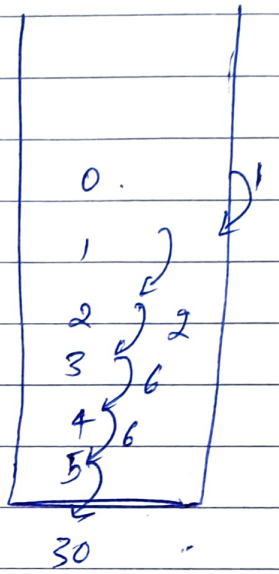
Date :
Page :

① Faith.
② Tree.
③ Code.

Ques 1.

```
f(n)
n==0 ? 1 : f(n-1) * n
```



Ques 2. $a^b = ?$ ⟶ n calls.

$a^b = (a^{b/2}) \cdot (a^{b/2})$.

2, 0
2, 1
2, 3
2, 6
2, 12
2, 25
2, 50
2, 100

let $x = a^b$.

$b, \frac{b}{2}, \frac{b}{4}$ . . . . . 4, 2, 1.

$r = 2$

$a = 1$

$a_n = b$.

total steps = $\log_2 b$

eg. $(a^{10})^{10000}$ =

total terms = $\log_2 (b) = \log_2 10^{10000}$

$= 10^4 \times \frac{\log_{10} 10}{\log_{10} 2} =$

// Code:

```
Public static int powerBtr(int a, int b){
    if(b==0) return 1;
    int ans = powerBtr(a, b/2);
    ans *= ans;
    return b%2 != 0 ? ans*a : ans;
}
```

Recursion with array.

~~faith~~ → ~~a tree~~.

① | 10 | 20 | 30 | 40 | 50 | 60 | 70 | 80. |

Print array in ↑ing order.

② odd index वाली पहले print करो फिर even index.

10	20	30	40	50	60.
0	1	2	3	4	5

5	6	2	5	2	10	11	12	5	6	12	18	5	19	20	15	*
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	

if (idx == arr.length

0	3	8	10 .

4k heap.

13, 5
12, 4
11, 3
10, 3
9, 3
8, 3
7, 2
6, 2
5, 2
4, 2
3, 2
2, 1
1, 1
0, 1

```
if(arr[idx] == data) count++;

int[] smallAns = allIndex(arr, idx, data, count);

if(arr[idx] == data) smallAns[count-1] =
```

arr [ ] [ ] [ ]

Set 3 :

Recursion
- void type → Base से ans मिलेगा।
- Return type → top पर ans मिलेगा।

vector (why use it) → (dynamic)	Arraylist → (dynamic)
① vector stl (google) करो to know	① add
① push_back().	② size
② size();	③ get() (idx).
③ arr [idx] ~~= val~~ (get)	④ arr.set(idx, val);
④ arr [idx] = val (put)	
string str = "abcd"	String str = "abc";
1. str.length().	① str.length();
2. char ch = str[idx];	② char ch = str.charAt(idx);
③ str.substring(idx, length) → O(n).	③ str.substring(si, ei+1); (3, 5+1)
eg. [3, 3] — idx, length	

(abc)

↓

(bc)

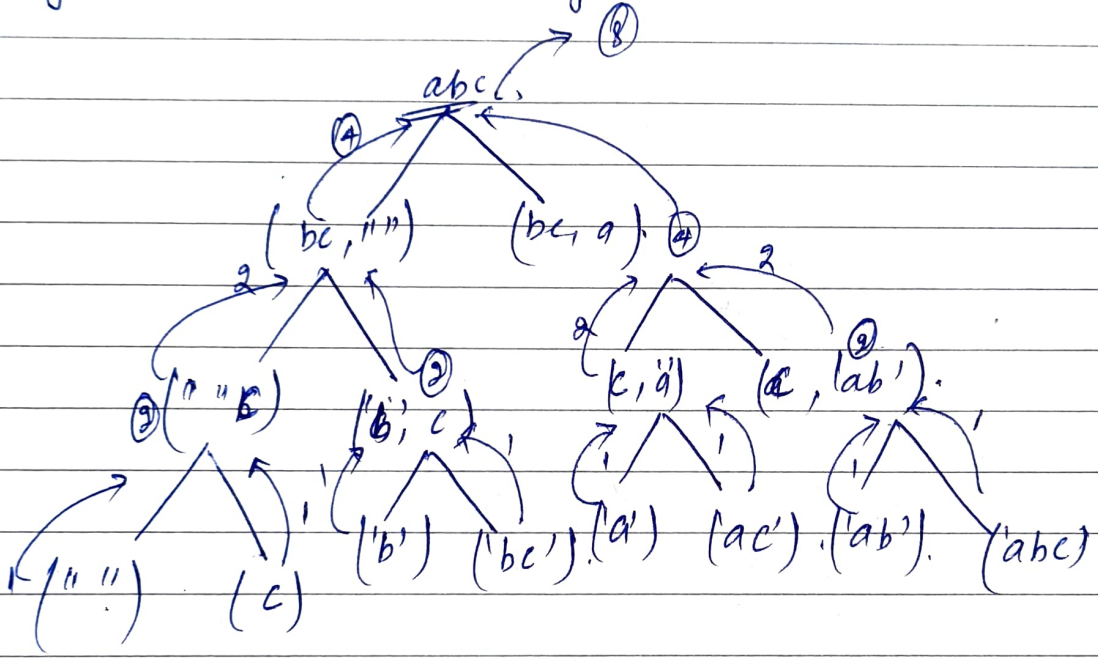
↓

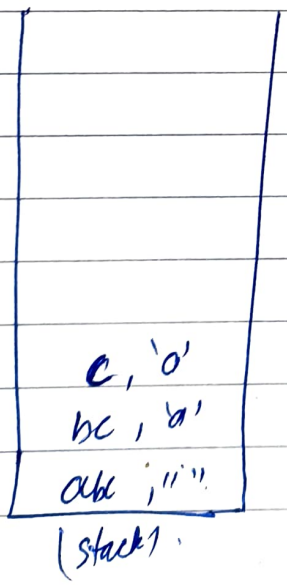
c

↓

("").

Arraylist can be printed directly.





APCO
Date :
Page :

keypad problem.

"568"

(17) Leetcode.
(90) Que.
78. Que.
91 Coun ways.

encoding.

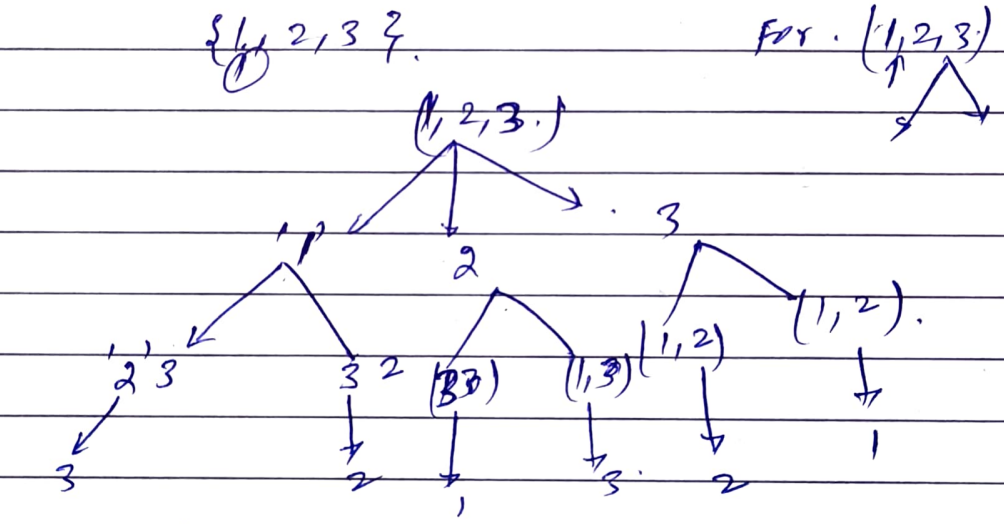
APCO
Date :
Page :

(28/08/2020) (lect-2)

1. 46, 47 *all permutation and combination of string.
2. Rat in maze, multiple jump., in all possible direction.

C.W.

{1, 2, 3}.

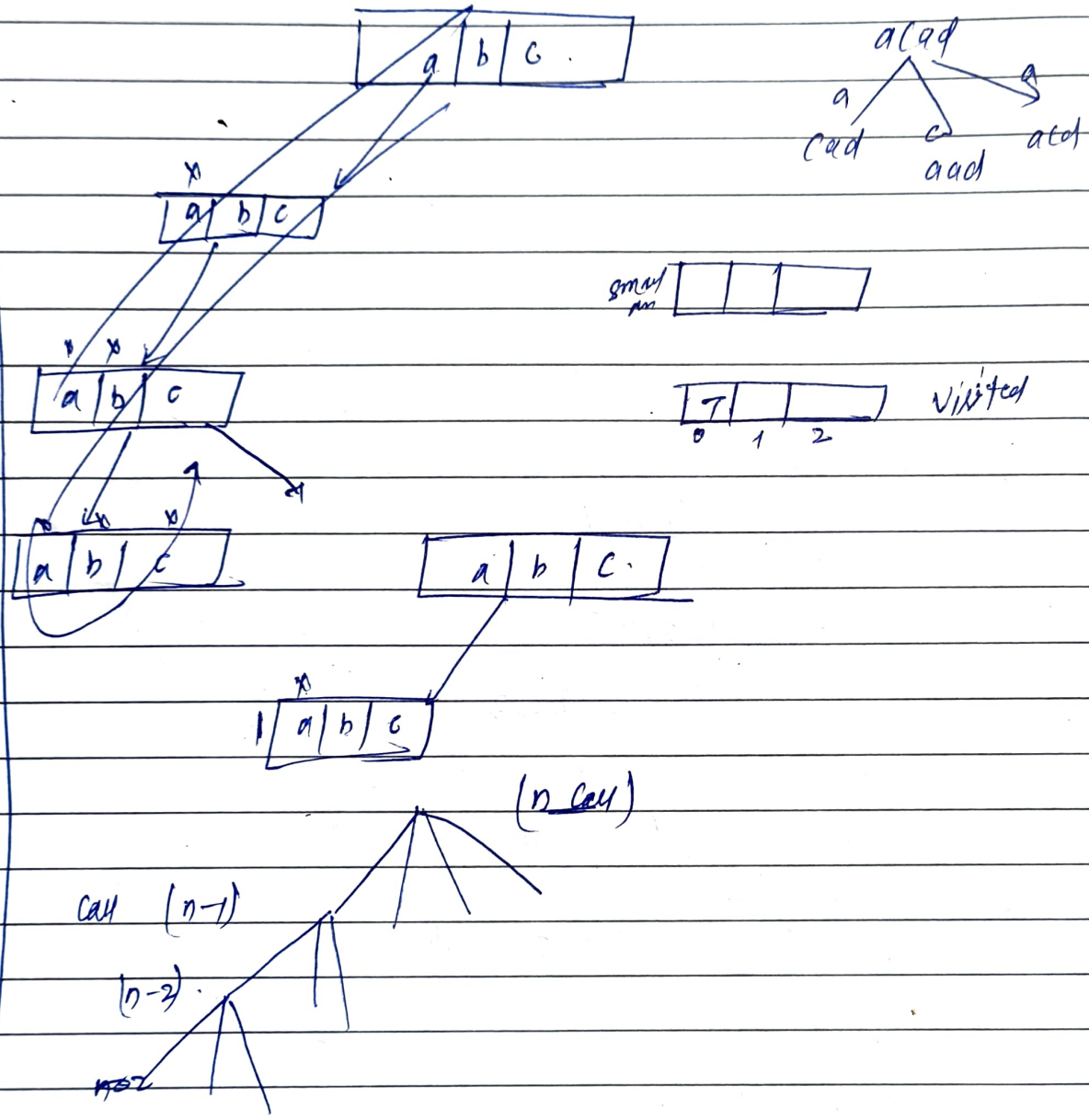


(1,2,3) (1,3,2).

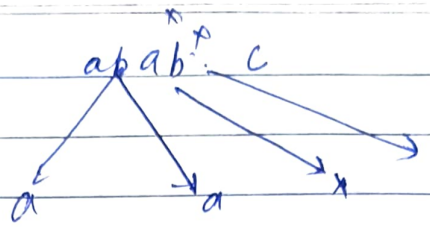
aedeaf (acad).

APCO
Date :
Page :

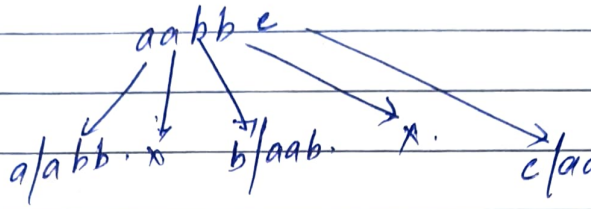
Faith → a से start होने वाली permutati



APCO
Date :
Page :



$\frac{4!}{2 \times 2!} = \frac{24}{4} = 6$



① count == length.
Result में add करो small one.

① String.
② Num के लिए using ① array
② Hash set
③ sorted array (prev, curr).

on class level java and cpp behave same.

M-2.

a	b	b	c	c

Counter is to check कितने character us हुए या नहीं

APCO
Date :
Page :

1. Qu. Print 'subsequence' of return type.

'abc'

f('abc', 0) → ["" | a | b | ab | c | ac | bc | abc] (all subsequence)

↓

f('abc', 0+1) → ["" | b | c | bc]

↓

f('abc', 1+1) → ["" | "" + c = c]

↓

f('abc', 2+1)

↓

return "" → [""]

faith if i'm at ith index तो i+1 index से उम्मीद करता हूँ कि वो अपने सारे subsequence ले आएगा तो मैं उन सब subsequence से अपने level पर subsequence बना दूँगा

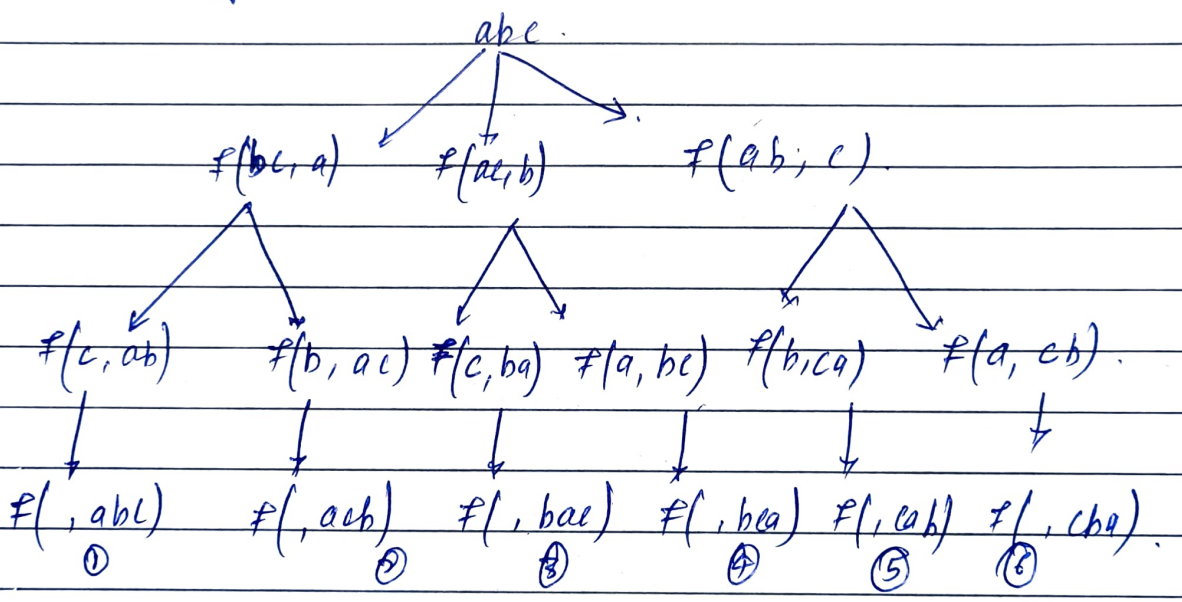
// code.

```
vector <string> subsequence (string &str, int idx){
        if(idx == str.length();
        vector <string> base;
        base.push_back("");
        return base;
    }
vector <string> smallAns = subsequence(str, idx+1);
vector <string> myAns;
```

```
for (string s: smallAns) {
    myAns.push_back(s);
    myAns.push_back(str[idx] + ch);
}
return myAns;
}
// Call ⟶ subsequence("abc", 0).
```

2. Count Permutation.

eg 'abc'.



permutation = 6 = 3! = 6.

a b c
(i).

ith index खुद अलग होकर बाकी को arrange होने के लिए भेज देता है & खुद को ans में add कर लेता है.

