Experiment3.2

NAME – RAJDEEP JAISWAL

UID 20BCS2761

BRANCH-BTECH CSE

SUB- CC LAB

1. Aim: ⇨ To implement the concept of backtracking.

2. Objective:

⇨ The objective is to build problem solving capability and to learn the basic concepts of data structures.

⇨ The implementation of Combination which shows and brushes up the concept of greedy.

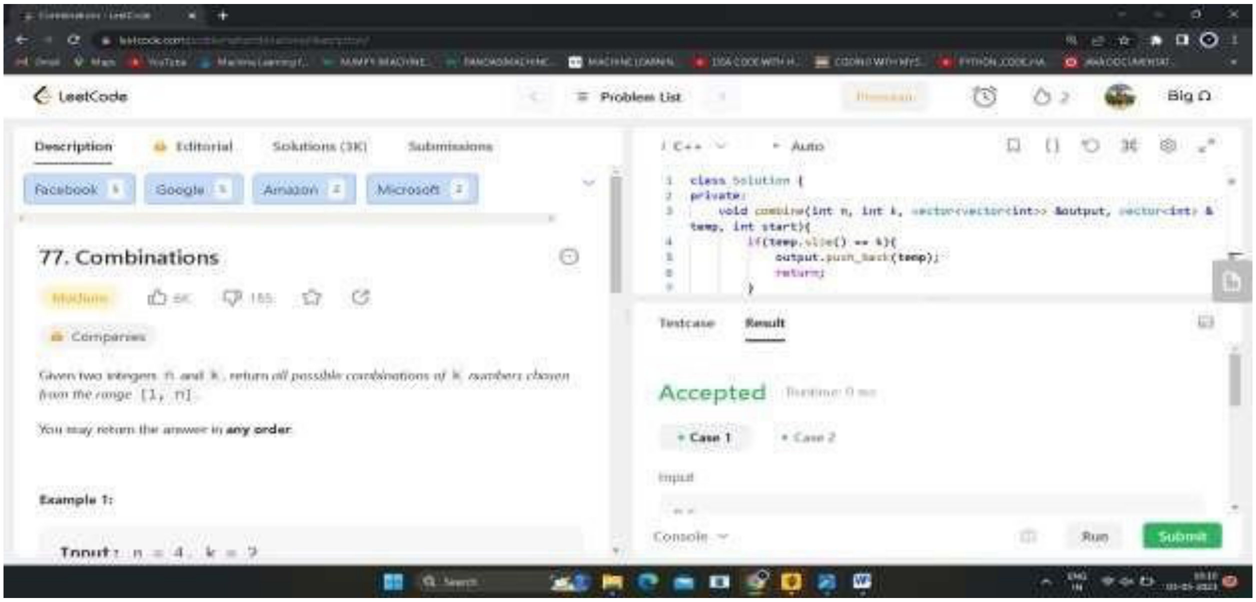
.

3. Leetcode code and output:

□ **COMBINATION CODE=**

```
class Solution {  private:      void combine(int n, int k, vector<vector<int>>
&output, vector<int> &temp, int start){         if(temp.size() == k){
output.push_back(temp);                return;
        }         for(int i=start;
i<=n; i++){
temp.push_back(i);              combine(n, k,
output, temp, i+1);              temp.pop_back();
        }
}      public:
    vector<vector<int>> combine(int n, int k) {
vector<vector<int>> output;         vector<int>
temp;
        combine(n, k, output, temp, 1);
return output;
} };
```

OUTPUT=



77. Combinations

Medium

Companies

Given two integers n and k, return all possible combinations of k numbers chosen from the range [1, n].

