**[1763. Longest Nice Substring](https://leetcode.com/problems/longest-nice-substring/)**

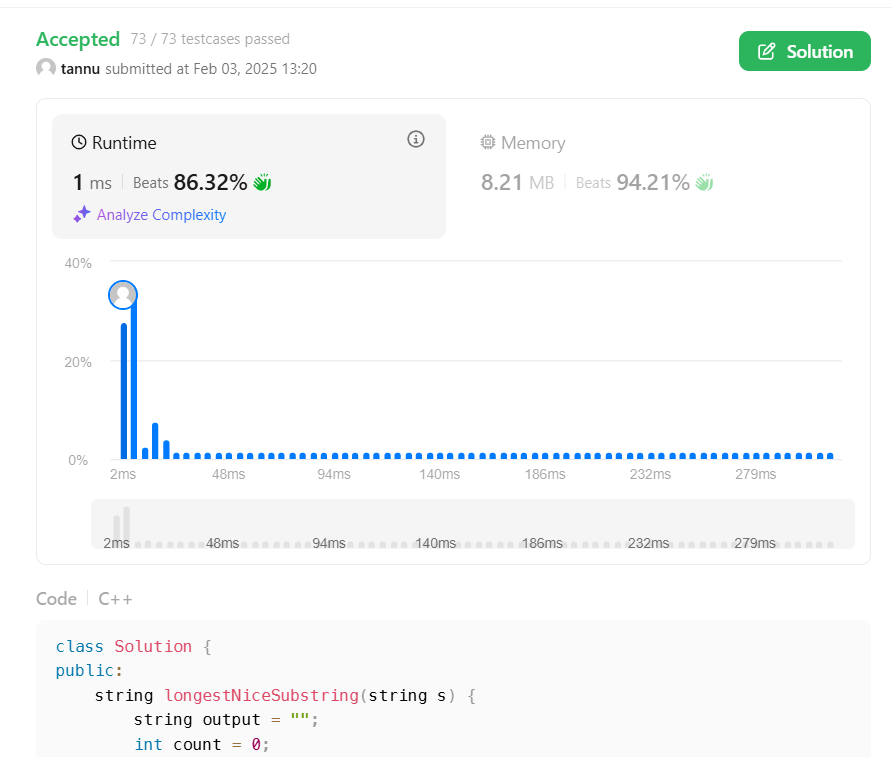
class Solution {

public:

    string longestNiceSubstring(string s) {

        string output = "";

        int count = 0;

        for(int i = 0;i<s.length();i++){

  int smallMask=0;

            int largeMask = 0;

            char ch = s[i];

            int chint = 0;

            if(ch>=65 && ch<=90){

                chint = ch-'A';

                largeMask = 1<<chint;

            }

            else{

                chint = ch-'a';

                smallMask = 1<<chint;

            }

            for(int j = i+1;j<s.length();j++){

                ch = s[j];

                if(ch>=65 && ch<=90){

                    chint = ch-'A';

                    largeMask |= 1<<chint;

                }

                else{

                    chint = ch-'a';

                    smallMask |= 1<<chint;

                }

                if((smallMask^largeMask) == 0){

                    if(count<j-i+1){

                        count = j-i+1;

                        string temp(s.begin()+i,s.begin()+j+1);

                        output = temp;

                    }

                }

            }

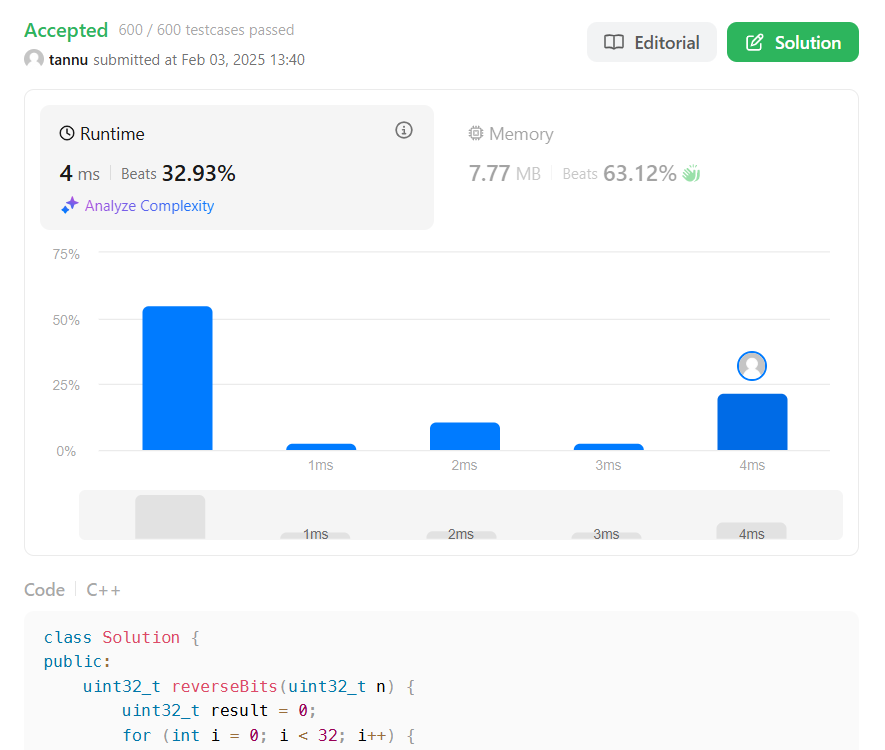
        }

        return output;

    }

};

[**190. Reverse Bits**](https://leetcode.com/problems/reverse-bits/)



class Solution {

public:

    uint32\_t reverseBits(uint32\_t n) {

        uint32\_t result = 0;

        for (int i = 0; i < 32; i++) {

            int bit = n & 1;

            result = (result << 1) | bit;

            n = n >> 1;