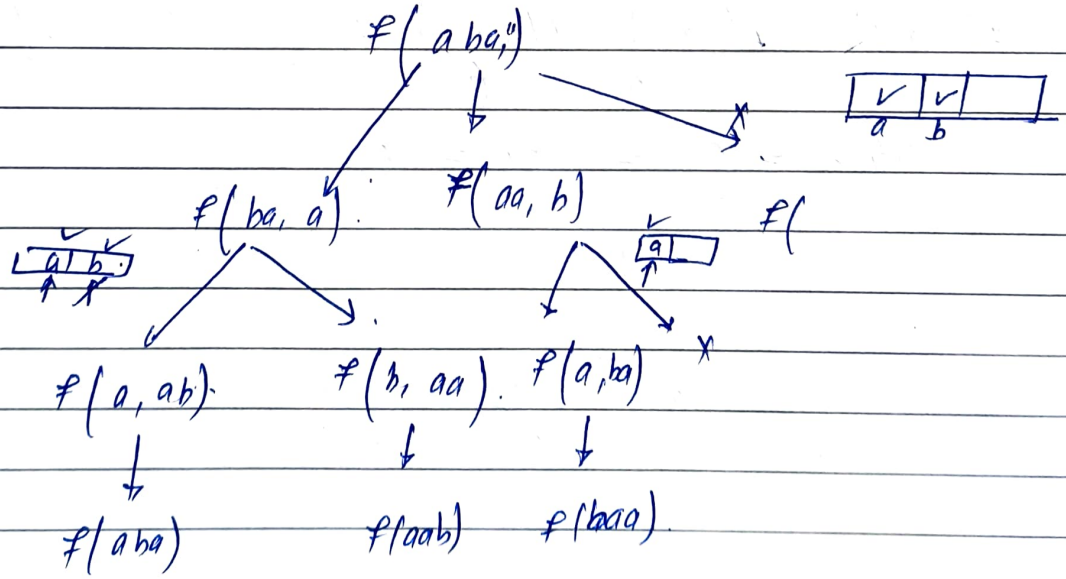
APCO
Date :
Page :

// Code :

```
int Permutation (String str, String ans){
    if( str.length() == 0 ) {
        cout << ans << endl;
        return 1;
    }
    int count = 0;
    For ( int i = 0; i < str.length(); i++ ) {
        char ch = str[i];
        String ros = str.substr(0,i) + str.substr(i+1);
        count += permutation( ros, ans+ch );
    }
    return count;
}
// Call permutation( 'abc', "" );
```

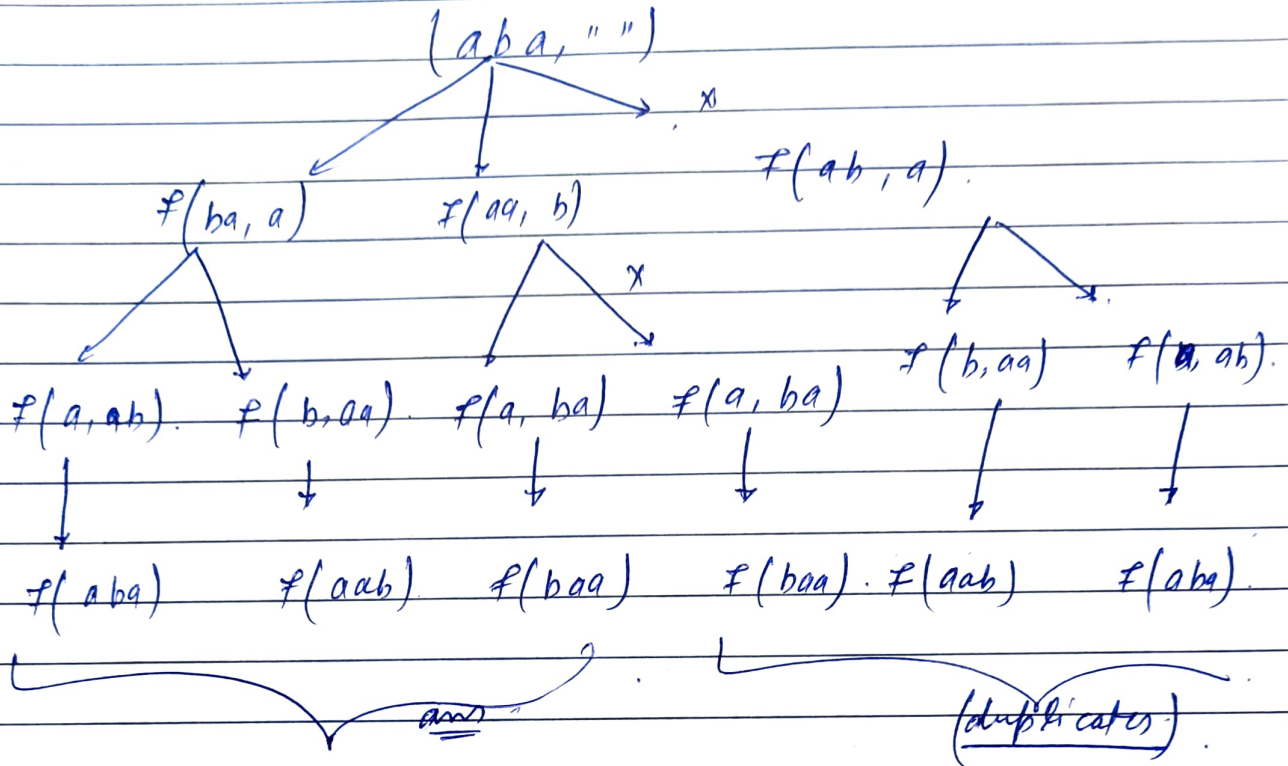
3. Que. Print Permutation without Duplication.

eg. aba.



APCO
Date :
Page :

Reason to take boolean array. to check whether ith character is included in ans or not.



// Code-

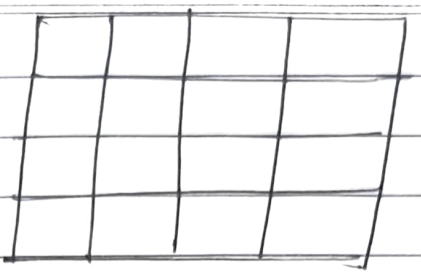
```
int printPermutationwithoutDuplicate (string str, string ans)
{
    if (str.length() == 0) {
        cout << ans << endl;
        return 1;
    }
    int count = 0;
    vector <bool> vis(26, 0);
    for (int i=0; i< str.length(); i++) {
        char ch = str[i];
        if (!vis[ch - 'a']) {
            vis[ch - 'a'] = true;
            string ros = str.substr(0, i) + str.substr(
```

```
        count += permutationWithoutDupli(ros, ans+ch);
    }
}
return count;
}
```

APCO
Date :
Page :

Qu.

① H, V



② H, V, D

③ H, V, D

jump = ∞

#0 V 3 H 1

④ V, L, R, D

⑤ V, N, E, L, R, W, S, D

⑥

jump ⇒ ∞

(4, 5, 6) (3×3)

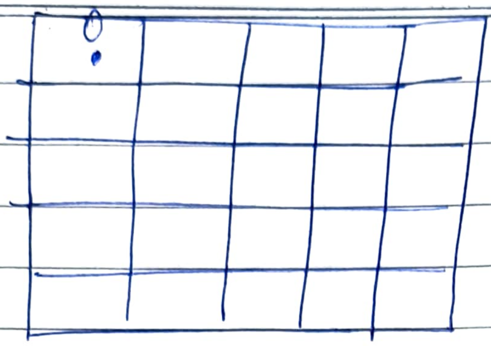
⑦

jump = ∞

6 6 6 6 6 6

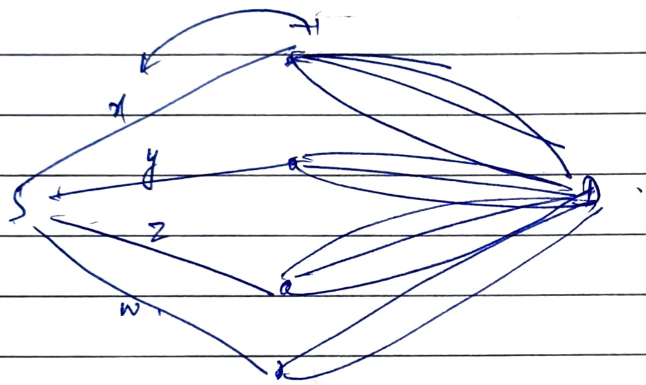
1 2 3 4 5 6

(3-4) hours



① Proactive call → soch समझ के call.

② Reactive Call → base से sb handle किया



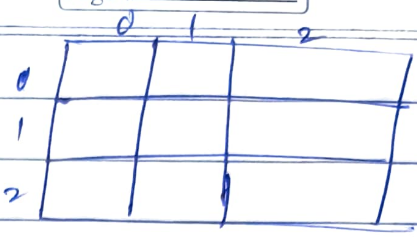
combination gives the complete path.

javac FileName.txt java > output.txt

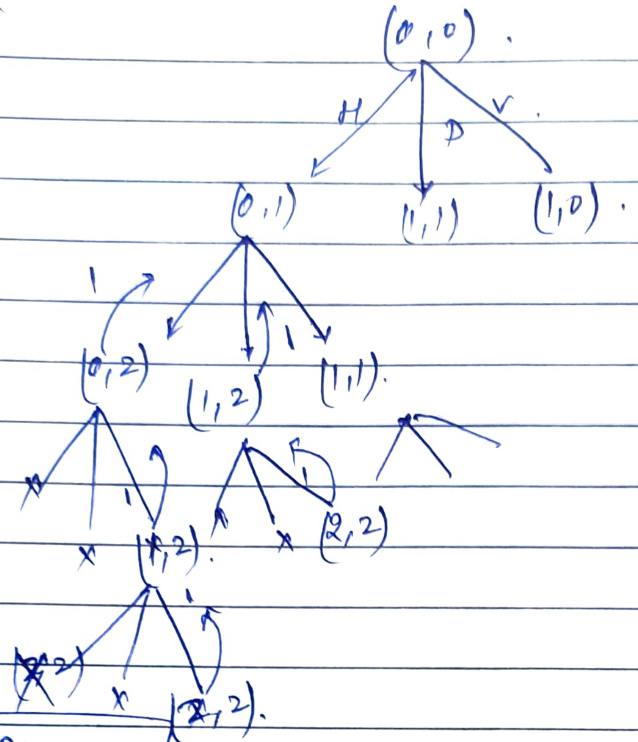
★ → return type वाला recursion is best

void type Recursion आसान बनाता है ★ Clear में Interest करो → benefit समझना आसान

(Code की reality return type होना चाहिए).



① Ex.



faith
(0,0) मुझे सारे path दे देगा

void ① faith

मैं सारे possible तरीके लगा दूँगा और अपने अगले node से expect करूँगा कि वो भी यही करेगा, तो जिस दिन base case hit करेगा वही मेरा ans होगा

Recursion में सारे possible case try होते हैं.

★ ~~Clear~~ // code-

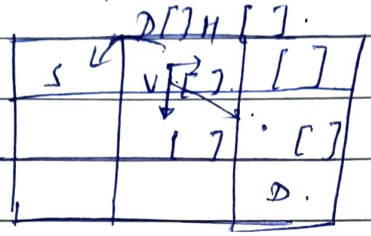
```
public static void mazepath_hv(int[][] arr,
int sr, ~~sc,~~ int sc, int dr, int dc, String path){
    if(sr > dr || sc > dc){
        return;
    }
    if(sr == dr && sc == dc){
        syso(path);
        return;
    }
    fn(arr, sr, sc+1, dr, dc, path+"H");
    fn(arr, sr+1, sc, dr, dc, path+"V");
```

APCO
Date :
Page :

② Arraylist of path का code :-

faith :- अगर मेरे से आगे वाला node अपने सारे path calculate कर लेता है तो मैं उसमें अपना path add करके ans वापस दूँगा।

eg.



// code.

```
Public static Arraylist<String> mazepath_HVD(
int sr, int sc, int er, int ec){
    if(sr==er && sc==ec){
        Arraylist<string> base = new Arraylist<>();
        base.add("");
        return base;
    }

Arraylist<String> myAns = new Arraylist<>();
if(sc+1<=ec){
AL<String> Horizontal = mazepath_HVD(sr, sc+1, er, ec);
for(String s: Horizontal){
    myAns.add("H"+s);
}
if(sr+1<=er){
AL<String> vertical = mazepath_HVD(sr+1, sc, er, ec);
for(String s: vertical){
    myAns.add("V"+s);
}
```

APCO
Date :
Page :

~~Void type of code~~

```
if(sr+1 <= er && sc+1 <= ec){
    Arr<String> diagonal = mazePath_HVD(sr+1, sc+1, er, ec);
    for(String s: diagonal){
        myAns.add("D"+s);
    }
    return myAns;
}
```

(3) HVD with infinite jump.

```
fn(int sr, int sc, int er, int ec, String str){
    if(sr == er && sc == ec){
        syso(ans);
        return 1;
    }

int count = 0;
for(int jump = 1; sc + jump <= ec; jump++){
    count += fn(sr, sc+jump, er, ec, ans+"H"+jump);
}

for(int jump = 1; sr + jump <= er; jump++){
    count += fn(sr+jump, sc, er, ec, ans+"V"+jump);
}

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

return count;
}
```

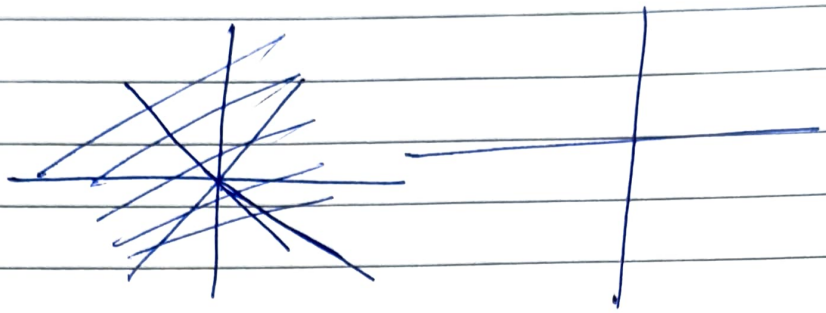
APCO
Date :
Page :

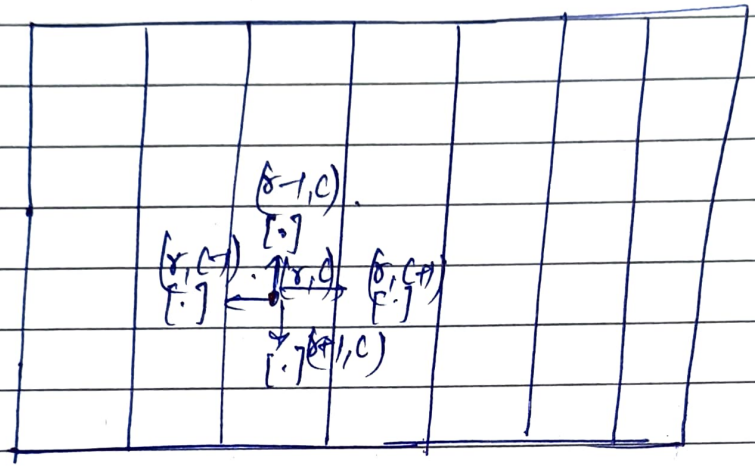
// jump वाला Code - Infinite.

