Recursive Implementation of atoi()

13 April 2025 03:32

Pow(x, n)

13 April 2025 03:33

Count Good numbers

13 April 2025 03:33

Sort a stack using recursion

13 April 2025 03:33

Reverse a stack using recursion

13 April 2025 03:33

Generate all binary strings

13 April 2025 03:33

Generate Paranthesis

13 April 2025 03:34

Print all subsequences/Power Set

13 April 2025 03:34

```
import java.util.*;

public class SubsetGenerator {

    public void find(int idx, List<List<Integer>> ans, List<Integer> res, int[] nums, int n) {
        if (idx == n) {
            ans.add(new ArrayList<>(res)); // Clone the current subset
            return;
        }

        // Not take
        find(idx + 1, ans, res, nums, n);

        // Take
        res.add(nums[idx]);
        find(idx + 1, ans, res, nums, n);
        res.remove(res.size() - 1); // Backtrack
    }

    public List<List<Integer>> subsets(int[] nums) {
        List<List<Integer>> ans = new ArrayList<>();
        List<Integer> res = new ArrayList<>();
        find(0, ans, res, nums, nums.length);
        return ans;
    }

    public static void main(String[] args) {
        SubsetGenerator obj = new SubsetGenerator();
        int[] nums = {1, 2, 3};
        List<List<Integer>> result = obj.subsets(nums);
        System.out.println(result);
    }
}
```

```
//{ Driver Code Starts
//Initial Template for Java

import java.util.*;
import java.lang.*;
import java.io.*;
class GFG
{
    public static void main(String[] args) throws IOException
```

```
    {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        int T = Integer.parseInt(br.readLine().trim());
        while(T-->0)
        {
            String s = br.readLine().trim();
            Solution ob = new Solution();
            List<String> ans = ob.AllPossibleStrings(s);
            for(String i: ans)
                System.out.print(i + " ");
            System.out.println();


System.out.println("~");
}
    }
}

// } Driver Code Ends


//User function Template for Java

class Solution
{
    public void find(int idx,List<String>ans,String res,String s,int n)
    {
        if(idx==n)
        {
            if(!res.isEmpty())
            {
                ans.add(res);

            }return;
        }

        find(idx+1,ans,res,s,n);
        //res+=s.CharAt(idx);
        find(idx+1,ans,res+s.charAt(idx),s,n);

    }
    public List<String> AllPossibleStrings(String s)
    {
        List<String>ans=new ArrayList<String>();
        //String s1="";
        find(0,ans,"",s,s.length());
        Collections.sort(ans);
        return ans;
    }
}
```

Learn All Patterns of Subsequences (Theory)

13 April 2025 03:34

Count all subsequences with sum K

13 April 2025 03:34

```
class Solution {
    int cnt = 0;

    public void find(int idx, int sum, int[] nums, int target, int n) {
        if (idx == n) {
            if (sum == target) {
                cnt++;
            }
            return;
        }

        // Take the current element
        find(idx + 1, sum + nums[idx], nums, target, n);

        // Do not take the current element
        find(idx + 1, sum, nums, target, n);
    }

    public int countSubsequenceWithTargetSum(int[] nums, int k) {
        cnt = 0; // Reset count in case method is reused
        find(0, 0, nums, k, nums.length);
        return cnt;
    }
}
```

Check if there exists a subsequence with sum K

13 April 2025 03:34

Combination Sum

13 April 2025 03:34

Combination Sum-II

13 April 2025 03:34

Subset Sum-I

13 April 2025 03:34

Subset Sum-II

13 April 2025 03:34

Combination Sum - III

13 April 2025 03:34

Letter Combinations of a Phone number

13 April 2025 03:35