Page No.
Date

imp

78 leetcode. (Subsets) // Powerset

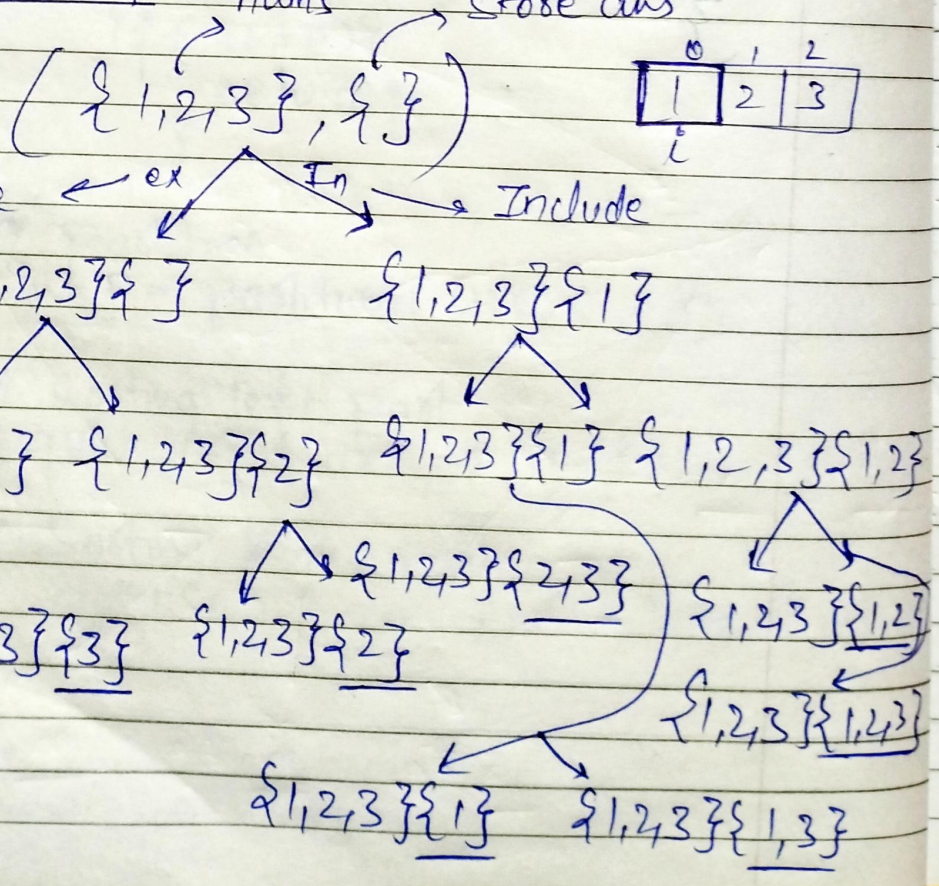
Q. Given an integer array nums of unique element, return all possible subset (the power set)

The sol^n must not contain duplicate subsets. Return in any order.

Eg:- nums = [1,2,3]

o/p = [[], [1], [2], [3], [1,2], [2,3], [1,3], [1,2,3]]

Sol^n Approach:



Page No.
Date

Code:

```
Class Solution{
Private:
void Solve(vector<int> nums, vector<int> output,
           int index, vector<vector<int>>& ans)
  {

      // base case
      if(index >= nums.size()){
       ans.push_back(output);
        return;
       }

      // exclude
      Solve(nums, output, index+1, ans);

      // Include
      int element = nums[index];
       output.push_back(element);
      Solve(nums, output, index+1, ans);

}
Public:

vector<vector<int>> Subsets(vector<int>& nums)
  {  vector<vector<int>> ans;
     vector<int> output;
     int index=0;
     Solve(nums, output, index, ans);
     return ans; } };
```