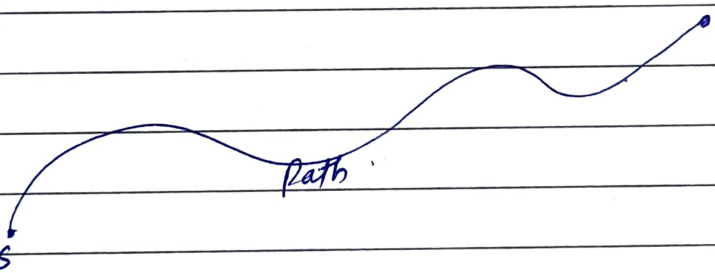
&	[=]	[=]	[=]	[=]	[=]
[=]	[=]	[ ]			
[=]	[=]		[ ]		
[=]	[=]			[ ]	
[=]	[=]				[ ]
[=]	[=]				[ ]

★ Permutation - वाला String को easy के लिए then convert it for array.

Four call.







DFS (src). [Means Recursion or Recursive call]

① Mark source.

② For all unvisited neighbour.
2.1 Call dfs for neighbour.

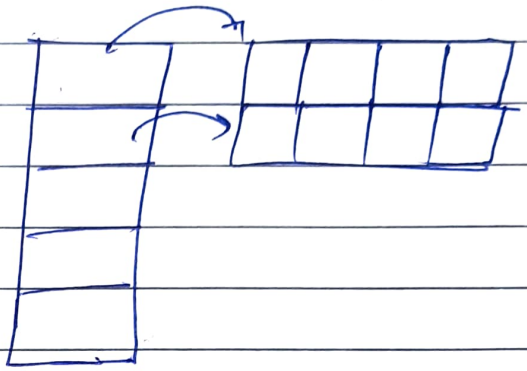
③ Unmark source.

APCO
Date :
Page :

⇒ Flood fill says उपले सारे paths explore कर लो and ans उसको (1 – infinite). is called flood fill algorithm.

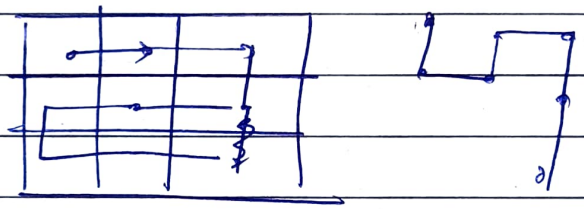
maze path small derivative of flood fill.

Vector < vector <int>

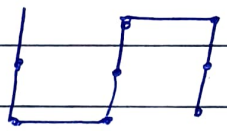


RRDLLDRR.

DRURDD.



DDRUURDD.

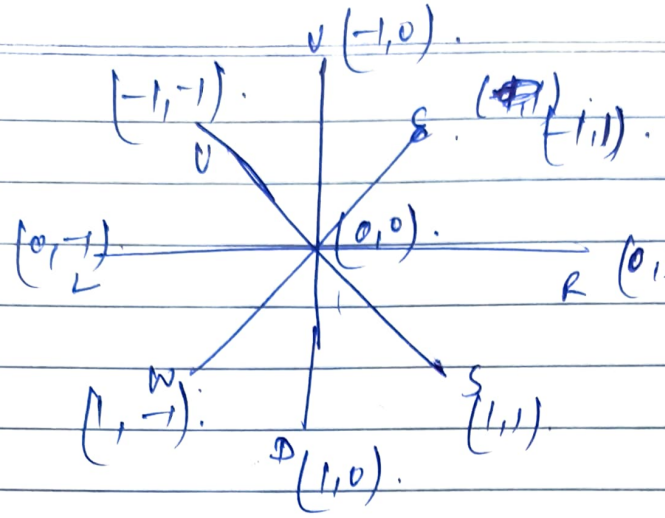


APCO
Date :
Page : 00

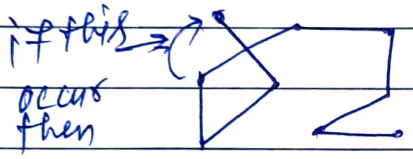
(3) // code. // move in down, left, up and right

```
Vector<Vector<int>> dir {{1,0},{0,-1},{-1,0},{0,1}};
Vector<string> dirs {"D","L","U","R"};

int floodfill(int sr, int sc, int er, int ec, string ans){
    if(sr==er && sc==ec){
        cout<<ans<<endl;
        return 1;
    }
    int count=0;
    vis[sr][sc]=1;
    for(int d=0; d<dir.size(); d++){
        int r = sr+dir[d][0];
        int c = sc+dir[d][1];
        if(r>=0 && c>=0 && r<vis.size() &&
           c<vis[0].size() && vis[r][c]==0){
            count+=floodfill(r,c,er,ec,ans+dirs[d]);
        }
    }
    vis[sr][sc]=0;
    return count;
}
```

APCO
Date :
Page :



~~NWUERDWR. EWUNRDNR.~~
SWUERDWR.



eg. SWRED.

① ② ③ ④
(1,0), (0,-1), (-1,0), (0,1).
[act as direction]

$(sr, sc) = (0,0)$.

jump = 1
$r = 0 + 1 \times 1 \Rightarrow (1,0)$.
$c = 0 + 1 \times 0$

jump = 2
$r = 0 + 1 \times (0) \Rightarrow 1$] $(1,-1)$.
$c = 0 + (-1)(1) = -1$

jump = 1
$(r,c) = (0 + 1(-1), 0 + 1(0)) = (-1, 0)$.

jump = 1
$(r,c) = ($

APCO
Date :
Page :

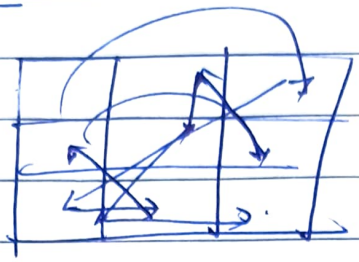
jump allowed in (d, l, u, r, e, w, n, s)

// code.

```
int floodfill_jump(int sr, int sc, int er, int ec, string ans){

    if(sr==er && sc==ec){
        cout << ans << endl;
        return 1;
    }
    int count=0;
    vis[sr][sc]=1;

    for(int jump=1; jump<=max(er,ec); jump++){

        for(int d=0; d<dir.size(); d++){
            int r = sr + jump*dir[d][0];
            int c = sc + jump*dir[d][1];

            if(r>=0 && c>=0 && r<vis.size() &&
               c<vis[0].size(); && vis[r][c]==0){

                count += floodfill_jump(r,c,er,ec, ans +
                         dirs[d] to_string(jump) + " ");
            }

        }
    }
    vis[sr][sc]=0;
    return count;
}
Vector<Vector<int>> dir{{1,0},{0,-1},{-1,0},{0,1},{1,1},{-1,-1},
                        {-1,1},{1,-1}};
Vector<string> dirs{"D","L","U","R","S","N","E","W"};
```

APCO
Date :
Page :

Multiple jumps-



geek fg. backtsaking.

H.W.

1.

Q1 ①

Matrix with hurdles -

1 → Safe

0 → unsafe.

Longest path & length

Q2 Find path from corner cell to middle cell in maze.

Q3 ② Longest possible route in a Matrix with hurdles.

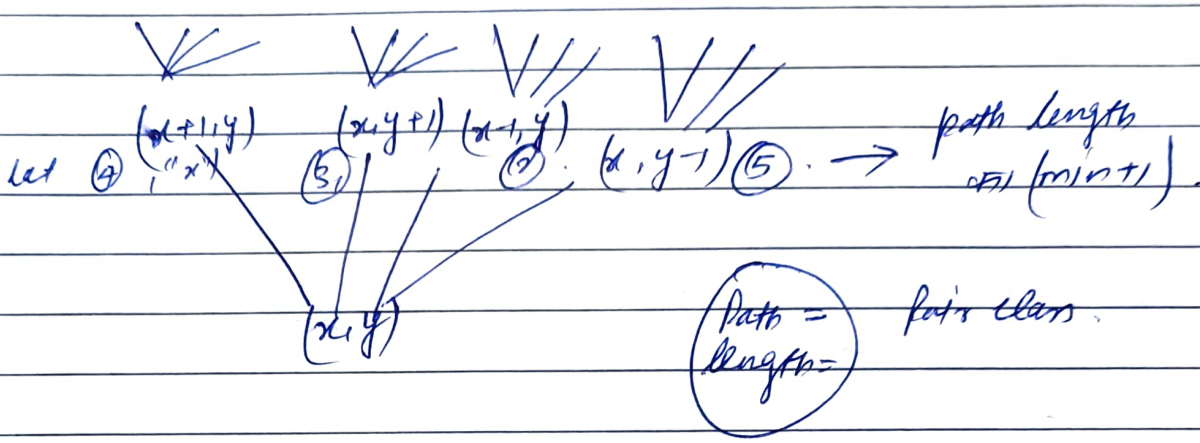
APC
Date :
Page :

(lect-3)

31/08/20

Recursion Question

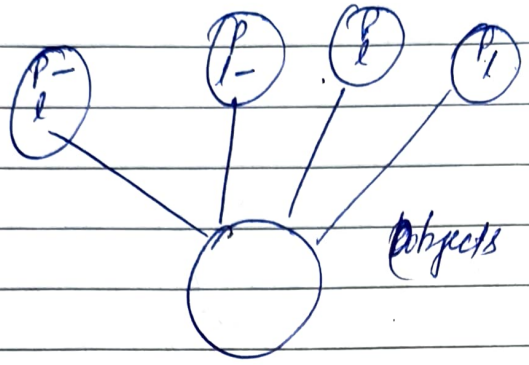
1. Find shortest path (infinite 8 direction with jump 1



Agenda :

1. Shortest path in maze.
2. Knight tour.
3. 1091 shortest path in binary matrix.
4. Permutation → for all call. } Infinite supply of
5. Combination → for all call } coin.
6. Permutation → } only one use of coin } using for loop.
7. Combination → }
8. Permutation → } in two call
9. Combination → }
10. Permutation → } Using subsequence method
11. Combination → } Again 2 call.
12. Make all subset ~~for~~ interviewbit.

—o out ↳ out >



① Knight's tour problem.
② 1091. Shortest path in Binary matrix.

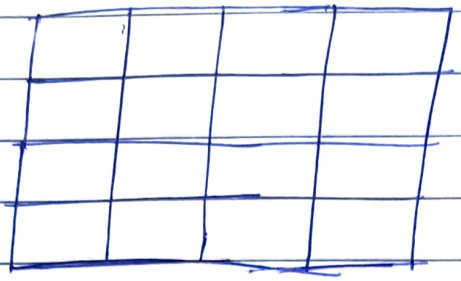
Note ⇒ class → private & struct में by default public होता है ☺

③ // Shortest path in maze code.

```
class pathPair {
public:
    string path;
    int len;

    PathPair(string path, int len) {
        this->path = path;
        this->len = len;
    }
};

PathPair floodfillShortestPath_jump(int sr, int sc, int er,
                                    int ec) {
    if (sr == er && sc == ec) {
        PathPair p("", 0);
        return p;
    }
```

APCO
Date :
Page :

Qu. Knight tour problem. gfg.



$O(8^{64})$ or (n^{64}).

```
int Count = 0;
vis[sr][sc] = 1;
pathpair ans ("", 1e8);
for(int jump = 1; jump <= max(er, ec); jump++){
    for(int d=0; d< dir.size(); d++){
        int r = sr + jump * dir[d][0];
        int c = sc + jump * dir[d][1];

    if(r>=0 && c>=0 && r< vis.size() && c<
        vis[0].size() && vis[r][c]==0){
        pathpair recAns = floodfill shortest path_jump
                          (r, c, er, ec);

        if(recAns.leng+1 < ans.len){
            ans.len = recAns.len+1;
            ans.path = dirS[d] + to_string(jump) + "," +
                       recAns.path;
        }
    }
}
}
```

```
vis[sr][sc] = 0;
return Ans;
}
```

Eg ① // (Knight Tour)

```
static int xmove = {2,1,-1,-2,-2,-1,1,2};
static int ymove = {1,2,2,1,-1,-2,-2,-1};
Public static boolean KnightTour(int[][] board, int sr, int sc, int steps) {
    board[sr][sc] = steps;
    if(steps == 63) return true;
    for(int d = 0; d < xMove.length; d++) {
        int r = sr + xmove[d];
        int c = sc + ymove[d];
        if(r >= 0 && c >= 0 && r < 8 && c < 8 && board[r][c] == -1) {
            boolean res = fn(board, r, c, steps+1);
            if(res) return true;
        }
    }
    board[sr][sc] = -1;
    return false;
}
```

