**Day-22**

**Problem1**-   
[**79. Word Search**](https://leetcode.com/problems/word-search/)

class Solution {

private:

    bool backtrack(vector<vector<char>>& board, string &word, int i, int j, int k, int m, int n) {

        if (k == word.size()) {

            return true;

        }

        if (i < 0 || i >= m || j < 0 || j >= n || board[i][j] != word[k]) {

            return false;

        }

        char temp = board[i][j];

        board[i][j] = '\0';  // mark as visited

        bool found = backtrack(board, word, i + 1, j, k + 1, m, n) ||

                     backtrack(board, word, i - 1, j, k + 1, m, n) ||

                     backtrack(board, word, i, j + 1, k + 1, m, n) ||

                     backtrack(board, word, i, j - 1, k + 1, m, n);

        board[i][j] = temp;  // restore original character

        return found;

    }

public:

    bool exist(vector<vector<char>>& board, string word) {

        int m = board.size();

        int n = board[0].size();

        for (int i = 0; i < m; i++) {

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

                if (backtrack(board, word, i, j, 0, m, n)) {

                    return true;

                }

            }

        }

        return false;

    }

};

**Problem2**-   
**All Unique Permutations of an array**

//{ Driver Code Starts

//Initial Template for C++

#include <bits/stdc++.h>

using namespace std;

// } Driver Code Ends

//User function Template for C++

class Solution {

public:

void solve(int i,int n,vector<int>&p,vector<vector<int>>&ans,vector<int> &arr ,bool v[])

{

if(i==n)

{

ans.push\_back(p);

return;

}

int j;

for(j=0;j<n;j++)

{

if(v[j]==false)

{

if(j>0)

{

if(arr[j]!=arr[j-1] || v[j-1]==true)

{

p.push\_back(arr[j]);

v[j]=true;

solve(i+1,n,p,ans,arr,v);

v[j]=false;

p.pop\_back();

}

}

else

{

p.push\_back(arr[j]);

v[j]=true;

solve(i+1,n,p,ans,arr,v);

v[j]=false;

p.pop\_back();

}

}

}

}

vector<vector<int>> uniquePerms(vector<int> &arr ,int n) {

// code here

vector<int>p;

vector<vector<int>>ans;

bool v[n];

for(int i=0;i<n;i++)

{

v[i]=false;

}

sort(arr.begin(),arr.end());

solve(0,n,p,ans,arr,v);

return ans;

}

};

//{ Driver Code Starts.

int main() {

int t;

cin >> t;

while (t--) {

int n;

cin>>n;

vector<int> arr(n);

for(int i=0 ; i<n ; i++)

cin>>arr[i];

Solution ob;

vector<vector<int>> res = ob.uniquePerms(arr,n);

for(int i=0; i<res.size(); i++)

{

for(int j=0; j<n; j++)

{

cout<<res[i][j]<<" ";

}

cout<<"\n";

}

}

return 0;

}

// } Driver Code Ends

**Problem 3**- [**381. Insert Delete GetRandom O(1) - Duplicates allowed**](https://leetcode.com/problems/insert-delete-getrandom-o1-duplicates-allowed/)

class RandomizedCollection {

public:

vector<int>v;

unordered\_map<int,int>m;

    RandomizedCollection() {

    }

    bool insert(int val) {

        if(m[val]==0)

        { v.push\_back(val);

            m[val]++;

            return true;

        }

        else if (m[val]>0)

        {   v.push\_back(val);

            m[val]++;

            return false;

        }

        return false;

    }

    bool remove(int val) {

        if(m[val]>0)

        {

            auto it=find(v.begin(),v.end(),val);

            v.erase(it);

            m[val]--;

            return true;

        }

        return false;

    }

    int getRandom() {

        return v[rand()%v.size()];

    }

};

/\*\*

 \* Your RandomizedCollection object will be instantiated and called as such:

 \* RandomizedCollection\* obj = new RandomizedCollection();

 \* bool param\_1 = obj->insert(val);

 \* bool param\_2 = obj->remove(val);

 \* int param\_3 = obj->getRandom();

 \*/

**Problem 4** [**84. Largest Rectangle in Histogram**](https://leetcode.com/problems/largest-rectangle-in-histogram/)

class Solution {

private:

    int n;

    vector<int> nextSmaller(vector<int> &nums) {

        vector<int> next(n, n);

        stack<int> s;

        s.push(n - 1);

        for(int i = n - 2; i >= 0; i--) {

            while(!s.empty() && nums[s.top()] >= nums[i]) s.pop();

            next[i] = s.empty() ? n : s.top();

            s.push(i);

        }

        return next;

    }

    vector<int> prevSmaller(vector<int> &nums) {

        vector<int> prev(n, -1);

        stack<int> s;

        s.push(0);

        for(int i = 1; i < n; i++) {

            while(!s.empty() && nums[s.top()] >= nums[i]) s.pop();

            prev[i] = s.empty() ? -1 : s.top();

            s.push(i);

        }

        return prev;

    }

public:

    int largestRectangleArea(vector<int> &heights) {

        n = heights.size();

        vector<int> next = nextSmaller(heights);

        vector<int> prev = prevSmaller(heights);

        int ans = 0;

        for(int i = 0; i < n; i++) {

            ans = max(ans, (next[i] - prev[i] - 1) \* heights[i]);

        }

        return ans;

    }

};