Page No.
Date

imp

***

Q

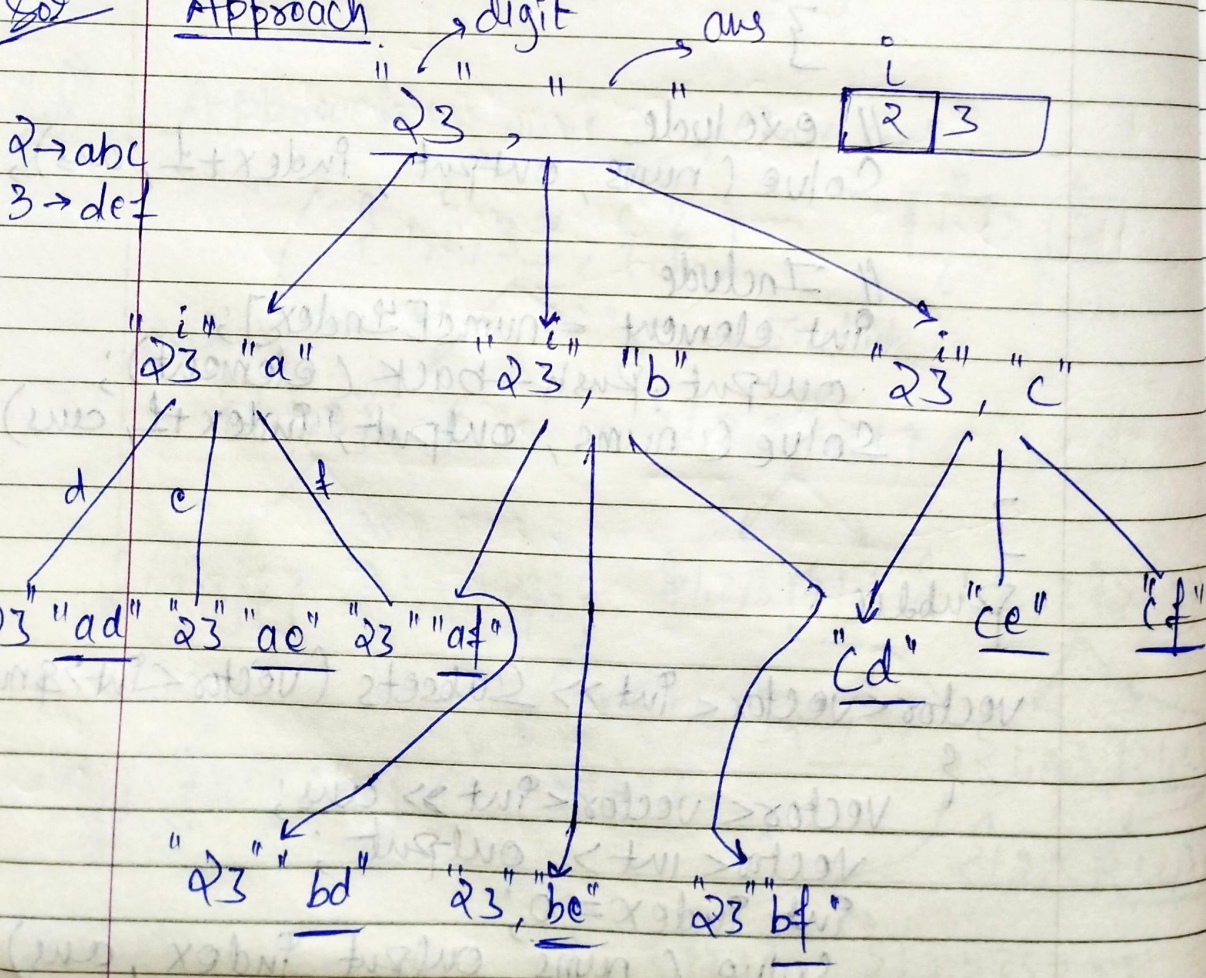
17 leetcode { Letter Combination of a Phone number }

Given a string containing digit from 2-9 inclusive, return all possible letter combination that the number could represent. Return answer in any order.

2	3	4	5	6	7	8	9
abc	def	ghi	JKL	MNO	Pqrs	tux	wxyz

Sol^n

Approach



Page No.
Date

Code:

```
Class Solution {

Private:
    void solve(string digit, string output,
        int index, vector<string>& ans, string
                                    mapping[])

{
    // base case
    if( index >= digit.length()) {
        ans.push_back( output);
        return;
    }

    int number = digit[index] - '0';
    string value = mapping[number];

    for(int i=0; i< value.length(); i++){

        output.push_back( value[i]);
        solve( digits, output, index+1, ans,
                    mapping);

        output.pop_back();

    }

}
```

Page No.
Date

```
public:
vector<string> letterCombinations(string digits)
{
    vector<string> ans;
    string output;
    int index = 0;

    if(digits.length()==0){
        return ans;
    }

    string mapping[10] = {"", "", "abc",
    "def", "ghi", "jkl", "mno", "pqrs", "tuv",
    "wxyz"};

    Solve(digits, output, index, ans, mapping);
    return ans;
}
};
```

Page No.
Date

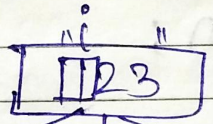
imp

Q*** 46 leetcode. (Permutations)