Eg ②

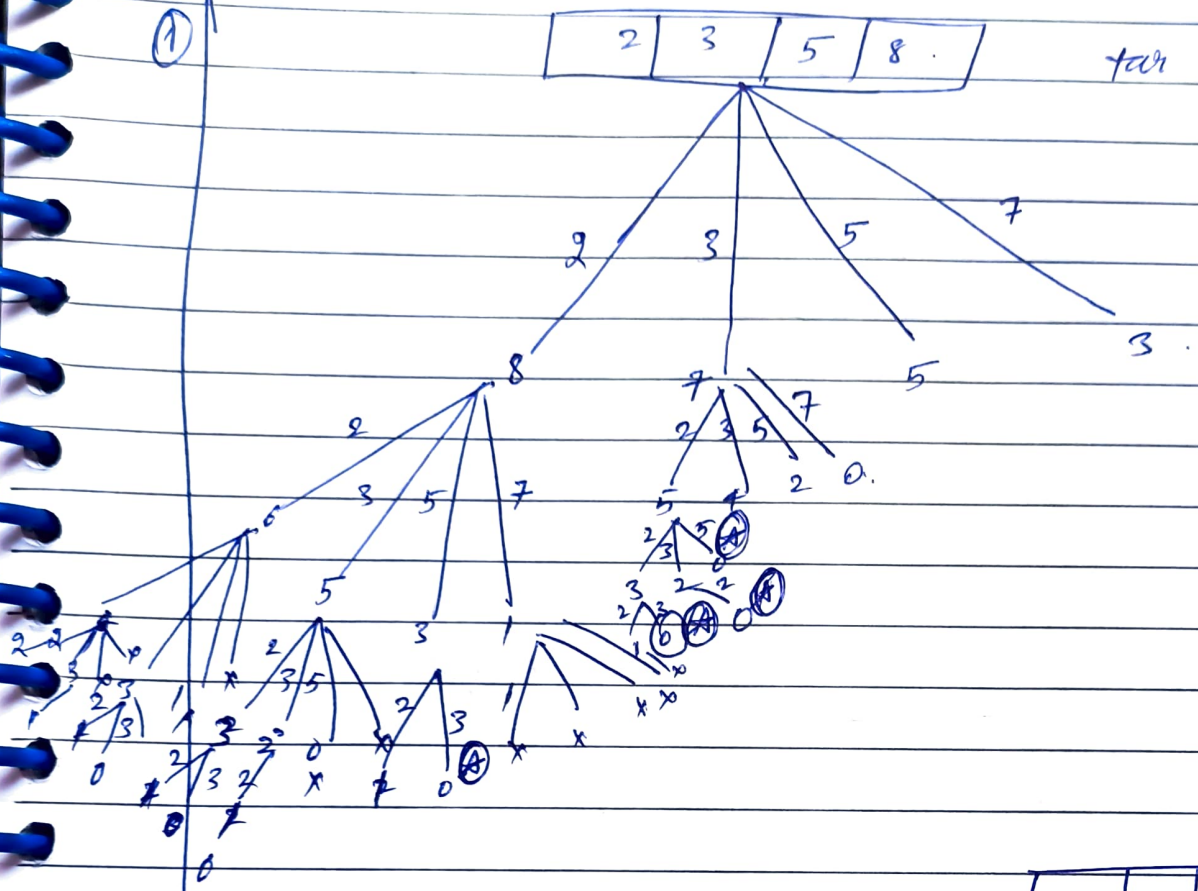
```
Public static void knightTour() {
    int[][] board = new int[8][8];
    for(int[] a: board){
        for(int ele: a){
            sysout(ele + " ");
        }
        sysout();
    }
}
```

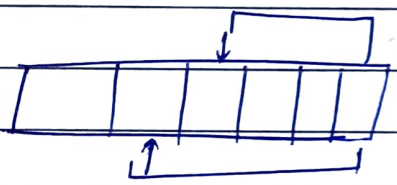
knightTour(board, 0, 0, 0);

APCO
Date :
Page :

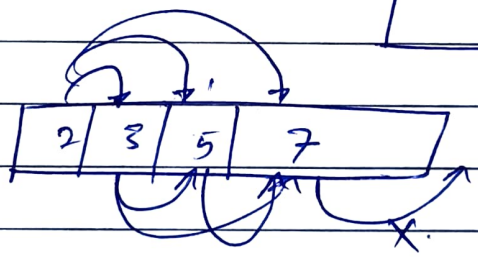
8 Que 8 Que (permutation)

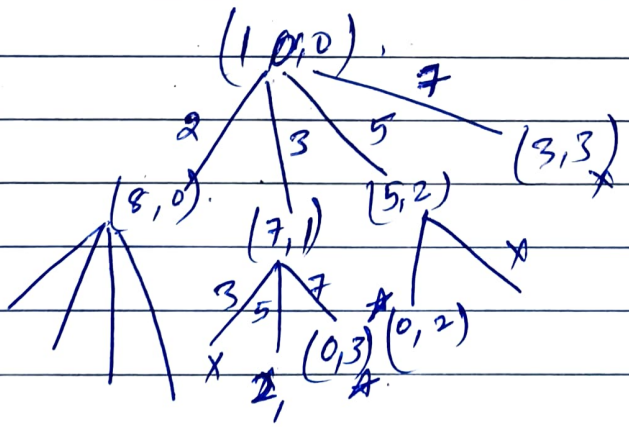
① 2 | 3 | 5 | 8 . ter = 10.



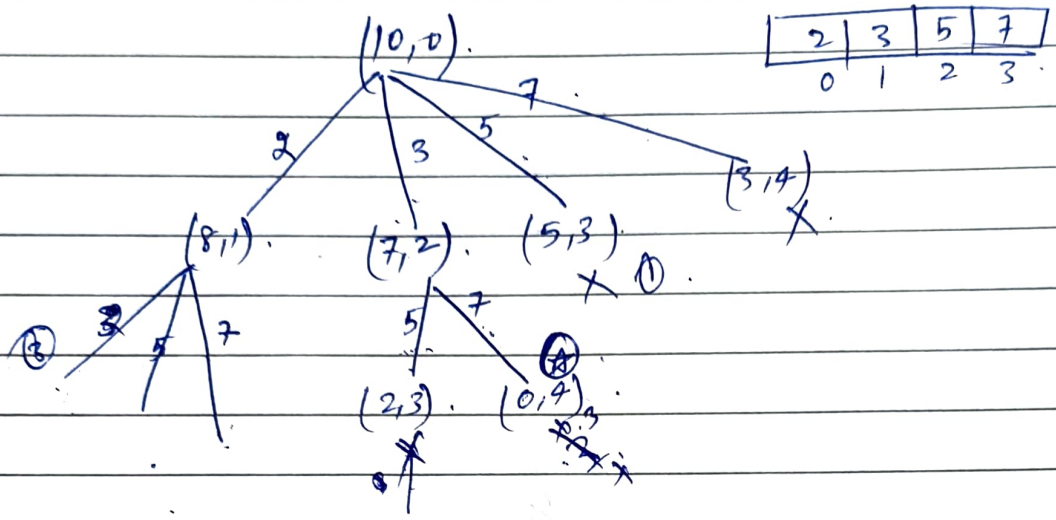


② Combination -





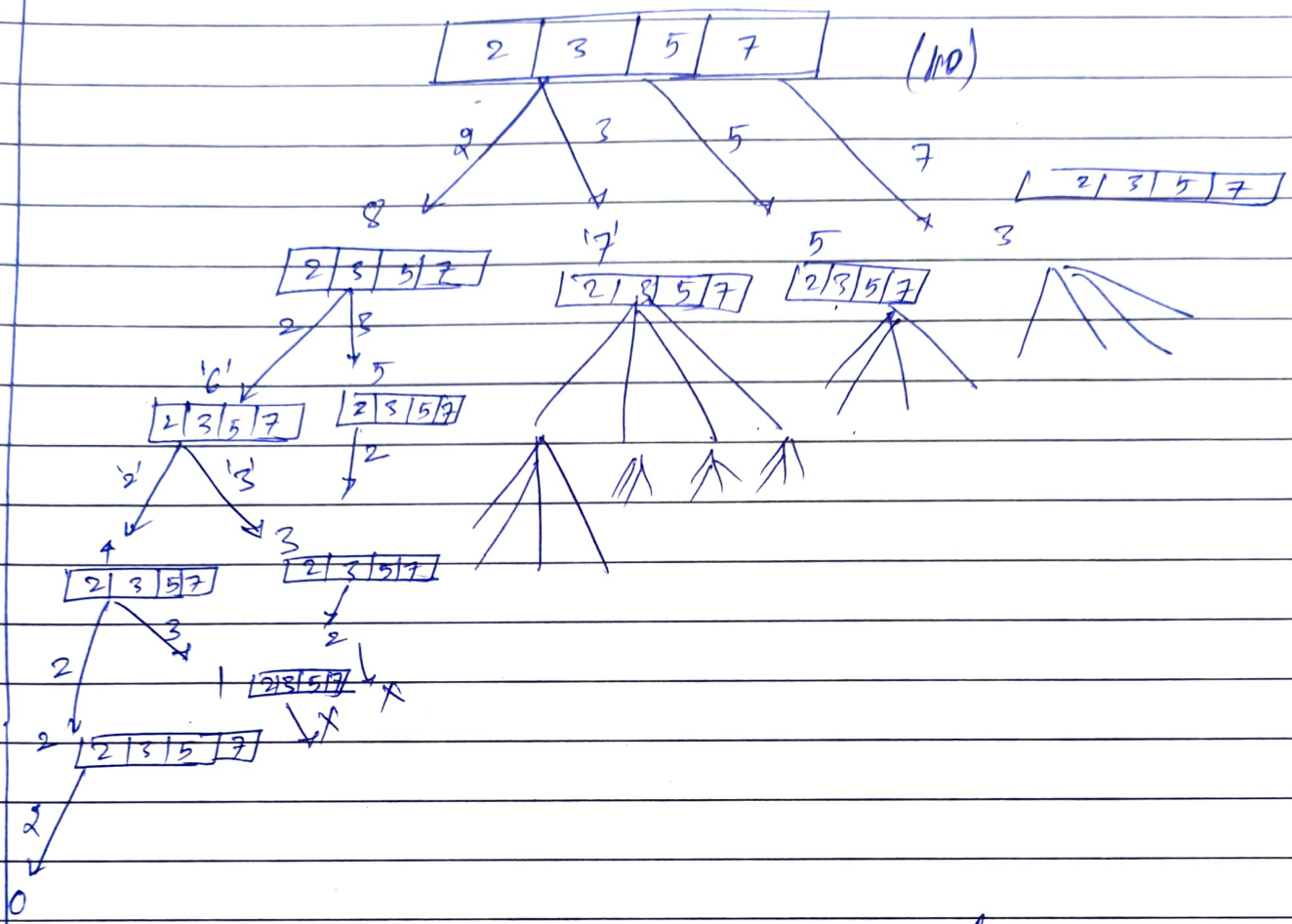
(3) Combination such that coin not use again.



(4) Permutation such that coin not used again.

APCO
Date :
Page :

① Coin change permutation such that infinite supply of coin.

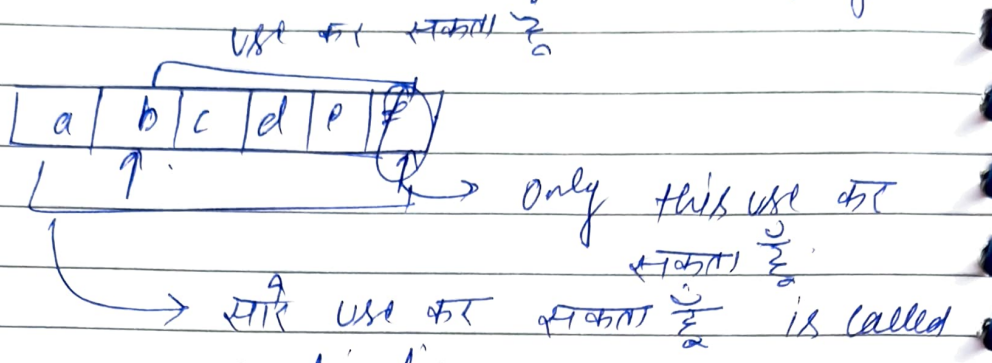


// Code:

```
Public static int CoinChangePer_01(int[] coins, int
    tar, String ans){
    if(tar == 0){
        System.out.println(ans);
        return 1;
    }
    int count = 0;
    for(int i=0; i< coins.length; i++){
        if(tar - coins[i] >=0){
            count += fn(coins, tar-coin[i], ans+coins
                [i]);
        }
    }
    return count;
}
```

APCO
Date :
Page :

(2) Coin change combination such that infinite supply of coin.




```
Public static int fn(int[] coins, int idx, int tar, String ans) {
    if (tar == 0) {
        syso(ans);
        return 1;
    }

    int count = 0;
    for(int i = idx; i < coins.length; i++){
        if(tar - coin[i] >= 0){
            count += fn(coins, i, tar - coins[i], ans + coins[i]);
        }
    }
    return count;
}
```

APCO
Date :
Page :

(3) // ~~temp~~ Coin change combination with single use of coin only.

```
Public static int ~~fn~~ coinChangeCombinationSingle_o(
    int[], coins, int idx, int tar, string ans) {
  if(tar == =0) {
    syso(ans);
    return 1;
  }

  int count = 0;
  for(int i=idx; i < coins.length; i++) {
    if(tar - coins[i] >=0) {
      count += fn(coins, i+1, tar - coins[i],
                  ans + coins[i]);
    }
  }
  return count;
}
```

APCO
Date :
Page :

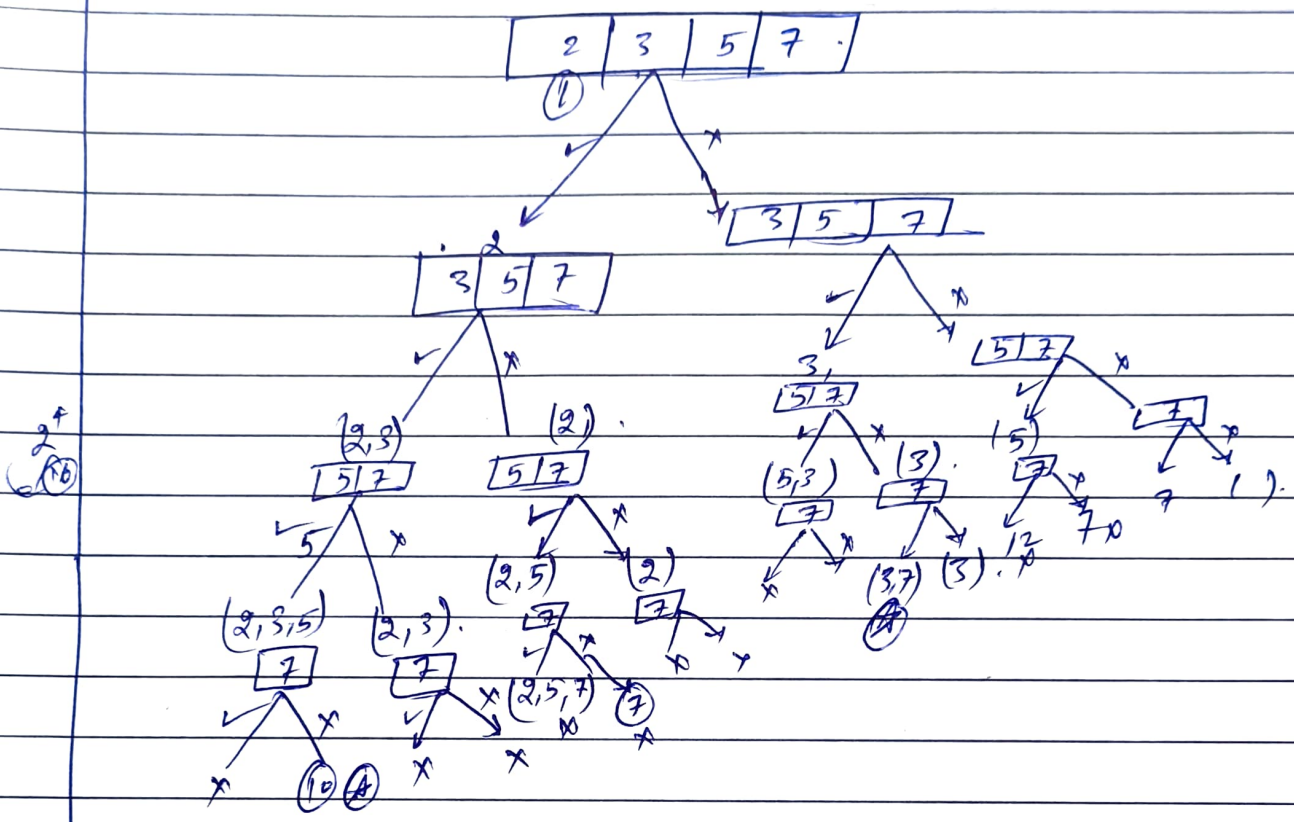
(4) Coin change permutation with single use of coin / limited supply.

```
Ps int f(coinChangePermutationSingle_01(int[] coins,
    int tar, boolean[] vis, String ans)){.

    if(tar == 0){
        syso(ans);.
        return 1;.
    }.

    int count = 0;.
    for(int i=0; i<coins.length; i++){.
        if(!vis[i] && tar - coins[i] >= 0){.
            vis[i] = true;.
            count += fn(coins, tar-coins[i]; vis,
                                ans + coin[i]);.
            vis[i] = false;.
        }.
    }.
    return count;.

}.
```

APCO
Date :
Page :

(Finite Supply of Coin)

(5) Q: coin change combination subsequence (By using subsequence).