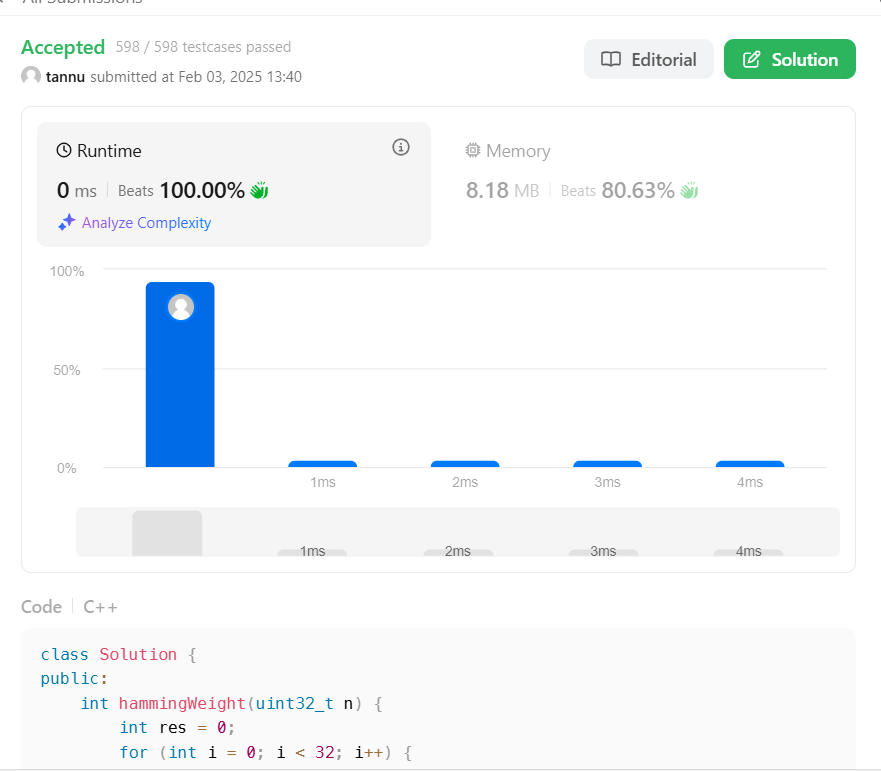
        }

        return result;

    }

};

[**191. Number of 1 Bits**](https://leetcode.com/problems/number-of-1-bits/)

class Solution {

public:

    int hammingWeight(uint32\_t n) {

        int res = 0;

        for (int i = 0; i < 32; i++) {

            if ((n >> i) & 1) {

                res += 1;

            }

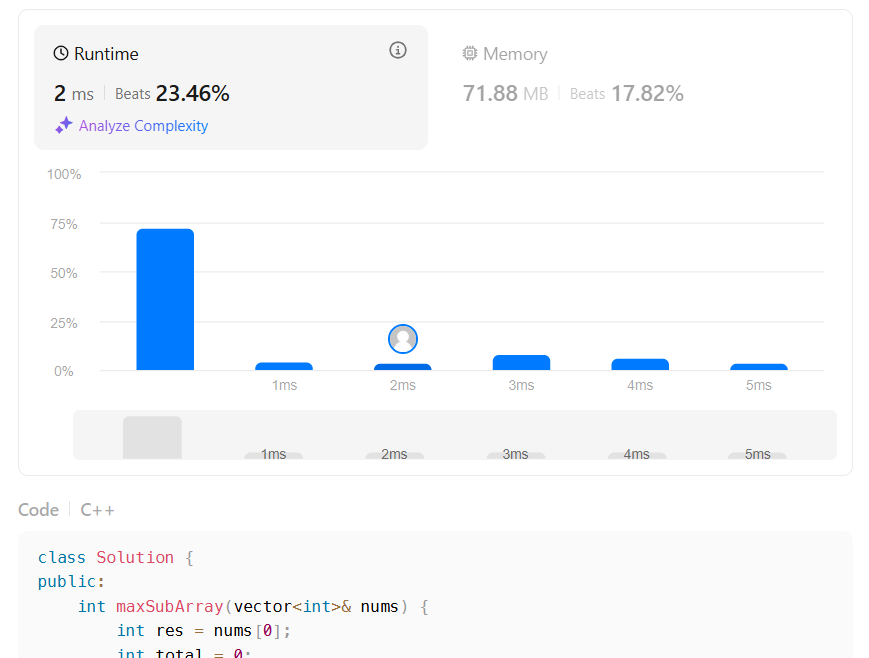
        }

        return res;

    }

};

[**53. Maximum Subarray**](https://leetcode.com/problems/maximum-subarray/)



class Solution {

public:

    int maxSubArray(vector<int>& nums) {

        int res = nums[0];

        int total = 0;

        for (int n : nums) {

            if (total < 0) {

                total = 0;

            }

            total += n;

            res = max(res, total);

        }

        return res;

    }

};

[**240. Search a 2D Matrix II**](https://leetcode.com/problems/search-a-2d-matrix-ii/)

class Solution {

public:

bool searchMatrix(vector<vector<int>>& matrix, int target) {

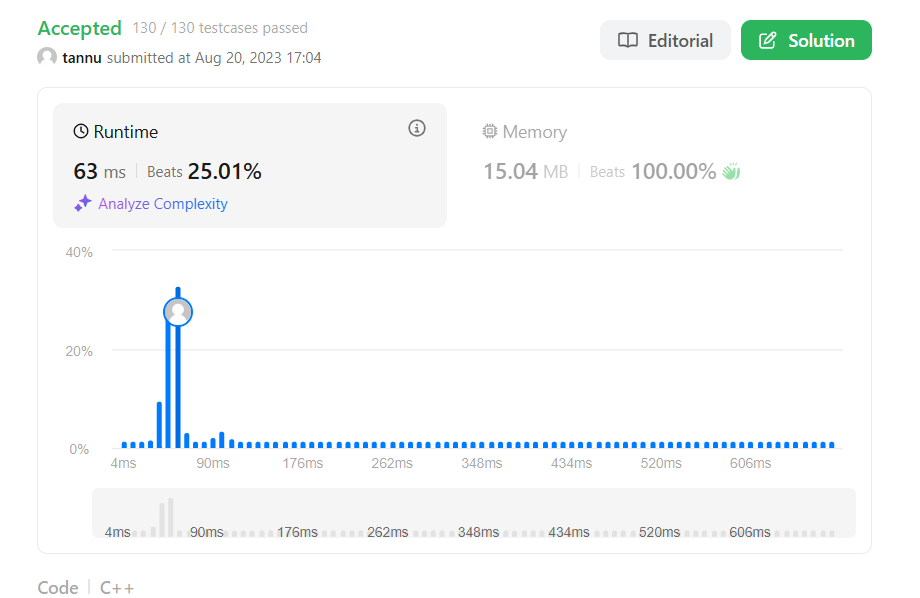
int row =matrix.size();

int col = matrix[0].size();

int rowIndex=0;

int colIndex= col-1;

while(rowIndex<row && colIndex>=0){

 int element=matrix[rowIndex][colIndex];

if(element == target){

return 1;

}

if(element<target){

rowIndex++;

}

else{

colIndex--;

