Date : 13th May
Mentor : Gladden Rumao
Topic : Backtracking Revision

Q1. Consider a rat placed at (0, 0) in a square matrix of order N * N. It has to reach the destination at (N - 1, N - 1). Find all possible paths that the rat can take to reach from source to destination. The directions in which the rat can move are 'U'(up), 'D'(down), 'L' (left), 'R' (right). Value 0 at a cell in the matrix represents that it is blocked and rat cannot move to it while value 1 at a cell in the matrix represents that rat can travel through it.
Note: In a path, no cell can be visited more than one time. If the source cell is 0, the rat cannot move to any other cell.
Problem Link : https://practice.geeksforgeeks.org/problems/rat-in-a-maze-problem/1

Input:
N = 4
m[][] = {{1, 0, 0, 0},
{1, 1, 0, 1},
{1, 1, 0, 0},
{0, 1, 1, 1}}
Output:
DDRDRR DRDDRR

Explanation:
The rat can reach the destination at
(3, 3) from (0, 0) by two paths - DRDDRR
and DDRDRR, when printed in sorted order
we get DDRDRR DRDDRR.

Solution :

```
class Solution {
    public static void solveMaze(ArrayList<String>
ans, int r, int c, int arr[][],int n, String p,
boolean vis[][]) {
        if (r == n-1 && c == n-1 && arr[r][c] != 0)
{
            ans.add(p);
            return;
        }
```

```
        if (r >= 0 && c >= 0 && r < n && c < n ) {
            if (vis[r][c]==true || arr[r][c] == 0)
                return;

            vis[r][c] = true;
            solveMaze(ans, r + 1, c, arr,n, p +
'D', vis);
            solveMaze(ans, r, c - 1, arr,n, p +
'L', vis);
            solveMaze(ans, r, c + 1, arr,n, p +
'R', vis);
            solveMaze(ans, r - 1, c, arr,n, p +
'U', vis);
            vis[r][c] = false;

        }

    }
    public static ArrayList<String>
findPath(int[][] m, int n) {
        ArrayList<String> ans = new ArrayList<>();
        boolean vis[][] = new boolean[n][n];
        solveMaze(ans, 0, 0, m,n, "", vis);
        return ans;
    }
}
```

Q2. Given an integer array nums of unique elements, return all possible

subsets (the power set).
Problem Link : https://leetcode.com/problems/subsets/

Example 1:

Input: nums = [1,2,3]
Output: [[],[1],[2],[1,2],[3],[1,3],[2,3],[1,2,3]]

Solution :

```
class Solution {
    public void backtrack(List<List<Integer>>
result , List<Integer> temp , int[] nums , int
start){
        result.add(new ArrayList<>(temp));
        for(int i= start ; i<nums.length ; i++){
            temp.add(nums[i]);
            backtrack(result,temp,nums,i+1);
            temp.remove(temp.size()-1);
        }
    }
    public List<List<Integer>> subsets(int[] nums)
{
        List<List<Integer>> result = new
ArrayList<>();
        backtrack(result , new ArrayList<>() , nums
, 0);
        return result;
    }
}
```