// Code:

```
Ps int coinChangeCombinationSingleSubseq(int[] coins,
    int idx, int tar, String ans) {

    if(tar == 0 || idx >= coins.length) {
        if(tar == 0) {
            syso(ans);
        } return 1;
        return 0;
    }
int count = 0;
```

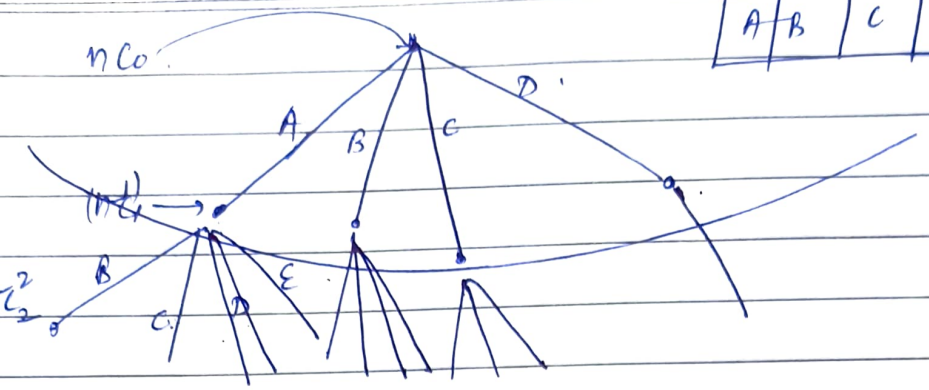
APCO
Date :
Page :

```
if( tar - coins[idx] >=0) {
    count += fn( coins, idx+1, tar - coins[idx], ans +
                                                coins[idx] );
    count += fn( coins, idx+1, tar, ans);

    return count;
}
```

APCO
Date :
Page :

Theo

A	B	C	D	E	F



$\frac{4!}{3!\times 1!}$

$= 4$

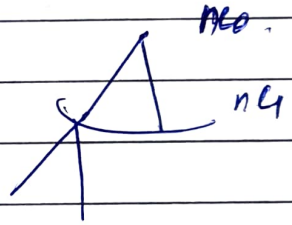
$4C_2 = \frac{4!}{2!\times 2!}$

$= \frac{12}{4} = 6.$

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

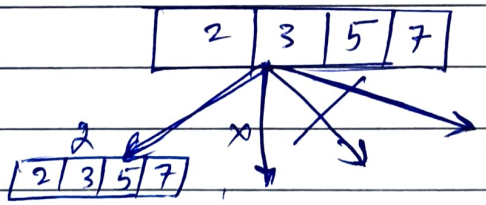
$(1+1)^n = \sum_{i=0}^{n} nC_i$

$$2^n = \sum_{i=0}^{n} nC_i$$



(1) nC_i
(A) Bit
(B) Binary inclusion exclusion.

2	3	5	7



Interviewbit Rec On-1

Note :- 2 [Array list add करो] for return type

15 × 50 = 7,500.

APCO
Date :
Page :

Coin change combination subsequence.

(b) Ans. Ps int Coin change Comb. Subseq (int [] coins, int idx, int tar, string ans) {.

```
if( tar == 0 || idx >= coins.length ) {.
    if( tar == 0 ) {.
        Syso (ans);.
        return 1;.
    }.
    return 0;.
}.

int count = 0;.
if( tar - coins[idx] >= 0 ).
    count += fn( coins, idx, tar - coin[idx],
                 ans + coins[idx] );.

count += fn( coins, idx+1, tar, ans );.

return count;.

}.
```

APCO
Date :
Page :

(7) Coin Change Permutation Subsequence 01. (2 call)

```
Pr int CoinChangePermutationSubseq_01(int[] coins,
    int idx, int tar, String ans) {

    if(tar == 0 || idx >= coins.length) {
        if(tar == 0) {
            syso(ans);
            return 1;
        }
        return 0;
    }

    int count = 0;
    if(tar - coins[idx] >= 0) {
        count += coinChangePermutationSubseq(coins, 0,
            tar - coins[idx], ans + coins[idx]);
    }

    count += fn(coins, idx+1, tar, ans);
    return count;

}
```

APCO
Date :
Page :

(8) ~~public static~~ Coin change Permutation Single Subsequence

```
Ps int Coin Change Permutation Single Subseq(int[] coins,
    int idx, int tar, boolean vis[], String ans){

    if(tar==0 || idx >= coins.length){
        if(tar == 0)
            sysO(ans);
            return 1;
        }
        return 0;
    }

    int count = 0;
    if(!vis[idx] && tar - coins[idx] >= 0){

        vis[idx] = true;
        count += fn(coins, 0, tar - coins[idx], vis,
                    ans + coins[idx]);
        vis[idx] = false;

    }
    count += fn(coins, idx+1, tar, vis, ans);

    return count;

}
```

APCO
Date :
Page :

Qu 1091 // leet code Binary Matrix

APCO
Date :
Page :

Qy Interview BIT Lec. 07 First Question.

APCO
Date :
Page :

(2/09/2020).

(Recursion) Lec-4.

Agenda :

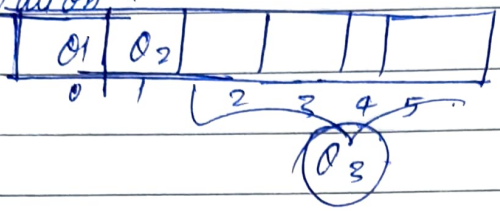
Queen Permutation and Combination.

1. Permutation to arrange 3 queen in 1-D box array.
2. Combination " " " 1-D
3. 2d Combination of all arrangement
4. 2d permutation of queen.
5. n queen combination.
6. n queen permutation.
7. with using ix safe (optimization). (Combination)
8. " " " " permutation

APCO
Date :
Page :

Queen Permutation and Combination:

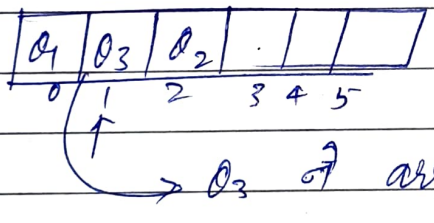
Q1 Queen Combination:



box0 Q1 box1 Q2 box2 Q3.
" " box3 Q3
box4 Q4
box5 Q5

Tree:

but in ~~Combi~~ Permutation -



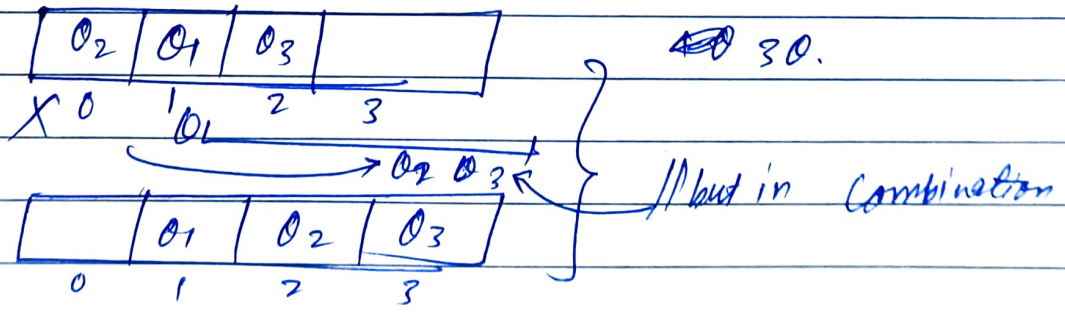
→ Q3 को arrange कर लिया फिर Q4 को

```
Public static int queenCombination(boolean[] box, int
idx, int qpsf, int tnq, String ans){

    if(qpsf == tnq){
        sysln(ans);
        return 1;
    }
    int count = 0;
    for(int i = idx; i < box.length; i++){
        coun += fn(box, i+1, qpsf+1, tnq, ans+"b"+i+"Q"
                                    + qpsf + " ");
    }
```

```
return count;
}
```

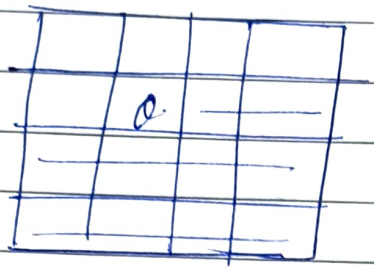
2. Qu Queen permutation in 1-D box.

```
Public static int Qqueen Permutation( boolean [] box, int qpsf
    int tnq, String ans) {

    if(qpsf == tnq) {
        sysO(ans);
        return 1;
    }
    int count = 0;
    for( int i=0; i< box.length; i++){
        if( ! box[i]) {
            box[i] = true;
            count += fn( box, qpsf+1, tnq, ans+"b"
                +i+'@'+qpsf+" ");
            box[i] = false;
        }
    }
    return count;
}
```

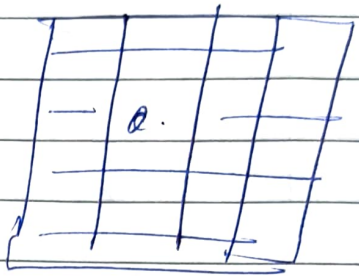


Queens.

2. Combination in 2-d array.



But in for



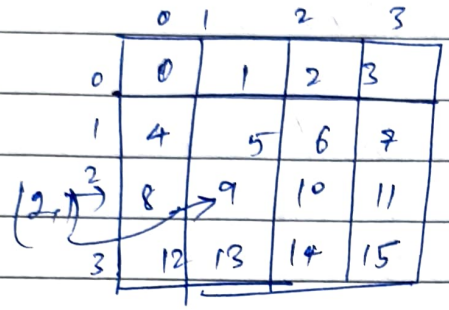
→ Same ques as 1D.

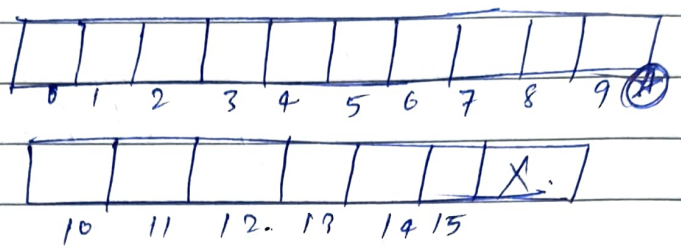
```
Public static int queenCombination2D(boolean[][] box,
    int idx, int qpsf, int tnq, String ans){
if(qpsf == tnq){
    sysoln(ans);
    return 1;
}
int count = 0;
int n = box.length;

for(int i = idx; i < n*n; i++){
    int r = i/n;
    int c = i%n;

    count += fn(box, i+1, qpsf+1, tnq, ans +
        "(" + r + "," + c + ")");

}
return count;
}
```

APCO
Date :
Page :

Qn ~~coin change for 2D~~. Queen permutation in 2D.

```
Public static int queenpermutation2D(boolean[][] box,
    int qpsf, int tnq, String ans){

    if(qpsf == tnq){
        sysoln(ans);
        return 1;
    }

    int count = 0;
    int n = box.length;
    for(int i=0; i< n*n; i++){
        int r = i/n;
        int c = i%n;

        if(!box[r][c]){
            box[r][c] = true;
            count += fn(box, qpsf+1, tnq, ans+"("+
                r+","+c+")");
            box[r][c] = false;
        }
    }
    return count;
}
```





$r = 9/4 = 2$

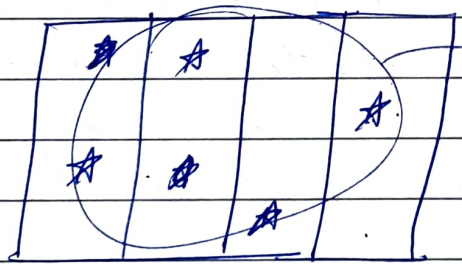
$c = 9 \% 4 = 1$

Qu Place Queen such that no kill each other.

Qu Ree., Soduko, Cross-word, Magnet, Cryptoarithmatic.

Note ☆.
Combination is the ans and in permutation arrangement of same take place at exact location.

eg.



4! ways ∄ permutation possible ∄.

APCO
Date :
Page :

Using Combination -

complexity $\boxed{(\text{no of call})^{\text{height}} \geq n!}$

$T(n) = (nT(n-1)) +$

$nT(n-1) = (n-1)(T(n-2)).$

$\boxed{n! < n^2}$ #

$= (p)T(n-1).$

$T(pH) = (N \times m)^H$

$(N \times m)^H + H(\text{isSafe}) \rightarrow 2(n+m).$

$+ H(2(n+m))$

n queen.

APCO
Date :
Page :

→ पिछले n बोर्ड से मंगाया गया.

$T(n) = n\,(T(n-1))$

$T(n-1) = (n-1)\,T(n-2) \cdot n(n)$.

$n(n-1)\,T(n-2) = (n-2)\,T(n-3)\,(n\times(n-1)$

$n(n-1)(n-2)\,T(n-3) = (n-3)\,T(n-4) \cdot (n(n-1)(n-2)$.

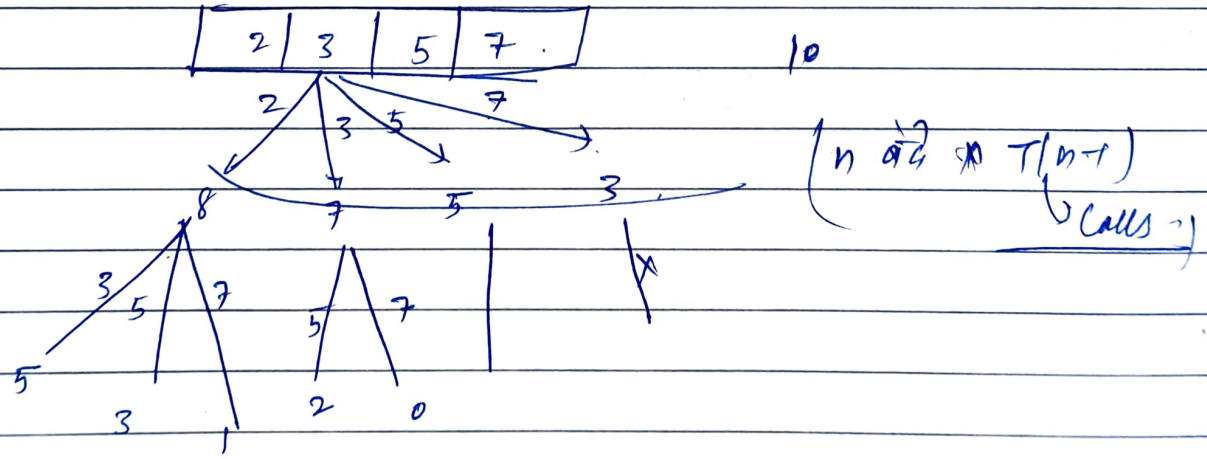
$T(n) = n(n-1)(n-2)(n-3) \ldots 4\cdot3\cdot2\cdot1\,(T(0))$.

