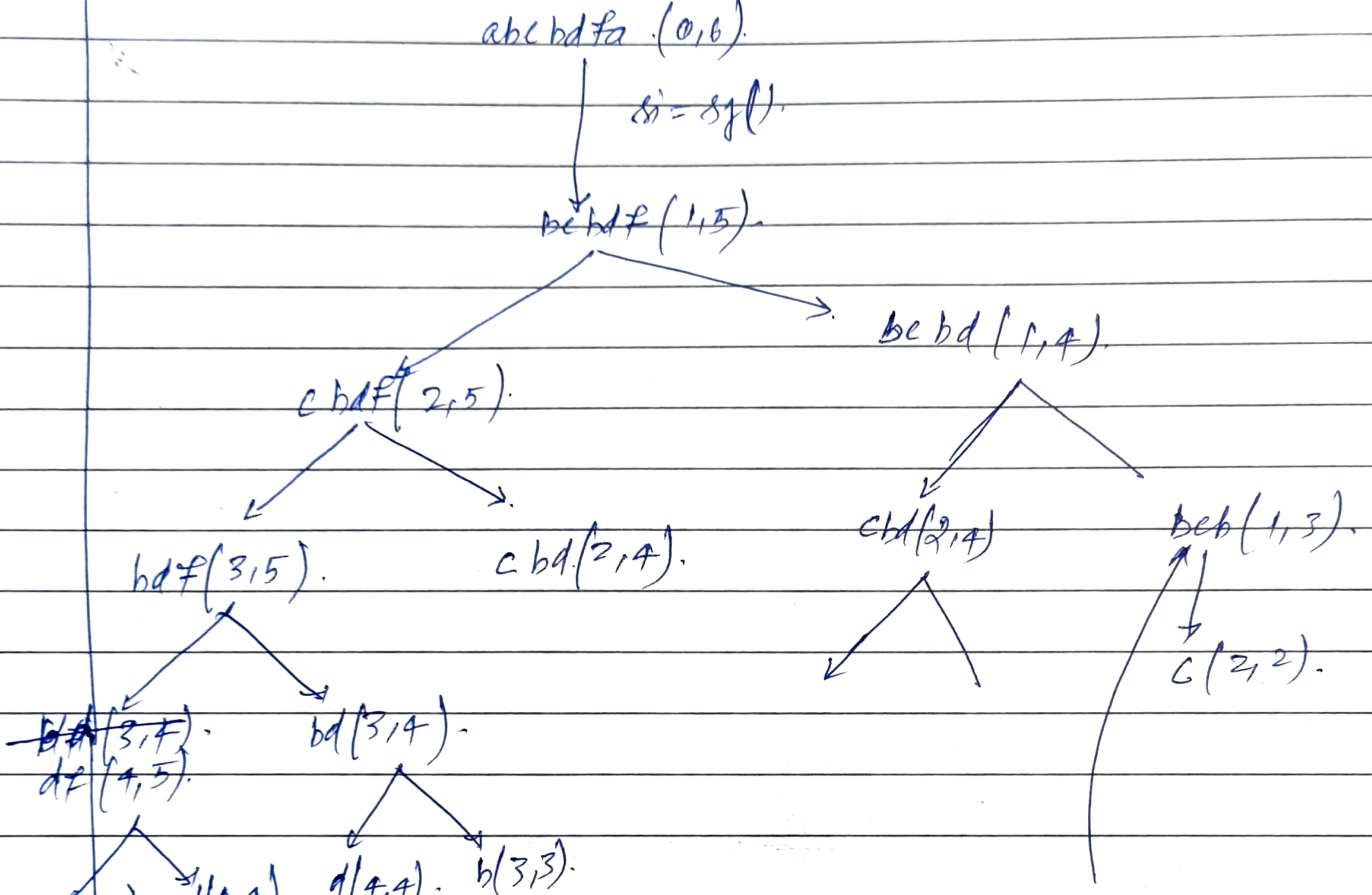
I/P = abcbdfa.
o/p = 5. (abcba).

How :-



APCO
Date: / /

Rec Tree

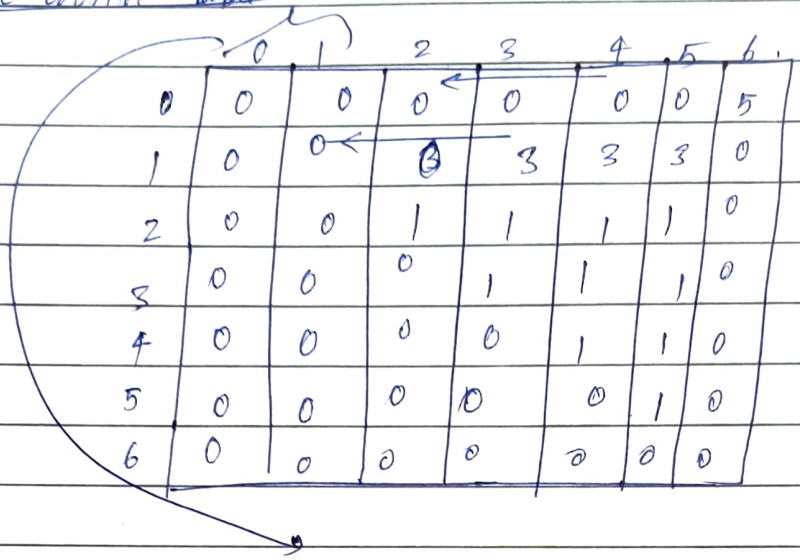


(1,3) state solve for 3 length and do not check for '1' so that cell remain vacant

→ ∴ here recursive work better than tabulation (∵ here need to solve for all states).

APCO
Date: / /

Recursive table:

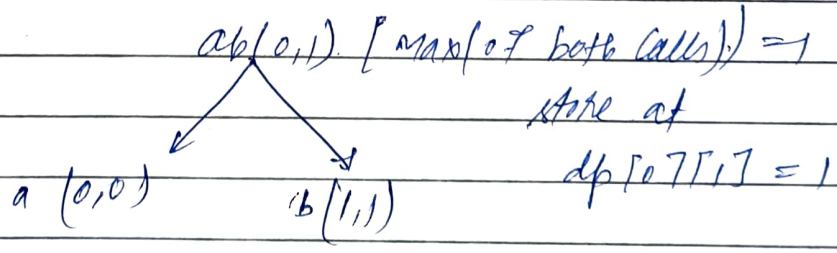


	0	1	2	3	4	5	6
0	0	0	0	0	0	0	5
1	0	0	0	3	3	3	0
2	0	0	1	1	1	1	0
3	0	0	0	1	1	1	0
4	0	0	0	0	1	1	0
5	0	0	0	0	0	1	0
6	0	0	0	0	0	0	0

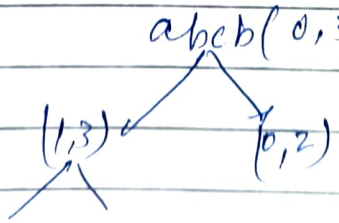
Here call never comes b
∵ s(0) = s(6). get ans and store at dp[0][6] so.
no need to go for lower values.

Similarly s(1,3) = 3
and not go for lower value.

① 0 length pallindrome check



Similarly check for other value like



APCO
Date: / /

Code:

```
Public int longestPallindromicSubseq(String s, int i, int j, int[][] dp){
    if(i>j || i==j)
        return dp[i][j] = (i==j) ? 1 : 0;

    if(dp[i][j] != 0) return dp[i][j];

    if(s.charAt(i) == s.charAt(j))
        dp[i][j] = fn(s, i+1, j, dp)+2;

    else {
        dp[i][j] = max(fn(s, i+1, j, dp), fn(s, i, j-1, dp));

        return dp[i][j];
    }
```

Note:- If have to tell string of max length. then store a pair of <int, String> at dp index.
which will tell max possible length of subseq and given subsequence at that index.

APCO
Date: / /

M-2 Tabulation.

0	1	2	3	4	5	6
a	b	c	b	d	f	a

i \ j	0	1	2	3	4	5	6
0	1	1	1	3	3	3	5
1	x	1	1	3	3	3	3
2	x	x	1	1	1	1	1
3	x	x	x	1	1	1	1
4	x	x	x	x	1	1	1
5	x	x	x	x	x	1	1
6	x	x	x	x	x	x	1

else

1

gap = 0

Note :-

Here we need to fill all the state while in memoization not need to fill all state so, that work little faster than ~~tabul~~ tabulation.

→ ~~Rem~~ leaving faith / state of dp is same in both d cases.

// Code :

```
Public int longestPallindromicSubseqDp(String s, int I,
    int J, int [][] dp){
    int n = s.length();
    String[][] sdp = new String[n][n]; //
    // save string at that state of dp.
    for(String[] d : sdp) Arrays.fill(d, "");
```

APCO
Date: / /

```
for(int gap=0; gap<n; gap++){
    for(int i=0, j=gap; j<n; j++,i++){

        if(i==j){
            dp[i][j] = (i==j)? 1:0;
            sdp[i][j] = s.charAt(i)+"";
            continue;
        }

        if(s.charAt(i) == s.charAt(j)){
            dp[i][j] = dp[i+1][j-1] + 2;
            sdp[i][j] = s.charAt(i)+ sdp[i+1][j-1]+
                        s.charAt(i);
        }
        else{
            if(dp[i+1][j] > dp[i][j-1]){
                dp[i][j] = dp[i+1][j];
                sdp[i][j] = sdp[i+1][j];
            }
            else{
                dp[i][j] = dp[i][j-1];
                sdp[i][j] = dp[i][j-1];
            }
        }
    } // inner for
} // outer
return dp[0][n];
}
```

APCO
Date: / /

// Back Engineering to print path of dp.

What:- to make string from given dp of max length.

How:- do same call as done in recursion just add char of that particular index in 'ans' string.

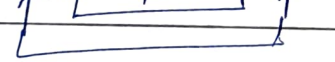
Table.

if have to find all ans (both Call).

	0	1	2	3	4	5	6
0	1	0	0	3	3	3	5 (i==j)
1	x	1	1	3	3	3	0 3
2	x	x	1	1	1	1	0 1
3	x	x	x	1	1	1	0 1
4	x	x	x	x	1	1	0 1
5	x	x	x	x	x	1	0 1
6	x	x	x	x	x	x	0 1

a	b	c	b	a	f	a
0	1	2	3	4	5	6

ans = a b c b a



APCO
Date: / /

① This ans can be obtain using ① string. $O(n^2)$.
② linked list $O(n)$.

n changes ($\frac{n}{2}$ times)

mean string operation
$O(n)$ complexity.

// code-

```
Public static void generateString(int [][] dp, String s, int i,
        int j, String ls, String rs){

    if(i>j || i==j){
        if(i==j){
            ls += s.charAt(i);
        }
        System.out.println(ls+rs);
        return;
    }

    if(s.charAt(i) == s.charAt(j)){
        fn(dp, s, i+1, j-1, s.charAt(i)+ ls, rs+s.charAt(i));
    }

    else if(dp[i+1][j] < dp[i][j-1])
        fn(dp, s, i, j-1, ls, rs);

    else{
        fn(dp, s, i, j-1, ls, rs);
        fn(dp, s, i+1, j, ls, rs);
    }
}
```

avoid this when one string ends.

APCO
Date: / /

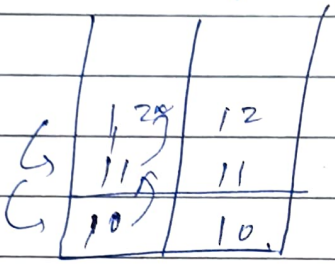
② Using LinkedList

// Code:

```
void generate fn(int[][] dp, String s, int i, int j, LL<Character> ls, LL<Ch
        rs){
    if(i>j || i==j){
        if(i==j){
            ls.addLast(s.charAt(i));
        }
        sysO(ls);
        sysO(rs);
        sysOln();

        if(i==j)
            ls.removeLast();

        return;
    }

    if(s(i) = s(j)){
        ls.addLast(s(i));
        rs.addLast(s(j));
        fn(dp, s, i+1, j-1, ls, rs);
        ls.removeLast();
        rs.removeLast();
    }
    else{
        if(dp[i+1][j] > dp[i][j-1]){ fn(dp, s, i+1, j, ls,
        else if(      < ) fn(dp, s, i, j-1, ls, rs);
        else{
            fn(dp, s, i, j-1, ls, rs);
            fn(dp, s, i+1, j
        }
    }
```

APCO
Date: / /

① In string we do not need to remove in call.
∴
String is formed on stack and every call copy pass parameters which create a new copy and changes are reflected there only not the base one.



Stack.

② while U is formed on heap and changes are reflected in heap so we have to remove them while coming back in recursion.

AP
Date: /

Lect - 5 (dp)

30/10/2020.

Agenda:

1. 115 (distinct subsequence).
2. Count palindromic subsequence (gfg).
3. Longest common subsequence.
4. 1035. Uncrossed lines.
5. Edit distance.
6. Wild card Matching (44)
7. Regex check.

APCO
Date: / /

Que. 115. distinct subsequence.

what:- count distinct subsequence of target string in . src string.

src target
I/P geeksforgeeks/ gks

How:-

for any index (n,m) if.

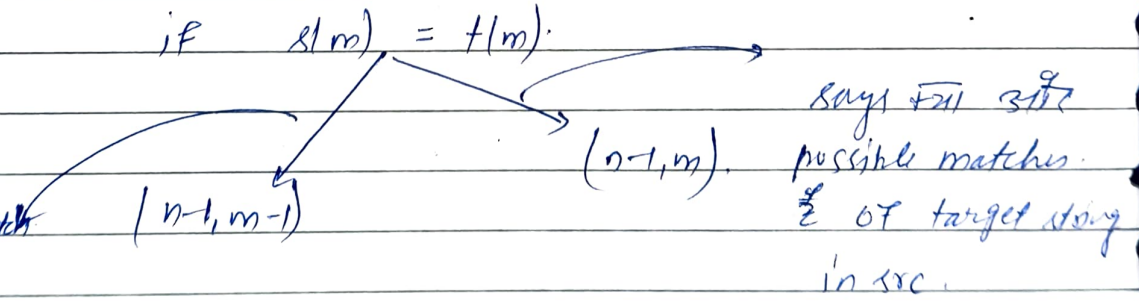
① s(n) = t(m) matches then

1.1 → find rem by eliminating them.
f(n-1, m-1).

1.2 → ~~Si~~ expect for chances to get same target in cutted subsequence.
f(n-1, m).
↑ ont ↑ Unint.

② if s(n) ≠ t(m).
pass by cutting nth and mth characters.
f(n-1, m-1).

C-1



APCO
Date: / /

C-2

(n, m)

s(n) ≠ s(m).

→ (eliminating character of src)

(n-1, m)

Que. (5, 3) tell what represent

Ans 5 length के subsequence by 3 length का target कितने तरीके से मिल रहा है.

Faith :-

m length का target 'src' में कितने तरीकों से मिल रहा है.

Note :- ① do dry run from आगे से eg.

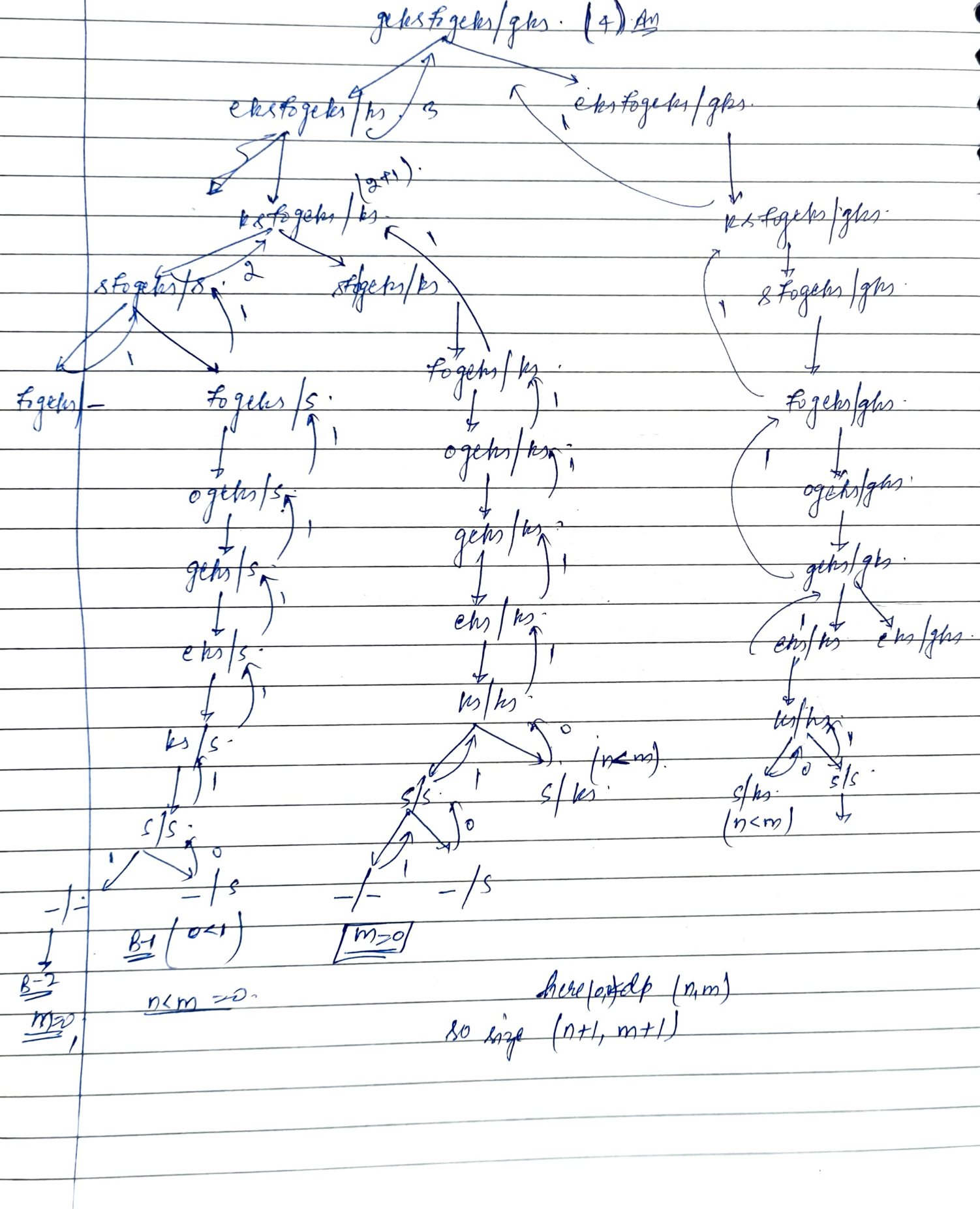
geksforgeks | gks.

eksforgeks | ks. eksforgeks | gks.

② But code from पीछे से because of Convenience because from आगे से have to maintain index as well as length.

APCO
Date: / /

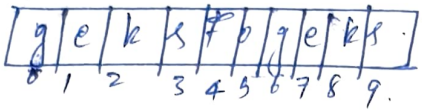
Recursive - tree



APCO
Date: / /

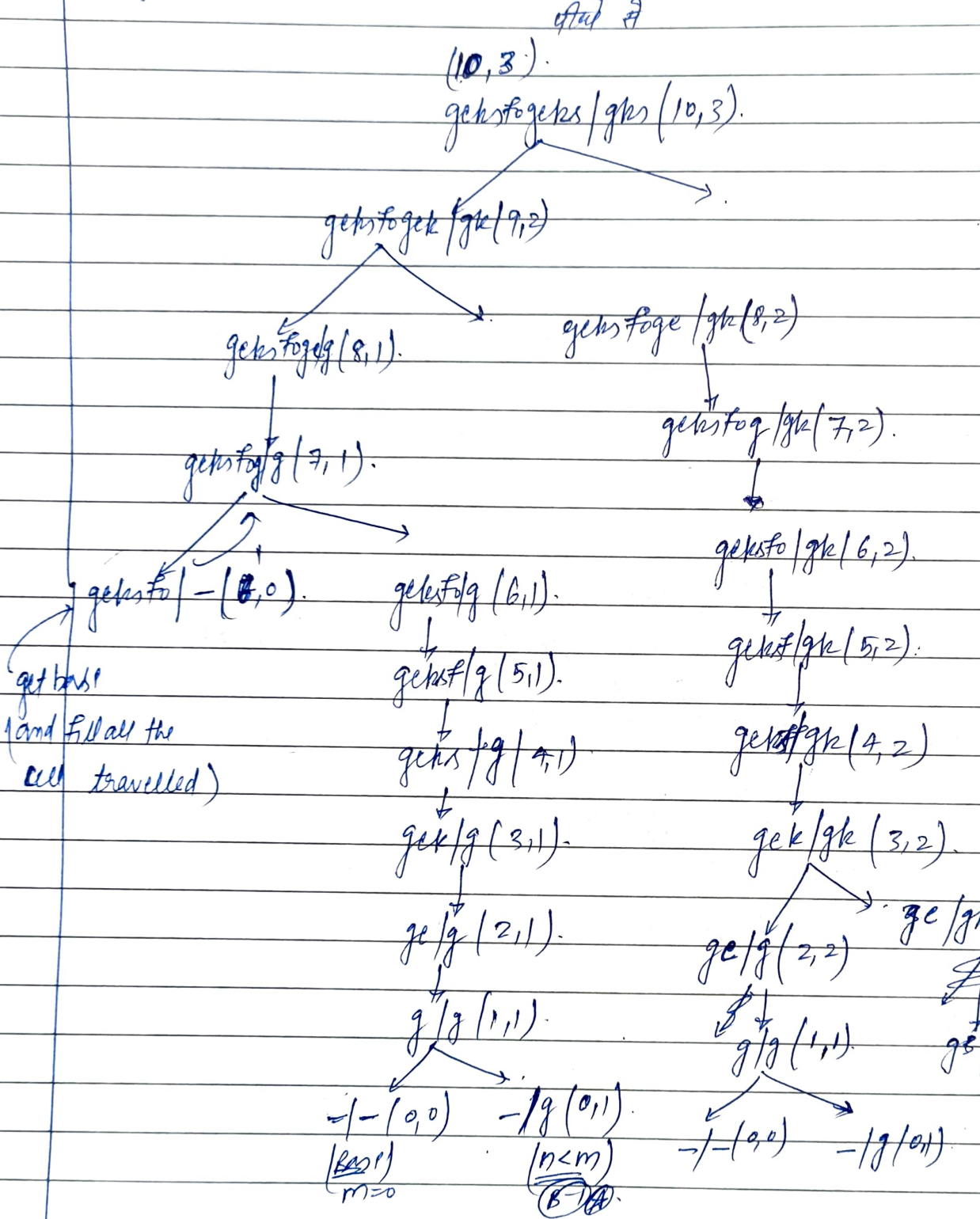
// Code.

```
Public int numDistinct (String s, String t, int n, int m, int[][] dp
    if (m == 0) {
        return dp[n][m] = 1;
    }
    if (n < m) {
        return dp[n][m] = 0;
    }

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

    int count = 0;

    if (s.(n-1) == t(m-1)) {
        count += fn(s, t, n-1, m-1, dp);
        count += fn(s, t, n-1, m, dp);
    }
    else {
        count += ~~num~~ fn(s, t, n-1, m, dp);
    }

    return dp[n][m] = count;

}
```



APCO
Date: / /

→ Using tabulation.