}

return 0; }};

[**372. Super Pow**](https://leetcode.com/problems/super-pow/)

class Solution {

private:

    int solve(int base, int power, int mod) {

        int ans = 1;

        while (power > 0) {

            if (power & 1) {

                ans = (ans \* base) % mod;

            }

            base = (base \* base) % mod;

            power >>= 1;

        }

        return ans;

    }

public:

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

        a%=1337;

        int n = b.size();

        int m = 1140;

        int expi = 0;

        for(int i : b){

            expi = (expi\*10+i)%m;

        }

        if (expi == 0) {

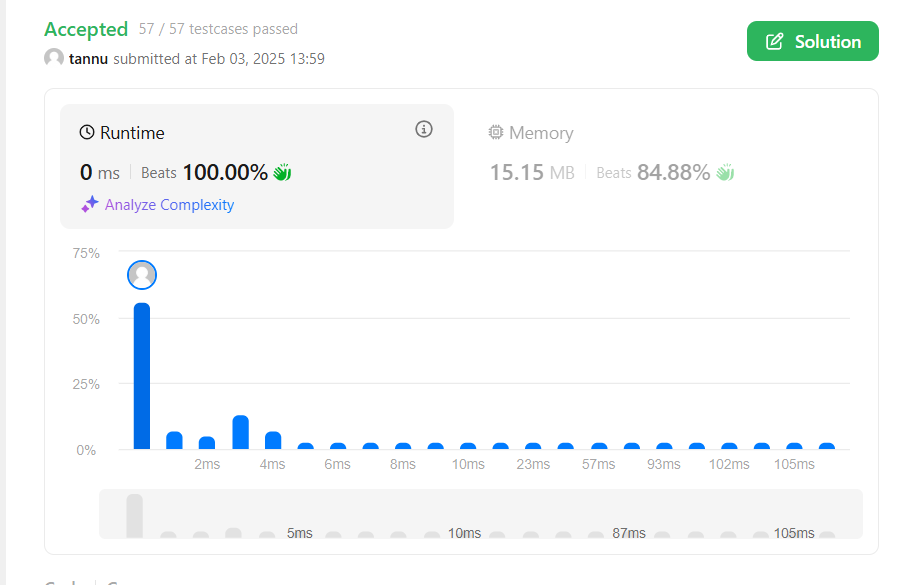
            expi = m;

        }

        return solve(a,expi,1337);

    }

};



[**932. Beautiful Array**](https://leetcode.com/problems/beautiful-array/)

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) {

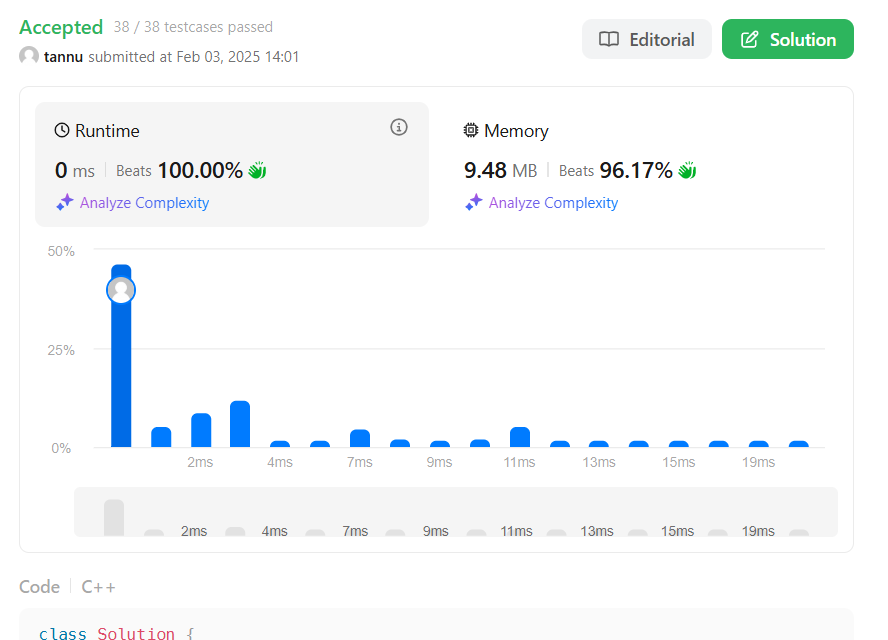
        vector<int> ans;

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

        sort(ans, 0, N - 1, 1);

        return ans;

    }

};

[**218. The Skyline Problem**](https://leetcode.com/problems/the-skyline-problem/)

class Solution {

public:

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

        int edge\_idx = 0;

        vector<pair<int, int>> edges;

        priority\_queue<pair<int, int>> pq;

        vector<vector<int>> skyline;

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

            const auto &b = buildings[i];

            edges.emplace\_back(b[0], i);

            edges.emplace\_back(b[1], i);

        }

        std::sort(edges.begin(), edges.end());

        while (edge\_idx < edges.size()) {

            int curr\_height;

            const auto &[curr\_x, \_] = edges[edge\_idx];

            while (edge\_idx < edges.size() &&

                    curr\_x == edges[edge\_idx].first) {

                const auto &[\_, building\_idx] = edges[edge\_idx];

                const auto &b = buildings[building\_idx];

                if (b[0] == curr\_x)

                    pq.emplace(b[2], b[1]);

                ++edge\_idx;

            }

            while (!pq.empty() && pq.top().second <= curr\_x)

                pq.pop();