$T(n) = n!$ → Height of tree.

$n! = n^H$ → nodes.

$n(n-1)(n-2)(n-3) \leq n\times n\times n\times n$.

$n! < n^2$.



10

(n बोर्ड ⇒ T(n-1)
↳ Calls)

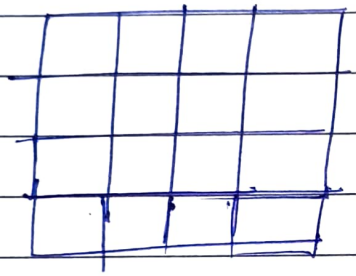
APCO
Date :
Page :

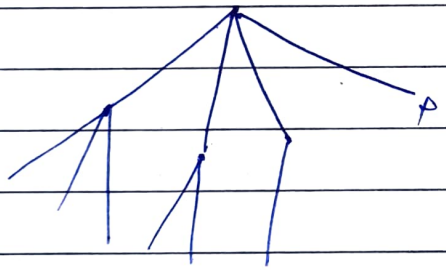
For N queen problem time complexity

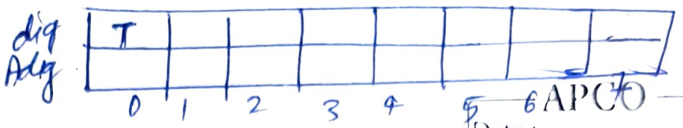
$p = n \times m$.

$T(p) = p\,T(n-1)$.
$T(n-1) = (p-1)\,T(n-2)$.

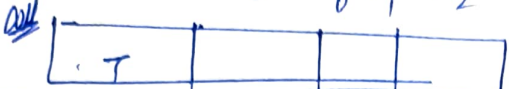
$$O(p^H) = (N \times m)^H + H(N+M).$$ *

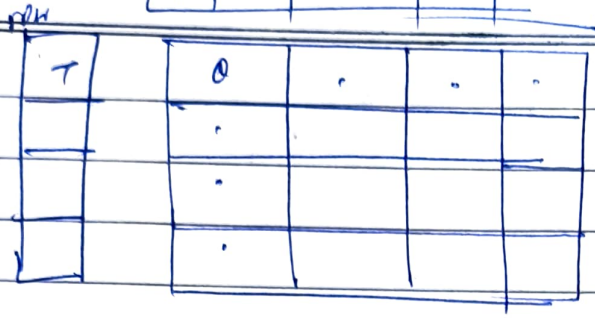




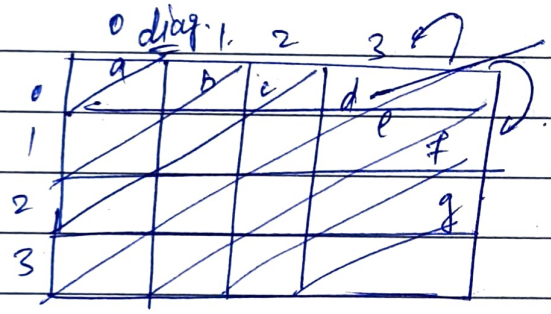


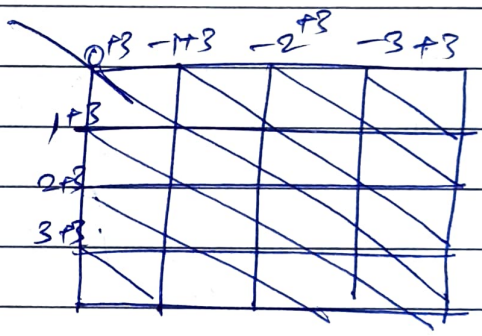
APCO
Date :
Page :





	Size.	formula.
row =	n	r
Col.	m.	c
diag.	n+m-1	r+c.
Antidiag -	n+m-1	r-c+(m-1)

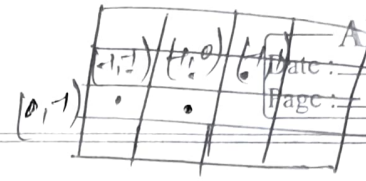




$(n \times m) \longrightarrow (n+m-1)$ diagonals होती है

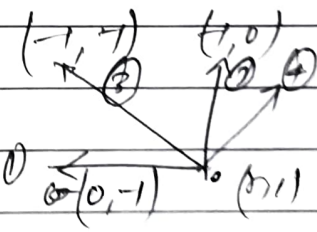
Now complexity = $(n \times m)^q + q(n+m)$ → $O(1)$

n^q



N Queen problem

```
Public static boolean[][] isSafeToPlaceQueen(boolean[][] box, int r, int c){
    // in combination
    int dirA[][] = {{0,-1},{-1,0},{-1,-1},{-1,1}};

    // in permutation
    int dirA[][] = {{0,-1},{-1,0},{-1,-1},{-1,1},{0,1},{1,0},
                    {1,1},{1,-1}};

    for(int d=0; d < dirA.length; d++){
        for(int rad=1; rad <= box.length; rad++){
            int x = r + rad * dirA[d][0];
            int y = c + rad * dirA[d][1];

            if(x>=0 && y>=0 && x < box.length && y <
                box[0].length){
                if(box[x][y]) return false;
            } else {
                break;
            }
        }
    }
    return true;
}
```



APCO
Date :
Page :

// Combination -

```
Public static int Nqueen_01(boolean[][] box, int idx, int
qsf, int tnq, String ans){

if(qsf == tnq){
    sysoln(ans);
    return 1;
}
int count = 0;
int n = box.length;
for(int i=idx; i< n*n; i++){
    int r= i/n;
    int c= i%n;

    if( isSafe( box, r, c)){
        box[r][c] = true;
        count += Nqueen_01(box, i+1, qsf+1,
            tnq, ans+"("+r+","+c+")");
        box[r][c] = false;

    }
}
return count;
}
```

APCO
Date :
Page :

// Permutation में only changes are.

```
fn( boolean box[][], int qpsf, int tnq, String ans){
    if( !box[r][c] && isSafe(box,r,c)){
        fn(                    );
    }
```

extra condition have to check in box whether there is any queen placed already there.

APCO
Date :
Page :

without using isSafe (optimization → O(n*m) !!)

```
static boolean[] rowA, colA, digA, adigA;

Public static int nQueen_03(int n, int idx, int qpsf, int
        tnq, string ans){

if(qpsf == tnq){
    sysodn(ans);
    return 1;
}

int count = 0;
for(int i = idx; i < n*n; i++){
    int r = i/n;
    int c = i%n;

    if(!rowA[r] && !colA[c] && !digA[r+c] &&
            !adigA[r-c+n-1]){

        rowA[r]       = !rowA[r];
        colA[c]       = !colA[c];
        digA[r+c]     = !digA[r+c];
        adigA[r-c+n-1] = !adigA[r-c+n-1];

        count += fn(n, i+1, qpsf+1, tnq, ans+"("+
                +r+","+c+")");
        rowA[r]       = !rowA[r];
        colA[c]       = !colA[c];
        digA[r+c]     = !digA[r+c];
        adigA[r-c+n-1] = !adigA[r-c+n-1];
    }
}
return count;
}
```

APCO
Date :
Page :

initialize all arrays rowA[7], colA, diagA, adiagA as boolean array at time of fn Call.

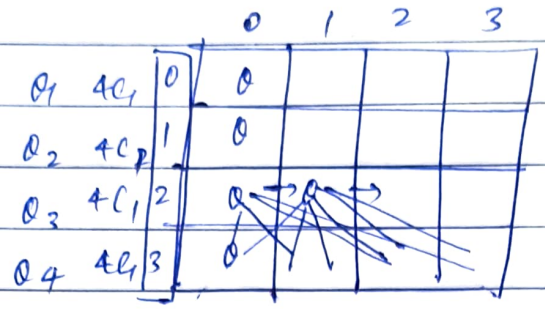
APCO
Date :
Page :

(4/09/2020)

Lect -5

Agenda :

1. optimization of nqueen . // 51 - // 52.
2. Solve 1 using subseq → this method will help when
3. word break. { we have 100×100 box and to place
4. leetcode 51 only 10 or less than n queen)
5. leetcode 52
6. Sudoku . // leetcode 37

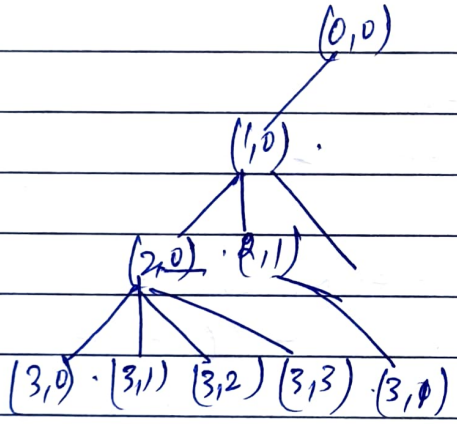


total way = $4C_1 \times 4C_1 \times 4C_1 \times 4C_1$

= 256. ways.

(n Queen + optimization).

→ 4 floor पर 4 queen place करने पर only one ways.



complexity (n²).

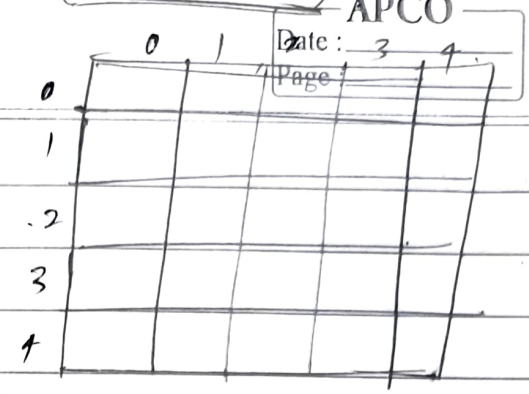
APCO
Date :
Page :

Optimized 04 $O(n^2)$ यहाँ $O(n \times n)^2$

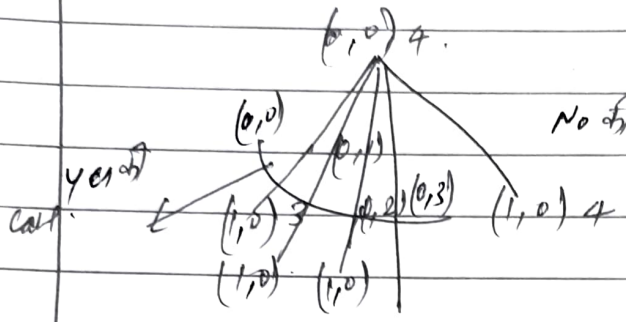
```
Public static int Nqueen_04(int n, int r, int tnq, String ans){
    if(tnq == 0){
        Sysoln(ans);
        return 1;
    }
    int count = 0;
    calls++;
    for(int c = 0; c < n; c++){
        if( !colA[c] && !diagA[r+c] && !adiagA[r-c+n-1]){

            colA[c] = !colA[c];
            diagA[r+c] = !diagA[r+c];
            AdiagA[r-c+n-1] = AdiagA[r-c+n-1];

            count += Nqueen_04(n, r+1, tnq-1,
                ans + "(" + r + "," + c + ")");

            colA[c] = !colA[c];
            diagA[r+c] = !diagA[r+c];
            AdiagA[r-c+n-1] = !adiagA[r-c+n-1];
        }
    }
    return count;
}
```

Best when ∞ q n ≤ box size (10 < 100 × 100)

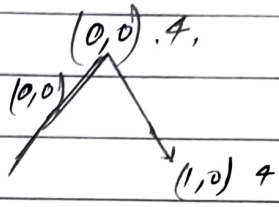
APCO
Date :
Page :



optimized Subsequence — 04.



(or).




```
Public static int Nqueen_04_Subseq(int n, int r, int tnq, String ans
    if(tnq==0 || r>=n){
        if(tnq==0){
            Syso(ans);
            return 1;
        }
        return 0;
    }
    int count = 0;
    for(int c=0; c<n; c++){
        if(!colA[c] && !diagA[r+c] && !adiagA[r-c+n-1]){
            colA[c] = !colA[c];
            diagA[r+c] = !diagA[r+c];
            adiagA[r-c+n-1] = !adiagA[r-c+n-1];
            count += fn(n, r+1, tnq-1, ans+"("+r+","+c+
                                        ");");
            colA[c] = !colA[c];
            diagA[r+c] = !diagA[r+c];
            adiagA[r-c+n-1] = !adiagA[r-c+n-1];
        }
    }
    count += fn(n, r+1, tnq, ans);
    return count;
}
```

APCO
Date :
Page :

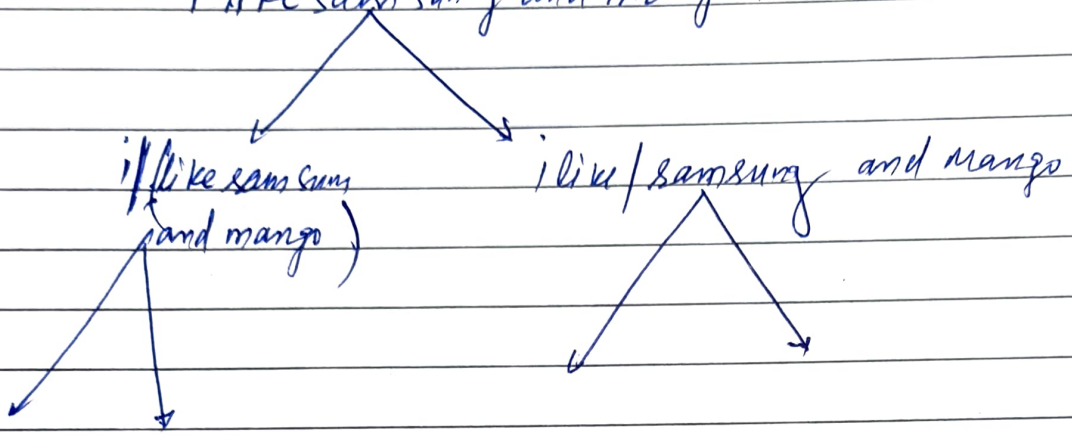
Qn 51 // b.c.

1152.

APCO
Date :
Page :

i like samsung and mango.
1 2 3 4 5

// word break :



complexity = (n^n)

$T(n) = n\,T(n-1) + n \times$ height times.

$T(n) = n(T(n-1)) + n \times n$.