APCO
Date: / /

"" \ ""	0	1	2	3
0	1	0	0	0
1		1		
2		1		
3		1		
4		1		
5	1	1		
6	1	1		
7				
8				
9				
10				

→ Here Recursion travel faster then tabulation
∴ In recursion all index are not travelled only required are travelled
→ In tabulation all cell are visited so take time.

APCO
Date: / /

// Code:

```
Public int numDistinctDp(string s, string t, int N, int M, int[][] dp){
    for(int n=0; n<=N; n++){
        for(int m=0; m<=M; m++){
            if(m==0){
                dp[n][m] = 1;
                continue;
            }
            if(n<m){
                dp[n][m] = 0;
                continue;
            }
            int count=0;
            if(s[n-1] == t[m-1]){
                count += dp[n-1][m-1];
                count += dp[n-1][m];
            }
            else{
                count += dp[n-1][m];
            }
            dp[n][m] = count;
        }
    }
    return dp[N][M];
}
```

APCO
Date: / /

2. Count pallindromic Subsequence (gfg).

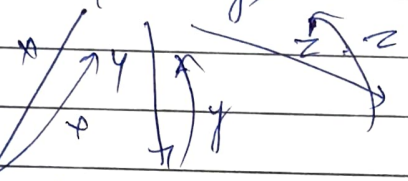
What :- Find all total pallindromic count of string

How :- faith :-

f(0, n-1) fn (0, n-1) के लिए सभी pallindromic count करके दे देगा.

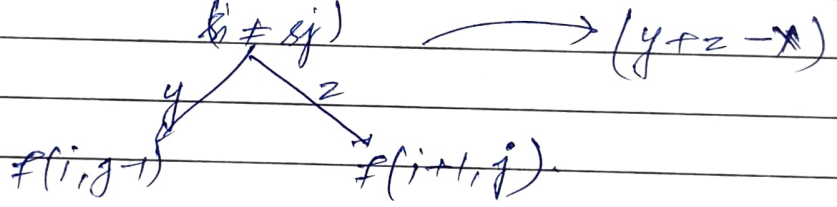
c-1

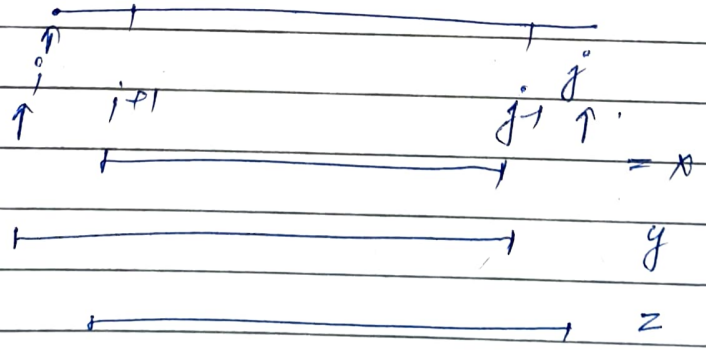
if (si = sj). (x+1+y+z-x).



f(i+1, j-1) f(i, j-1) f(i+1, j).

c-2.





if (equal) = 1 + x + y + z - x

else y + z - x

eg = | a | b | c | c | b | a | d | a | X |
0 1 2 3 4 5 6 7 8.

C-1

abccbada
f(i+1, j-1). j

x =

$a_0 a_7$ + { $b_1, c_2, c_3, b_4, d_5, a_6$; $b_1 b_4, c_2 c_3$; $b_1 c_2 b_4, b_1 c_3 b_4$; $b_1 c_2 c_3 b_4$ }

a_0 { $b_1, c_2, c_3, b_4, d_5, a_6$; $b_1 b_4, c_2 c_3$; $b_1 c_2 b_4, b_1 c_3 b_4$; $b_1 c_2 c_3 b_4$ } a_7.

C-2.

f(i, j-1).

abccbad a_7

y {
$a_0, b_1, c_2, c_3, b_4, d_5, a_6$ — ①
$a_0 a_6, b_1 b_4, c_2 c_3$. — ②
$a_0 b_1 a_6, a_0 c_2 a_6, a_0 c_3 a_6, a_0 a_4 a_6, a_0 d_5 a_6$; $b_1 c_2 b_4, b_1 c_3 b_4$ — ③
$a_0 b_1 b_4 a_3, a_0 c_2 c_3 a_5, b_1 c_2 c_3 b_4$ — ④
$a_0 b_1 c_4 a_5, a_0 b_1 c_3 b_4 a_5$ — ⑤
$a_0 b_1 c_2 b_4 a_5, a_0 b_1 c_3 b_4 a_5$ — ⑥
}

C-3.

abccbada

f(i+1, j)

$b_1, c_2, c_3, b_4, a_5, d_6, a_7$ — ①
$b_1 b_4, c_2 c_3, a_5 a_7$ — ②
$b_1 c_2 b_4, b_1 c_3 b_4, a_5 d_6 a_7$ — ③
$b_1 c_2 c_3 b_4$ — ④

all underline element are common in C-2, C-3 in x no of quantity so.
nett $2x - x = x$

$x = f(i+1, j-1)$
$y = f(i, j-1)$.
$z = f(i+1, j)$

APCO
Date: / /

// Code:

```
Public int countPS(string str, int i, int j, int[][] dp){
    if(i>=j){
        return dp[][] = (i==j)?1:0;
    }

    if(dp[i][j] != 0) return dp[i][j];

    int x = fn(str, i+1, j-1, dp);
    int y = fn(str, i, j-1, dp);
    int z = fn(str, i+1, j, dp);

    if(str(i) == str(j))
        dp[i][j] = (x+1) + (y+z-x);

    else dp[i][j] = y+z-x;

    return dp[i][j];
}
```

Complexity :- As here gap strategy is followed ($O(n^2)$)

$\therefore$ it comp = $O(n^2)$.

APCO
Date: / /

Using dp:-

cell of dp(1,2) represent total count of pallindrome in range (1,5) and store at (1,3) cell.

Code:-

```
Public int CountPS_DP (string str, int I, int J, int[][] dp) {

    for ( int gap=0 ; gap < str.length() ; gap++) {
        for ( int i=0, j=gap ; j < str.length() ; i++, j++) {

            if (i>=j) {
                dp[i][j] = (i==j) ? 1 : 0 ;
                continue ;
            }
            int x = dp[i+1][j-1] ; // fn(str, i+1, j-1, dp);
            int y = dp[i][j-1] ; // fn(str, i, j-1, dp);
            int z = dp[i+1][j] ; // fn(str, i+1, j, dp);

            if ( str(i) == str(j))
                dp[i][j] = (x+1) + y + z - x
            else
                dp[i][j] = y + z - x ;
        }
    }
    return dp[I][J];
}
```

T: O(n²) complexity.

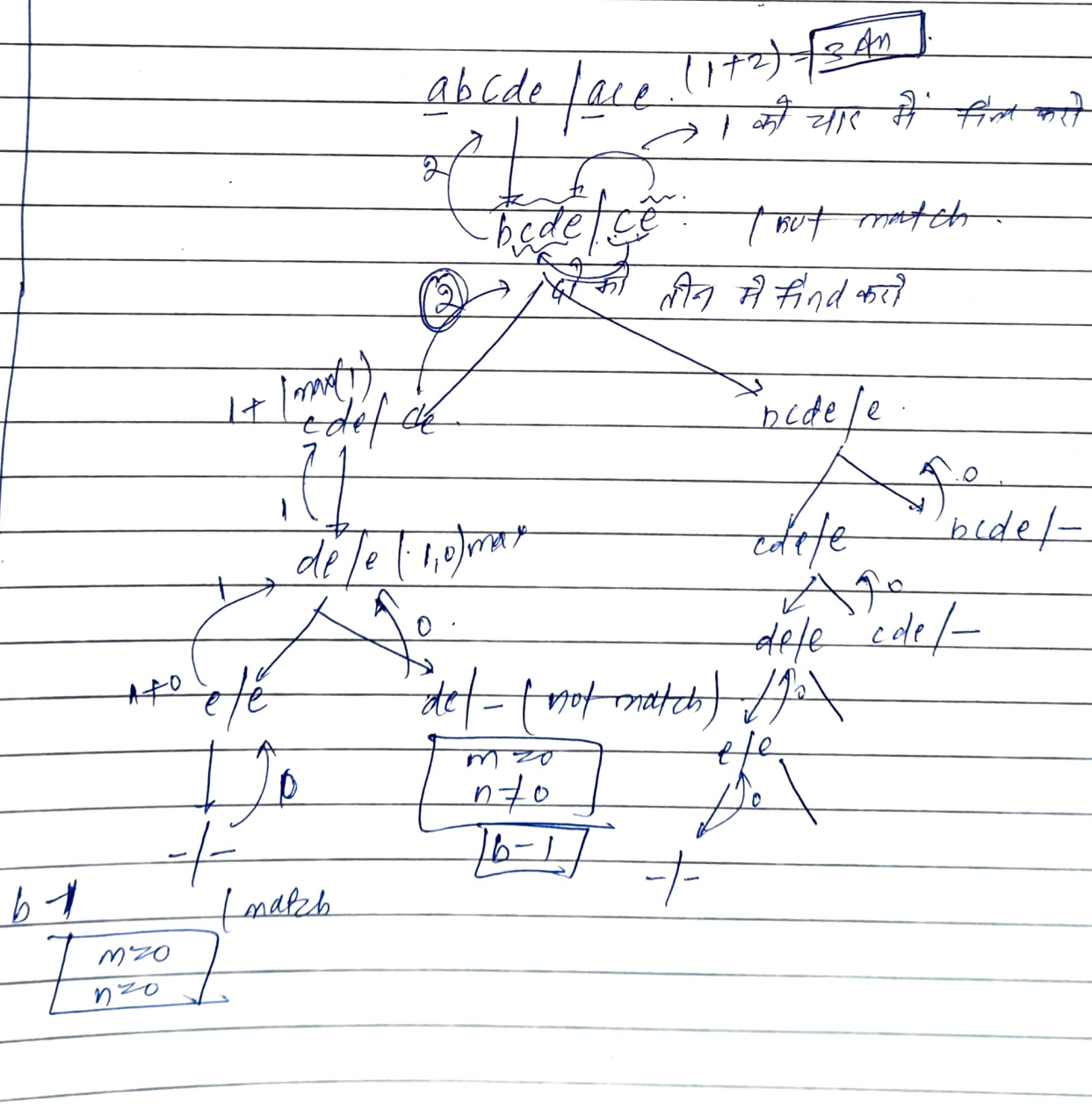
APCO
Date: / /

3. Q1143 (Longest Common Subsequence.

What :- given two string text1 and text2 return length of longest common subsequence.

How :- if character matches: then make call of
1 + f(n-1, m-1)

else else. एक बार target string को काट कर आओ
& एक बार src को eg.



APCO
Date: / /

// Code :

```
Public int ~~longest common~~ fn( String s1, String s2, int n, int m,
                                  int [][] dp) {.

    if(n==0 || m==0). return 0.

    if( dp(n)(m) ≠ -1). return dp(n)(m);.

    if( s1(n-1) == s2(m-1))
        dp(n)(m) = fn(s1,s2, n-1, m-1, dp) +1;.

    else
        dp(n)(m) = max( fn(s1, s2, n, m-1, dp) , fn(s1, s2,
                                  n-1, m, dp));.

    return dp(n)(m);.

}
```

qa6

APCO
Date: / /

Using dp:

```
for( string s1, string s2, int N, int M, int [][]dp){
    for( int n=0; n<=N; n++){
        for(int m=0; m<M; m++){
            if( n==0 || m==0 ){
                dp[n][m] = 0;
                continue;
            }
            if( s1[n-1] == s2[m-1])
                dp[n][m] = dp[n-1][m-1] + 1;
            else{
                dp[n][m] = max(dp[n][m-1], dp[n-1][m]
                                                    );
            }
        }
    return dp[N][M];
}
```

APCO
Date: / /

tabulation table:-

S1 = | a | b | c | d | e | $n = 5$
0 1 2 3 4

S2 = | a | c | e | $m = 3$
0 1 2

$= n+1 = 6$.

$m = 4$.

$m \rightarrow$

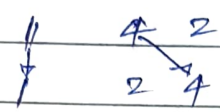
n ↓	0	1	2	3
0	0	0	0	0
1	0	1	1	1
2	0	1	1	1
3	0	1	1+1=2	2
4	0	1	2	2
5	0	1	2	(3) → Ans.

→ (3, 2) → mean.
(3 elements and 2 len S2)

APCO
Date: / /

Que 1035 Uncrossed lines.

Here we have to find lcs & and uncrossed mean.

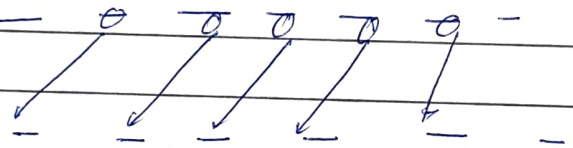
I/P 

a b c.

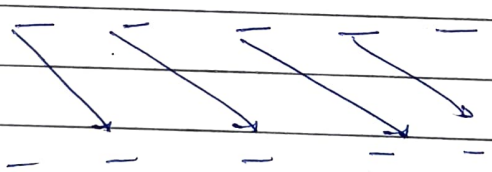
(each index is not possible in subseq). that is called uncrossed lines).

→ I/P is in form of arrays.

same as subsequence (lcs).

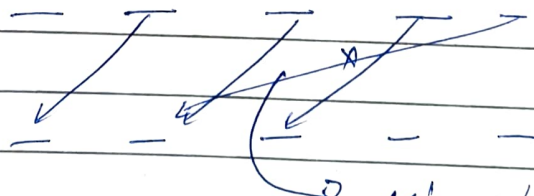


either this.



or this.

kind of subsequence are generated.



this kind of subsequence never possible (that is restriction of uncrossed lines.

APCO
Date: / /

// Code:

```
maxUncrossedLines(int[] A, int[] B){
    int N = A.length;
    int M = B.length;
    int[][] dp = new int[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;
                continue;
            }

            if( A[n-1] == B[m-1] ) {
                dp[n][m] = dp[n-1][m-1]+1;
            }
            else{
                dp[n][m] = max(dp[n-1][m], dp[n][m-1]);
            }

        }

    return[N][M];

}
```

APCO
Date: / /

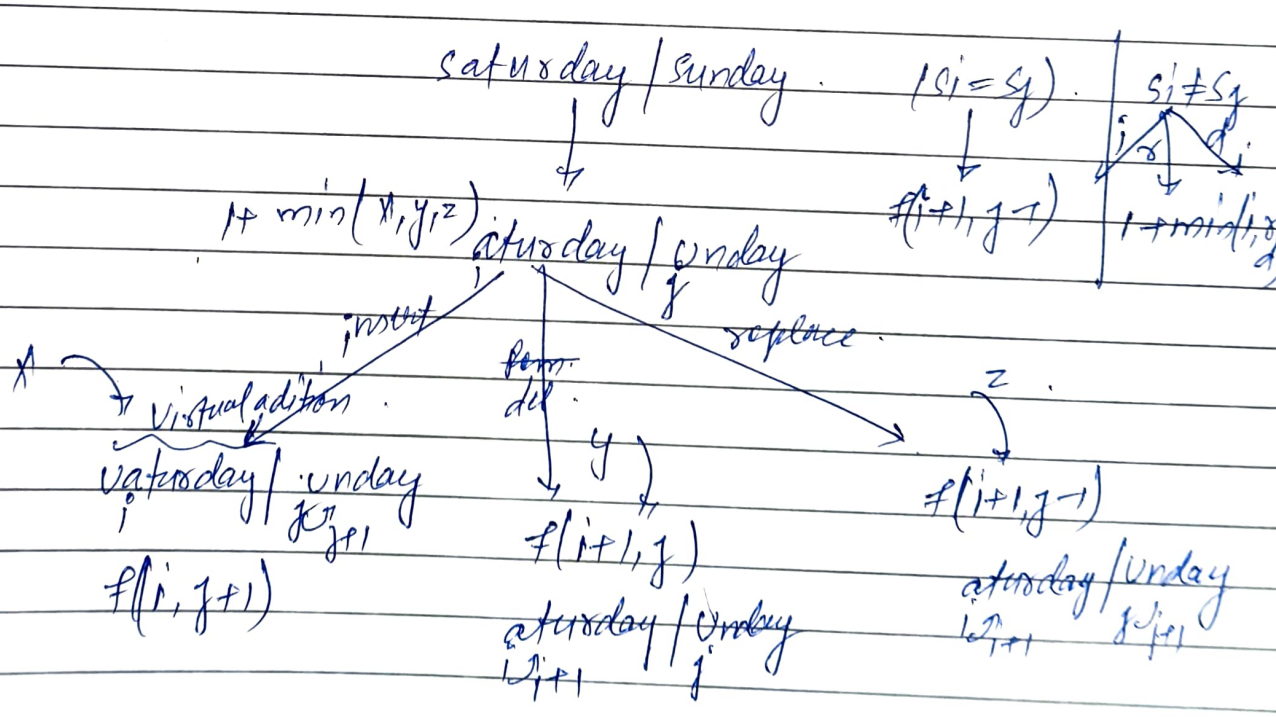
Qno 72. edit distance.

what :- given two strings word1 and word2 find (calculate) min no. of operation required to convert word1 → word2.

How approach:

faith :- faith from child is that that will give me the min operation to convert 1→2. and adding '1' into it i will calculate total operation of convertion.

Recursive - tree



APCO
Date: / /

- if (-, unday). [0, 5).
  only ~~del~~ insert possible.

  if (unday, -) (5, 0).
  ↳ only delete operation. possible.

APCO
Date: / /

Code -

```
Public int minDis fn( String word1, String word2, int n, int m, int[][] dp);

    if( n == 0 || m == 0 ){
        if( n == 0 && m == 0 ) return dp[n][m] = 0;
        return dp[n][m] = n != 0 ? n : m;
    }
    if( dp[n][m] != 0 ) return dp[n][m];

    if( word1(n-1) == word2(m-1))
        dp(n)(m) = fn(word1, word2, n-1, m-1, dp);

    else
        dp(n)(m) = 1+ min( fn (word1, word2, n, m-1, dp),
                           min( fn( word1, word2, n-1, m dp),
                                fn( word1, word2, n-1, m-1, dp)));

    return dp(n)(m);

}
```

APCO
Date: / /

Edit distance dp:-

```
Public int minDistDP(String word1, String word2, int N, int M){
        int[][] dp){

    for(int n=0; n<=N; n++){
        for(int m=0; m<=M; m++){
            if(n==0 || m==0){
                dp[n][m] = n!=0 ? n : m;
                continue;
            }
            if(word1[n-1] == word2[m-1])
                dp[n][m] = dp[n-1][m-1];

            else{
                dp[n][m] = 1+min(dp[n-1][m],
                                 dp[n][m-1],
                                 dp[n-1][m-1]);
            }
        }
    return dp[N][M];
}
```

APCO
Date: / /

44. Wild Card Matching.

What ? given input 's' and pattern 'p' implement wild card pattern matching with support for '?' and '*'

→ ? single character match.

→ * match any no. of character including space.

faith.

How :- Here we will make ~~there~~ three calls.

① if ~~char~~ s(i) = p(i) then increase both the index of s and p.

② else. once increase for s(i+1) ~~and~~ p(i) and s(i) ~~and~~ p(i+1)

Recursive tree.

s / p.

s(i) = p(i).

s(i+1) / p(i+1).

if s(i) ≠ p(i).

if (*)

then (* = ε)

s(i+1) / p(i) s(i) / p(i+1).

if there is contiguous star then convert it into one.