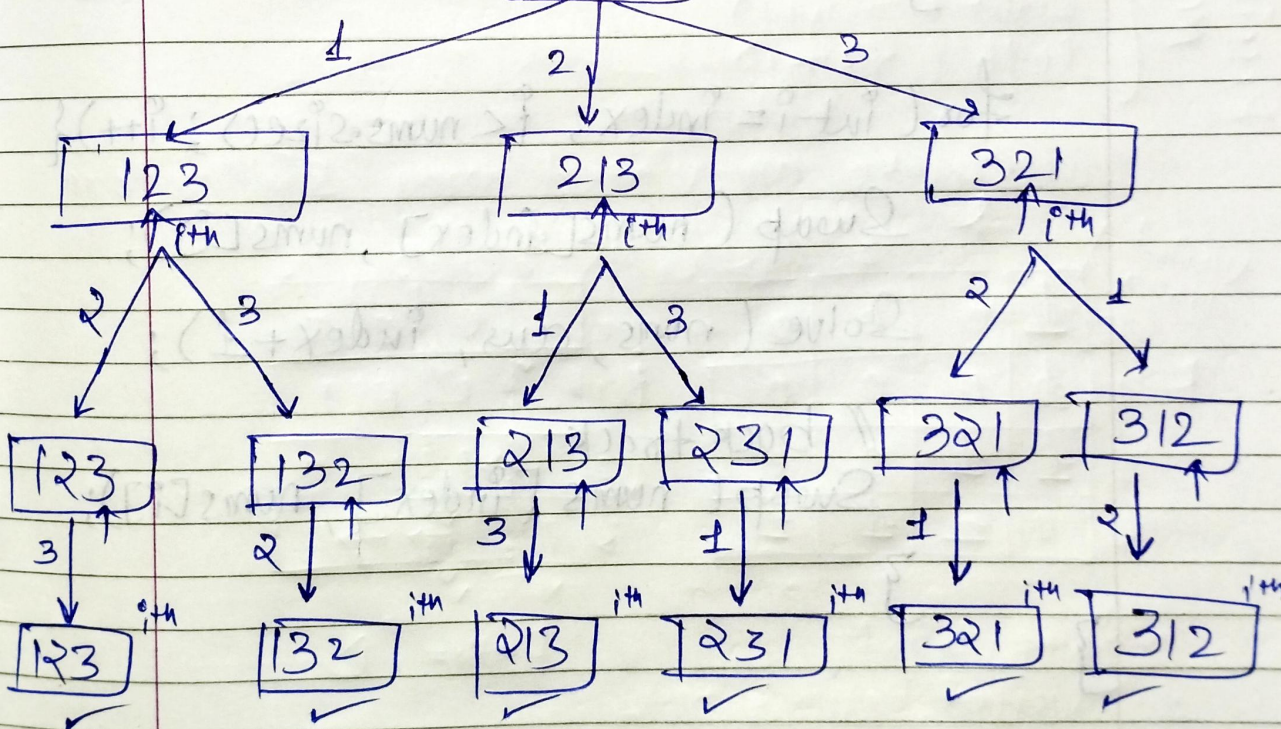
Given an Integer array nums of distinct integers, return the answer in any order.

Eg:- nums = [1,2,3]

o/p = [[1,2,3], [1,3,2], [2,1,3], [2,3,1], [3,1,2], [3,2,1]]

Approach: 

for the i^{th} position



Page No.
Date

Code :

```
class solution{
Private :
void Solve( vector<int>nums , vector<vector<
                int>> &ans , int index)
{
    // base case
    if(index >= nums.size()){
        ans.push_back(nums);
        return ;
    }

for( int i= index; i<nums.size() ; i++){

    Swap (nums[index] , nums[i]);

    Solve ( nums, ans, index+1);

    // backtrack
    Swap( nums [index], nums[i]);

    }
}
```

Page No.
Date

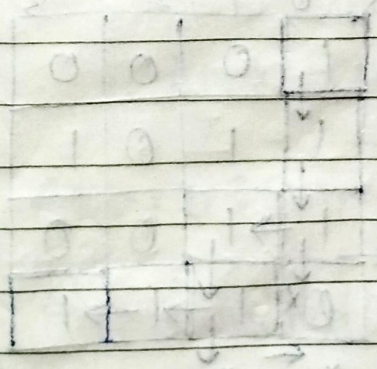
```
public:
    vector<vector<int>> permute(vector<
        int>& nums)
    {

        vector<vector<int>> ans;
        int index = 0;

        Solve(nums, ans, index);
        return ans;

    }

};
```