APCO
Date :
Page :

// Code - (leetcode).

```
Public static void wordBreak(){
    String Dictionary[] = {"mobile", samsung", "sam",
                           "sung", "man", "mango",
                           "icecream", "and", "go", "i", "like", "ice",
                           "cream", "ilikesamsung"};

Hash set < String > words = new HashSet<>();
int len = 0;
for( String str : dictionary ){
    word.add(str);
    len = Math.max( len, str.length() );
}

String ques = "ilikesamsung and Mango";
sysoln( wordBreak(ques, 0, "", words, len);

}

Public static int wordBreak (string ques, int idx,
string ans, HashSet < String> words, int Len ){
    if( idx == ques.length() ){
        sysoln( ans);
        return 1;
    }
    int Count = 0;
    for( int i = idx; i - idx <= len && i <= ques.
        length(); i++) {
        String s = ques.substring (idx, i);
        if(words.contain(s)) {
            count += fn( ques, i, ans+s+" ", words, len)
    } return count ; }.
```

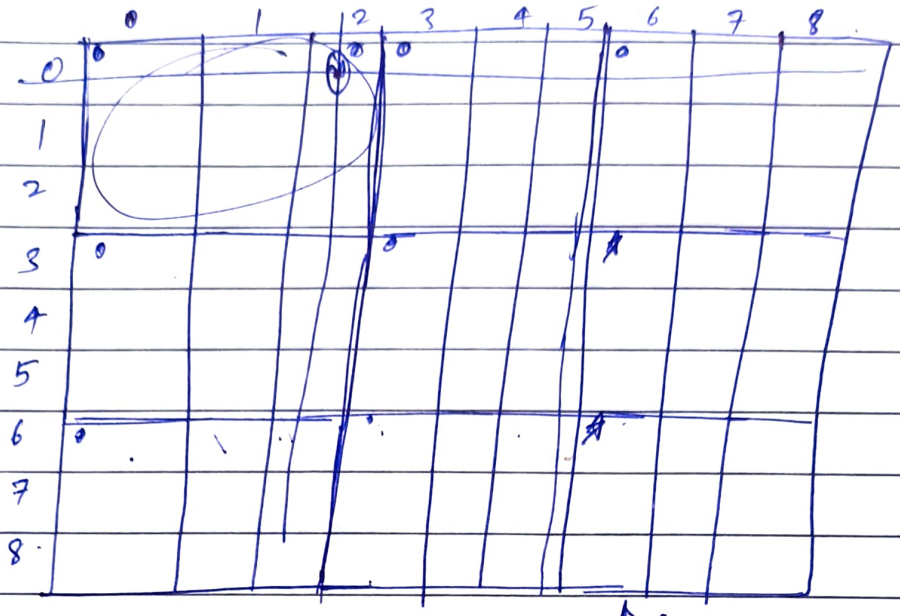
APCO
Date :
Page :

M-2

```
Public static int wordBreak_02(String ques, String ans,
    HashSet<String> words, int len){

    if(ques.length()==0){
        sysout(ans);
        return 1;
    }
    int count = 0;
    for(int i=0; i<=len && i<=ques.length(); i++){
        String s = ques.substring(0,i);
        if(words.contain(s)){
            count += wordBreak_02(ques.substring(i)
                , ans + s + " ", words, len);
        }
    }

    return count;

}
```

APCO
Date :
Page :

Qu // Soduko Solver - //37



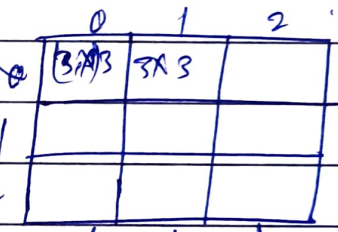
2,1

5,5

compress:

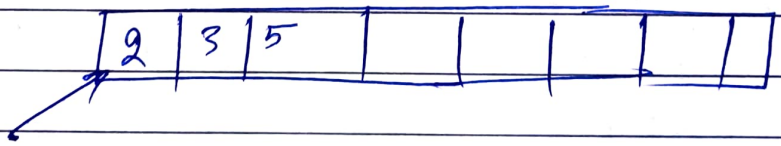
$(6,5)/3 =$

$(r,c)/3 \ * 3 =$



eg $(7,7)/3 * 3 = (2,2) \times 3 = (6,6)$.

* Try to keep tree height as low as possible.

① 

array of vacant location.

$(81)^{81}$ total Combination

// question code.

fn ①

```
void Soduko() {
    vector<int> loc;
    for(int i=0; i<9; i++) {
        for(int j=0; j<9; j++) {
            if(board[i][j] == 0) {
                loc.push_back(i*9+j);
            }
        }
    }
    SodukoSolver(0, loc);   // contain only empty space eg in array.
}
```

✱	9	✱
5	✱	6

fn ②

```
int SodukoSolver(int vidx, vector<int>& loc) {
    if(vidx == loc.size()) {
        Print2D();
        return 1;
    }
    int idx = loc[vidx];
    int r = idx/9;
    int c = idx%9;
```

APCO
Date :
Page :

```
int count = 0;
for(int num = 1; num <= 9; num++){
    if(isSafeToPlaceNo(r, c, num)){
        board[r][c] = num;
        count += fn(vidx+1, loc);
        board[r][c] = 0;
    }
}
return false;
}
```

(3)

```
bool isSafeToPlaceNo(int r, int c, int num){
    // Row check
    for(i=0; i<9; i++){
        if(board[i][c] == num) return false;
    }
    // column check
    for(int i=0; i<9; i++){
        if(board[r][i] == num) return false;
    }
    // Matrix
    r = (r/3)*3;
    c = (c/3)*3;
```

gives starting of each matrix (index).

APCO
Date :
Page :

```
for(int i=0; i<3; i++){
    for(int j=0; j<3; j++){
        if(board[sr+i][sc+j] == num)
            return false;
    }
}
return true;
}
```

④

```
void print2D(){
    for(vector<int> a : board){
        for(int ele : a){
            cout << ele << " ";
        }
        cout << endl;
    }
}
```

make static board eg.

```
vector < vector<int>> board;
```

$(9^2) = (n^2)$

Complexity $O(n^3)$.

how → isSafe = $\Theta(n+n+n)$ n^2

= $O(n^3)$

APCO
Date :
Page :

6/09/2020.

Agenda:

① Qn n queen ~~of~~ optimization.

Bit manipulation. → on. ~~self~~ 2-2

⑦ → 136, ✓ // single Number.

⑧ → 137 ✓ // single Number-II

④ → 191 ✓ // Number of 1 bits.

⑤ → 260 ✗ match v tube. // single Number - III

⑥ → 338. // Counting bit.

⑦ → 342. // power of 4

⑧ On-~~off~~ / ~~off~~-on.

⑨ 201 // Bit wise AND of Numbers Range.

APCO
Date :
Page :

Bit manipulation :-

int a = 5;.

32 bit ही occupy करेगा in ज्ञ कम ज्ञ उपयोग

eg.

0 0 - 1 0 1

29.

long
~~float~~ 64 bit में store होगा (64 bit).

8 byte.

operators - &, |, ^, ~, !, - → not, 2's comp.

~ → ones complement

2's - ones comp. में 1 add कर दो (यानि negative num)

1's $\xrightarrow{+1}$ (-ve number).

→ ~~&~~ << left shift >> right shift

→ bracket लगाओ → eg (num << 4).

APCO
Date :
Page :

Bit operator से programme fast होता है.

right shift

eg. x = 0 1 0 1 ... 0
(32 bit)
↑ msb. ↑ lsb.

num = num >> 3.
↳ if 1 है तो 1 append होता
otherwise 0.

→ left shift :- (2 * numb) से होता है

1 — 2^0
10 — 2^1
100 — 2^2
1000 — 2^3.
10000 — 2^4.

eg.
101 — 5
1010 — 10.
10100 — 20.
101000 — 40.

ax Math.pow(2, b) ⟶ log(n).
a << b ⟶ O(1)

⁂. >>> (right shift). only append '0' at left

Unsigned int → only ~~assig~~ ~~unset~~ assign '0'

Q. int num = x.
pos = k.

①	off → on · on on → on	② on – off off – off

1*
at kth pos^n.

Fn.
① int offTOON_setTrue (int n, int k) {
int mask = (1<<k);
return n|mask;
}

Fn.
② int onTOOFF_setFalse (int n, int k) {
int mask = (~(1<<k));
return (n & mask);
}

APCO
Date :
Page :

④ // 191 LeetCode (Number of 1 bits).

i/p = 00000000000 1011
o/p = 3.

algo-1 do left shift and take '&' for n time and count.

```
int hammingweight(uint32_t n) {
    // n = 32.
    int count = 0;
// o(n). Algo-1
    for(int i=0; i<32; i++) {
        int mask = 1<<i;
        if((n & mask) != 0) count++;
    }.
```

// (algo-2) o(logn).

```
while(n != 0){
    if((n & 1) != 0) count++;
    n >>= 1;
}.
```

// algo-3. O(No. of set bit time)

```
while(n != 0){
    count++;
    n &= (n-1);
}
return count;
}.
```

xor → toggle करता है.

APCO
Date :
Page :

x = | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 |

32 bit का array of '0' and '1'

int num = 0 → boolean में 32 false.

Bool array operation

① int idx = k.

② set arr[idx] = true
" = false.

③ get arr[idx].

④ toggling arr[idx] = ! arr[idx].

⑤ if(arr[idx]){

}. else {

}.

integer array.

arr = 0.

① to goto any k posn.
int mask = (1 << k).

② to set true. arr |= mask.
" false arr &= ~mask.

③ int val = (arr & mask) != 0 ? 1 : 0;

④ arr ^= mask (toggling).

⑤ if(!(arr & mask)) {

}
else {

}

APCO
Date :
Page :

Q. nqueen using bit :

~~stat~~ only diff in bit is we get better space complexity.
Instead of using array of boolean we used num of bits.

```
Static int colN = 0, diagN = 0, adiagN = 0;
Public static int { Nqueen_04_bits( int n, int r,
    int tnq, String ans ) {
    if( tnq == 0 ) {
    sysoln (ans);
    return 1;
    }
    int count = 0;
    for ( int c = 0; c < n; c++ ) {
    if ((colN & (1<<c)) == 0 && ((diagN & (1 << (r+c))))
        && (adiagN & ((1 << (r-c+n-1)))) == 0 ) {
        colN ^= (1<<c);
        diagN ^= (1 << (r+c));
        adiagN ^= (1 << (r-c+n-1));
        count += Nqueen04_bits(n, r+1, tnq-1,
            ans + "(" + r + "," + c + ")");
        colN ^= (1<<c);
        diagN ^= (1 << r+c);
        adiagN ^= (1<<(r-c+n-1));
    }
    }
    return count;
}
```

APCO
Date :
Page :

10 D00 N 70
100
= 7000
= 2000k → 1200

Bit question.

Conting bit, SN-2, SN-3.
338. 137.
191, 136, 342, 260,

n = n & (n-1) → Remove last set bit

last significant bit (last set bit).

lsb = n & (-n).

(n & (n-1)) | (n & (-n)) = n. *

(n & (n-1)) → Remove right set bit.
(n & (-n)) → add r-set bit

APCO
Date :
Page :

$4^n = (2^2)^n = 2^{2n}$.

4 — 100
~~8 — 10~~
16 — 10000
64 — 1000000
128 — 100000000
256 — 1000000000000

* use to check (n&1) == 0. even
even / odd.

(136) Single Number.

ans = 0 ^ a ^ b ^ c ^ d ^ c ^ b ^ a
ans = 0 ^ d.
= d.

(137)

101 — 5
~~0~~ 010 → 2
+1
011 → 3.

APCO
Date :
Page :

Q 137

1 5 2 3 1 2 8 13 5 3 1 5 3 2 1 7 7 7

1 → 00001
2 → 00010
3 → 00011
5 → 00101
13 → 01101
7 → 00111

ε < k.

(k(A + B + C ... D) + ε) % k ⇒ 0

only this left.

- Sodoku puzzle-

Q 201 // leetcode // Power of two

~~n!(n<0)~~.

```
Public boolean isPowerOfTwo(int n){
    if(n<0)
        return false;
    return (n!=0 && n&(n-1)==0);
}
```

APCO
Date :
Page :

Q) // 342 leetcode.

I/P = 16.

O/P = true;

algorithm :-

① check whether it is power of 2 or not
② if yes then count no. of set bits.
③ take & with '1' and set bit.

// code.

```
bool isPowerOfFour(int num){
    if(num>0 && (num&(num-1))==0){
        int count=0;
        while(num!=1){
            num>>=1;
            count++;
        }
        return (count&1)==0;
    }
    return false;
}
```

Q) // 136 leetcode: (Single Number).

```
int singleNumber(vector<int>& nums){
    int ans=0;
    for(int ele:nums) ans^=ele;
    return ans;
}
```

APCO
Date :
Page :

Q4 // 137.

Counting setbit II.

{2, 2, 2, 3, 5, 5, 5}.

```
0 1 0
0 1 0   } 3.
0 1 0.
0 1 1   } 2.
1 0 1
1 0 1   } 5
1 0 1
-------
3 1 4
```

4%3 = 1 ⇒ 011 ans.

4%3 = 1

3%3 = 0

↑ (Count set bit for each place using mask) take modulus and add that into ans.

// code

```
Public int singleNumber(int[] nums){
    int k = 3;
    int res = 0;
    for(int i=0; i<32; i++){
        int mask = (1<<i);
        int count = 0;
        for(int ele : nums){
            if((ele & mask) != 0) count++;
        }
        if(count % k != 0) res |= mask;
    }
    return res;
}
```

Time : $O(32 * N) \neq O(N)$.

M-2. Sorting method is better.

WC (sorting) $= O(32 * N)$.

↑ when no. of elements = INT_MAX.

* Previously done counting bits method *.

Time : WC = Avg = Best = $O(32 * N)$.

M-2 is better.

M-④ {2, 2, 2, 3}

APCO
Date :
Page :

Qu 260 / Leetcode.

```
vector<int> singleNumber(vector<int>& nums){
    int xy=0;
    for(int n: nums) xy ^= n;
    xy &= -xy;
    vector<int> result = {0,0};
    for(int n: nums){
        if(xy & n) result[0] ^= n;
        else result[1] ^= n;
    }
    return result;
}
```

→ global में calculation का part नहीं आना चाहिए।
→ static वाले

APCO
Date :
Page :

(Recursion) (7/09/2020)

Agenda:-

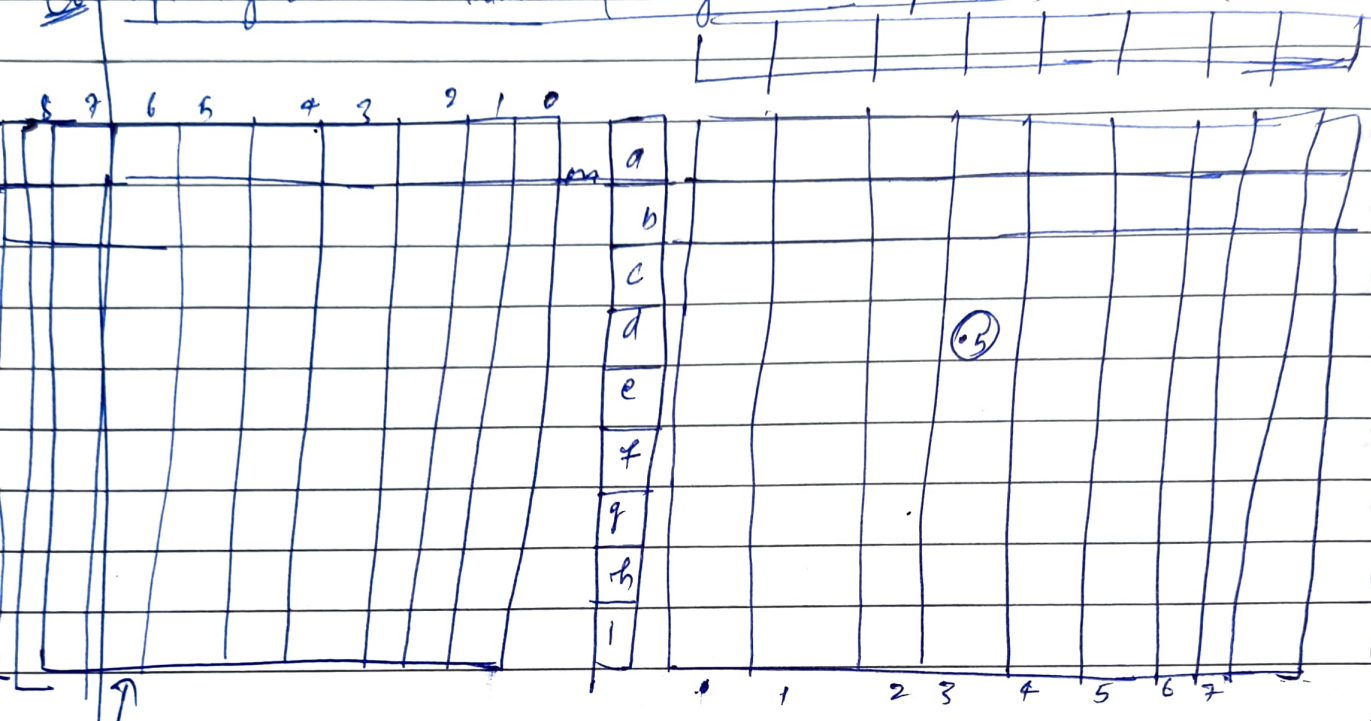
1. Soduko Validator
2. Soduko solver (with bit optimization).
3. Crypto arithmatic.
4. Cross word.

