**ASSIGNMENT 6**

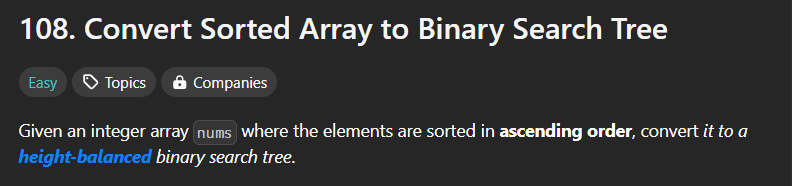
**Student Name: Mehak Chauhan UID: 22BCS12675**

**Branch: CSE Section: 22BCS\_IOT\_605 B**

**Semester: 6th DOP:19-03-2025**

**Subject: Advanced Programming Lab-II Subject Code: 22CSP-351**

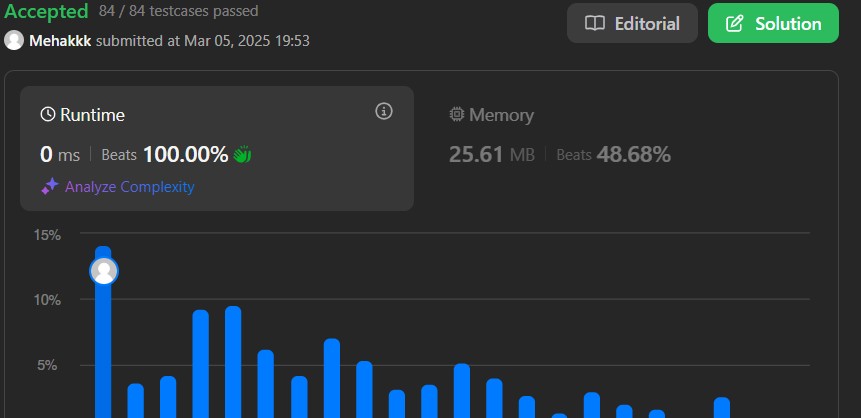
**Question 1**

****

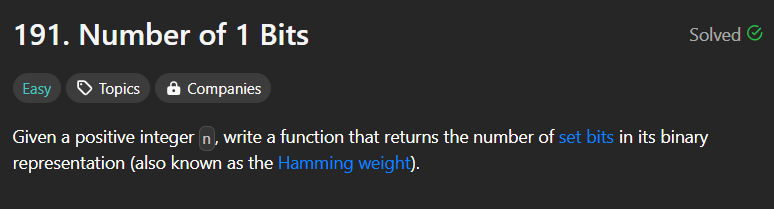
**Code:**

|  |
| --- |
| #include <vector>  using namespace std;  class Solution {  public:      TreeNode\* sortedArrayToBST(vector<int>& nums) {          return helper(nums, 0, nums.size() - 1);      }  private:      TreeNode\* helper(vector<int>& nums, int left, int right) {          if (left > right) return nullptr;          int mid = left + (right - left) / 2;          TreeNode\* root = new TreeNode(nums[mid]);          root->left = helper(nums, left, mid - 1);          root->right = helper(nums, mid + 1, right);          return root;      }  }; |

**Output:**



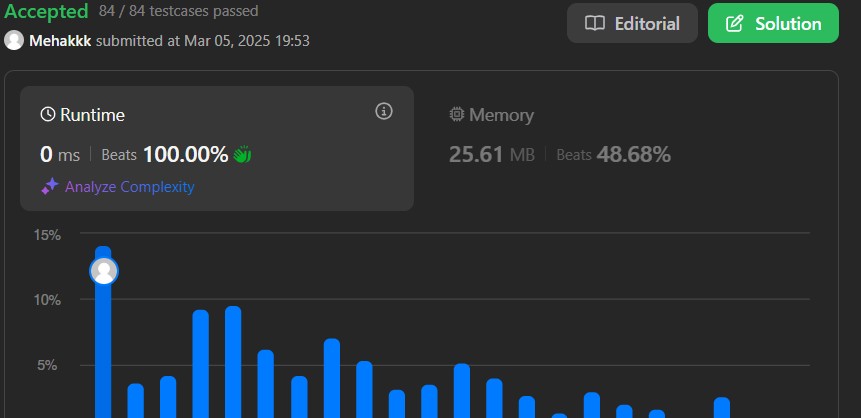
**Question 2**



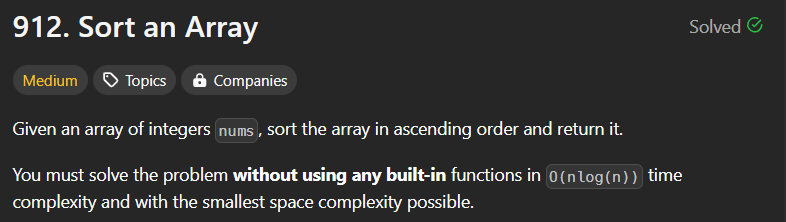
**Code:**

|  |
| --- |
| class Solution {  public:      int hammingWeight(uint32\_t n) {          int res = 0;          for (int i = 0; i < 32; i++) {              if (((n >> i) & 1) ==1) {                  res += 1;              }          }          return res;      }  }; |