* * * * → *

#code Public static string removeStar(string str){

if(str.length() == 0) return "";

StringBuilder sb = new StringBuilder();

sb.append(str.charAt(0));

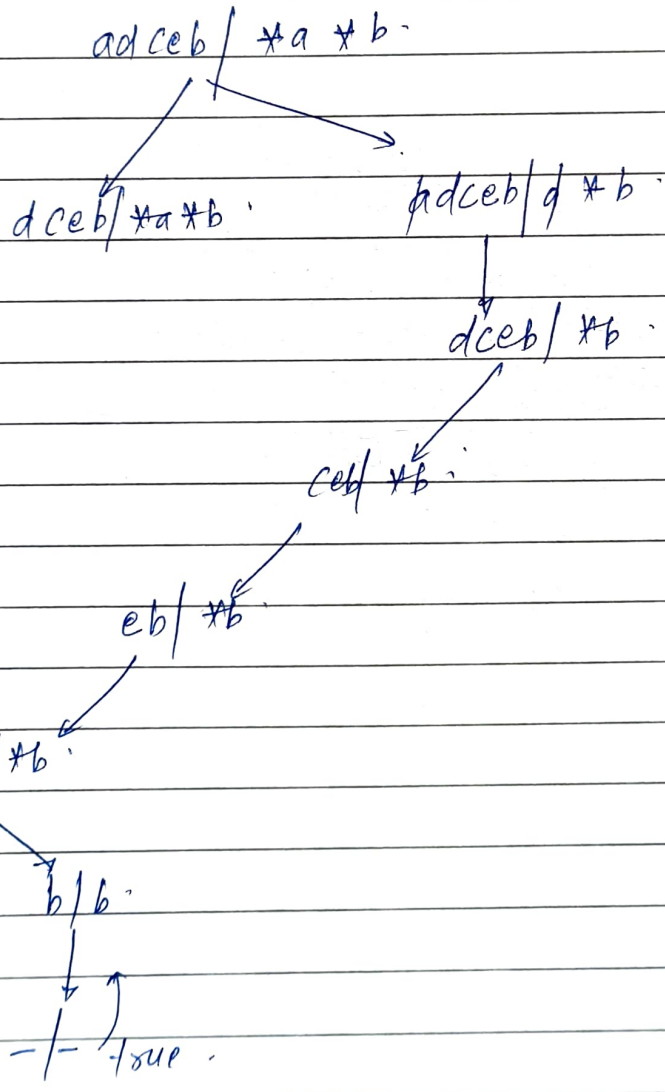
int i = 1;

-1 = not resolved.
0 = false.
1 = True.

APCO
Date: / /

```
while( i < str.length()){
    while( i < str.length() && str.charAt(i-1) == '*' &&
        str.charAt(i) == '*') i++;
    if (i < str.length()) sb.append(str.charAt(i));
    i++;
}

return sb.toString();
}
```

Recursive Tree.



Date:

Lecture-6 (dp). 1/11/2020 (sun).

Agenda

1. coin change permutation.
2. coin change combination.
3. 377 / ~~combination~~ permutation Sum-IV
4. 322 / Min Count
5. ~~416 / partition equal subset sum~~.

(∞ supply of coin)

Target Sum (gfg) / or Subset sum dp-25 (gfg.

6. 416 partition equal subset sum.
7. print path of subset sum using back engineering knapsack (0/1).
8. Unbounded knapsack (based infinite supply of coin/coin change combination).

APCO
Date: / /

1. Que What :- calculation all permutation to achieve target (10)

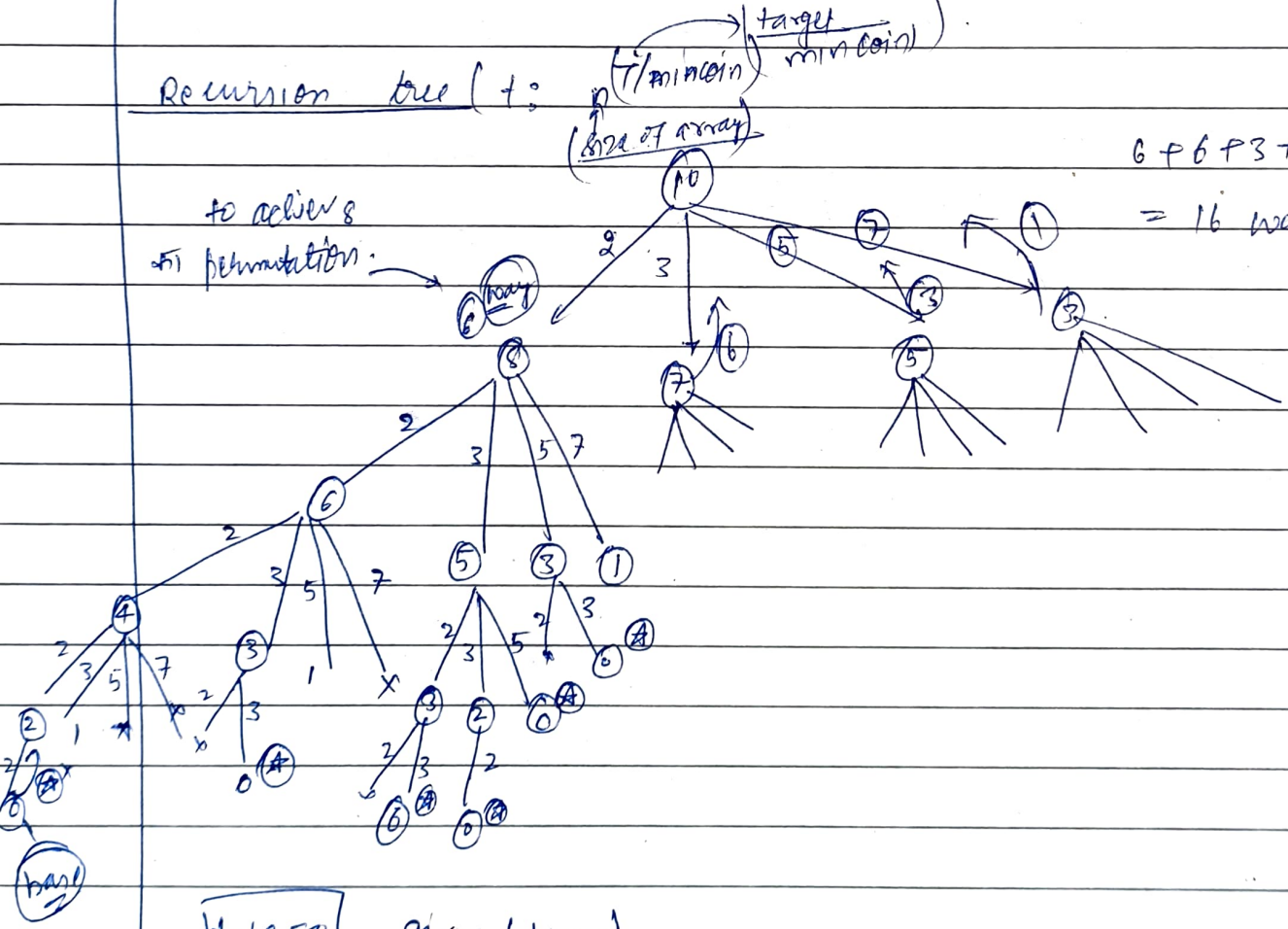
coin[] [2, 3, 5, 7].

How :-

Faith : मैं i th coin use करके tar - arr[i] से faith करता हूँ कि वो मुझे remaining target का permutation दे देगा मैं उन सबसे बढ़ने वाले permutation बना लूँगा।

Recursion tree (↑: T/minCoin) (target / min coin)

(size of array)



Harg = 0 gives (1 way)

T : O(nT)

(size of coin array)

APCO
Date: / /

→ Any state of dp store count to recive that target

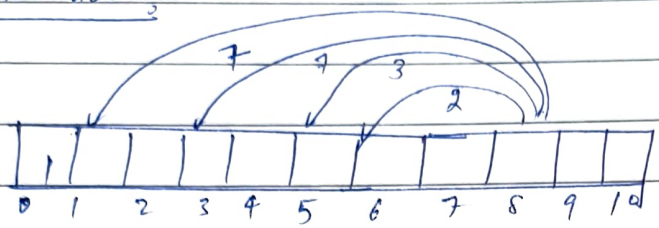
// Code :-

```
Public static int coinChangePermutation(int[] arr, int tar, int[] dp){
    if(tar == 0)
        return dp[tar] = 1;
    }.

    if(dp[tar] != 0)  return dp[tar];.

    for(int ele : arr){.
        if(tar - ele > 0){.
            dp[tar] += fn(arr, tar-ele, dp);
        }.
    }.

    return dp[tar];.
}.
```

APCO
Date: / /

(Here also memoization run faster than tabulation)

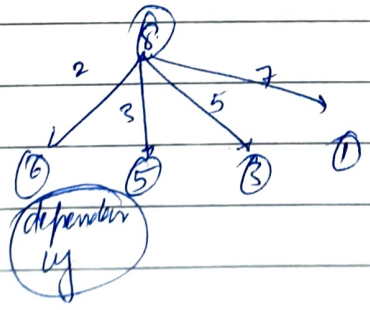
using tabulation :-



→ all dependencies are on left so.
for loop left → right

// Code

```
Public static int coinchangePermutationDp(int[] arr, int Tar, int[] dp){
    dp[0] = 1;
    for(int tar = 1; tar <= Tar; tar++){
        for(int ele : arr){
            if(tar - ele >= 0){
                dp[tar] += dp[tar-ele];
            }
        }
    }
    return dp[Tar];
}
```



t: o(n T)
↑ target
coin array

7(9-7)
5 (9-5) APCO
3 (6-3) Date: / /
2(9-2)

0	1	2	3	4	5	6	7	8	9	10
1	0	1	1	1	3	2	6	6	10	15
		2	3	22	23	222	23	2222		
					32	33	232	233		
					5		25	323		
							322	332		
							52	35		
							7	53		

→ Here we do as · (6 − 2). को achieve करने के all ways. के आगे 2 (MJ) के.

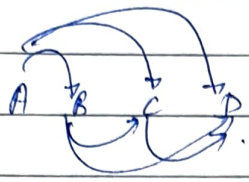
→ similarly 6 − 3, 6 − 5 and so on.

→ This is done to achieve any other target like, 2, 3, 10 & etc. any.

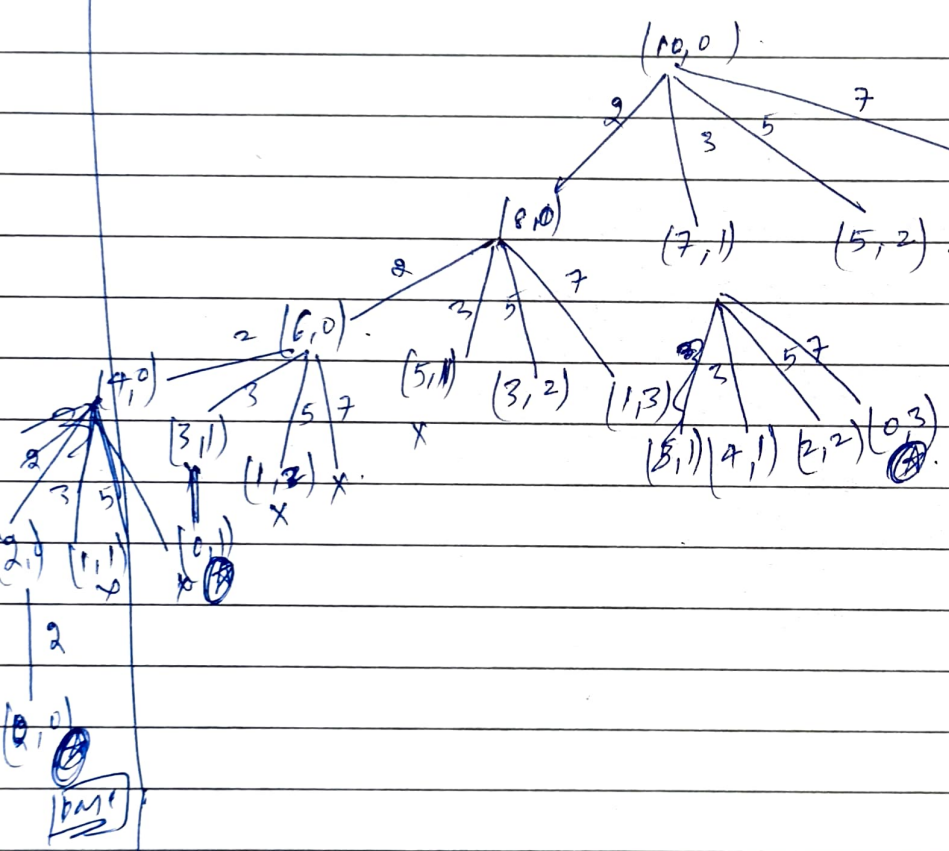
APCO
Date: / /

2. One combination .. of coins. Infinite coin.

Combination :- As in combination we always look in fwd index.



2	3	5	7
0	1	2	3.



→ Here we can't use previous index coin of that particular index.

APCO
Date: / /

In Recursion will have to make 2-d dp of target vs index and resolve dependency which is slower than combination.

So, we will prefer 1-D combination using tabulation.

t: 0

APCO
Date: / /

M-2 tabulation:-

	0	1	2	3	4	5	6	7	8	9	10
	1										5
2	11		2		22		222		2222		22222
5			2	3	22	23	222 33	223	2222 233	2223 333	22235
5			2	3	22	23 5	222 25	223 25	2222 233 35	2223 333 225	22222 2233 235 55
7	11		2	3	22	23 5	222 25 7	223 25 7	2222 233 35	2223 333 225 27	22222 2233 235 55 37

t: O(nT).

arraySize (target)

APCO
Date: / /

```
// Code  Public static int CoinChangeCombinationDp(int arr, int Tar,
        int[] dp){
            dp[0] = 1;
            for(int ele: arr){
                for(int tar = ele; tar <= Tar; tar++){
                    dp[tar] += dp[tar-ele];
                }
            }
            return dp[Tar];
        }
```

APCO
Date: / /

Que 377. ~~Combination~~ Permutation Sum - IV.

Given integer array with positive no. find permutations that add to given target.

I/P : [1, 2, 3]; targ = 4

O/P : (1,1,1,1), (1,1,2), (1,2,1), (1,3), (2,1,1), (2,2), (3,1)

Same as on. 1. done before.

// code. Public int combinationSum4(int[] arr, int Tar){

if (arr.length() = 0) return 0;
int dp[] = new int[n+1]; // (0, n) → Varies.
dp[0] = 1;

for(int tar = 1; tar <= Tar; tar++){
for(int ele: arr){
if(tar - ele) >= 0){
dp[tar] += dp[tar - ele];
}
}
}

return dp[Tar];
}

APCO
Date: / /

Que 322:-

Using coin of different denomination. tell a total amount of money make using min coin.

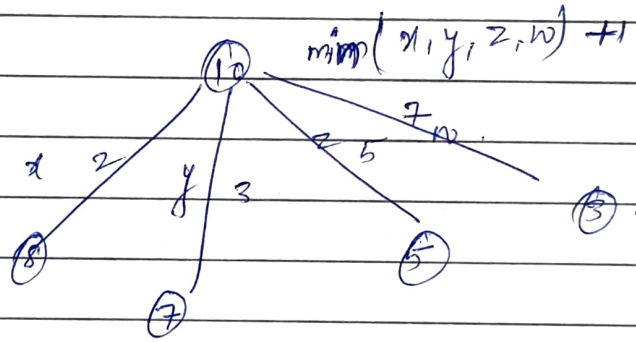
I/P coin = [1, 2, 5] amount = 11

O/P = 3.

5+5+1

How:- Faith:-

मैं पर- (11) को faith करना है कि की मुझे count of coin दे देगा to achieve पर 11 min coin will be my ans.



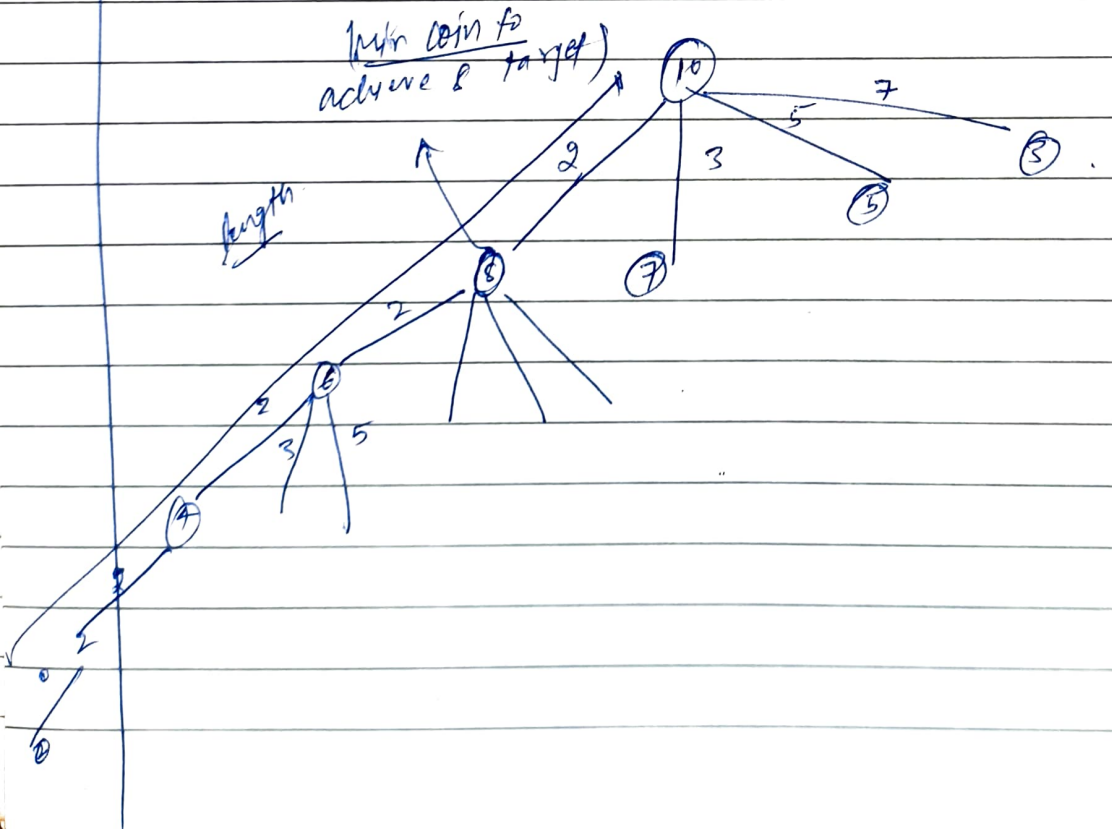
APCO
Date: / /

11 Code -

```
public int minCoinRequired(int[] arr, int tar, int[] dp){
    if(tar==0){
        return 0;
    }
    if(dp[tar] != -1) return dp[tar];
    int minCoin = (int)1e9;

    for(int ele: arr){
        if(tar-ele>=0){
            int val = fn(arr, tar-ele, dp);
            if(val != (int)1e9 && val+1 < minCoin){
                minCoin = val+1;
            }
        }
    }

    dp[tar] = minCoin;
    return minCoin;
}
```

If val = ∞ here then infinity return.
& ∞+1 = −∞ due to infinity cycle occur)

But not in (int)1e9.