Super imp (Recursion)

Page No.
Date

***
**

Q. Rat in a Maze : y(tol)

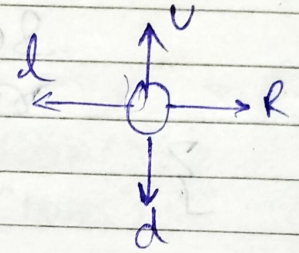
IIP

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

n (Rows)

Src → (0,0)
dest → ((n-1),(n-1))

1 → open path
0 → closed path



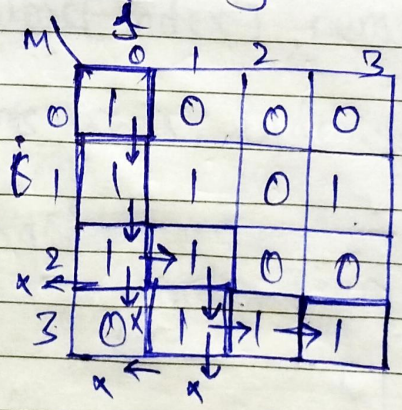
Possible moves
- up (x-1, y)
- down (x+1, y)
- Right (x, y+1)
- left (x, y-1)

lets make an visited array similar to m

Visited array

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

0 → Not visited
1 → visited.



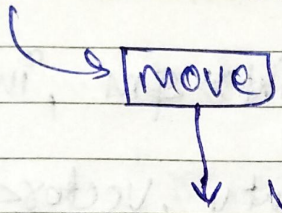
Path = "D "

M(i,j) → M(K,l)

Conditions

Page No. Date

(•) within the matrix
(•) M(K,l) = 1 // Allowed to move
(•) visited [K][l] = 0 // Not visited.



→ when fun^n call return
↳ visited[K][l] = 0

Code :

```
Class Solution {
Private:
bool issafe (int x, int y, int n, vector<vector
        <int>> visited, vector<vector<int>>&m)
 {   //check conditions
if( (x>=0 && x<n) && ( y>=0 && y<n)
    && (visited[x][y]==0) && ( m[x][y]==1))
  {
    return true;
  }
else{
    return false;
  }
```

Page No.
Date

```
}

Void Solve (vector<vector<int>> &m , int n,
vector<string> &ans, int x, int y , vector<vector<
int>> visited, string path) {
  // reached at the Positon of x and y
   // base case
if( (x == n-1) && (y == n-1)){
      ans.push_back( path);
      return;
}
   // visited the Positions
   visited[x][y] = 1;
  // 4 choices to move. ort -> D, U, R, L
  // Down
int newx = x+1;
int newy = y;
```

Page No.
Date

```
if( issafe (newx,newy ,n ,visited, m)){
        Path.push_back('D');
        // recursive call
    Solve(m,n, ans ,newx,newy ,visited,Path);
        Path.pop_back();
}
// Left
newx=x
newy=y-1;
if( issafe( newx,newy, n, visited,m)){
        Path.push_back('L');
        // recursive call
    Solve(m,n, ans ,newx,newy, visited,Path);
        Path.pop_back();
}
// Right
newx=x;
newy=y+1;
```

Page No.
Date

```
if(issafe(newx, newy, n, visited, m)){
        Path.Push_back('R');
        // recursive call
    Solve(m, n, ans, newx, newy, visited, Path);
        Path.Pop_back();
}
// up
newx = x-1;
newy = y
if(issafe(newx, newy, n, visited, m)){
        Path.Push_back('U');
        // recursive call
    Solve(m, n, ans, newx, newy, visited, Path);
        Path.Pop_back();
}
// set to Not visited
visited[x][y] = 0;
```

Page No. | Date

Public:

```
vector<string> findPath(vector<vector>> int &m,
                        int n)
{
    vector<string> ans;
    if(m[0][0]=0){
        return ans;
    }
    int srcx=0;
    int srcy=0;

    vector<vector<int>> visited=m;
    // Initization of visited to 0
    for(int i=0; i<n; i++){
        for(int j=0; j<n; j++){
            visited[i][j]=0;
        }
    }
    string path="";
    // function call
    solve(m,n,ans,srcx,srcy,visited,path);
```

Page No.
Date

```
// sorted in lexographically order
sort(ans.begin(), ans.end());
return ans;
}
};
```

Eg:- N = 4

M[][] =

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

4 x 4

O/P = DDRDRR , DRDDRR.