**Output:**



**Question 3**

****

**Code:**

class Solution {

public:

    void merge(vector<int>& nums,int s, int e){

        int m=(s+e)/2;

        vector<int>first(m-s+1),second(e-m);

        for (int i=0;i<first.size();i++){first[i]=nums[s+i];}

        for (int i=0;i<second.size();i++){second[i]=nums[m+1+i];}

        int i1=0,i2=0,maindex=s;

        while (i1<first.size() && i2<second.size()){

            if (first[i1]<second[i2]){nums[maindex++]=first[i1++];}

            else {nums[maindex++]=second[i2++];}

        }

        while (i1<first.size()){nums[maindex++]=first[i1++];}

        while (i2<second.size()){nums[maindex++]=second[i2++];}

    }

    void mergesort(vector<int>&nums, int s, int e){

        if (s>=e){return ;}

        int m=(s+e)/2;

        mergesort(nums,s,m);

        mergesort(nums,m+1,e);

        merge(nums,s,e);

    }

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

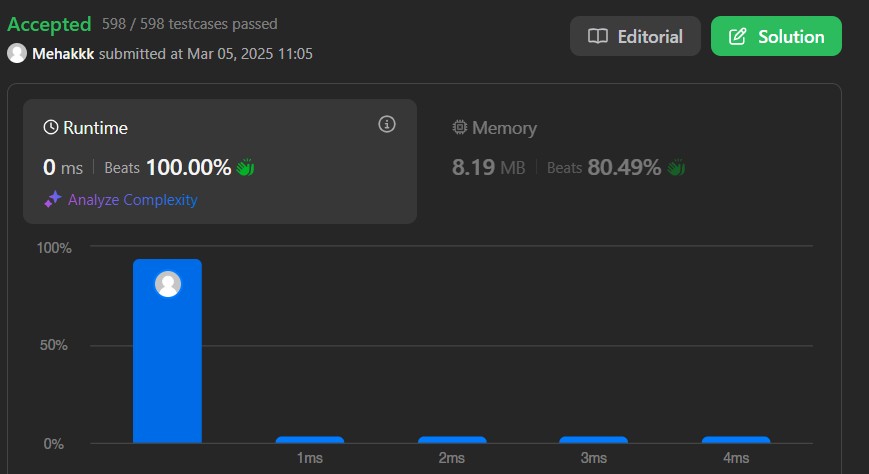
        mergesort(nums,0,nums.size()-1);

        return nums;

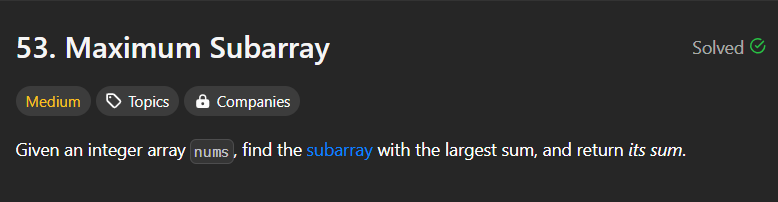
    }

};

**Output:**



**Question 4**

****

**Code:**

class Solution {

public:

    int maxSubArray(vector<int>& nums) {

        int result = nums[0];

        int curr\_sum = nums[0];

        for(int i=1;i<nums.size();i++){

            if(nums.size()==1){

                return nums[i];

            }

            curr\_sum= max(nums[i],curr\_sum+nums[i]);

            result= max(result,curr\_sum);