APCO
Date: / /

Tabulation:-

```
Public int minCoinRequired Dp(int[] arr, int Tar, int[] dp){
    dp[0] = 0;

    for(int tar = 1; tar <= Tar; tar++){
        int minCoin = (int) 1e9;
        for(int ele : arr){
            if( tar - ele >= 0 ){
                int val = dp[tar - ele];
                if( val != (1e9) && val+1 < minCoin){
                    minCoin = val+1;
                }
            }
        }
        dp[tar] = minCoin;
    }

    dp[Tar];
}
```

APCO
Date: / /

Eve **Subset sum problem dp-25 (gfg)**

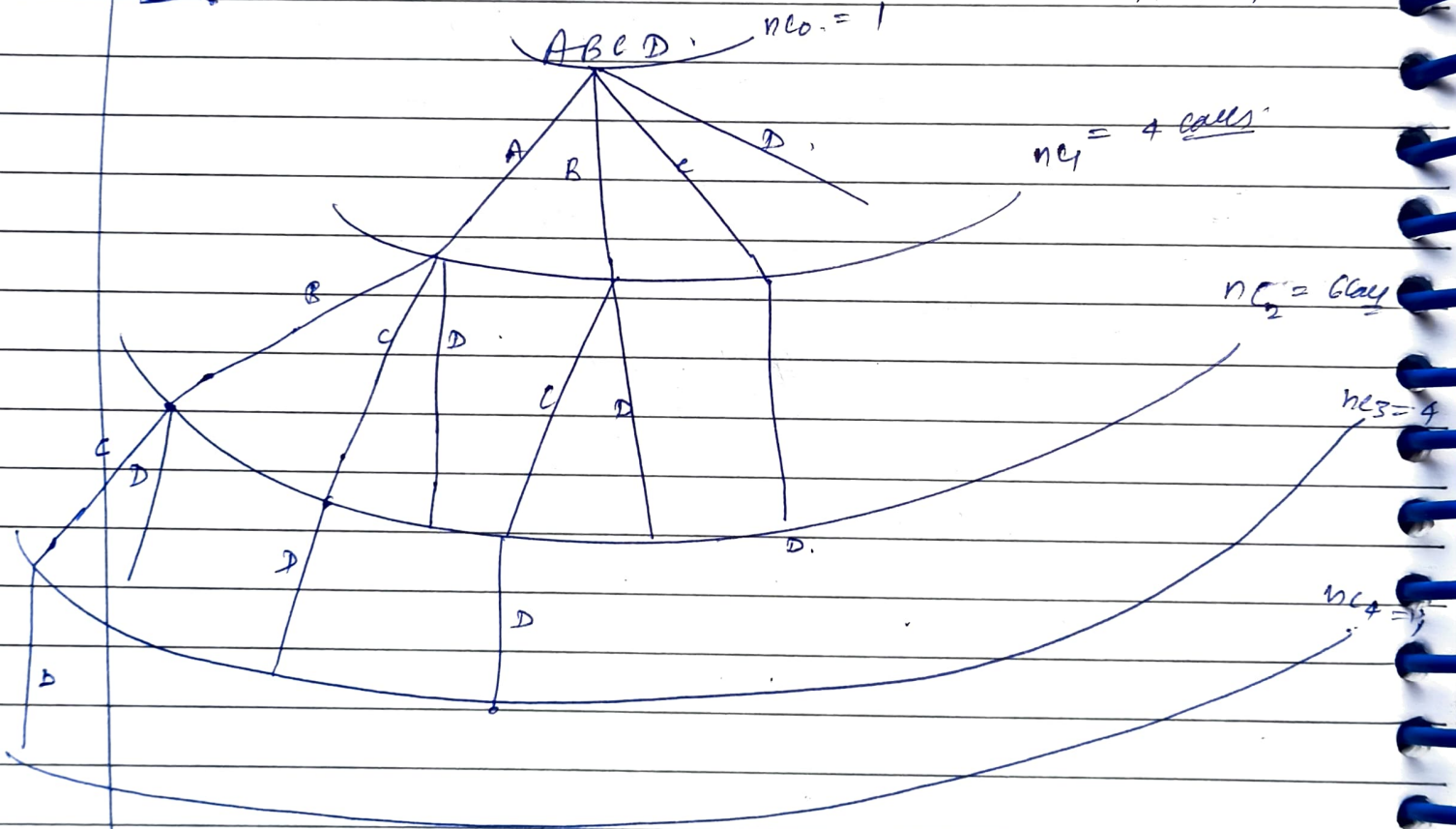
What? We have a set of non-negative integer.

arr[] = {3, 34, 4, 12, 5, 2}

Sum = 9.

Find ~~Sub~~ Subset having sub = 9, {4, 5}

How :-



$$T(n) = nC_0 + nC_1 + nC_2 \cdots + nC_n$$

$$(x+1)^n = x^0 nC_0 + x^1 nC_1 + x^2 nC_2 + \cdots x^n nC_n$$

$x=1$

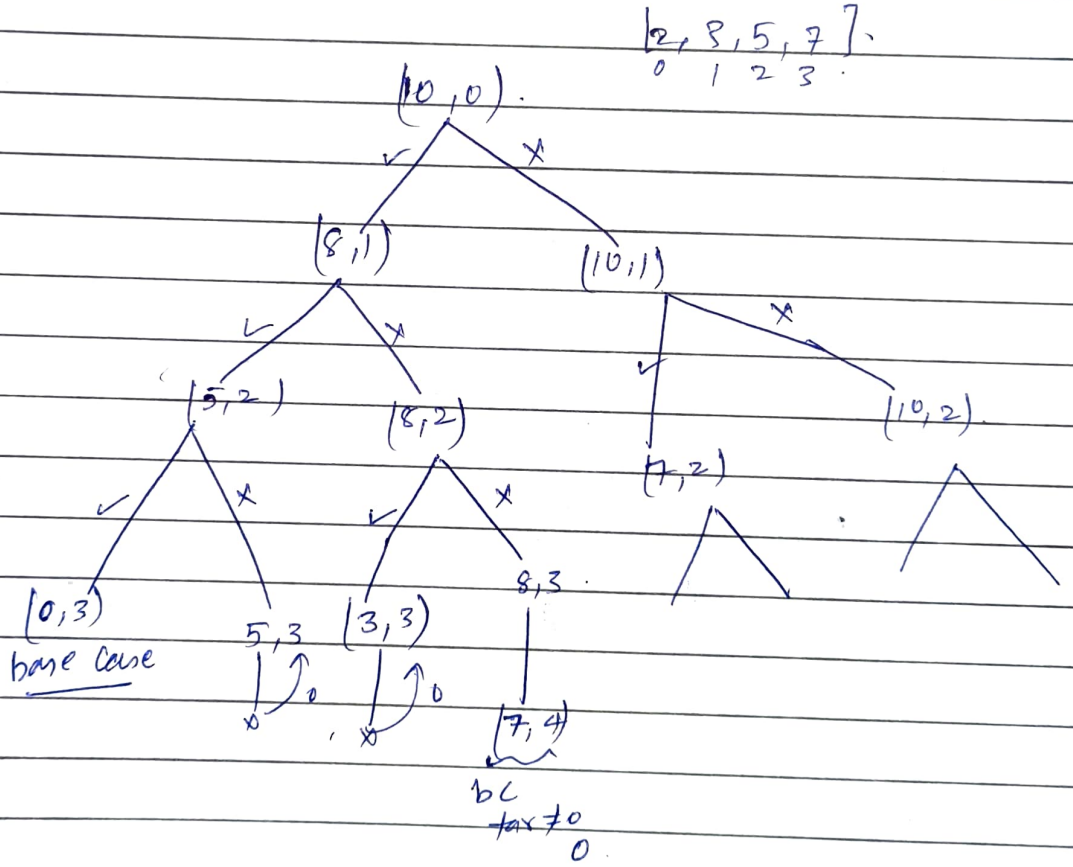
$$\boxed{2^n = nC_0 + nC_1 + nC_2 + \cdots + nC_n}$$

APCO
Date: / /

Means. we can do this problem using
① Subsequence } . answer will be same.
② For loop call }

→ But in knapsack using Subsequence is easy so, we use that.

Recursion tree of subsequence.



APCO
Date: / /

// code :-

```
Public static int targetsum(int[] arr, int idx, int tar, int[][] dp){
    if( tar == 0 || idx == arr.length){
        return dp[idx][tar] = tar==0 ? 1 : 0;
    }

    if( dp[idx][tar] != -1) return dp[idx][tar];

    int Count=0;
    if( tar - arr[idx] >= 0 ){
        count += fn( arr, idx+1, tar-arr[idx], dp);
    }
    Count += fn(arr, idx+1, tar, dp);

    return dp[idx][tar]= Count;

}
```

dp of fee.

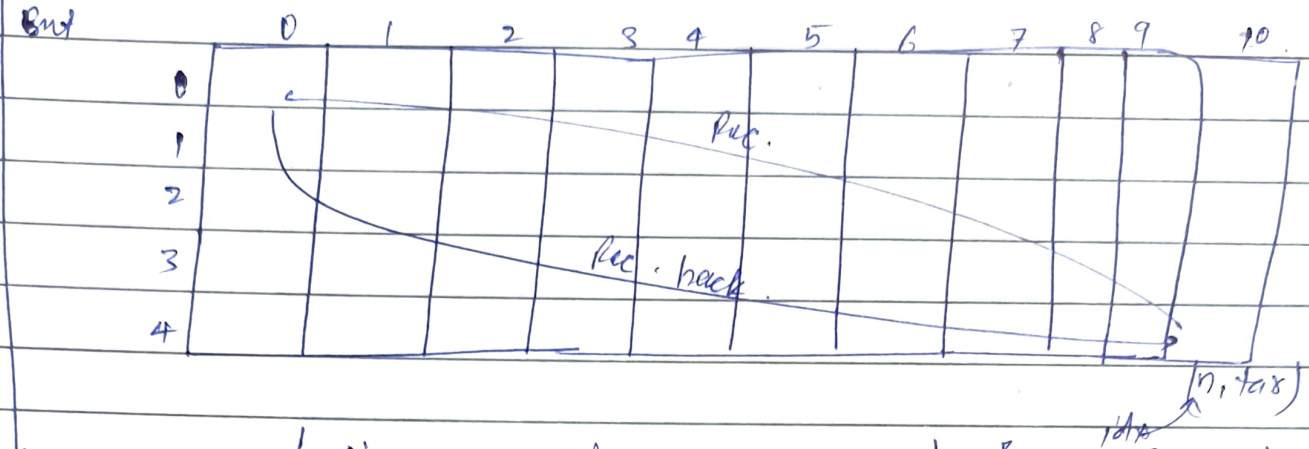
	0	1	2	3	4	5	6	7	8	9	10 (start)
0	-1	-1	-1	-1	recur 1	-1	-1	-1	-1	-1	(2) → ans.
1	-1	-1	-1	-1	-1	-1	-1	-1	1	-1	1
2	-1	-1	-1	-1	-1			+1	0	0	0
3	(1)	-1	0	0	-1	0	1	1	0	-1	0
4	(0)	0	0	0	-1	0	1	0	0	-1	0

recurse back.

-1 state are never visited.

tar=0, idx 4 base case normal

APCO
Date: / /



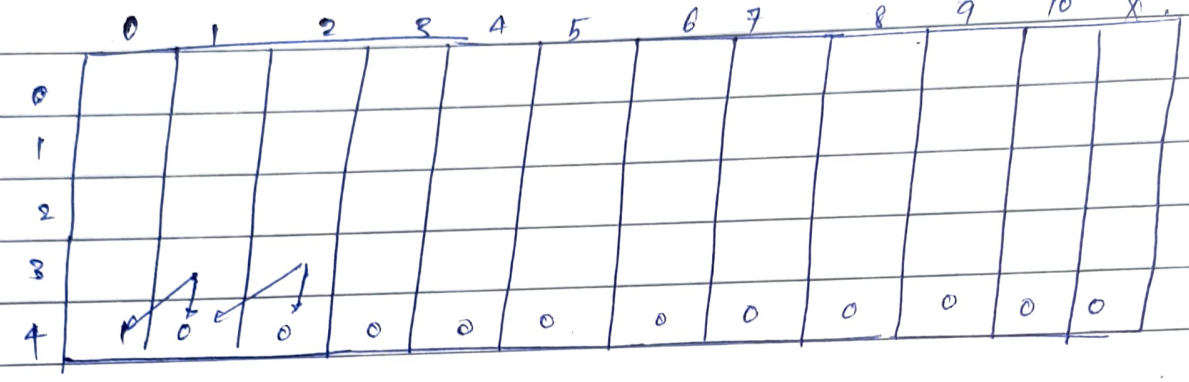
→ (this method is best for code). [back ⇄ code).

→ while (prev is best for dry run) (front ⇄ dry run n-1).

→ do this for array as well as string.

APCO
Date: / /

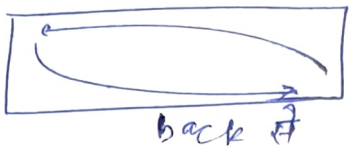
Tabulation :- A table :

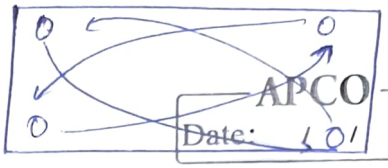


→ Tabulation is worst than memoization but it help in optimization of code.

// Code :-

```
Public static int dp.( int [] arr, int Idx, int Tar, int [][] dp){
    for( int idx = arr.length ; idx >=0; idx--){
        for( int tar =0; tar <= Tar ; tar++){
            if( tar ==0 || idx = arr.length){
                dp [idx] [tar] = (tar ==0)? 1:0;
                continue ;
            }
            int count =0;
            if( tar - arr [idx] >=0){
                Count += dp [idx+1] [tar - arr [idx]];
            count += dp [idx+1] [tar];

            dp [idx] [tar] = Count;

        }
    }
    return dp [Idx] [Tar];
}
```





APCO
Date: 101

M-2.

Back ✓ code:

```
n = arr.length.
targ = 10.

Public static int targetSum2(int[] arr, int n, int tar, int[][] dp
    if(tar == 0 || n == 0){
        return dp[n][tar] = (tar == 0) ? 1 : 0;
    }
    if (dp[n][tar] != -1) return dp[n][tar];

    int count = 0;
                    → because of f(n+1) we got dp.
    if( tar - arr[n-1] >= 0){
        count += fn(arr, n-1, tar-arr[n-1], dp);
    }
    count += fn( arr, n-1, tar, dp);

    return dp[n][tar] = count;
}
```

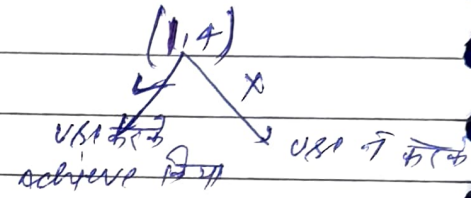
APCO
Date: / /

tabulation back से

used		0	1	2	3	4	5	6	7	8	9	10
	0	1	0	0	0	0	0	0	0	0	0	0
(2)	1	1	0	1	0	0	0	0	0	0	0	0
(3)	2	1	0	1	1	0	1	0	0	0	0	0
(5)	3	1	0	1	1	0	2	0	1	1	0	1
(7)	4.	1	0	1	1	0	2	0	2	1	1	(2)

ans.

(1,4)



// Code :-

~~for (int n=0;~~

Public static int targetSum dp2(int[] arr, int N, int Tar,
int[][] dp){

for(int n=0; n<=N; n++){

for(int tar=0; tar<=Tar; tar++){

if(tar==0 || n==0){

dp[n][tar] = (tar==0) ? 1 : 0;

continue;

}

int count = 0;

if(tar - arr[n-1] >=0){

count += dp[n-1][tar-arr[n-1]];

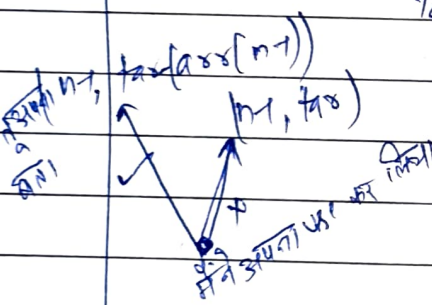
count += dp[n-1][tar];

}

}

return dp[N][Tar];

}

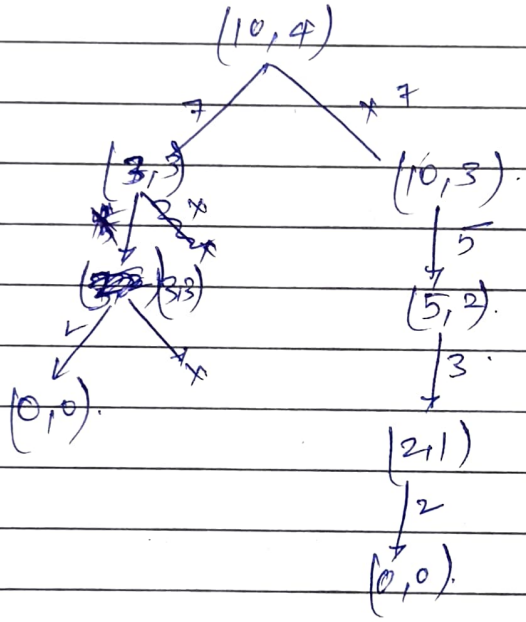


APCO
Date: / /

Q) Now print the coin which are used using back engineering

① all path

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



APCO
Date: / /

// Code:

```
Public static int targetSumPath(int[] arr, int idx, int tar, int[][] dp,
    int psf) {
    if( tar == 0 || n == 0 ) {
        if( tar == 0 ) sysout(psf);
        return 1;
    }
    int count = 0;
    if( tar - arr[n-1] >= 0 && dp[n-1][tar-arr[n-1]] >= 0 ){
        count += fn(arr, n-1, tar-arr[n-1], dp, psf + arr[n-1]
                    + ", ");

    if( dp[n-1][tar] > 0 ).
        count += fn(arr, n-1, tar, dp, psf);

    return count;

}.
```

② To print only one path.

```
P.S boolean targetSumPath( int[] arr, int n, int tar,
    int[][] dp, String psf){
    if( tar == 0 || n == 0 ) {
        if( tar == 0 ) {
            sysout( psf);
            return true;
        }
        return false;
    boolean res = false;
    if( tar - arr[n-1] >= 0 && dp[n-1][tar] > 0 ){
        res = res || fn( arr, n-1, tar - arr[n-1] dp, psf +
                         arr[n-1] + ", ");
    if( dp[n-1][tar] > 0 )
        res = res || fn(arr, n-1, tar, dp, psf);
    return res;
}
```

APCO
Date: / /

Qn 416 Partition into equal set

What :- given non-empty array containing only +ve numbers. find if array can be partioned into two subsets such that sum of element in both subset is equal.

I/p = [1, 5, 11, 5].

O/p = true

{11}, {1, 5, 5}.

How :- ① Calculate sum using loop.

② if can be partioned into two then check. subset sum as previous qn.

// Code :-

```
Public boolean canPartition(int[] arr) {
    int N = arr.length;
    if(N==0) return false;
    int sum=0;

    for(int ele : arr) sum += ele;
    if(sum%2 != 0) return false;

    int Tar = sum/2;
    boolean[][] dp = new boolean[N+1][Tar+1];
```

APCO
Date: / /

```
for(int n=0; n<=N; n++){
    for(int tar=0; tar<=Tar; tar++){
        if(tar==0 || n==0){
            dp[n][tar] = (tar==0)? true : false;
            continue;
        }
        int count=0;
        if(tar - arr[n-1] >=0){
            dp[n][tar] = dp[n][tar] || dp[n-1][tar-arr[n-1]];
        }
        dp[n][tar] = dp[n][tar] || dp[n-1][tar];
    }
}
return dp[N][Tar];
}
```

$\sum set1 = \sum set2$

$\sum set1 = x$

$\sum set1 + \sum set2 = \sum arr$

$2x = \sum arr$

$$\boxed{x = \left(\frac{\sum arr}{2}\right)}$$

APCO
Date: / /

Que Knapsack 0/1 :- (gfg).

Date:

Lecture - 7 date - 2/11/2020 (Mon)

Stock Buy and sell:

Agenda:

1. 121 //
2. 122 //
3. 123
4. 309
5. 188 (H.W.)
6. 714

APCO
Date: / /

Stock buy and sell.

Here we have simulated 3-d ~~for~~ array (dp). whose index are represented as.

i time → k - transaction → [0] या [1].
portfolio (कि item)

→ sell condition. (Hold, sell)

① $T[i][k][0] = \max(T[i-1][k][0], T[i-1][k][1] + price)$

② $T[i][k][1] = \max(T[i-1][k][1], T[i-1][k-1][0] - price)$

Buy: (Hold)

Base condition.

Market not open and want to sell

① $T[-1][k][0] = 0$

② zero transaction and zero stock

$T[i][0][0] = 0$.

③ market not open & want to buy k transaction के साथ / k

$T[-1][k][1] = -\infty$.

④ zero transaction कि / stock want to buy.

$T[i][0][1] = -\infty$.

APCO
Date: / /

Que 121. Best time to buy and sell stock.

What : you have an array for which ith element is price of given stock on ith day.

How :- We'll do it using previous two formula of stock. take two variable not need to take array can be done using variable only.

12	6	7	19	4	1	6	28	9.
0	1	2	3	4	5	6	7	8.

sell →	0 (0)	0	0	1	13	13	13	27	27	27 → profit
buy.	1 (∞)	-12	-6	-6	-4	-1	-1	-1	-1	-1 → buying cost
	↑ market not open.	0	1	2	3	4	5	6	7	8

$T_i[0] = \max(0, -\infty + 12) = 0.$

$T_i[1] = \max(-\infty, -12) = -12$

} similarly check for other as well.

APCO
Date: / /

// code.

```
Public int maxprofit (int[] arr){.
    if(arr.length==0) return 0;.

    int T10 =0;.
    int T11 = -(int)1e9;

    for(int ele: arr){.
        T10 = ~~Mat~~ max(T10, T11+ele);.
        T11 = max(T11, 0-ele);.
    }.

    return T10;.
}.
```

APCO
Date: / /

Que 123. problem is same as prev.
But can do atmost two transaction to earn max profit.

I/P = [3, 5, 5, 0, 0, 3, 1, 4]
O/P = 6.

How :-

Here we'll form 4 eq using prev ② and do.

① $T[i][2][0] = \max(T[i-1][2][0], T[i-1][2][1] + price)$.
↑ 2nd

② $T[i][2][1] = \max(T[i-1][2][0], T[i-1][1][0] - price)$.
↑ 2nd (↑ diff).

③ $T[i][1][0] = \max(T[i-1][1][0], T[i-1][1][1] + price)$;
↑ 1st

④ $T[i][1][1] = \max(T[i-1][1][1], 0 - price)$.
↑ 1st

→ 4 transaction.

30	40	45	50	45	20	26	40	80	50	30
0	1	2	3	4	5	6	7	8	9	10

20	0	0	10	13	20	20	20	26	40	80	80	80 → profit combined
21	-∞	-30	-30	-30	-30	-25	0	0	0	0	0	0
10	0	0	10	13	20	20	20	20	20	60	60	60 → single transaction profit
11	-∞	-30	-30	-30	-30	-30	-20	-20	-20	-20	-20	-20
		0	1	2	3	4	5	6	7	8	9	10

APCO
Date: / /

Code.

```
Public int maxProfit(int[] arr){
    if(arr.length == 0) return 0;

    int Ti20 = 0;
    int Ti21 = -(int)1e9;

    int Ti10 = 0;
    int Ti11 = (-int)1e9;

    for(int ele : arr){
        Ti20 = max(Ti20, Ti21 + ele);
        Ti21 = max(Ti21, Ti10 - ele);

        Ti10 = max(Ti10, Ti11 + ele);
        Ti11 = max(Ti11, 0 - ele);

    }

    return Ti20;

}
```

APCO
Date: / /

Que 122 // Best time to buy and sell stock II.