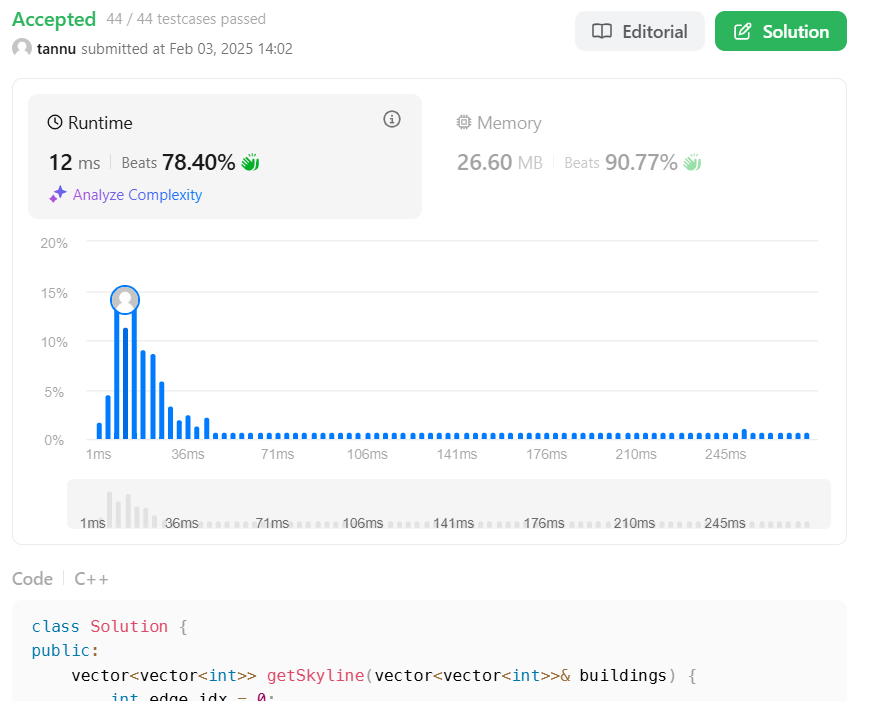
            curr\_height = pq.empty() ? 0 : pq.top().first;

            if (skyline.empty() || skyline.back()[1] != curr\_height)

                skyline.push\_back({curr\_x, curr\_height});

        }

        return skyline;

    }

};

[**493. Reverse Pairs**](https://leetcode.com/problems/reverse-pairs/)

class SegTree {

private:

    int tree\_size;

    vector<int> tree;

    void update(int lx, int rx, int ni, int idx) {

        if (rx - lx == 1) {

            tree[ni]++;

            return;

        }

        int m = (lx + rx) >> 1;

        if (idx < m)

            update(lx, m, ni \* 2 + 1, idx);

        else

            update(m, rx, ni \* 2 + 2, idx);

        tree[ni] = tree[ni \* 2 + 1] + tree[ni \* 2 + 2];

    }

    int query(int l, int r, int lx, int rx, int ni) {

        if (l >= rx || r <= lx)

            return 0;

        if (l <= lx && r >= rx)

            return tree[ni];

        int m = (lx + rx) >> 1;

        return query(l, r, lx, m, ni \* 2 + 1) + query(l, r, m, rx, ni \* 2 + 2);

    }

public:

    SegTree(int n) {

        tree\_size = 1;

        while (tree\_size < n)

            tree\_size <<= 1;

        tree = vector<int>(tree\_size \* 2);

    }

    void update(int idx) {

        update(0, tree\_size, 0, idx);

    }

    int query(int l, int r) {

        return query(l, r + 1, 0, tree\_size, 0);

    }

};

class Solution {

public:

    int reversePairs(vector<int>& nums) {

        int n = nums.size();

        set<long long> values;

        for(const auto& num : nums) {

            values.insert(num);

            values.insert(2LL \* num);

        }

        int last\_index = 0;

        unordered\_map<long long, int> values\_indices;

        for(const auto& val : values)

            values\_indices[val] = last\_index++;

        SegTree seg\_tree(last\_index);

        int ans = 0;

        for(int i = 0; i < n; ++i) {

            ans += seg\_tree.query(values\_indices[2LL \* nums[i]] + 1, last\_index);

            seg\_tree.update(values\_indices[nums[i]]);

        }

        return ans;

    }

};

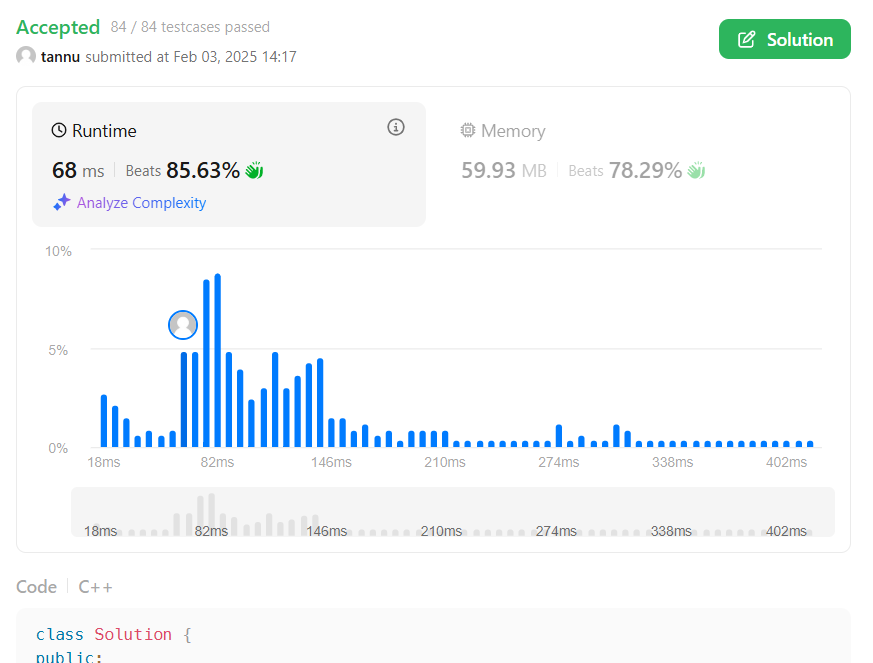
[**2407. Longest Increasing Subsequence II**](https://leetcode.com/problems/longest-increasing-subsequence-ii/)

class Solution {

public:

    vector<int>tree;

    void update(int node,int st,int end,int i,int val){

        if(st==end){

            tree[node]=max(tree[node],val);

            return;

        }

        int mid=(st+end)/2;

        if(i<=mid){

            update(node\*2,st,mid,i,val);

        }else{

            update(node\*2+1,mid+1,end,i,val);

        }

        tree[node]=max(tree[node\*2],tree[node\*2+1]);

    }

    int query(int node,int st,int end,int x,int y){

        if(x>end || y<st) return -1e9;

        if(st>=x && end<=y){

            return tree[node];

        }

        int mid=(st+end)/2;

        int left=query(2\*node,st,mid,x,y);

        int right=query(2\*node+1,mid+1,end,x,y);

        return max(left,right);

    }

    int lengthOfLIS(vector<int>& nums, int k) {

        int n=nums.size();

        if(n==1) return 1;

        int m=\*max\_element(nums.begin(),nums.end());

        tree.clear();

        tree.resize(4\*m+10);

        for(int i=n-1;i>=0;i--){

            int l=nums[i]+1,r=min(nums[i]+k,m);

            int x=query(1,0,m,l,r);

            if(x==-1e9) x=0;

            update(1,0,m,nums[i],x+1);