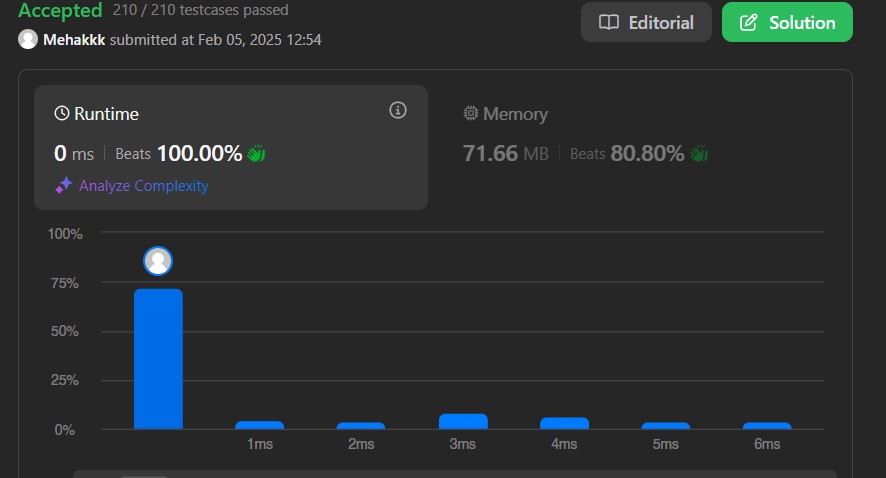
        }

        return result;

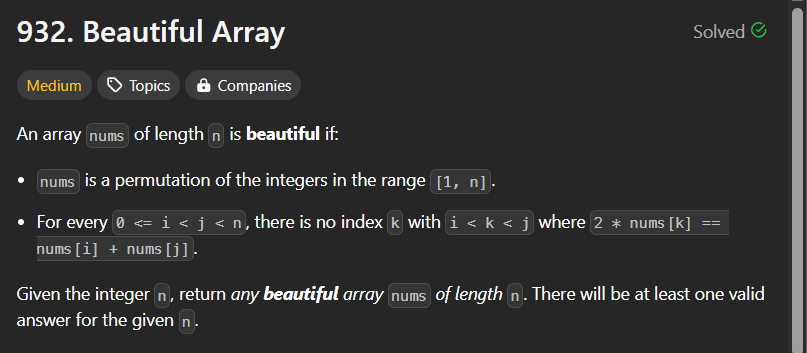
    }

};

**Output:**



**Question 5**



**Code:**

class Solution {

public:

    int partition(vector<int> &v, int start, int end, int mask)

    {

        int j = start;

        for(int i = start; i <= end; i++)

        {

            if((v[i] & mask) != 0)

            {

                swap(v[i], v[j]);

                j++;

            }

        }

        return j;

    }

    void sort(vector<int> & v, int start, int end, int mask)

    {

        if(start >= end) return;

        int mid = partition(v, start, end, mask);

        sort(v, start, mid - 1, mask << 1);

        sort(v, mid, end, mask << 1);

    }

    vector<int> beautifulArray(int N) {

        vector<int> ans;

        for(int i = 0; i < N; i++) ans.push\_back(i + 1);

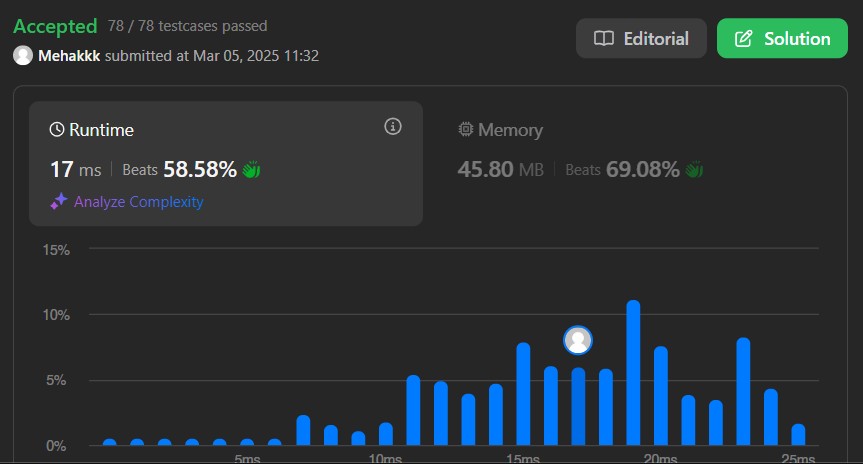
        sort(ans, 0, N - 1, 1);

        return ans;

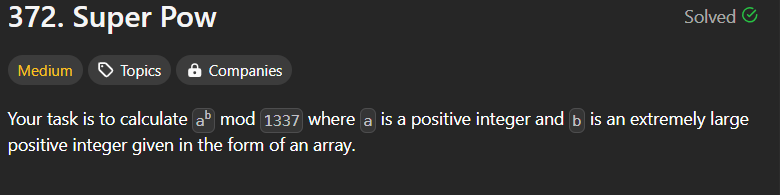
    }

};