What :- you have array of prices where ith element represent price on day "i"
calculate max profit using as many txn you want.

I/P = [7, 1, 5, 3, 6, 4].
O/P = 7.

How :-

① $T[i][k][0] = \max(T[i-1][k][0], T[i-1][k][1] + price)$
② $T[i][k][1] = \max(T[i-1][k][1], T[i-1][k-1][0] - price)$

↳ on (∞) txn we use prev sell price incase of default '0'

APCO
Date: / /

// code

```
Public int maxProfit(int[] arr){
    if(arr.length == 0) return 0;

    int Ti0 = 0;
    int Ti1 = -(int)1e9;

    for(int ele : arr){
        Ti0 = max(Ti0, Ti1+ele);
        Ti1 = max(Ti1, Ti0-ele);
    }
    return Ti0;
}
```

APCO
Date: / /

Que 714 Best time to Buy and Sell Stock wit txn fee.

What :-

You are given array of integers of prices. For which ith element is the price of stock on day 'i' and non-negative txn fee.

→ you may complete as many txn you want.
→ calculate max profit.

I/P : [1, 3, 2, 8, 4, 9], fee = 2
O/P = 8.

How :-

① Here we keep track of prev sell and at buying time. we subtract fee and price.

Code :-

```
Public int maxProfit (int[] prices, int fee) {
    int Ti0 = 0;
    int Ti1 = -(int)1e9;

    for (int ele : arr) {
        int prevSell = Ti0;
        Ti0 = max(Ti0, Ti1 + ele);
        Ti1 = max(Ti1, prevSell - ele - fee);
    }
    return Ti0;
}
```

APCO
Date: / /

Low

309. Best time to Buy and Sell Stock with Cooldown

What you have an array for which ith element is price of element on day i
you can do multiple txn with following restriction
① not engage in multiple txn at same time.
② Cool down of 1day.

I/P = [1, 2, 3, 0, 2].
O/P = 3.

APCO
Date: / /

//code -

```
Public int maxprofit (int[] arr){
    if(arr.length ==0) return 0;

    int Ti0 = 0;
    int Ti1 = -(int)1e9;
    int Ti2 = 0;

    for(int val : arr){
        int temp = Ti0;
        Ti0 = max( Ti0, Ti1 + val);
        Ti1 = max(Ti1, Ti2 - val);

        Ti2 = max;

    }
    return Ti0;

}
```

APCO
Date: / /

lecture-7 (dp). 2/11/2020 (Mon).

1. 494
2. Find no. of solution of linear eq. of n variable.

new segment of dp. [Lis].

1. Lis.
2. Lds.

} not asked but base of lbs.

3. LBS — dp-15 (gfg) v. Imp. [always asked] (DTo) lis
4. max sum ↑ing sub.sequence dp-14 (gfg) } [have to check sum
5. Max sum bitonic subsequence. } H.W.
6. min no. of deletion. ✓
7. 673 / No. of longest ↑ing subsequence ✓.

APCO
Date: / /

1. Dec 494. Target Sum:

You are given non-negative no. $a_1, a_2 \ldots a_n$ and target S. ~~Now~~ you have 2 symbol '+' and '−' For each integer you should choose '+' and '−' and its new symbol.

Find out how many ways to assign symbol to make sum of integer equal to S.

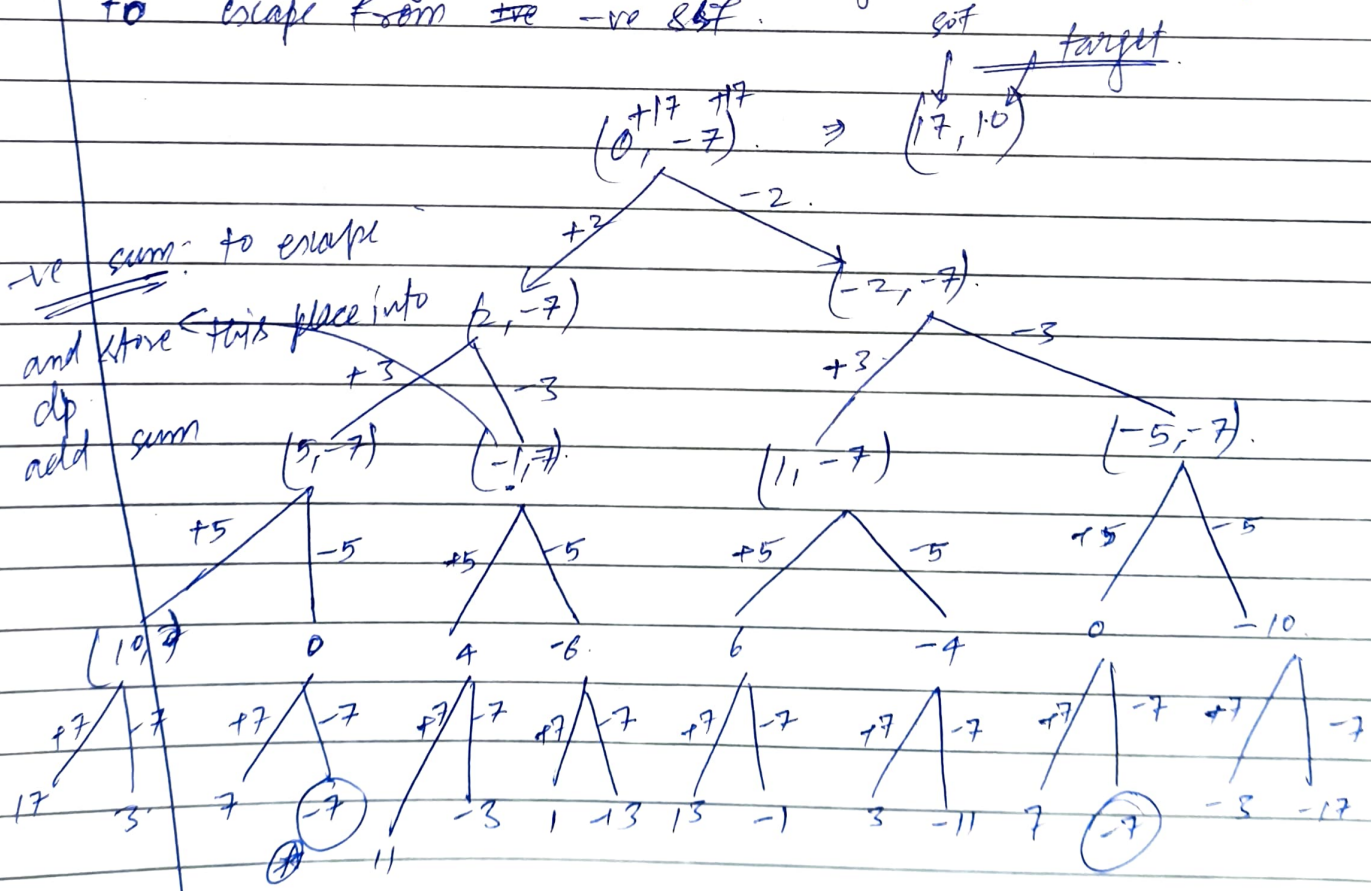
I/P:

nums = [1, 1, 1, 1, 1] S = 3.

How :- [2, 3, 5, 7] Sum = 17.

tar = −7

① Here we do by shifting our target as well as origin (0) to escape from −ve.



-ve sum: to escape and store this place into dp add sum

APCO
Date: / /

dp arr index:

dp[n] [sum]

↑ ↑

index. sum obtained using nth index.

// Code-

```
Public int Find Target sumways( int [] nums, int tar,
    int n, int sum, int [] [] dp) {.
        if (n==0) {.
            return dp[n] [sum] = (tar==sum)? 1: 0;.
        }.

        if (dp[n] [sum] != -1)  return dp[n] [sum];.

        int count=0;

        count += findTsway( nums, tar, n-1, sum+nums[n-1]
                                dp); // positive no.
        count += findTsways( nums, tar, n-1, sum+(-nums[n-1])
                                ,dp);

        return dp[n] [sum] = count;

    }.
```

APCO
Date: / /

```
Public int findTsWays(int[] nums, int s) {
    int n = nums.length;
    if(n==0) return 0;

    int sum=0;
    for(int ele : nums) sum += ele;

    if(s > sum || s < -sum) return 0;

    int[][] dp = new int[n+1][2*sum+1];
    for(int[] d: dp) Arrays.fill(d,-1);

    int ans = fn(nums,(s+sum),n,(0+sum),dp);

    return ans;
}
```

APCO
Date: / /

2. Qu: Find number of solution of linear equation of n variables.

What ? given linear equation of n variable. find non-negative solution of it.

$$x + 2y = 5$$

coff [] = [1, 2] , rhs = 5.

o/p = 3.

solution (3,1), (1,2), (5,0).

How :- It is similar to coin change combination.

Array is coin.

LIS :-

1. longest ↑ing subsequence :-

Faith: nth no. से छोटे सभी no. को बोला कि तुम seq. बना के दे दो जो ↑ing हो & तुम पर खत्म हो
→ तो मैं तुम्हारे में खुद को add करके lis की length बता दूंगा.

Note :- Tabulation of lis is very important.

1. In recursion we'll have to call each element of array to check its lis.

t: $O(n^2)$.

APCO
Date: / /

//code

① int LisRec() {
    vector<int> arr = {0,8,4,12,2,10,6,14,1,9,5,13,3,11,7,15,14};
    int maxlen = 0;
    vector<int> dp(arr.size(),0);
    for(int i = arr.size(); i>=0; i--){
        maxLen = max(maxLen, LisRec(arr,i,dp);
    }
    return maxLen;
}

② int Lis_rec(vector<int> &arr, int idx, vector<int> &dp){
    if(dp[idx]!=0){
        return dp[idx];}
    int maxlen = 1;
    for(int i=idx-1; i>=0; i--){
        if(arr[i] < arr[idx]){
            int len = Lis_rec(arr,i,dp);
            maxlen = max(maxlen, len+1);
        }
    }
    return maxlen;
}

APCO
Date: / /

(7) Ans Tabulation of LIS: (Very Important tabulation method).

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
arr	0	8	4	12	2	10	6	14	1	9	5	13	3	11	7	15	14
len	1	2	2	3	2	3	3	4	2	4	3	5	3	5	4	6	6
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	8	4	4	2	4	4	4	1	4	1	4	1	4	1	4	4	4
			12		10	6	10		6	5	6	3	6	5	6	6	6
							14		9		9		9	7	9	9	9
											13		11		11	11	11
															15	14	14

7 1 5 7 3 7 7 14 14
11 9 5 13 11 15
14

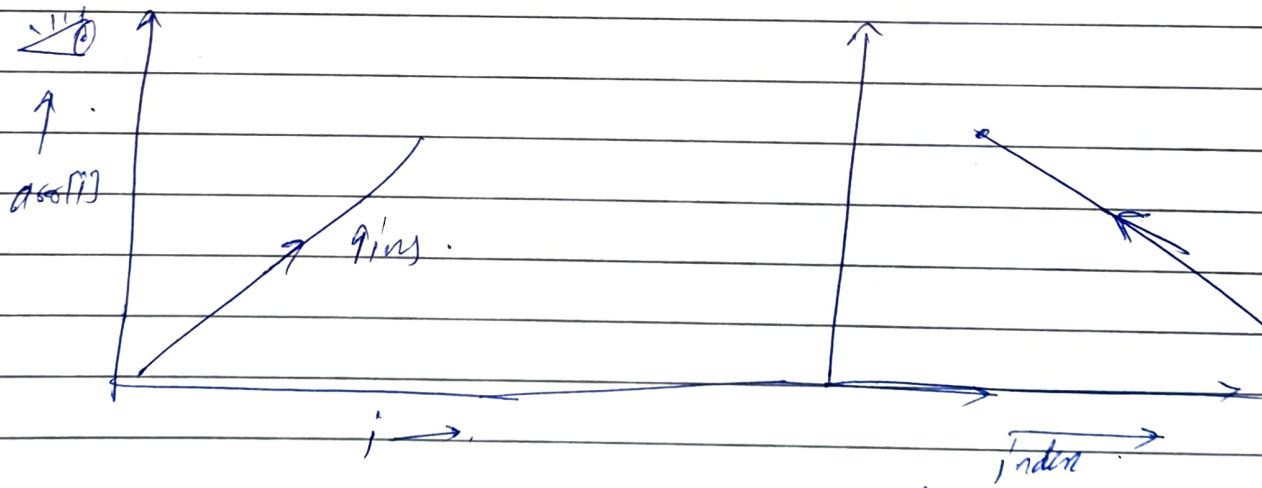
lis = 0, 4, 10, 14, 11, 7 len = 6. Ans

① & print ans

APCO
Date: / /

// Code:

```
int LIS( vector<int> &arr, vector<int> &dp){
    int n = arr.size();
    int maxlen = 0;
    for(int i=0; i<n; i++){
        dp[i] = 1;
        for(int j= i-1; j>=0; j--){
            if(arr[i] > arr[j]){
                dp[i] = max(dp[i], dp[j]+1);
            }
        }
    }
    maxlen = max(maxlen, dp[i]);
}
return maxlen;
}
```

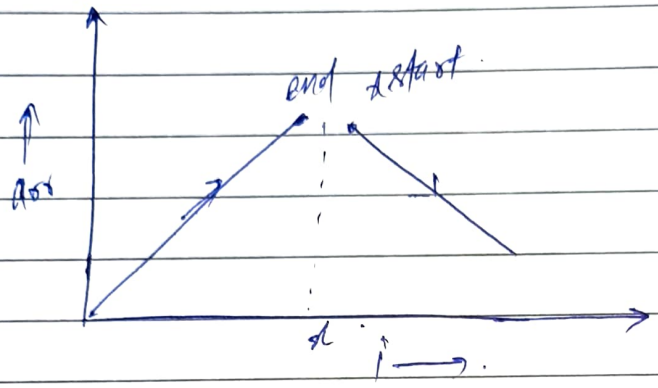


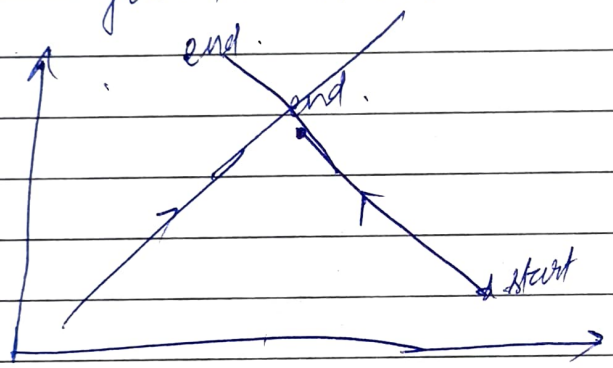
Find after ~~3~~ 6s

it will be 4s from front

APCO
Date: / /

Note: ① x point पर खत्म होने वाला lim. end.
② x point से start होने वाला lim. ही
guarantee to get lim.



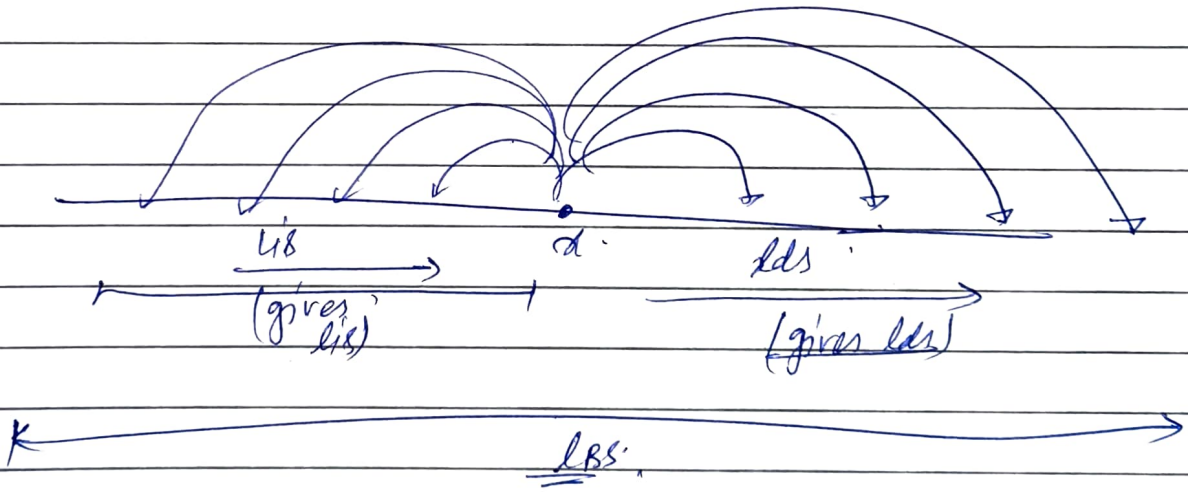


→ doesn't guarantee ~~lim.~~ lim.

APCO
Date: / /

4 lds:

```
int LBS(vector<int> &arr, vector<int> &dp){
    int n = arr.size();
    int maxlen = 0;
    for(int i=n-1; i>=0; i--){
        dp[i] = 1;
        for(int j=i+1; j<n; j++){
            if(arr[i] > arr[j]){
                dp[i] = max(dp[i], dp[j]+1);
            }
        }
        maxlen = max(maxlen, dp[i]);
    }
    return maxlen;
}
```



① faith
② Trap
③ Time complexity

APCO
Date: / /

```
int LBS(vector<int> &arr){
    int n = arr.size();
    vector<int> DP_LIS(n,0);
    vector<int> DP_LDS(n,0);

    LIS(arr, LIS_DP);
    LDS(arr, LDS_DP);

    int maxlen=0;
    for(int i=0; i<n; i++){
        maxlen = max(maxlen, DP_LIS[i]+DP_LDS[i]-1);
    }
    return maxlen;
}
```

T: $O(n^2)$

APCO
Date: / /

4. Qus Max Sum Ung Subseq.

APCO
Date: / /

5 Qn Max Sum Bitonic Subsequence.

APCO
Date: / /

Q: Min. no of deletion to make sorted seq. (gfg)

What :- given array of 'n' integers. The task is to remove or delete the min no. of elements from array so that remaining elements are placed in same seq. order to form a Ping sorted seq.

I/P :- [5, 6, 1, 7, 4]

o/p : 2

(5, 6, 7) Ans

How - ① Find LIS with equality (not strictly Ping

②

ans = arr length - maxlen.

APCO
Date: / /

// Code :-

```
int minNo.of deletions( vector<int>&arr){
    int n = arr.size();
    vector<int> dp(n,0);

    int maxlen=0;
    for(int i=0; i<n; i++){
        dp[i] = 1;
        for(int j = i-1; j>=0; j--){
            if( arr[i] >= arr[j]){
                dp[i] = max(dp[i], dp[j]+1);
            }
        }
        maxlen = max( maxlen, dp[i]);
    }
    return n - maxlen;
}
```

APCO
Date: / /

7-Que 673 | No. of longest Ping subsequence.

APC

Date: /

Lecture-8:- 4/11/2020:-

Agenda:-

1. Russian doll envelop (leetcode) 354.
2. Building bridges. (gfg).

Cut Type:-

1. MCM (Matrix chain Multiplication). Rec.
2. dp (MCM).
3. Print which order of MCM (using Parenthesis)

} gfg.

APCO
Date: / /

Q. 354 Russian doll envelop.

what :- you given no. of envelop with width and height (w, h).
One envelop can fit into another if and only if both width and height of one is greater than other

Find stack of max no. of envelop you can build.

~~o/p~~ = (w, h).

How :- ① Sort the array (2-d) on the basis of (h, w) any here sorting on height & sorting is done on basis of width if height equal.
② Run lis on width
③ Max length of lis will give max stack of envelop.

I/p :- [[5,4], [6,4], [6,7], [2,3]];
o/p :- 3.

(2,3), (5,4), (6,7) (stack of envelop.) Ans.

dry run :-

I/p : (5,4), (6,4), (6,7), (2,3), (1,2), (1,4), (3,8)

sorted [(1,4), (1,2) (2,3), (3,8), (5,4), (6,4), (6,7)]
0 1 2 3 4 5 6.

1	1	2	3	3	4	4 3	
0	1	2	3	4	5	6	

APCO
Date: / /

② Here we have done reverse sorting of width because doing so we get surity larger width cannot become in smaller one
eg.

(1,2), (1,4) can come because smaller
↓ reverse sorting.

(1,4), (1,2) cannot come.

// Code -

```
Public int maxEnvelopes (int[][] envelopes){
    int n = envelopes.length;
    int dp[] = new int[n];
    Arrays.sort( envelopes, (a,b) ->{
        if(a[1] == b[1])
            return b[0] - a[0];

        return a[1] - b[1];
    });

    int maxstack = 0;
    for(int i=0 ; i<n ; i++){
        dp[i] = 1;
        for( int j= i-1; j>=0; j--){
            if( envelopes[i][0] > envelopes[j][0]){
                dp[i] = max(dp[i], dp[j]);
            }
        }
        maxstack = max(maxstack, dp[i]);
    }
    return maxstack;
}
```

APCO
Date: / /

2. Building Bridges (gfg).

What :- given starting and ending index of bridge. Tell how many bridges can be formed without doing intersection.

I/P: (8,1), (1,2), (4,3), (3,4), (5,5), (2,6), (6,7), (7,8).
o/p = 5

① Sort on the basis of index 0. (~~Start~~ ending index).

