

Assignment-04

1. Longest Nice Substring: <https://leetcode.com/problems/longest-nice-substring/description/>

The screenshot shows the LeetCode interface for the 'Longest Nice Substring' problem. On the left, the 'Runtime' section indicates a submission by 'Priyanshu Chaurasia' that is 'Accepted' with a runtime of 347 ms, beating 11.68% of other submissions. The memory usage is 134.52 MB, beating 5.09%. A bar chart shows the runtime distribution. The 'Code' section displays the C++ solution:

```
class Solution {
public:
    bool check(string s){
        set<char>cfreq;
        set<char>sfreq;
        for(auto i:s){
            if(i>='a' && i<='z') cfreq.insert(i);
            else sfreq.insert(i);
        }
        for(auto i:cfreq){
            if(cfreq.count(toupper(i))==0) return false;
        }
        for(auto i:sfreq){
            if(sfreq.count(tolower(i))==0) return false;
        }
        return true;
    }
    string longestNiceSubstring(string s) {
        int maxi=0;
        string ans="";
        for(int i=0; i<s.length(); i++){
            for(int j=i+1; j<s.length(); j++){
                string st=s.substr(i, j-i+1);
                if(check(st)){
                    if((int)st.size()>maxi){
                        maxi=st.size();
                        ans=st;
                    }
                }
            }
        }
        return ans;
    }
};
```

2. Reverse Bits: <https://leetcode.com/problems/reverse-bits/description/>

The screenshot shows the LeetCode interface for the 'Reverse Bits' problem. On the left, the 'Runtime' section indicates a submission by 'Priy...' that is 'Accepted' with a runtime of 0 ms, beating 100.00% of other submissions. The memory usage is 7.66 MB, beating 87.24% of other submissions. A bar chart shows the runtime distribution. The 'Code' section displays the C++ solution:

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        n = (n >> 16) | (n << 16);
        n = ((n & 0xffff00ff) >> 8) | ((n & 0x00ff00ff) << 8);
        n = ((n & 0xf0f0f0f0) >> 4) | ((n & 0x0f0f0f0f) << 4);
        n = ((n & 0xcccccccc) >> 2) | ((n & 0x33333333) << 2);
        n = ((n & 0xaaaaaaaa) >> 1) | ((n & 0x55555555) << 1);
        return n;
    }
};
```

3. Number of 1 Bits: <https://leetcode.com/problems/number-of-1-bits/description/>

191. Number of 1 Bits Solved

Easy Topics Companies

Given a positive integer n , write a function that returns the number of set bits in its binary representation (also known as the **Hamming weight**).

Example 1:
Input: $n = 11$
Output: 3
Explanation:
The input binary string 1011 has a total of three set bits.

Example 2:
Input: $n = 128$
Output: 1
Explanation:
The input binary string 10000000 has a total of one set bit.

Example 3:
Input: $n = 2147483645$
Output: 30
Explanation:

```
class Solution {
public:
    int hammingWeight(int n) {
        int cnt=0;
        for(int i=0; i<31; i++){
            int num=1<<i;
            if((num&n)!=0){
                continue;
            }
            else{
                cnt++;
            }
        }
        return cnt;
    }
};
```

4. Maximum Subarray: <https://leetcode.com/problems/maximum-subarray/description/>

Maximum Subarray

Accepted 210 / 210 testcases passed
Priyanshu Chaurasia submitted at Mar 17, 2025 11:25

Runtime: 0 ms | Beats 100.00% | Memory: 71.63 MB | Beats 80.95%

Code | C++

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        int ans=INT_MIN, sm=0;

        for(auto i:nums){
            sm+=i;
            ans=max(ans, sm);
        }
        return ans;
    }
};
```

5. Search a 2D Matrix II: <https://leetcode.com/problems/search-a-2d-matrix-ii/description/>

The screenshot displays the LeetCode interface for the problem "Search a 2D Matrix II". The solution is accepted, with a runtime of 149 ms (beating 10.38%) and a memory usage of 18.81 MB (beating 36.77%). The code is written in C++ and implements a search algorithm that iterates through the matrix rows and uses binary search on each row.

Runtime: 149 ms | Beats 10.38%
Memory: 18.81 MB | Beats 36.77%

```
class Solution {
public:
    bool f(vector<vector<int>>& matrix, int s, int e, int row, int target){
        int mid=s+(e-s)/2;

        while(s<=e){
            if(matrix[row][mid]==target){
                return true;
            }
            else if(matrix[row][mid]<target){
                s=mid+1;
            }
            else{
                e=mid-1;
            }
            mid=s+(e-s)/2;
        }
        return false;
    }

    bool searchMatrix(vector<vector<int>>& matrix, int target) {
        for(int i=0; i<matrix.size(); i++){
            if(f(matrix, 0, matrix[i].size()-1, i, target)) return true;
        }
        return false;
    }
};
```

6. Super Pow: <https://leetcode.com/problems/super-pow/description/>

The screenshot displays the LeetCode interface for the problem "Super Pow". The solution is accepted, with a runtime of 7 ms (beating 7.17%) and a memory usage of 15.32 MB (beating 14.83%). The code is written in C++ and implements a recursive function to calculate the super power of a base and an exponent.