        }

        return tree[1];

    }

};

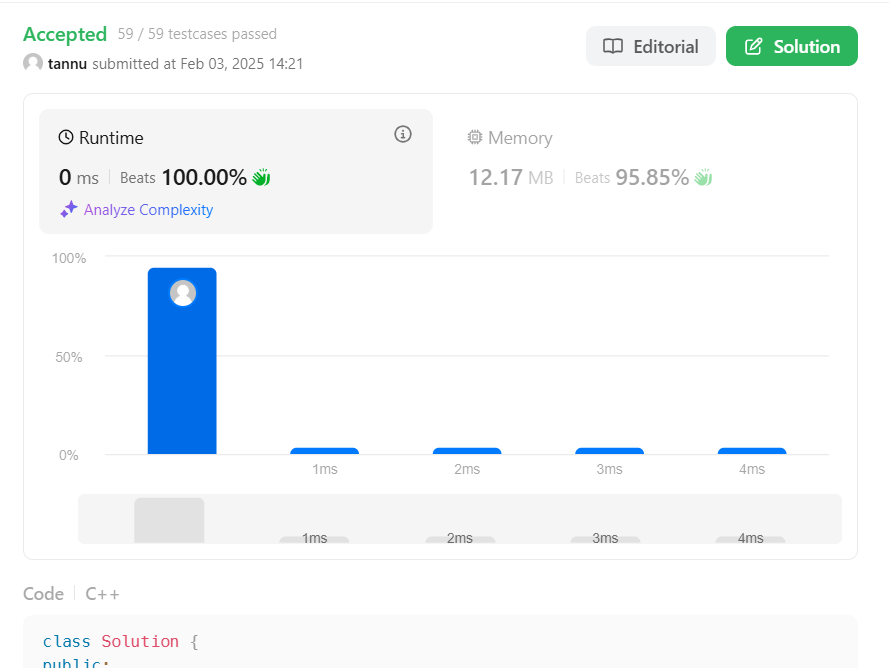
[**88. Merge Sorted Array**](https://leetcode.com/problems/merge-sorted-array/)

class Solution {

public:

    void merge(vector<int>& nums1, int m, vector<int>& nums2, int n) {

        int midx = m - 1;

****        int nidx = n - 1;

        int right = m + n - 1;

        while (nidx >= 0) {

            if (midx >= 0 && nums1[midx] > nums2[nidx]) {

                nums1[right] = nums1[midx];

                midx--;

            } else {

                nums1[right] = nums2[nidx];

                nidx--; }right--; }}};

[**278. First Bad Version**](https://leetcode.com/problems/first-bad-version/)

class Solution {

public:

    int firstBadVersion(int n) {

        int first = 1;

        int last = n;

        while (first < last) {

            int mid = first + (last - first) / 2;

            if (isBadVersion(mid)) {

                last = mid;

            } else {

                first = mid + 1;

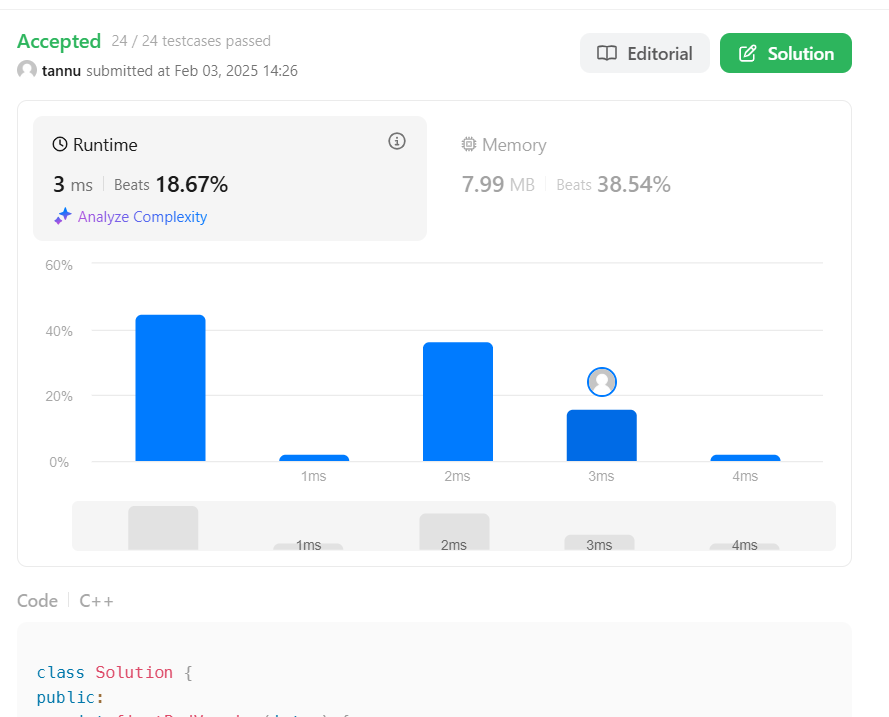
            }

        }

        return first;

    }

};



[**75. Sort Colors**](https://leetcode.com/problems/sort-colors/)

class Solution {

public:

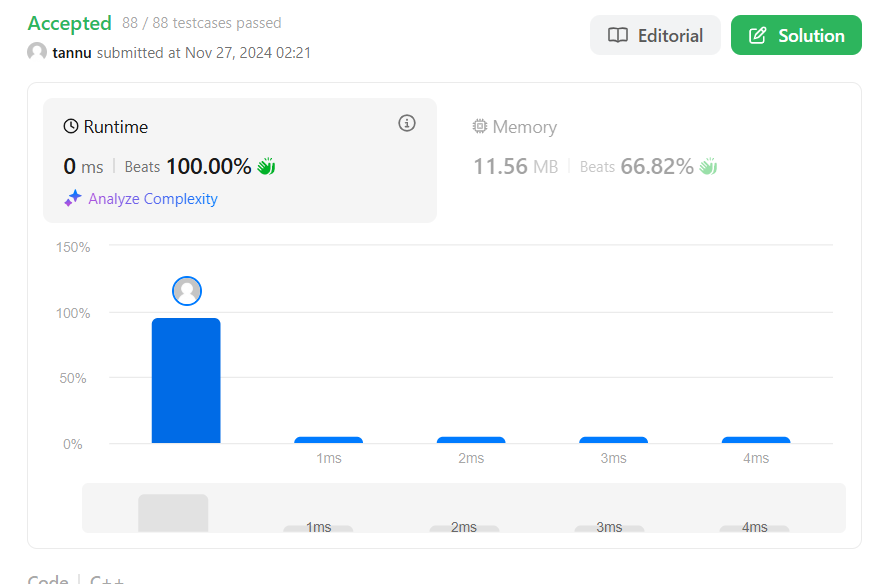
void sortColors(vector<int>& nums) {

int count0=0;

int count1=0;

int count2=0;