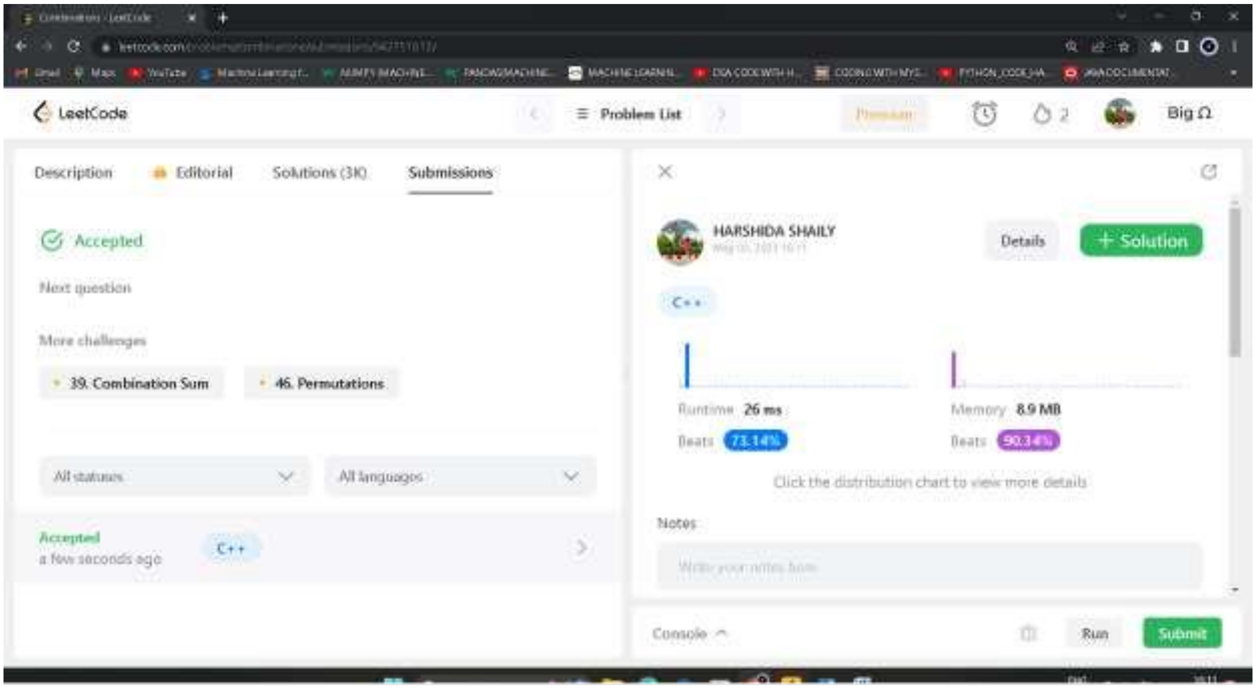
You may return the answer in any order.

Example 1:

Input: n = 4, k = 2

```
class Solution {
private:
    void combine(int n, int k, vector<vector<int>> &output, vector<int> &
temp, int start){
        if(temp.size() == k){
            output.push_back(temp);
            return;
        }
```