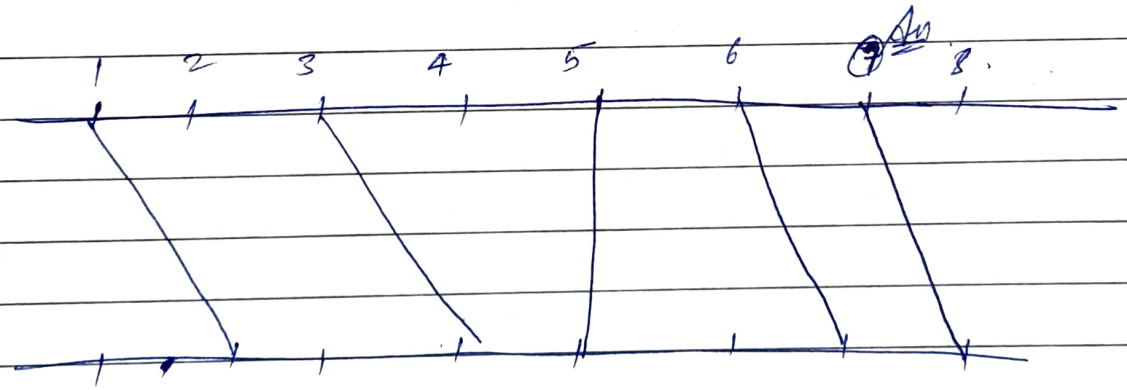
(1,2) (2,6) (3,4) (4,3) (5,5) (6,7) (7,8) (8,1).

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

Map

② if same starting point then sort on basis on ending point (reverse order).

Same as ~~tv~~ envelops.



Fib
Pall Pall LCS

① Two pointer	⑤ stockBuy Sell.
② _ stringset	⑥ Cut Type.
③ Targetset	
④ LIS.	

APCO
Date: / /

Cut Type.

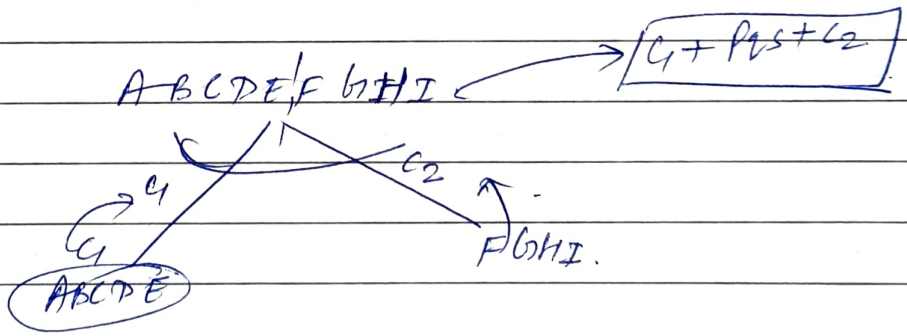
① Matrix Chain Multiplication (very Important).

What: given matrix ABCD. with dimension in form of array. return min cost of multiplication.

40	20	30	10	30.
A	B	C	D.	

A = 40 X 20.
B = 20 X 30
C = 30 X 10.
D = 10 X 30
E.

How:- Faith:- min cost के आधार पर Multiply करके मैं आऊंगी
& my cost will be.

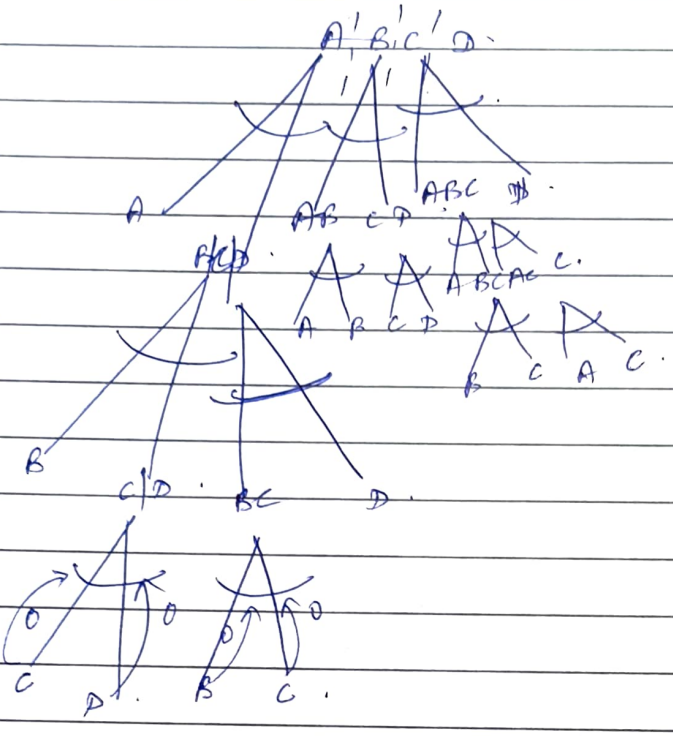


ABCDE = A'pq = cost = C1

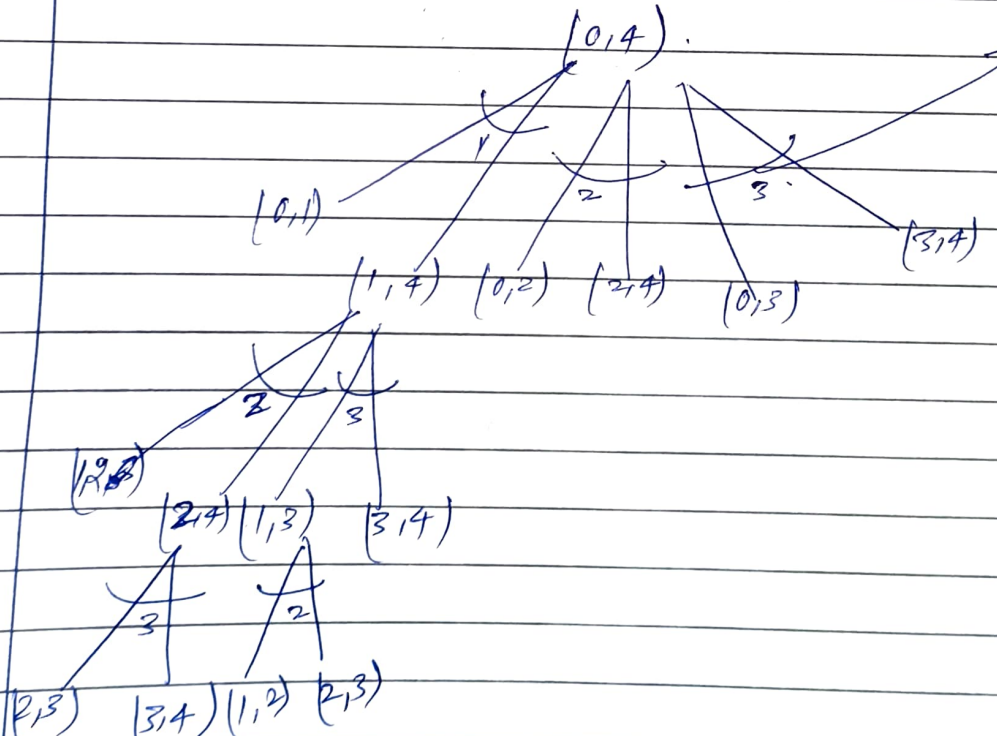
FGHI = B'qs = cost = C2.

APCO
Date: / /

Recursion Tree -



For Index.



APCO
Date: / /

// Code :- Recursive + Memoization.

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

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

    int minCost = (int)1e9;

    for(int cut = si+1; cut < ei; cut++){
        int leftTree = MCM(arr, si, cut, dp);
        int rightTree = MCM(arr, cut, ei, dp);

        int cost = leftTree + (arr[si] * arr[cut] *
                   arr[ei]) + rightTree;
        minCost = Math.min(minCost, cost);
    }

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

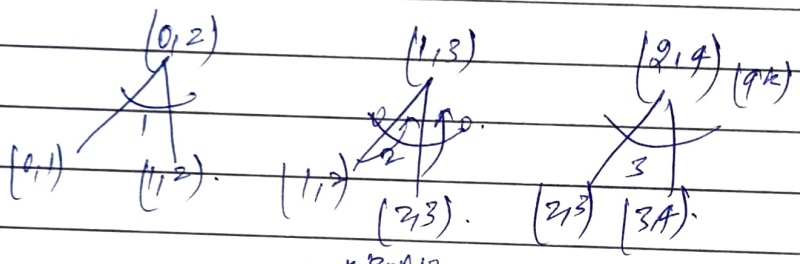
APCO
Date: / /

(2) Tabulation :- Here loop is of gap one

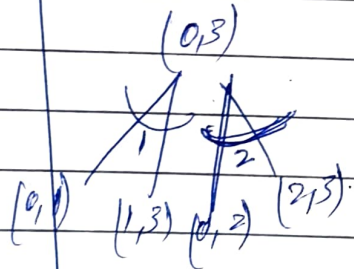
{40, 20, 30, 10, 30}.

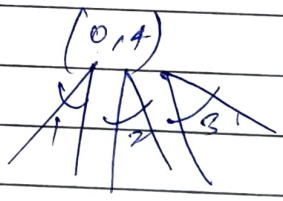
0 1 2 3 4.

	0	1	2	3	4
0	-1	0	24k.	14k.	
1	-1	-1	0	6k.	12k.
2	-1	-1	-1	0	9k.
3	-1	-1	-1	-1	0
4	-1	-1	-1	-1	-1



20 × 30 × 10





(0,1) = 0 +

(1,3) = 6k. +

(0,1,3) = 40 × 20 × 10 = 8k +

= 14k.

0,2 = 24k

2,3 = 0

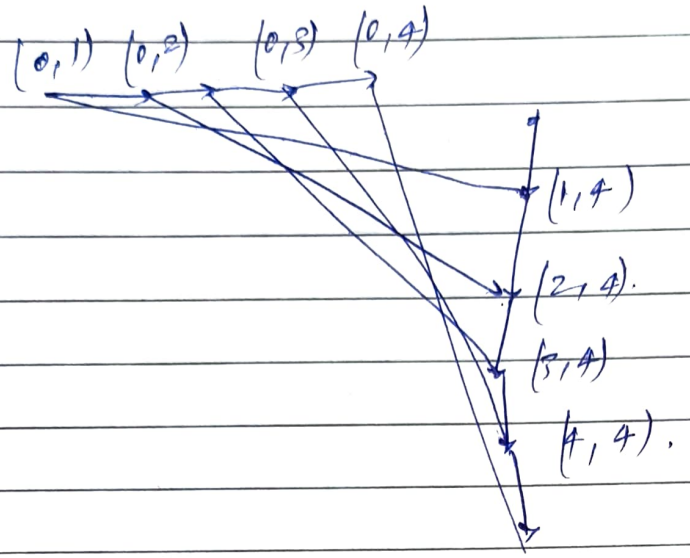
(0,2,3) = 12k

= 36k

APCO

Date: / /

Pattern of ~~Multiple~~ dp.

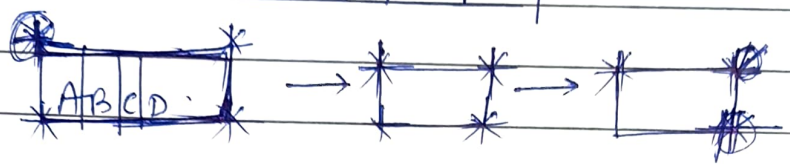


Code:

APCO
Date: / /

Qu For String printing.

	0	1	2	3	4
0	-1	(AB)	(AB)		
1	-1	-1	B	(BC)	
2	-1	-1	-1	C	(CD).
3	-1	-1	-1	-1	D.
4	-1	-1	-1	-1	-1



How:

① Make dp of string and store Min Cost Matrix string.
eg A B C) or (AB)C.
0 1 2

② Now calculate Min of Matrix product ~~from~~ from dependency.
for → (A B C) - depends(0,3)
0 1 2 → neg

// Code: Public static int MCM_DP_string(int[] arr, int SI, int EI,
int[][] dp){.
int n = arr.length;
String[][] sdp = new String[n][n];
for(String[] s: sdp) Arrays.fill(s, "");.

for(int gap=0; gap<n; gap++){.
for(int si=0; ei==gap; ei<n; si++; ei++){.
if(si+1 == ei){.
dp[si][ei]=0;.
sdp[si][ei]=(char)(si+'A')+"";
continue;.
}.

APCO
Date: / /

si=0, ei = arr.length.

```
int minCost = (int)1e9;
for (int cut=si+1; cut<ei; cut++){
    int leftTree = dp[si][cut]; // MCM(arr, si, cut, dp);
    int rightTree = dp[cut][ei]; // MCM(arr, cut, ei, dp);

    int costOfMUL = leftTree + (arr[si] * arr[cut] * arr[ei]) +
                    rightTree;
    if(costOfMUL < minCost){
        minCost = costOfMUL;
        sdp[si][ei] = "(" + sdp[si][cut] + sdp[cut][ei] + ")";
    }
}
dp[si][ei] = minCost;

}
System.out.println(sdp[SI][EI]);
return dp[SI][EI];
}
```

APCO
Date: / /

Lect - 9 (dp)

6/11/2020.

Agenda -

1. Max and min calculation - gfg.
2. Burst balloon - lc - 312.
3. optimal bst - gfg.
4. Min score triangulation - (leetcode - 1039)

APCO
Date: / /

1. Que Max and Min Value of an expression.

What :- We will have expression and seperate it to num array and operator array, then find max/min value of the expression.

I/P: 1+2*3+4*5

Min=27 / Max = 105

num :- | 1 | 2 | 3 | 4 | 5 |

operator : | + | * | + | * |

How :- Faith if is similar like matrix chain multiplication we expect from (si, cut) and (cut, ei) to give it max and min value.
My ans would be operator at cut index.
and left(M/m) & right (M/m).

// Code :-

```
Public static class pair {
    int minVal = 0;
    int maxVal = 0;

    Pair (int minVal, int MaxVal){
        this.minVal = minVal;
        this.maxVal = maxVal;
    }
}

Public static int Evaluate(char ch, int a, int b){
    if(ch == '*') return a*b;
    else return a+b;
}
```

APCO
Date: / /

```
Public static pair MinMaxEvaluation(int[] num, char[] operator,
    int si, int ei, pair[][] dp){
        if(si == ei){
            return new pair(num[si], num[si]);
        }

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

        pair myAns = new pair((int)1e9, -(int)1e9);

        for(int cut = si; cut < ei; cut++){
            pair left = minmaxEvaluation(num, operator,
                                si, cut, dp);
            pair right = fn(num, operator, cut+1, ei, dp);

            char ch = operator[cut];

            myAns.minVal = Math.min(myAns.minVal,
                    Evaluate(ch, left.minval, right.minval));
```

if there will be '−' sign then to calculate Max and we have to 4 operation of M,m and operator.

(m,m), (M,M), (m,M), (M,m).

while in '+' & '*'

(m,m) ⟶ min
(M,M) ⟶ max }.

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

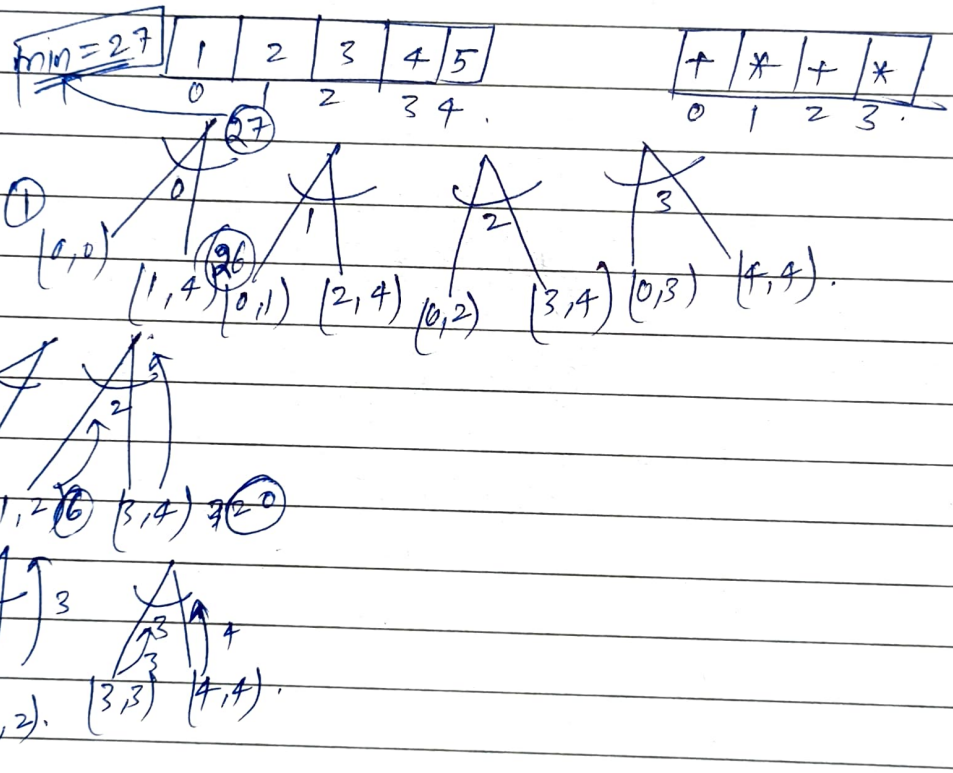
APCO
Date: / /

→ Here: dp will of 3d but we have managed it in 2-d using pair

→ ∵ because if we take integer then, we will have to make dp like:

```
int[][][] dp = new int[n][n][2];
                          for max/min.
```

Rec. Tree:



Similarly MAX = 105

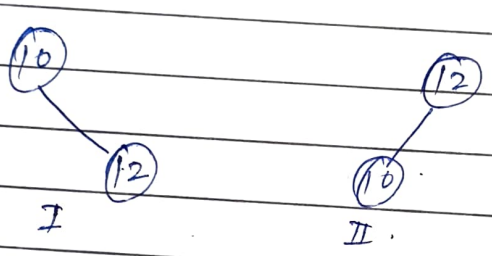
Here also cut diff (Ans evaluated at last) same in balloon problem.

APCO
Date: / /

Q. Our optimal BST :-

What ? given a sorted array keys [0, n-1] and array ~~and~~ freq [0, n-1]. of freq count. where freq[i] is the no. of search to keys[i].
construct BST of all keys such that total cost of all the searches is as small as possible.

I/P. keys : {10, 12,} freq = {34, 50}



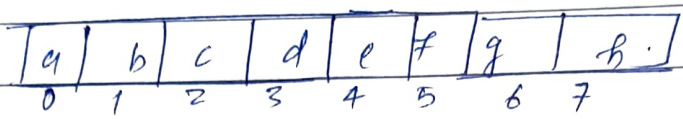
Cost - II = 50 × 1 + 10 2 × 34 = 118.
Cost - I = 34 × 1 + 50 × 2 = 134.

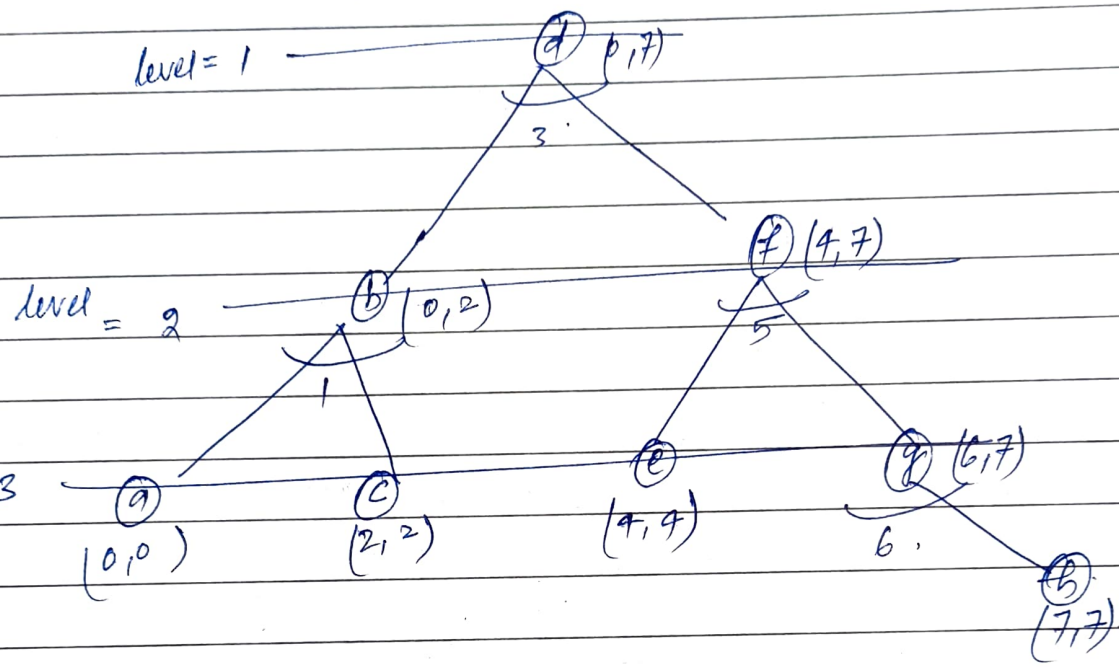
How :- Faith :- मैं जहाँ भी cut मारूँगा (cut index will be ~~node~~ root). ~~इसका~~ left free ~~इसके~~ बनने की min cost देगा and right part will also give minimal cost of construction.