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

 if(nums[i]==0){

count0++;

}

else if(nums[i]==1){

count1++;

}

else{

count2++;

}

int index=0;

for(int i=0;i<count0;i++){

nums[index++]=0;

}

for(int i=0;i<count1;i++){

nums[index++]=1;

}

for(int i=0;i<count2;i++){

nums[index++]=2;

}}

}};

[**347. Top K Frequent Elements**](https://leetcode.com/problems/top-k-frequent-elements/)

class Solution {

public:

    vector<int> topKFrequent(vector<int>& nums, int k) {

        unordered\_map<int, int> counter;

        for (int n : nums) {

            counter[n]++;

        }

        auto comp = [](pair<int, int>& a, pair<int, int>& b) {

            return a.second < b.second;

        };

        priority\_queue<pair<int, int>, vector<pair<int, int>>, decltype(comp)> heap(comp);

        for (auto& entry : counter) {

            heap.push({entry.first, entry.second});

        }

        vector<int> res;

        while (k-- > 0) {

            res.push\_back(heap.top().first);

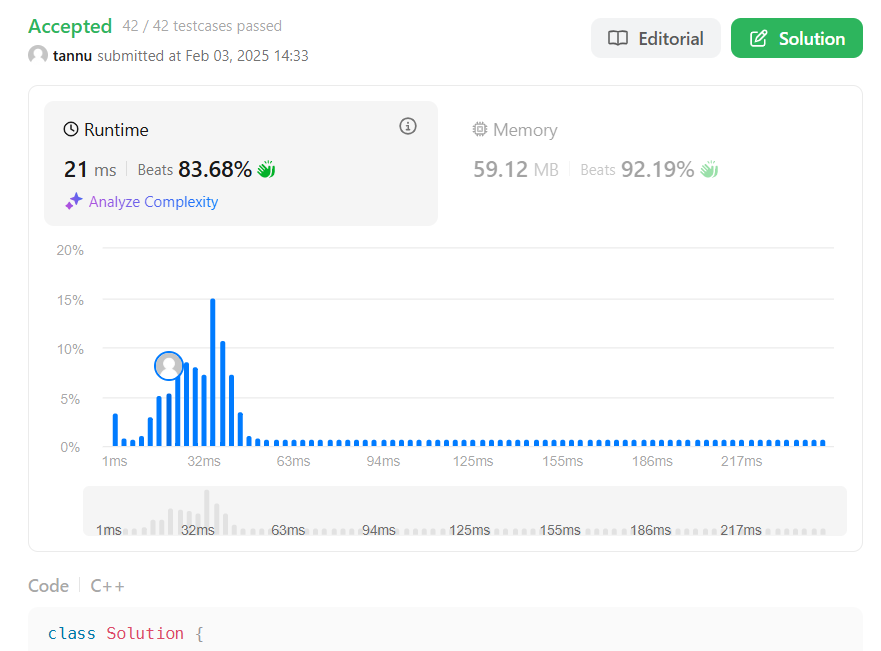
            heap.pop();

        }

        return res;

    }

};

[**215. Kth Largest Element in an Array**](https://leetcode.com/problems/kth-largest-element-in-an-array/)

class Solution {

public:

    int findKthLargest(vector<int>& nums, int k) {

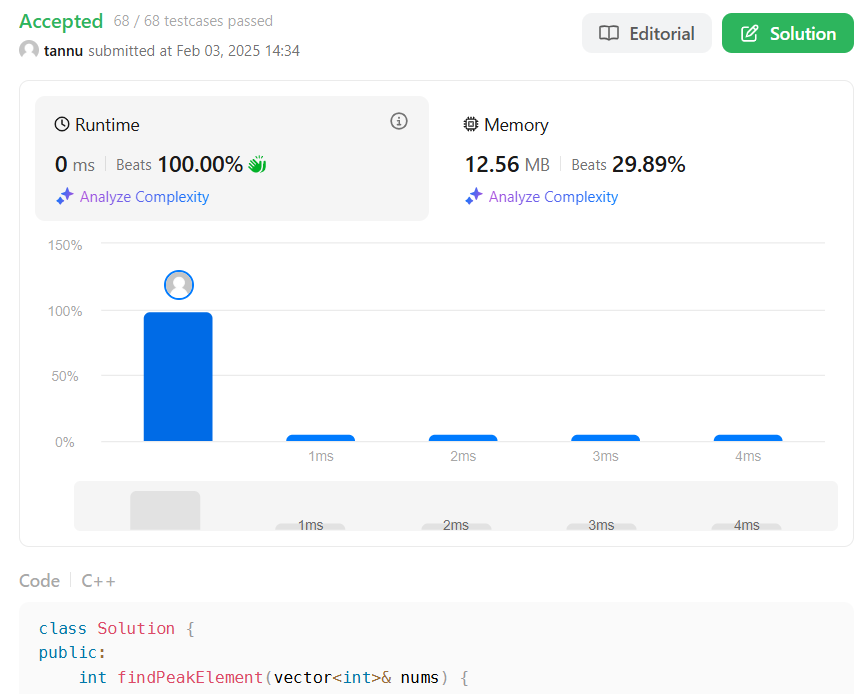
        sort(nums.begin(), nums.end());

        return nums[nums.size() - k];

    }

};

[**162. Find Peak Element**](https://leetcode.com/problems/find-peak-element/)

class Solution {

public:

    int findPeakElement(vector<int>& nums) {

        int left = 0;

        int right = nums.size() - 1;

        while (left < right) {

            int mid = left + (right - left) / 2;

            if (nums[mid] > nums[mid + 1]) {

                right = mid;

            }

else {

                left = mid + 1;

            }

        }

        return left;

  }

};