**Output:**



**Question 6**

****

**Code**

int n=1337;

int phi=1140;

class Solution {

public:

    int Chinese\_Remainder(int a,  int x, vector<int>& b){

        if (x==n) return 0;

        int p=n/x;

        int M;//modInverse i.e. x\*M==1%p

        if (x==7) M=82;//can be computed by extended euclidean algorithm

        else M=4;

        int s=b.size();

        int exp=0;

        for(int i=0; i<s; i++)

            exp=(b[i]+10\*exp)%(p-1);

        bitset<12> e(exp);

        int y=1;

        a%=n;

        for(int i=11; i>=0; i--){

            y=y\*y%n;

            if (e[i]==1) y=y\*a%n;

        }

        int ans=y\*M\*x%n;

        while( ans<0)

            ans+=n;

    //    cout<<ans<<endl;

        return ans;//Chinese Remainder Theorem

    }

    int superPow(int a, vector<int>& b) {

        int g=gcd(a, n);

    //    cout<<"gcd="<<g<<endl;

        if (g!=1) return Chinese\_Remainder(a, g, b);

        int s=b.size();

        int exp=0;

        for(int i=0; i<s; i++)

            exp=(b[i]+10\*exp)%phi;

        bitset<12> e(exp);

        int y=1;

        a%=n;

        for(int&& i=11; i>=0; i--){

            y=y\*y%n;

            if (e[i]==1) y=y\*a%n;

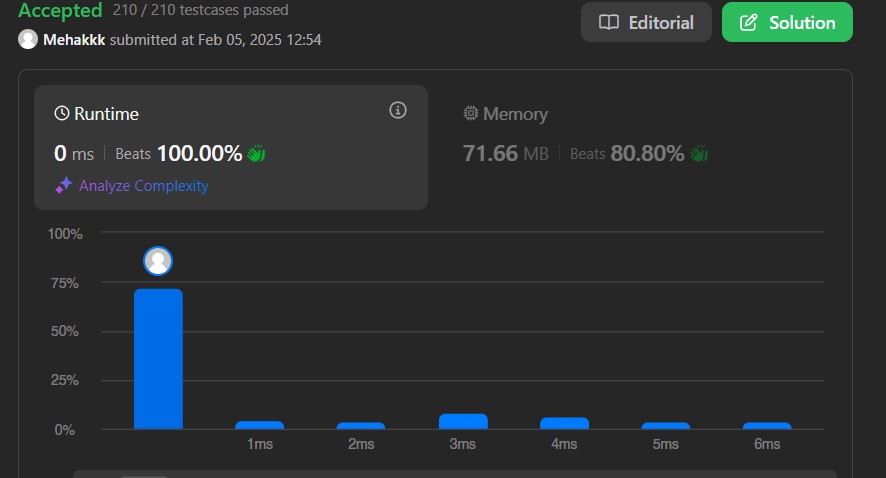
        }

        return y;

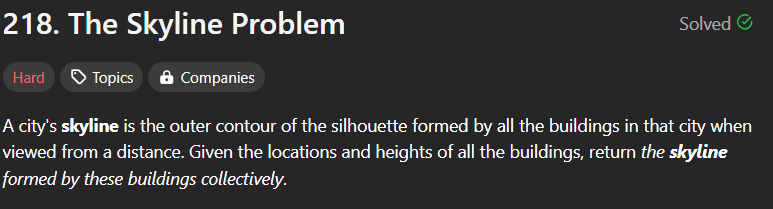
    }

};

**OUTPUT:**



**Question 7**

****

**CODE:**

class Solution {

public:

    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {

        vector<pair<int, int>> h;

        // Convert each building into two segments.

        for (auto b : buildings) {

            h.push\_back({b[0], -b[2]});

            h.push\_back({b[1], b[2]});

        }

        // Sort the segments.

        sort(h.begin(), h.end());

        int prev = 0, cur = 0;

        multiset<int> m;

        vector<vector<int>> res;

        m.insert(0);

        for (auto i:h) {

            // If i.second is less than zero, then it means it is left boundary.

            if (i.second < 0) {

                m.insert(-i.second);

            } else { // else it is right boundary.

                m.erase(m.find(i.second));

            }

            cur = \*m.rbegin();

            // If current maximum height is not equal to maximum previuous height, it is a key point.

            if (cur != prev) {

                res.push\_back({i.first, cur});

                prev = cur;

            }

        }

        return res;

}

};

**Output:**