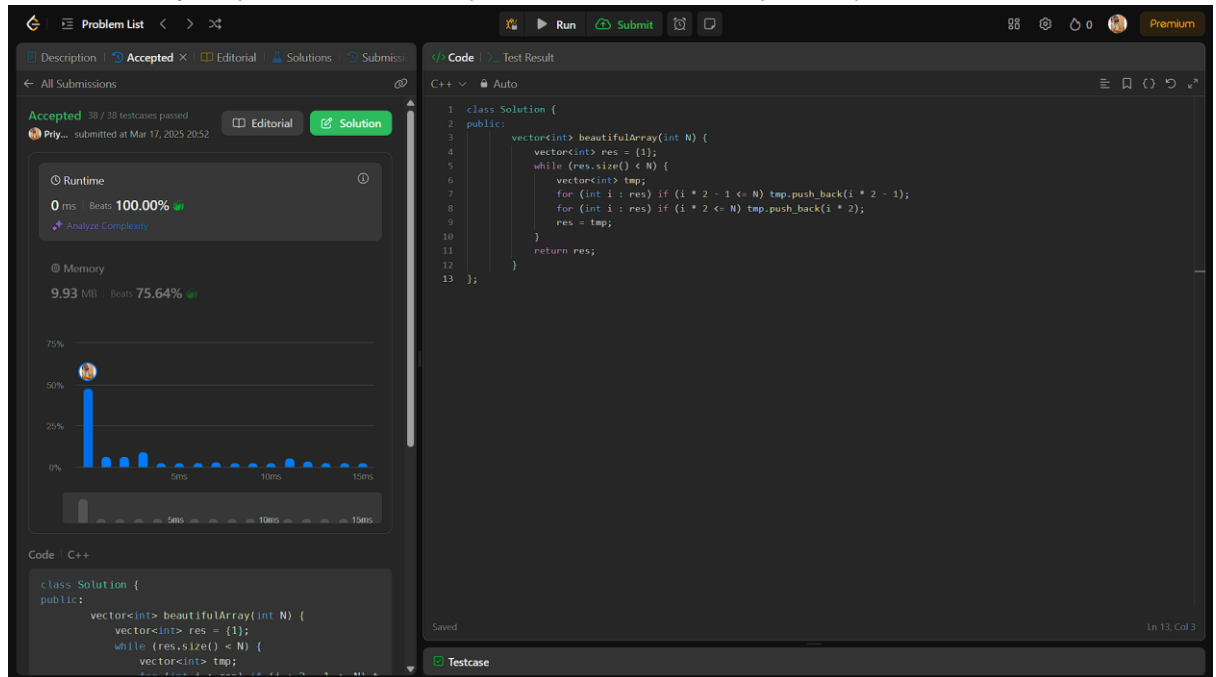
Runtime: 7 ms | Beats 7.17%
Memory: 15.32 MB | Beats 14.83%

Test Case Details:

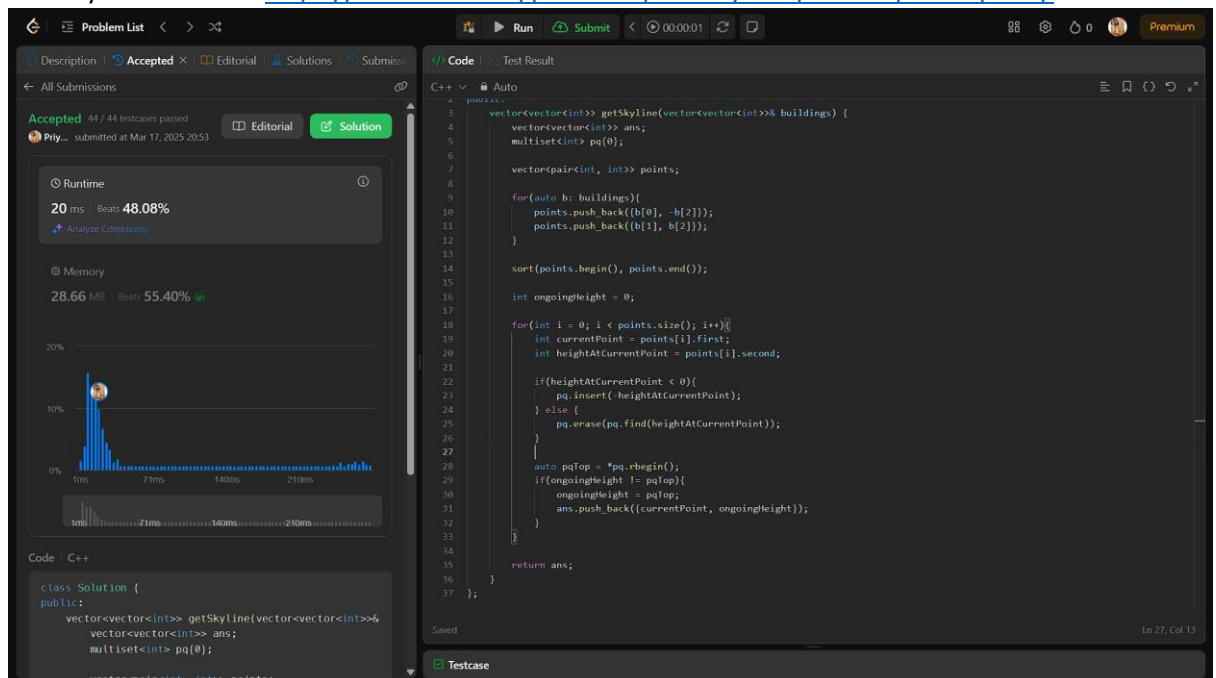
- Case 1: Input a = 2, b = [3], Output = 8, Expected = 8

```
class Solution {
    const int base = 1337;
    int powmod(int a, int k) //a^k mod 1337 where 0 <= k
    {
        a %= base;
        int result = 1;
        for(int i=0; i<k; i++){
            result = (result * a) % base;
        }
        return result;
    }
};
```

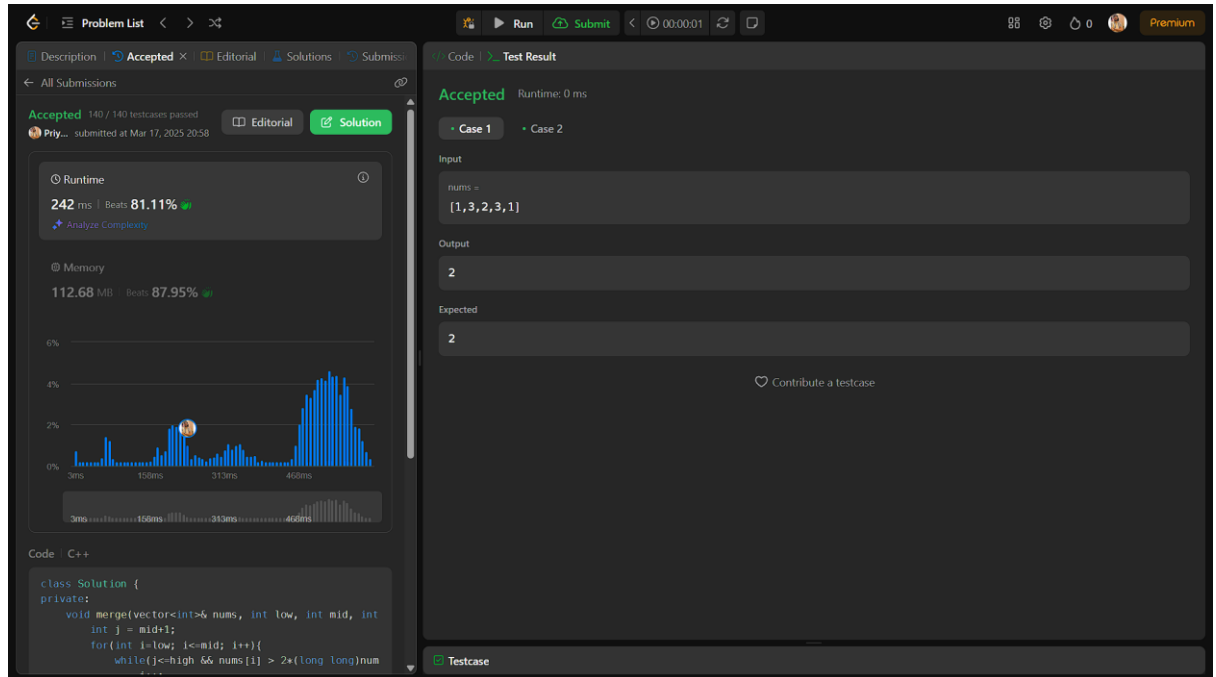
7. Beautiful Array: <https://leetcode.com/problems/beautiful-array/description/>



8. The Skyline Problem: <https://leetcode.com/problems/the-skyline-problem/description/>



9. Reverse Pairs: <https://leetcode.com/problems/reverse-pairs/description/>



10. Longest Increasing Subsequence II: <https://leetcode.com/problems/longest-increasing-subsequence-ii/description/>