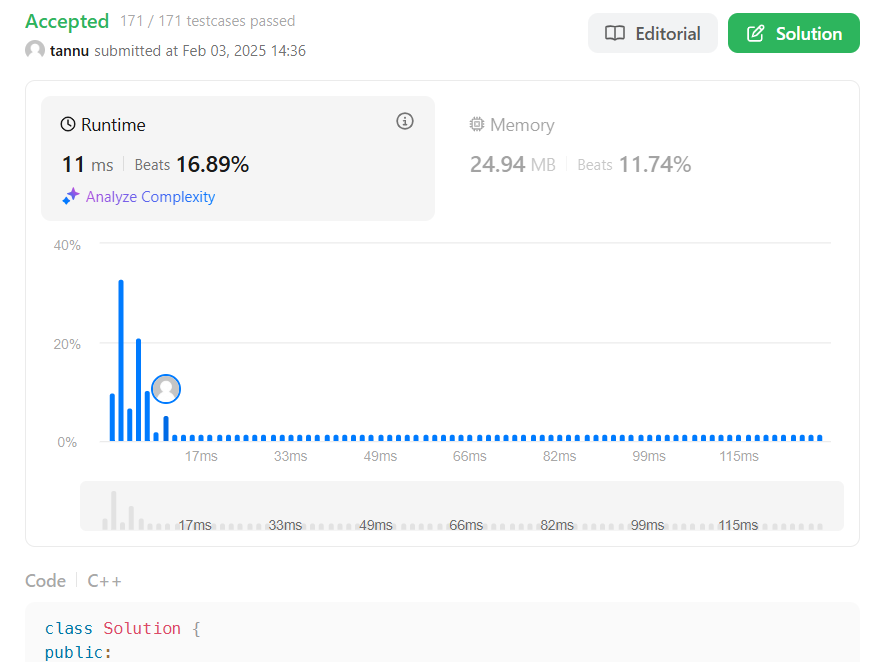
[**56. Merge Intervals**](https://leetcode.com/problems/merge-intervals/)

class Solution {

public:

vector<vector<int>> merge(vector<vector<int>>& intervals) {

sort(intervals.begin(), intervals.end(), [](const vector<int>& a, const vector<int>& b) {

 return a[0] < b[0];

});

vector<vector<int>> merged;

vector<int> prev = intervals[0];

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

vector<int> interval = intervals[i];

if (interval[0] <= prev[1]) {

prev[1] = max(prev[1], interval[1]);

} else {

merged.push\_back(prev);

prev = interval;

}

}

merged.push\_back(prev);

return merged;

}

};

[**33. Search in Rotated Sorted Array**](https://leetcode.com/problems/search-in-rotated-sorted-array/)

class Solution {

public:

    int search(vector<int>& nums, int target) {

        int left = 0;

        int right = nums.size() - 1;

        while (left <= right) {

            int mid = (left + right) / 2;

            if (nums[mid] == target) {

                return mid;

            } else if (nums[mid] >= nums[left]) {

                if (nums[left] <= target && target <= nums[mid]) {

                    right = mid - 1;

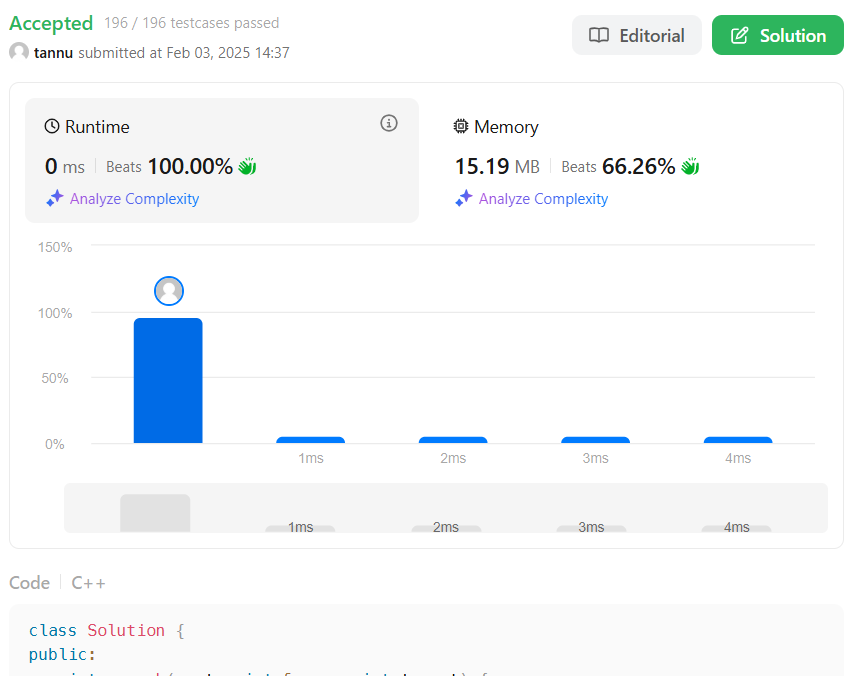
                } else {

                    left = mid + 1;

                }

            } else {

                if (nums[mid] <= target && target <= nums[right]) {

                    left = mid + 1;

                } else {

                    right = mid - 1;

                }

            }

        }

        return -1;

    }

};

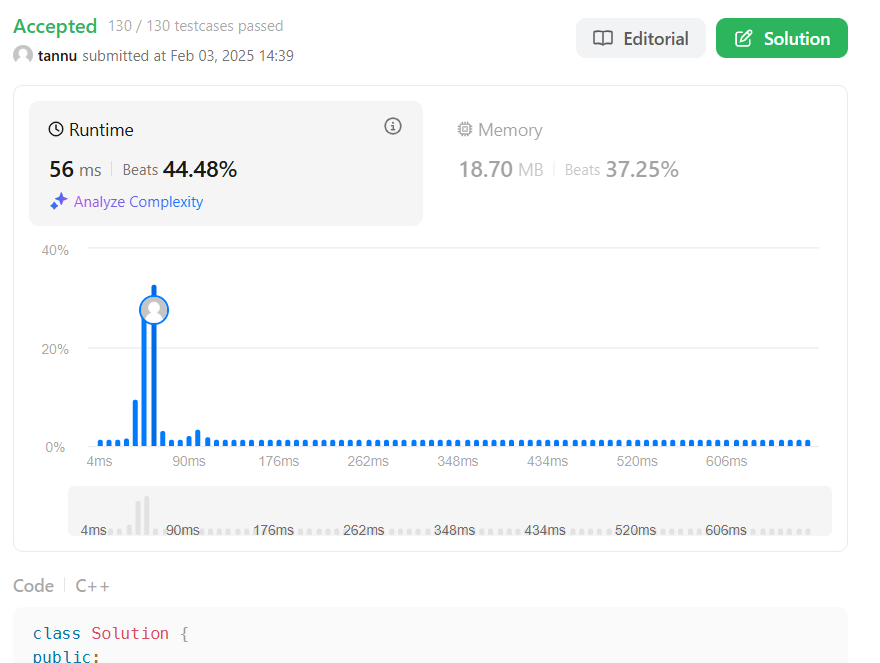
[**240. Search a 2D Matrix II**](https://leetcode.com/problems/search-a-2d-matrix-ii/)

class Solution {

public:

bool searchMatrix(vector<vector<int>>& matrix, int target) {

int n = matrix.size(), m = matrix[0].size();

 int row = 0, col = m - 1;

while (row < n && col >= 0) {

if (matrix[row][col] == target) return true;

else if (matrix[row][col] < target) row++;

else col--;