APCO
Date: / /

M-1 Recursion tree:

on Rec. cost = leftcost + levelx arr[cut] + rightcut.



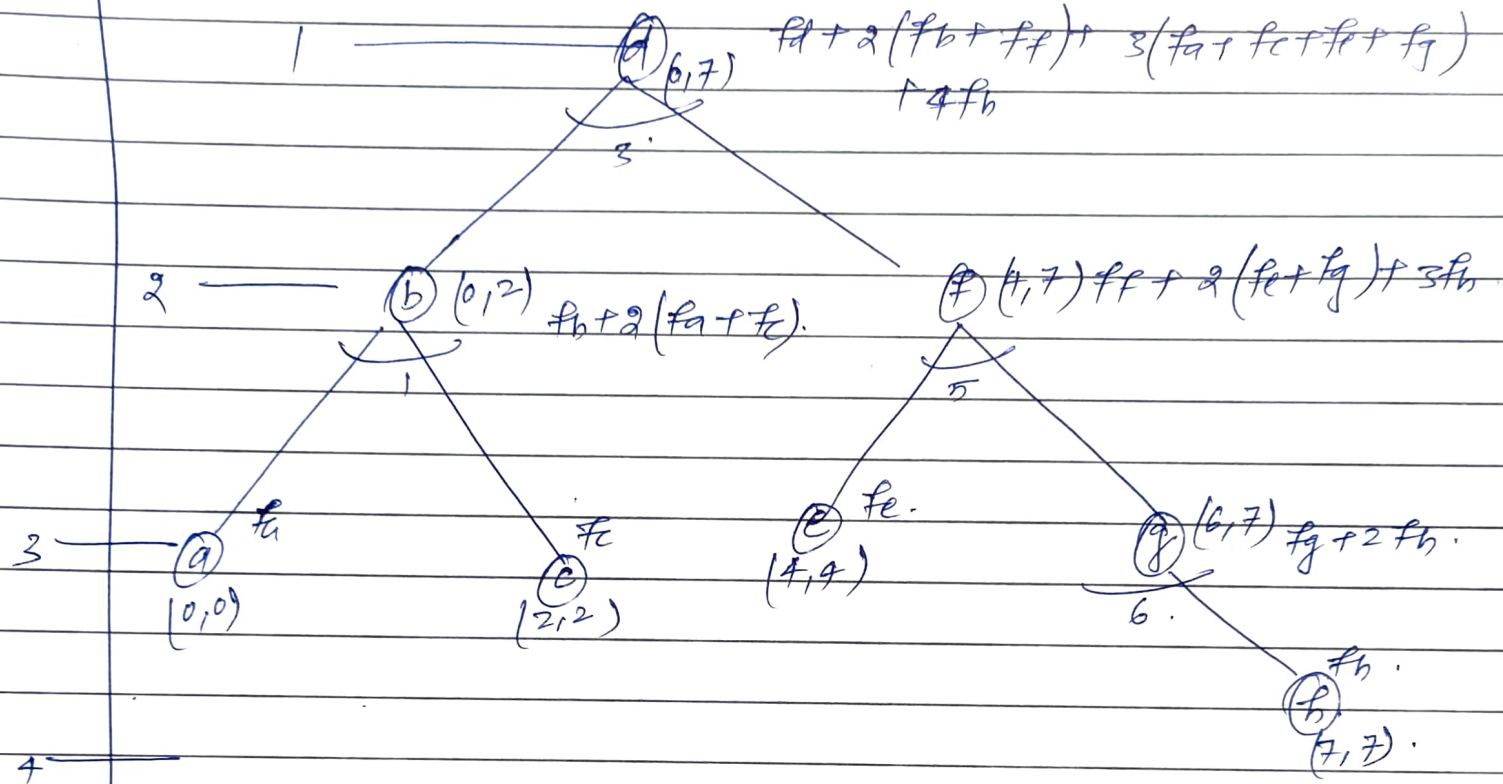


// Code:

```
int OBST(int[] keys, int[] freq, si, ei, level, int[][] dp){
    if(dp[si][ei] != -1) return dp[si][ei];
    int minCost = (int)1e9;

    for(int cut = si; cut <= ei; cut++){
        int leftTree = (cut == si) ? 0 : fn(keys, freq,
            si, cut-1, level+1, dp);
        int rightTree = (cut == ei) ? 0 : fn(., cut+1, ei,
            level+1 );
        minCost = min(minCost, costFromAbove+0);
    }
    return dp[si][ei] = minCost;
}
```

APCO
Date: / /

M-2. In tabulation if we use level kind of thing then complexity will be higher so, to remove dependency of level we have new trick of keeping sum variable.



cost of node (b) = (fa + f(0,2) + fc).

= fa + [fa + fb + fc) + fc.

↑fl ↑fr

Cost = fb + 2(fa + fc).

↑fl

By using prefix array -

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

APCO
Date: / /

Ans. sum(1,7) = presum(7) - presum(0).

//Code:-

```
int OBST_01(int[] keys, int[] freq, int si, int ei, int[] psum,
            int[][] dp){
    if(dp[si][ei] != -1) return dp[si][ei];

    int minCost = (int) 1e9;
    for(int cut = si; cut <= ei; cut++){
        int leftTree = (cut == si) ? 0 : fn(keys, freq, si, cut-1, psum, dp);
        int rightTree = (cut == ei) ? 0 : fn(keys, freq, cut+1, ei, psum, dp);

        minCost = Math.min(minCost, leftTree +
                  psum[ei] - (si == 0 ? 0 : psum[si-1]) + rightTree);
    }
    return dp[si][ei] = minCost;
}
```

APCO
Date: / /

M-3 How to replace prefix Sum Array.

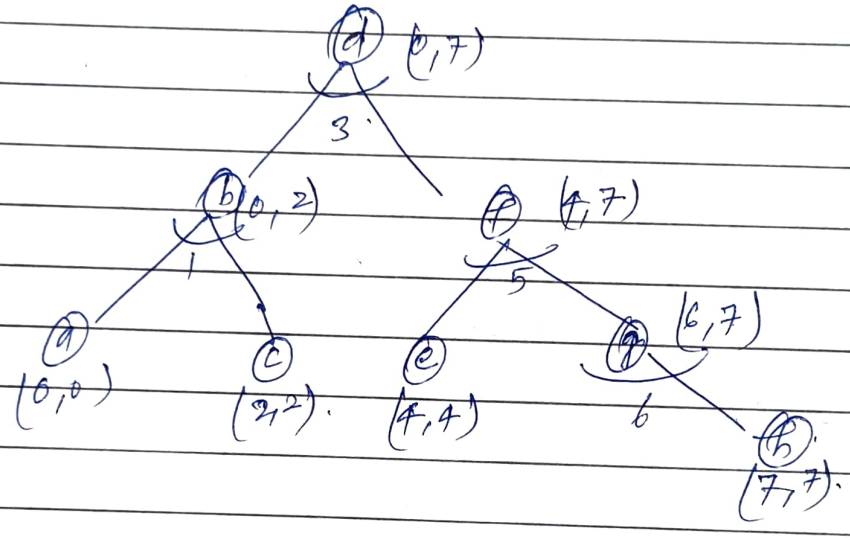
As we've seen:

$((l_0 + S + r_0), (l_1 + S + r_1) \ldots)$ — ①.

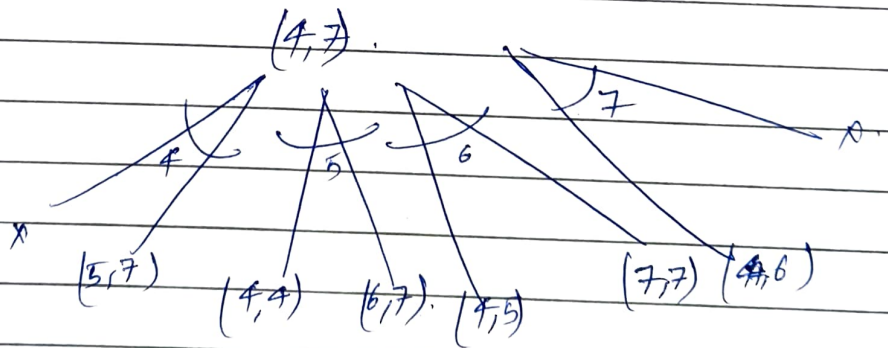
$\Rightarrow ((l_0 + r_0), (l_1 * r_1), (l_2 + r_2) \ldots) + S$. — ②.

of for loop

& we can calculate sum before / last and add them into answer



for :-



Sum = $f_e + f_f + f_g + f_h$.

same as prefix sum calculation.

APCO
Date: / /

A code in

```
int fn(int keys, freq, si, ei, dp){
    if(dp[si][ei] != -1) return dp[si][ei];
    int minCost = (int)1e9;

    int sum = 0;
    for(int cut = si; cut <= ei; cut++){
        int leftTree = (cut == si) ? 0 : OBST(keys, freq, si, cut-1,
                                              dp);
        int rightTree = (cut == ei) ? 0 : fn(keys, freq, cut+1,
                                             ei, dp);

        sum += arr[cut];
        minCost = min(minCost, leftTree + rightTree);
    }

    return dp[si][ei] = minCost + sum;
}
```

APCO
Date: / /

Tabulation:

Same as gap strategy.

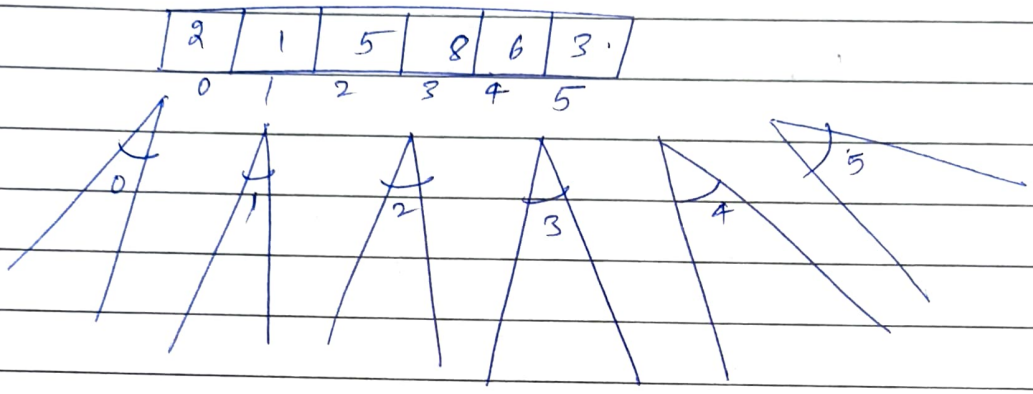
3	4	5	2	3	X
0	1	2	3	4	

	0	1	2	3	4
0	3.				
1	X	4			
2	X	X	5		
3	X	X	X	2	
4	X	X	X	X	3.

APCO
Date: / /

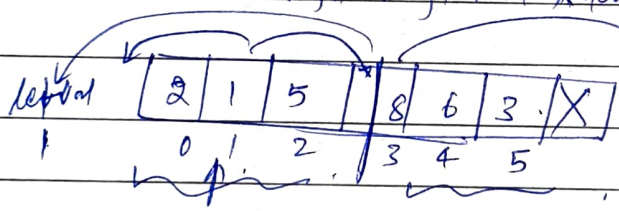
2. Que Burst Balloons.

what :- generate max coin by burst of balloons.



faith :- मैं ith index पर cut मारा तो (0, i-1) and (i+1, ei) की max coin calculate करने को कहा गया

फिर ans = leftVal * leftVal[cut] * rightVal + rightAns



APCO
Date: / /

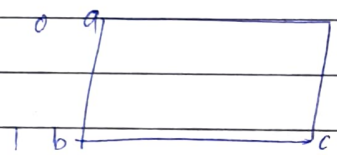
4. Min score triangulation.

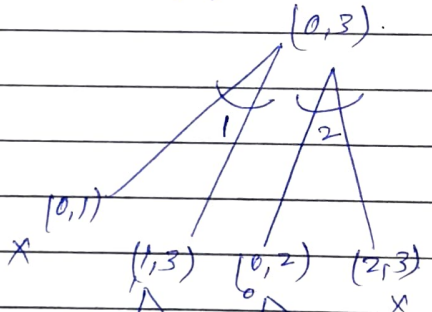
given N sided polygon with vertices A[0], A[1], A[2] ... A[N-1] in clockwise order.
return smallest possible score you can achieve with triangulation of polygon.

I/P = [1, 2, 3].
(0, 2) base.

eg. Path : consider given side (0, n) as base and make cut starting from ~~0~~, and left and right ~~of~~ answer आएगा जो considering (0, ~~2~~) as base. ~~return~~ and path वाला left and right मिन answer देगा।

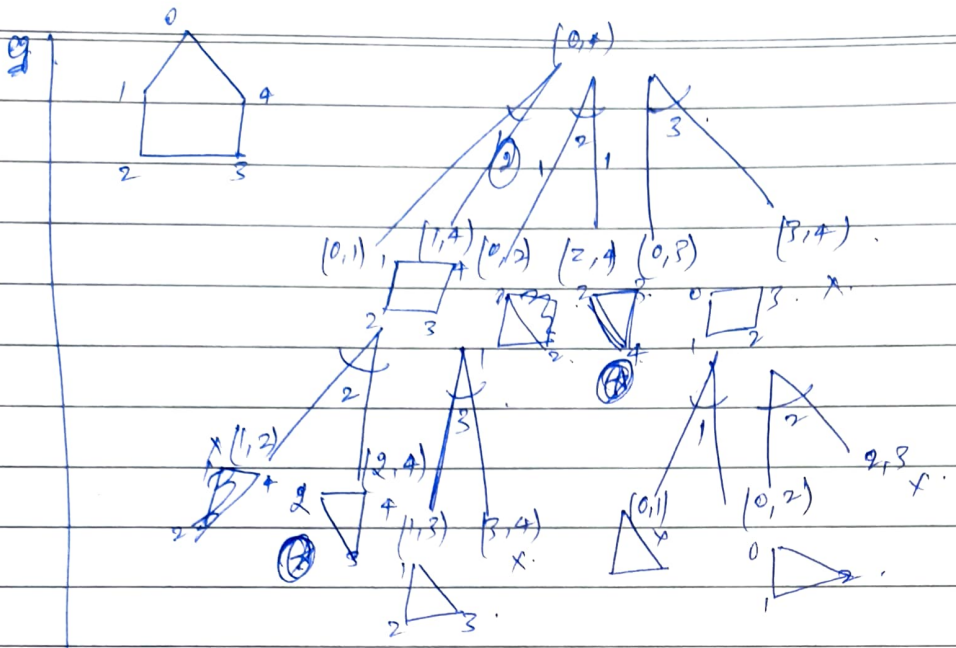
eg 2



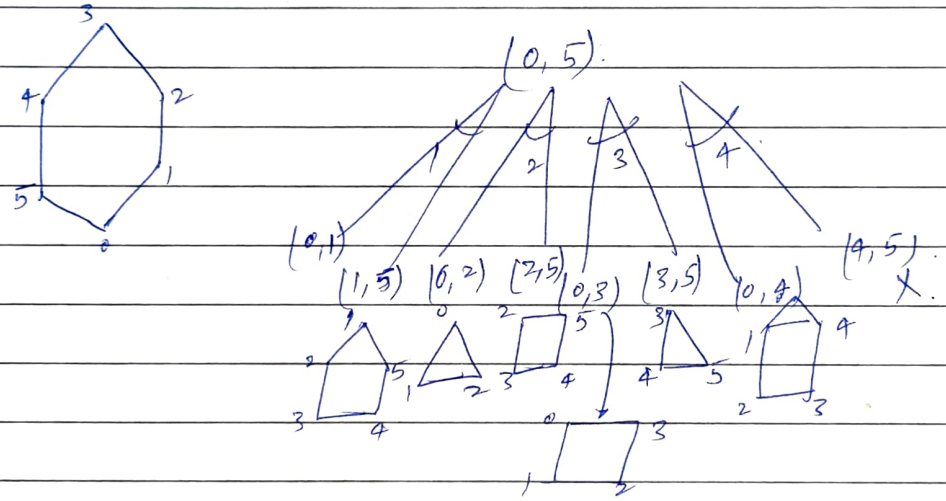


(si+2 = ei) 3 वाला triangle है।

return arr(si) * arr(ei) * arr(si+1)
or cut



overlapping



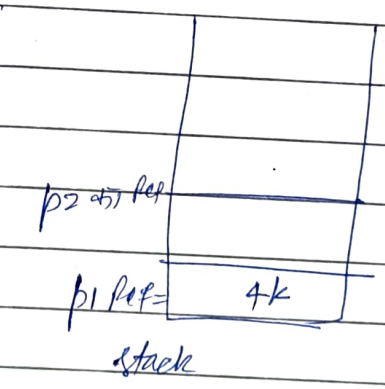
and given all
dig are calculated in

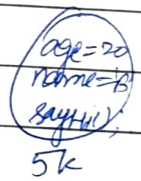
oops (Agenda):

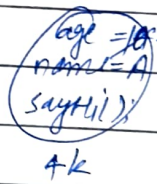
Ref → `Person p1 = new person();` ← object creation
`p1.age = 10;`
`p1.name = "A";`
`p1.sayHi();`

`Person p2 = new person();`
`p2.age = 20;`
`p2.name = "B"`
`p2.sayHi();`

`p3 = p1`
and change in p3.
reflect in p1
`p3.age = 40`



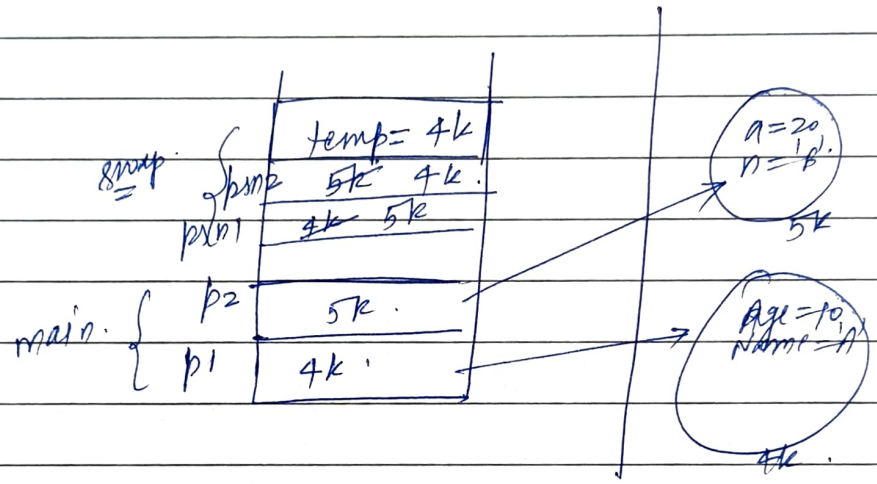




APCO
Date: / /

Swap - 1

```
Public static void swap1( Person psn1, Person psn2){
    Person temp = psn1;
    psn1 = psn2;
    psn2 = temp;
}
```




```
p1.sayHi();
p2.sayHi();
swap(p1, p2);
p1.sayHi();
p2.sayHi();
```

→ after this references get erased and come to initial stage.

APCO
Date: / /

swap2.

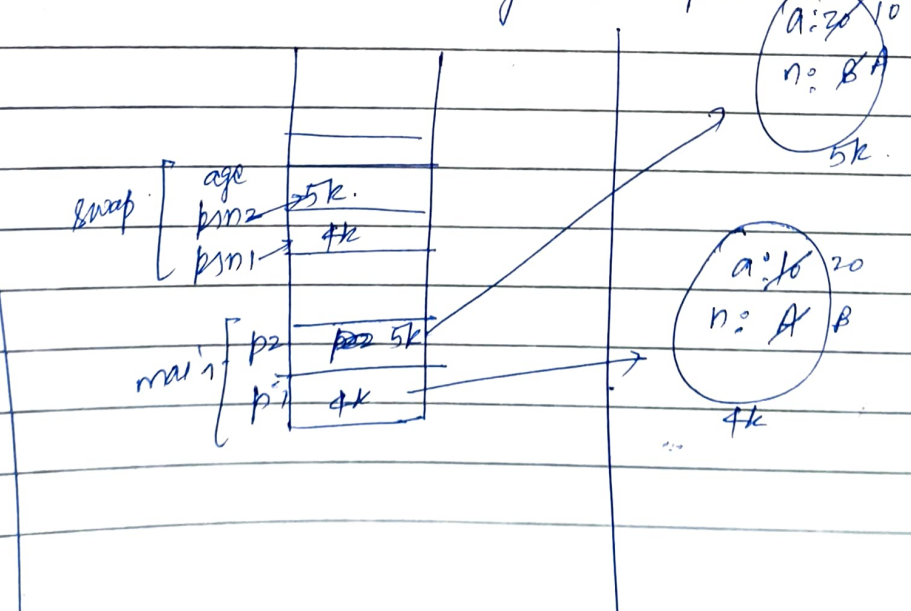
```
Public static void swap1(Person psn1, Person psn2){

    int age = psn1.age;
    psn1.age = psn2.age;
    psn2.age = age;

    String name = psn1.name;
    psn1.name = psn2.name;
    psn2.name = name;

}
```

o/p. A [10] says Hi
B [20] says Hi

swap1().

B [20] say Hi
A [10] says.

get swaped because reference get accen to memory of heap.



APCO
Date: / /

Swaps:-

Public static void swap1 (Person psn1, Person psn2){

Person psn1 = new person();
int age = psn1.age;
psn1.age = psn2.age;
psn2.age = aage;

a=0, n=""
p1 does not change since new instance created.
p2 change same instance.