① Recursion की complexity varies करती है faith पर

② so try only those faith which gives you the lower complexity.

Date :
Page :

① optimization of Sudoku (using Bit manipulation).



similar make for col^n and 3×3 matrix.

let say hame to check for 5 so (1<<5) की position पर check करेंगे (faster than loop).

Valid sudoku - // 36.

M-① nqueen.

M-② hashMap और set के help of element से ??

$\lfloor 3,3 \rfloor / 3 = (1,1)$.

$\lfloor 4,3 \rfloor / 3 = (1,1)$ ×

row.

9	8	7	6	5	4	3	2	1	0
				1					

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

	0	1	2	3	4	5	6	7	8
		1							

col.

0	1	2	3	4	5	6	7	8	9
	1								

0
1
2
3
4
5
6
7
8
9

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

a	b	c
d	e	f
g	h	i

$\left(\frac{\lfloor 4,7 \rfloor}{3}\right) \times 3$

& we reach to the starting coordinate

If multiply by $\lfloor 3,3 \rfloor / 3 = (1,1) \times 3$ pos n of offset matrix & vice.

APCO

Date :

Page :

APCO
Date :
Page :

Sodoku Validator - || Leetcode - 36.

```
bool isValidSudoku (vector<vector<char>> &board){
    for(int i=0; i< board.size(); i++){
        for (int j=0; j< board[0].size(); j++){
            if( board[i][j] != '.'){
                int mask = 1<<(board[i][j]-'0');
                if((rowA[i] & mask) == 0) &&
                   (colA[j] & mask) == 0) &&
                   (matA[i/3][j/3] & mask) == 0)){
                    rowA[i] |= mask;
                    colA[j] |= mask;
                    matA[i/3][j/3] |= mask;
                }
                else{
                    return false;
                }
            }
        }
    }
    return true;
}
```

APCO
Date :
Page :

2. Qu 1137 leetcode.

```
void sodoku(){
    vector<int> loc;
    for(int i=0; i<9; i++){
        for(int j=0; j<9; j++){
            if(board[i][j]==0){
                loc.push_back(i*9+j);
            }
            else{
                int mask = 1<<board[i][j];
                rowA[i] ^= mask;
                colA[j] ^= mask;
                matA[i/3][j/3] ^= mask;
            }
        }
    }
    sodukoSolver_Best(0, loc);
}
```

fn (2)

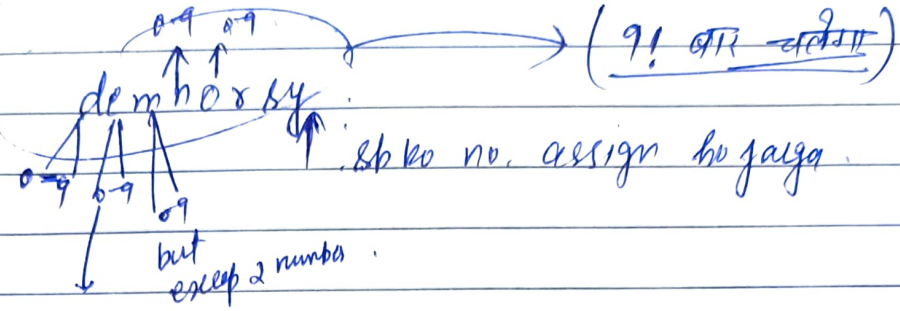
```
int sodukoSolver_Best(int vidx, vector<int>&loc){
    if(vidx==loc.size()){
        print2D();
        return 1;
    }
    int idx = loc[vidx];
    int i = idx/9;
    int j = idx%9;
```

APCO
Date :
Page :

```
int count = 0;
for (int num = 1; num <= 9; num++) {
    int mask = 1 << num;
    if ((rowA[i] & mask) == 0 && (colA[j] & mask) == 0 &&
        (mat[i/3][j/3] & mask) == 0) {
        rowA[i] ^= mask;
        colA[j] ^= mask;
        mat[i/3][j/3] ^= mask;
        board[i][j] = num;
        count += SodukoSolver_Best(visited, loc);
        board[i][j] = 0;
        rowA[i] ^= mask;
        colA[j] ^= mask;
        mat[i/3][j/3] ^= mask;
    }
}
return false;
}
```

APCO
Date :
Page :

8. Q. Crypto arithmatic



idx → index of string.

Send
more
money

7531
0825
8356

① zero आ गया check in recursive call.
② or check in base case.

Eg:

```
Void Crypto1(){
    String str = s1+s2+s3;
    int freq = 0;
    for(char ch: str){
        int mark = 1 << (ch-'a');
        freq |= mark;
    }
    str = "";
    for(int i=0; i<26; i++){
        int mark = 1<<i;
        if((freq & mark) != 0){
            str += string(1,(char)(i+'a'));
        }
    }
    cout << cryptoSolver(str,0) << endl;
}
```

1 leng की str add हो रही है

APCO
Date :
Page :

(2)

```
int cryptoSolver(string str, int idx){
    if(idx == str.length()){
        int x = StringToNumber(s1);
        int y = StringToNumber(s2);
        int z = StringToNumber(s3);

        if( x+y == z){
            cout << " " << x << "\n+" << y <<
                    "\n---\n" << z << endl;
            cout << endl;
            return 1;
        }

        return 0;
    }

    char ch = str[idx];
    int count = 0;
    for(int num = 0; num <= 9; num++){
        int mask = 1 << num;
        if((numUsed & mask) == 0){
            numUsed |= mask;
            mapping[ch-'a'] = num;

            count += cryptoSolver(str, idx+1);
            numUsed ^= mask;
            mapping[ch-'a'] = -1;
        }
    }
    return count;
}
```

APCO
Date :
Page :

Eg. (3)

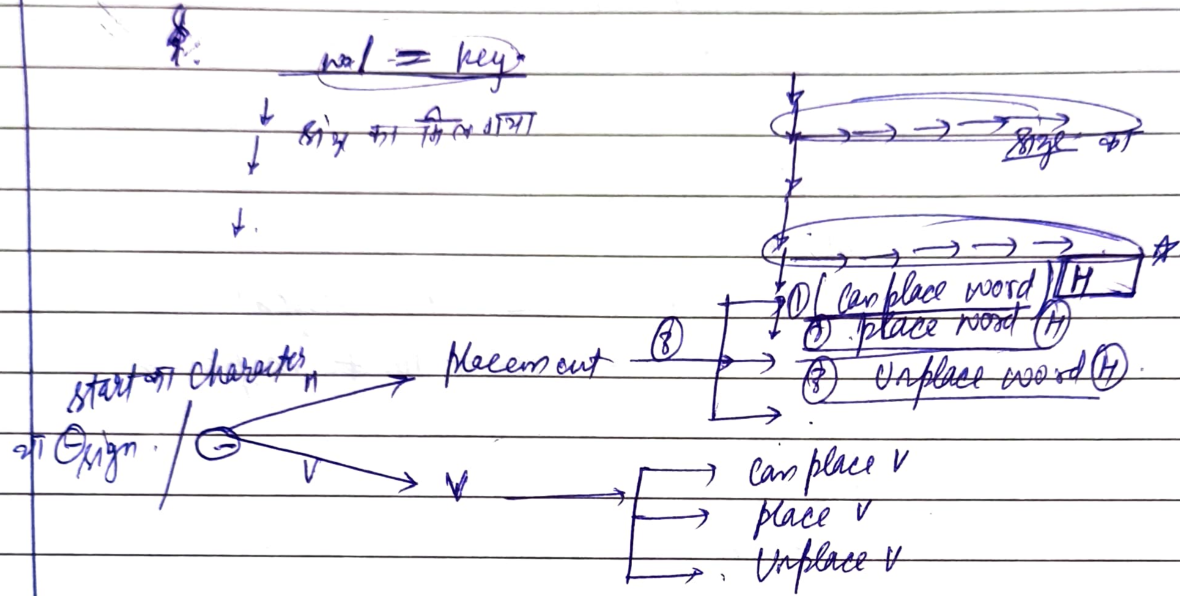
```
String s1 = "send";
String s2 = "more";
String s3 = "money";
Vector<int> mapping(26, -1);
int numUsed = 0;

int stringToNumber(String str){
    int res = 0;
    for(int i=0; i< str.length(); i++){
        res = res * 10 + mapping[str[i] - 'a'];
    }
    return ans;
}
```

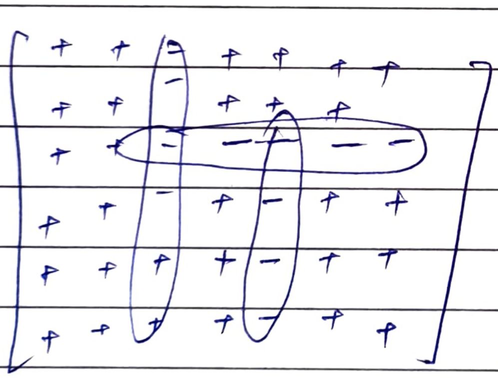
47, 77, 39, 216, 78, 57,
52, 37

APCO
Date :
Page :

4 Q. Place, implace, कौन सा word चाहिए || Cross word (Hackerrank)

(agra, norway, england, gwalior).





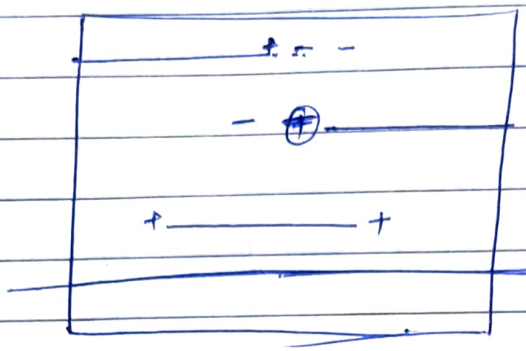
$O(n^{2 \times l})$ (length of words) → total word.

n → size of board;

O → (complexity).

APCO
Date :
Page :

optimization :



(is handled by for loop)

पहले

Ex 77, 39, 58 . Vector < vector < char >> box =

	0	1	2	3	4	5	6	7	8	9	
{{0	+	-	+	+	+	+	+	+	+	+	},
{1	+	-	+	+	+	+	+	+	+	+	},
{2	+	-	-	-	-	-	-	-	+	+	},
{3	+	-	+	+	+	+	+	+	+	+	},
{4	+	-	+	+	+	+	+	+	+	+	},
{5	+	-	-	-	-	-	-	+	+	+	},
{6	+	-	+	+	+	-	+	+	+	+	},
{7	+	+	+	+	+	-	+	+	+	+	},
{8	+	+	+	+	+	-	+	+	+	+	},
{9	+	+	+	+	+	+	+	+	+	+	}};

APCO
Date :
Page :

fn ①

```
void crossword(){
vector<string> words={"agra","norway","england",
                      "gwalior",};
crossword(words,0);
for(int i=0;i<box.size();i++){
    for(int j=0;j<box[0].size();j++){
        cout<<box[i][j]<<" ";
    }
    cout<<endl;
}
}
```

fn ②

```
bool crossword(vector<string>&words, int idx){
if(idx==words.size()){
    return true;
}

string word = words[idx];
int count=0;
for(int i=0; i<box.size(); i++){
    for(int j=0; j<box[0].size(); j++){

if(box[i][j]=='-'||box[i][j]==word[0]){
    if(canPlacewordH(i,j,word)){
        int loc = placewordH(i,j,word);
    if(crossword(words,idx+1)){
        return true;
    unplaceword(i,j,word,loc);
}
```

```
        if (canPlaceWordV(i, j, word)){
            int loc = placeWordV(i, j, word);
            if(crossWord(words, idx+1))
                return true;
            unplaceWordV(i, j, word, loc);
        }
    }
}
return false;
}
```

fn (3)

```
bool canPlaceWordH(int r, int c, String word){
    int l = word.length();
    int m = box[0].size();
    if( c==0 && l<m) if(box[r][c+l] != '+') return false;
    else if( c+l == m) if(box[r][c-1] != '+')
                            return false;
    else {
        if( (c-1>=0 && box[r][c-1] != '+') ||
            (c+l<m && box[r][c+l] != '+') )
            return false;
    }
```

APCO
Date :
Page :

```
for( int i=0; i< word.length(); i++){
    if( c+i >= box[0].size())
        return false;

    if( box[r][c+i] != '-' && box[r][c+i] !=
        word[i])
        return false;
}
return true;
}
```

fn. ④

```
int PlaceWordH (int r, int c, string word){

// vector<bool> loc(word.length(), false);
to keep track of location where word place
// M-2 by integer using bit

    int loc=0;
    for(int i=0; i< word.length(); i++){
        if( box[r][c+i] == '_'){
            box[r][c+i] = word[i];
            loc |= 1<<i;
        }
    }
    return loc;
}
```

APCO
Date :
Page :

⑤ void UnplaceWordH(int r, int c, String word, int loc){

for(int i=0; i<word.length(); i++){
if((loc & (1<<i)) != 0){
box[r][c+i] = '-';

}

}

fn.

⑥ bool CanPlaceWordV(int r, int c, String word){

int l = word.length();
int n = box.size();

if(r==0 && l<n) if(box[r+1][c] != '+') return false;

else if(r+l==n) if(box[r-1][c] != '+')
return false;

else {
if((r-1>=0 && box[r-1][c] != '+') ||
(r+l<n && box[r+l][c] != '+'))
return false;
}

APCO
Date :
Page :

```
for( int i=0; i< word.length(); i++){
    if( r+i >= box.size())
        return false;

    if( box[r+i][c] != '-' && box[r+i][c] !=
        word[i])
        return false;
}
return true;
}
```

fn.

(1)

```
void UnplaceWordV( int r, int c, String word, int loc){
    for( int i=0; i< word.length(); i++){
        if( (loc & (1<<i)) != 0)
            box[r+i][c] = '-';
    }
}
```

(2)

```
int PlaceWordV( int r, int c, String word){
    // vector<bool> loc(word.length(), false);
    int loc = 0;
    for( int i=0; i< word.length(); i++){
        if( box[r+i][c] == '-'){
            box[r+i][c] = word[i];
            loc |= 1<<i;
        }
    }
return loc;
}
```

APCO
Date :
Page :

H.W on Recursion -

(1) 77, 47, 39, 216, 78, 51, 52, 37