}return false; }};

[**324. Wiggle Sort II**](https://leetcode.com/problems/wiggle-sort-ii/)

class Solution {

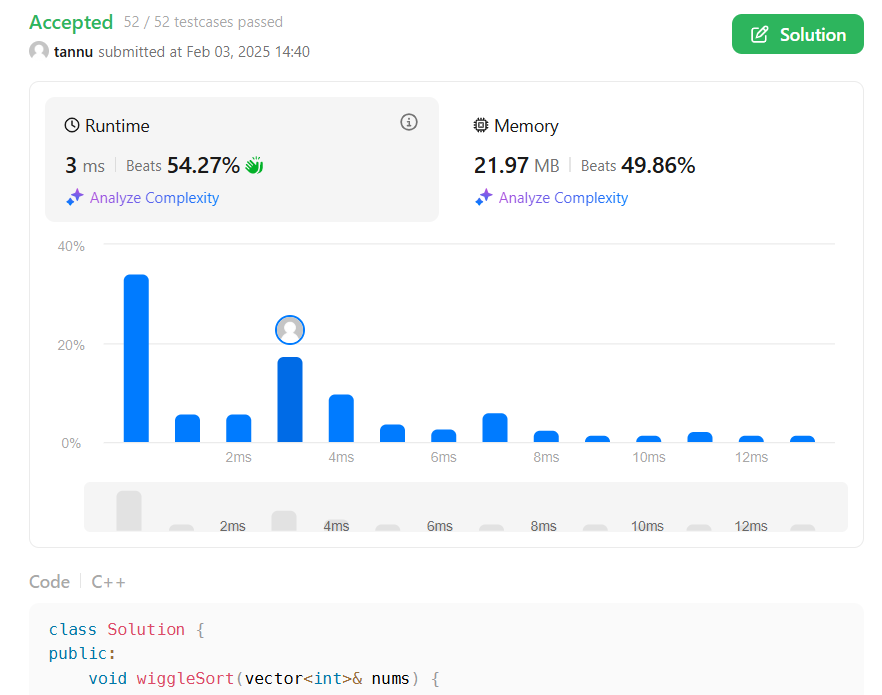
public:

void wiggleSort(vector<int>& nums) {

int n = nums.size();

vector<int> nums1(nums);

sort(nums1.begin(), nums1.end());

 int i = n-1;

int j = 0;

int k = i/2 + 1;

while(i >= 0)

{if(i % 2 == 1) {

nums[i] = nums1[k];

k++;

}else

{

nums[i] = nums1[j];

j++;

}

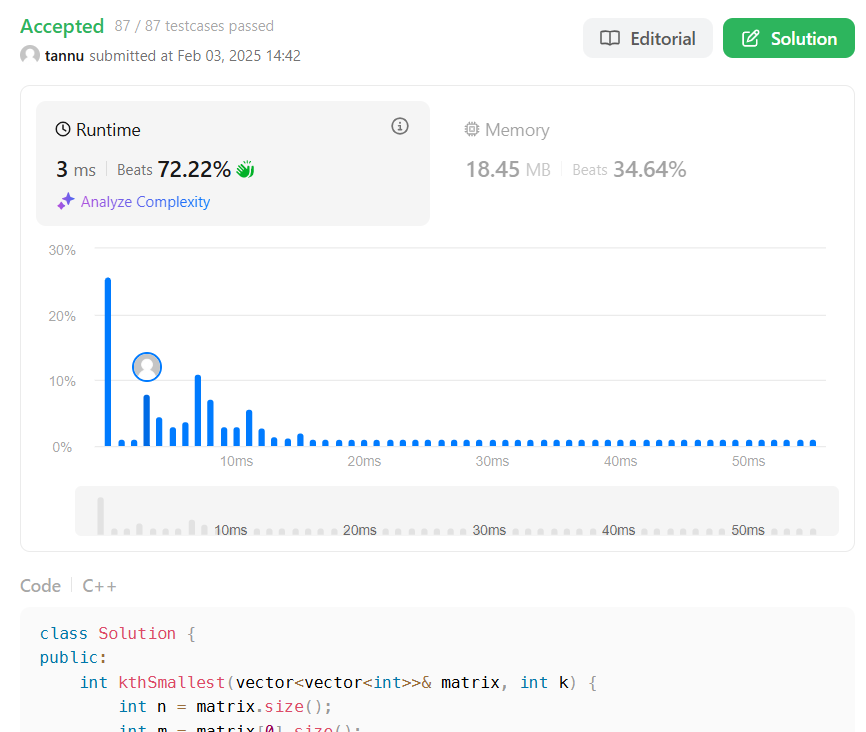
i--;}}};

3[**78. Kth Smallest Element in a Sorted Matrix**](https://leetcode.com/problems/kth-smallest-element-in-a-sorted-matrix/)

class Solution {

public:

    int kthSmallest(vector<vector<int>>& matrix, int k) {

        int n = matrix.size();

        int m = matrix[0].size();

        vector<int> p;

        for (int i = 0; i < n \* m; i++) {

            p.push\_back(matrix[i / m][i % m]);

        }

        sort(p.begin(), p.end());

        return p[k - 1];

    }

};

[**4. Median of Two Sorted Arrays**](https://leetcode.com/problems/median-of-two-sorted-arrays/)

class Solution {

public:

    double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {

        int n = nums1.size();

        int m = nums2.size();

        vector<int> merged;

        for (int i = 0; i < n; i++) {

            merged.push\_back(nums1[i]);

        }

        for (int i = 0; i < m; i++) {

            merged.push\_back(nums2[i]);

        }

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

        int total = merged.size();

        if (total % 2 == 1) {

            return static\_cast<double>(merged[total / 2]);

        } else {

            int middle1 = merged[total / 2 - 1];

            int middle2 = merged[total / 2];

            return (static\_cast<double>(middle1) + static\_cast<double>(middle2)) / 2.0;

        }

    }

};