Person psn2 = new Person();
String nam = psn1.name();
psn1.name = psn2.name();
psn2.name = name;

name doesn't swap because new instance created.

}

APCO
Date: / /

lect-10. (dp).

Agenda (30/11/2020)

1. ~~Partition of sample space.~~
2. ~~Total probability theorem.~~
3. Boolean Paranthezation.
4. Pallindromic cut (132).
5. 1278 / Pallindromic cut-3.

APCO
Date: / /

(1) Leetcode 9fg df-57 (Boolean parenthesization).

Count no. of ways of paranthesization of expression, so that expression evaluates to true.

eg $\{T, F, T\}$ $\{\wedge, \&\}$.

symbol (operator).

$\{T \wedge F \& T\}$.

eg $(T|T \& F \wedge T)$

$((T|T) \& (F \wedge T))$

$(T \& T)$

T.

$((T|(T \& F)) \wedge T)$

$((T|T) \wedge T)$

$(T \wedge T)$.

F

$T|((T \& F) \wedge T))$.

$(T|(F \wedge T))$.

$(T|T)$

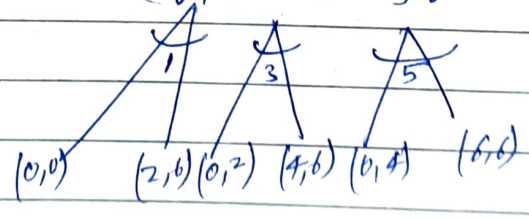
T Ans.

we do it using our strategy:-

when data is given in string format

$(T|T \& F \wedge T)$.

0 1 2 3 4 5 6

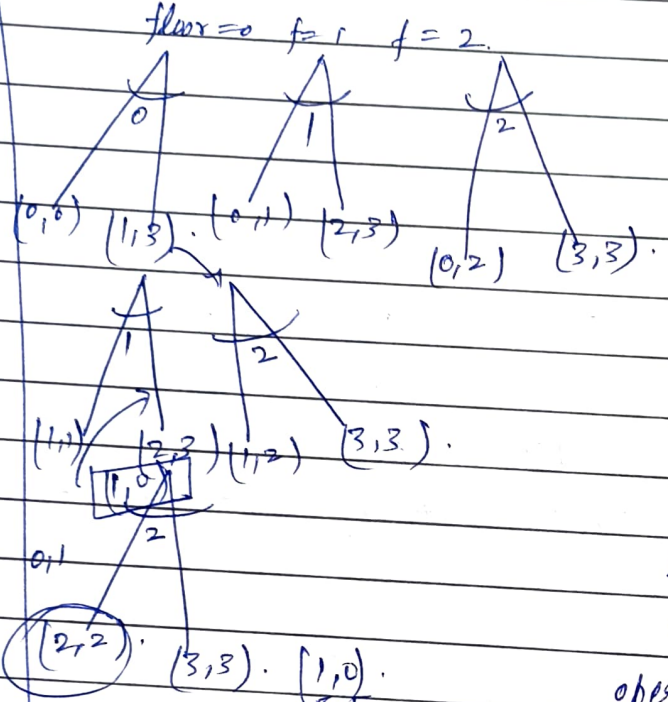


APCO
Date: / /

if given in array form.

T	T	F	T.
0	1	2	3.

0.5 1.5 2.5



1	^	1.
0	1	2

l = (0,1)

r = (1,0).

operator = ^

TC = 0+1 = 1

FC = 0+0 = 0.

base case

(0,1)

T, F

return T = 0 (bcoz 2 index = F)

F = 1 (bcoz 2 index is F).

faith :- left वाला ने total ways of true generation and total ways of false generation दिया होगा same for right call ने.

APCO
Date: / /

if '^'

^ → T → $T_{lc} * F_{rc} + F_{lc} * T_{rc}$.

F → $T_{lc} * T_{rc} + F_{lc} * F_{rc}$.

if 'v'

v → T → $T_{lc} * T_{rc}$.

F → Total count − ways of true generation.

Total Count = $(T_{lc} + F_{lc}) * (T_{rc} + F_{rc})$

Total Count = $T_{lc} * T_{rc} + T_{lc} * F_{rc} + F_{lc} * T_{rc} + F_{lc} * F_{rc}$.

if '!'

'!' → T → Tc − ways of false generation

→ $F_{lc} * F_{rc}$.

		^	v	!
0	0	(0)	0	(0)
1	0	1	0	1
0	1	1	0	1
1	1	1	(1)	(0)

Calculate Min of T & F & Subtract to Calculate other.

APCO
Date: / /

Modulus property

do modulus after any operation. if given to keep answer in range.

① $(a+b) = (a\%m + b\%m)\%m$.

② $(a-b) = (a\%m - b\%m) + m)\%m$.

Here 'm' is added becoz.
if $b > a \Rightarrow$ -Value. $+ m \Rightarrow [0, m)$.

③ $(a * b) = (a\%m * b\%m)\%m$.

// Code :-

```
Public static void main (String [] args) throws IOException
    Scanner scn = new Scanner(new BufferedReader(new
    while(t-->0){  InputStreamReader(System.in));
    int n = scn.nextInt();
    scn.nextLine();
    String str = scn.nextLine();
    int N = str.length();

    Pair[][] dp = new pair[N][N];
    sysoln(boolean Paren (str, 0, N-1, dp). Tcount % mod);
}

}.
```

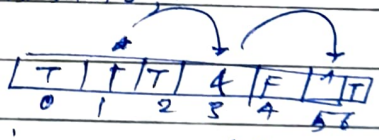
```
Public static class pair {
    int TCount = 0;
    int FCount = 0;

    Pair (int T, int F) {
        this.TCount = T;
        this.FCount = F;
    }
}
    static int Mod = 1003;

# Public static void EvaluateBooleanAns(Pair leftAns, pair rightAns,
    char oper, pair Ans) {

int totalways = ((leftAns.TCount % mod + leftAns.FCount % mod) % mod
                * (rightAns.TCount % mod + rightAns.FCount % mod)
                ) % mod ) % mod.

if (oper == '|') {
    int fCount = ((leftAns FCount % mod * rightAns FCount % mod)
                % mod;
    ans.TCount += (totalways % mod - fCount % mod + mod) % mod;

    ans.fCount = fCount;

} else if (oper == '&') {
    int tCount = ((leftAns TCount % mod * rightAns.TCount % mod)
                ) % mod;
    ans.TCount += tCount;
```

APCO
Date: / /

```
ans.Fcount += (totalways % mod - tcount % mod + mod) % mod;
} else {
    int tcount = ((leftAns.Tcount % mod * rightAns.Fcount % mod
                 % mod + (leftAns.Fcount % mod * rightAns.Tcount
                 % mod) % mod;
    ans.Tcount += tcount;
    ans.Fcount += (totalways % mod - tcount % mod + mod) % mod;
}
}
```

Public static pair_ boolean Paren (string str, int si, int ei, pair_ [][] dp) {

```
if (si == ei) {
    int t = str.charAt(si) == 'T' ? 1 : 0;
    int f = str.charAt(si) == 'F' ? 1 : 0;
    Pair_ base = new Pair_ (t, f);
    return dp[si][ei] = base;
}
if (dp[si][ei] != null) retdp[si][ei];
```




```
pair_ Ans = new pair (0,0);
for (int cut = si+1; cut < ei; cut += 2) {
    pair_ leftAns = booleanfn (str, si, cut-1, dp);
    pair_ rightAn = boolefn (str, cut+1, ei; dp);
    Char oper = str.charAt(cut);
    Evaluate BooleanAns (leftAns, rightAn, oper, ans);
}
Return dp[si][ei] = ans;
}
```

APCO
Date: / /

2. Que 132 / Pallindromic cut Partitioning

what :- given a string 's' partition s. such that every substring of the partition is a pallindrome.

return min cut need for a pallindrome partitioning of s.

eg.

a	a	a	b	b	a	a	c.
0	1	2	3	4	5	6	7

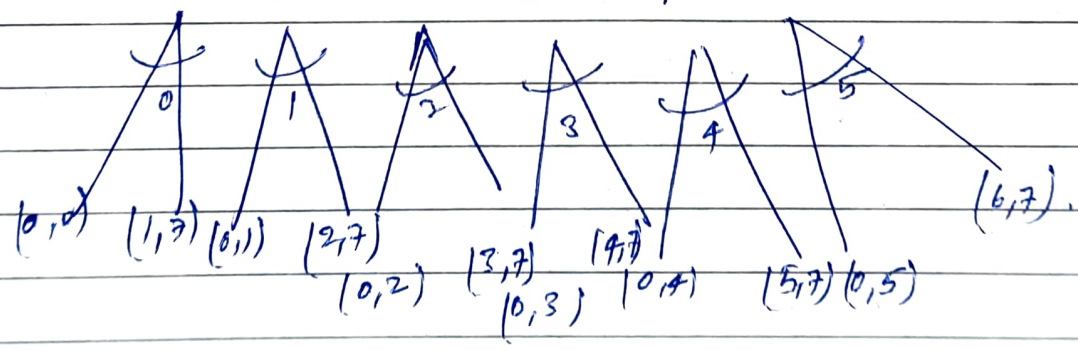
faith :- left = (0, cut).
right call = (cut+1, ei)

ans = left cut + 1 + right cut

if we do cut at every point of string. then. ans is calculated as.

dp(si)(ei) = Min((1 + leftans + rightans), myans)

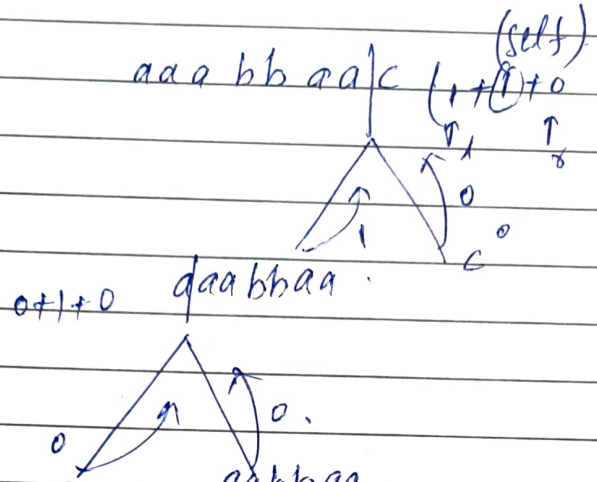
a	a	a	b	b	a	a	c
0	1	2	3	4	5	6	7



APCO
Date: / /

t: O(n³).

eg.



Approach:-

Make cut at every point and send left and right to count pallindromic cut

$$ans = M((lcut + 1 + rcut), ans).$$

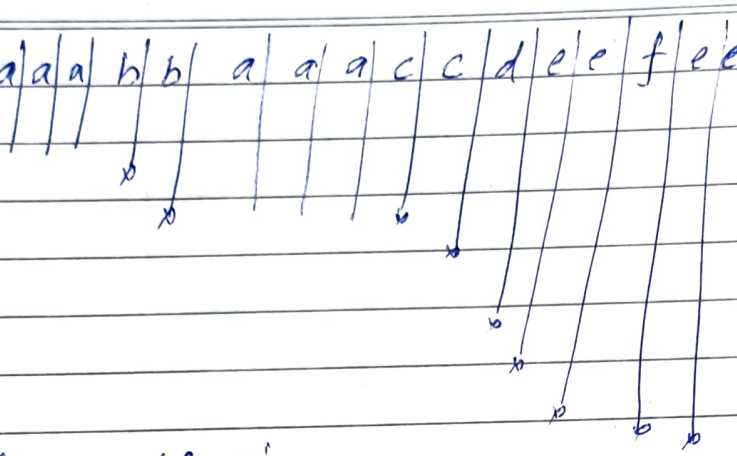
(2.) Approach.

Make cut at only those index left part is already pallindromic and send right part to calculate pallindromic cut.

to check pallindrome use concept of pallindromic substring [lect - 5]

APCO
Date: / /

eg.



✓ are valid cut if $\delta i = 0$.

✗ not valid cut as left part is not pallindrome.

$t = O(n^2)$.

if we pass $f(0, 7)$.

then as only δi is only varying so we need to make only 1-d dp. here.

APCO
Date: / /

// code -

① Public int mincut (String str, int si, int [] dp, boolean [] [] isPallindrome) {
if (si == str.length() || isPallindrome[si][str.length()-1]) {
return dp[si] = 0;
}

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

int ans = (int)1e8;
for (int cut = si; cut < str.length; cut++) {
if (isPallindrome[si][cut]) {
int MinCutCount = Mincut(str, cut+1, dp, isPalin);
ans = min(ans, MinCutCount+1);
}
}

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

② To check substring is pallindrome and to fill its dp. and it gives access in O(1) time of a range of substring whether pallindrome / not

APCO
Date: / /

```
Public int MinCut(String str){
    if(str.length() < 1) return 0;
    int n = str.length();

    boolean[][] isPallindrome = new boolean[n][n];
    for(int gap=0; gap<n; gap++){
        for(int i=0, j=gap; j<n; i++, j++){
            if(gap==0)
                isPallindrome[i][j] = true;
            else if(gap==1)
                isPallindrome[i][j] = str.charAt(i)==
                    str.charAt(j);
            else{
                isPallindrome[i][j] = isPallindrome[i+1][j-1]
                    && (str.charAt(i)==str.
                    charAt(j));
            }
        }

    int[] dp = new int[n+1];
    Arrays.fill(dp,-1);
    return minCut(str,0,dp,isPallindrome);

}
```

APCO
Date: / /

3. ~~One~~ 1278. / Pallindrome Partitioning III.

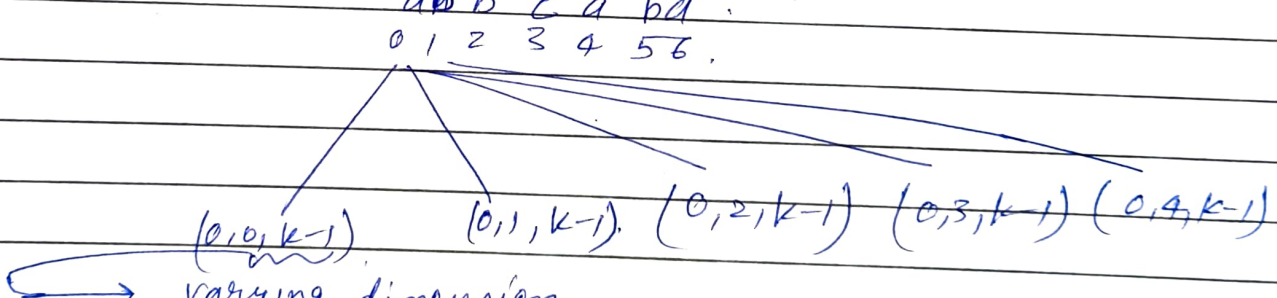
What :- You are given a string 's' containing lowercase letters 's' and integer 'k'.
(1) changing some characters of 's'.
(2) divide 's' into 'k' subset so that they are all pallindrome.

return minimal no. of characters need to change.

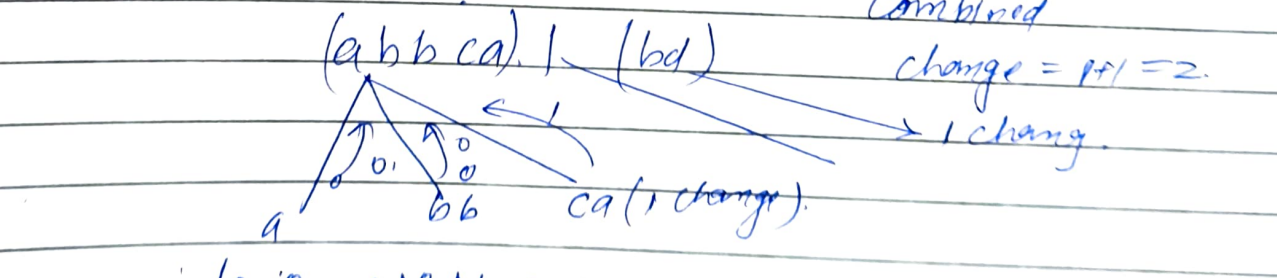
eg $s = "abc"$ $k = 2$

a a c
change = 1

How :-



→ varying dimension.
→ will return minimal changes to convert into pallindromic substring.



~~min~~ :0, 0+0+1 = 1

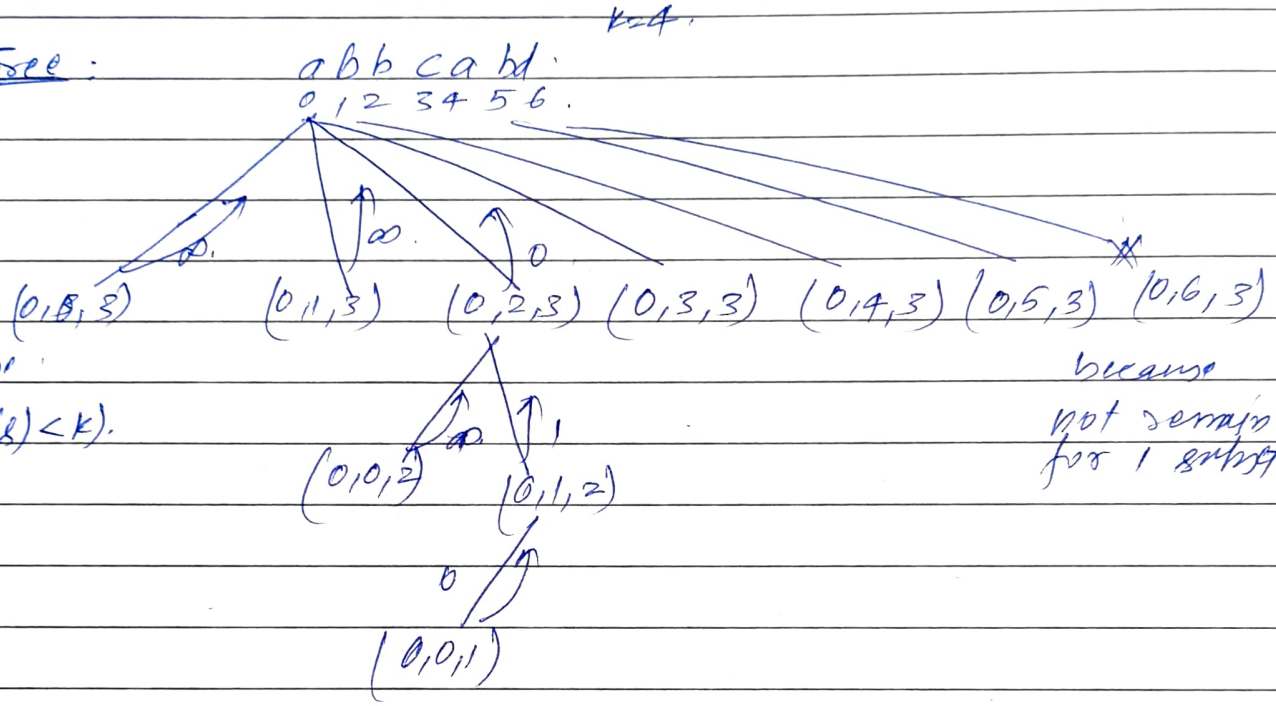
APCO
Date: / /

bd $\xrightarrow{1 \text{ change}}$ (bb).

(abbca) (bb).

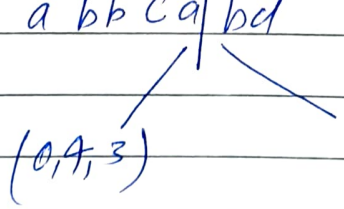
total change = 1+1 = 2

Tree:



faith:- हर बार cut मारा कर कुछ को min changes के साथ (k-1) subset वाले palindrome में convert करके दे आओ

& return min changes required to convert into (k-1) subset palindrome.



→ we need to make min change Required to convert string into pallindrome at dp.

a bb c c d e f f

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

if
s(i) ≠ s(j). मेरा change 1 + x होगा
↓
(i+1, j-1) 'x' change.
↳ To convert into pallindrome.

if
s(i) == s(j). 'x' change
↓
s(i+1, j-1) x change.

APCO
Date: / /

// code.

① Public int PallindromePartition(int si, int ei, int k, int[][] dp, int[][] pdp){

store changes required to convert

if(k==0) return dp[k][ei]=0;
if(k==1) return dp[k][ei]= pdp[si][ei];
if(ei-si+1<=k) return dp[k][ei] = (ei-si+1<k)?1e8:0;

if(dp[k][ei] !=-1) return dp[k][ei];

int min = int(1e8);

for(int cut = si; cut < ei; cut++) {

(not cut till ei because faith does not cut & ① subseq not made).

int recAns = pallindromePartition(si, cut, k-1, dp, pdp);
min = min(min, pdp[cut+1][ei] + recAns);

(1 rem. subseq) faith ✓ Cal.

}
return dp[k][ei] = min;

}

② Public int PallindromePartition(string str, int k) {
if(str.length() ==0 || k==0 || str.length() <=k)
return 0;
int n = str.length();
int[][] pdp = new int[n][n];

APCO
Date: / /

```
for (int gap = 1; gap < n; gap++) {
    for (int i=0; j = gap; j<n; i++, j++) {
        pdp[i][j] = pdp[i+1][j-1];
        if (str.charAt(i) != str.charAt(j)) pdp[i][j]++;
    }
}
int[][] dp = new int[k+1][n+1];
for (int[] d: dp) Arrays.fill(d, -1);
return palindromePartition(0, n-1, k, dp, pdp);
}
```