Accepted Runtime: 0 ms



Accepted

Next question

More challenges

39. Combination Sum

46. Permutations

All statuses

All languages

Accepted a few seconds ago C++

HARSHIDA SHAILY

C++

Runtime 26 ms Beats 73.14%

Memory 8.9 MB Beats 90.34%

Click the distribution chart to view more details

Notes

Write your notes here

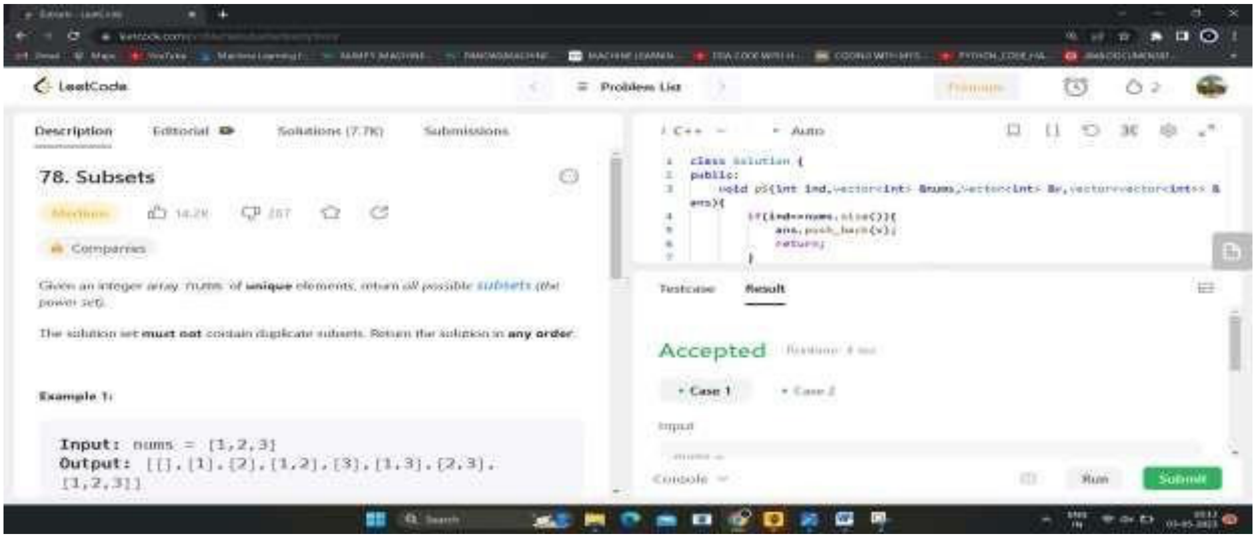
□ **SUBETS**

CODE

```
class Solution { public:
    void pS(int ind,vector<int> &nums,vector<int> &v,vector<vector<int>> &ans){
if(ind==nums.size()){              ans.push_back(v);             return;


        }
        v.push_back(nums[ind]);
pS(ind+1,nums,v,ans);
        v.pop_back();
pS(ind+1,nums,v,ans);
    }        vector<vector<int>>
subsets(vector<int>& nums) {
vector<vector<int>> ans;         vector<int> v;
pS(0,nums,v,ans);         return ans;
    } };
```

OUTPUT=



78. Subsets

Given an integer array nums of **unique** elements, return *all possible* subsets *(the power set)*.

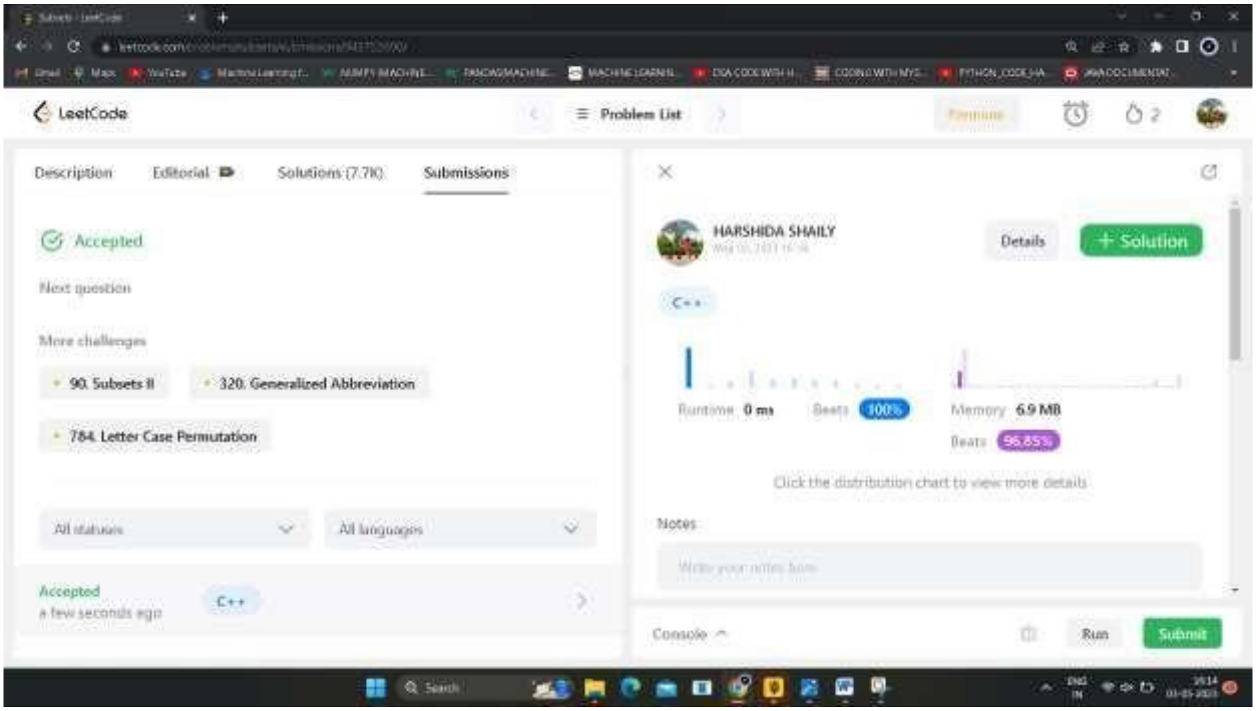
The solution set **must not** contain duplicate subsets. Return the solution in **any order**.

Example 1:

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

```
class Solution {
public:
    void pS(int ind,vector<int> &nums,vector<int> &v,vector<vector<int>> &
ans){
        if(ind==nums.size()){
            ans.push_back(v);
            return;
        }
```

Accepted Runtime: 4 ms



Accepted

HARSHIDA SHAILY

Runtime 0 ms Beats 100%

Memory 6.9 MB Beats 96.85%

Click the distribution chart to view more details

Accepted
a few seconds ago C++