

Name :- Ashish Kumar

Uid :- 22BCS15516

Sec :- 607-B

AP_Assignment4

1763. 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;

}

//checking for nice

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;

}

};

```

Editorial | Solutions | (Click) | Submissions | Accepted

← All Submissions

Accepted 73 / 73 testcases passed

Ashish Kumar submitted at Mar 17, 2025 23:09

[Solution](#)

Runtime 4 ms | Beats 78.51%

[Analyze Complexity](#)

Memory 14.34 MB | Beats 31.05%

[Analyze Complexity](#)

190. Reverse Bits

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        string str = bitset<32>(n).to_string();
        reverse(str.begin(),str.end());
        uint32_t num = bitset<32>(str).to_ulong();
        return num;
    }
};
```

Solutions | 0 Ms Runtime | Submissions | Accepted

← All Submissions

Accepted 600 / 600 testcases passed

Ashish Kumar submitted at Mar 17, 2025 23:10

[Editorial](#) [Solution](#)

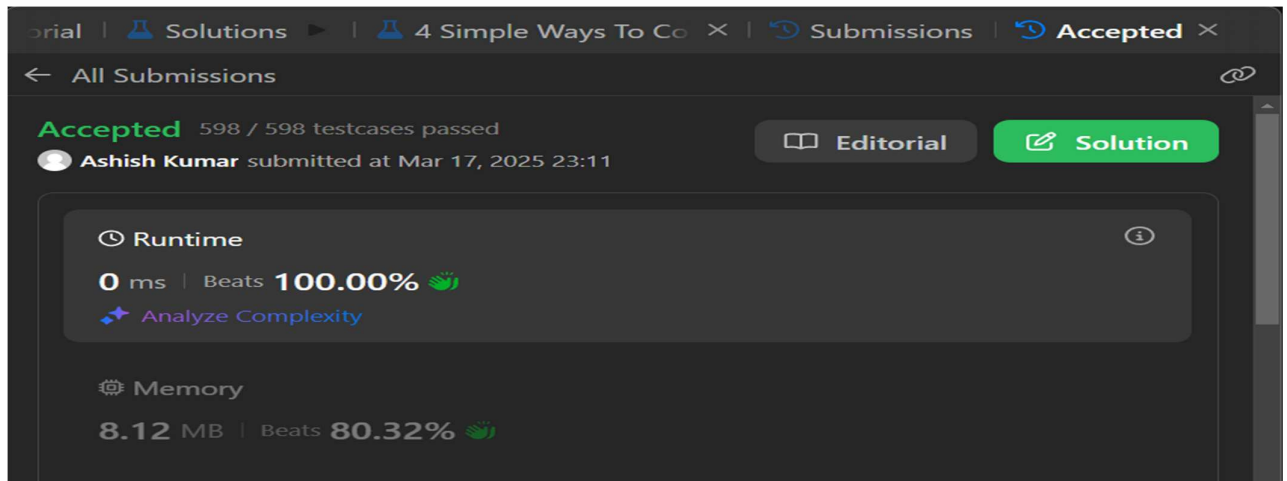
Runtime 0 ms | Beats 100.00%

[Analyze Complexity](#)

Memory 7.91 MB | Beats 7.66%

191. Number of 1 Bits

```
class Solution {  
public:  
    int hammingWeight(int n) {  
        int count = 0;  
        for(int i = 31; i >= 0; i--){  
            if(((n >> i) & 1) == 1)  
                count++;  
        }  
        return count;  
    }  
};
```



53. Maximum Subarray

```
class Solution {  
public:  
    int maxSubArray(vector<int>& nums) {  
        int max = INT_MIN;
```

```

int sum = 0;

int n = nums.size();

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

    sum = sum + nums[i];

    if(sum> max){

        max = sum;

    }

    if(sum<0){

        sum = 0;

    }

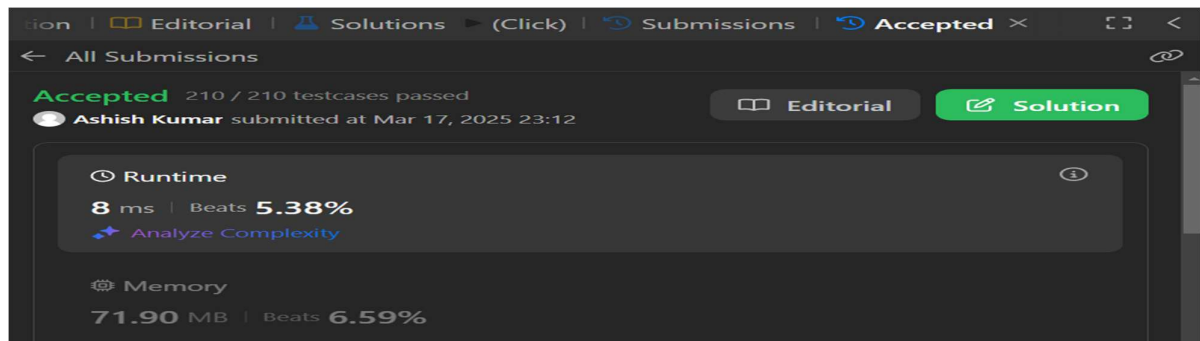
}

return max;

}

};

```



[240. Search a 2D Matrix II](#)

```

class Solution {

public:

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

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

```

```

int m = matrix.size();

int cols = n-1; //last col

int rows =0; //1st row

while(rows<m && cols>=0){

    if(target==matrix[rows][cols]) return true;

    else if(target<matrix[rows][cols]) cols--;

    else if(target>matrix[rows][cols]) rows++;

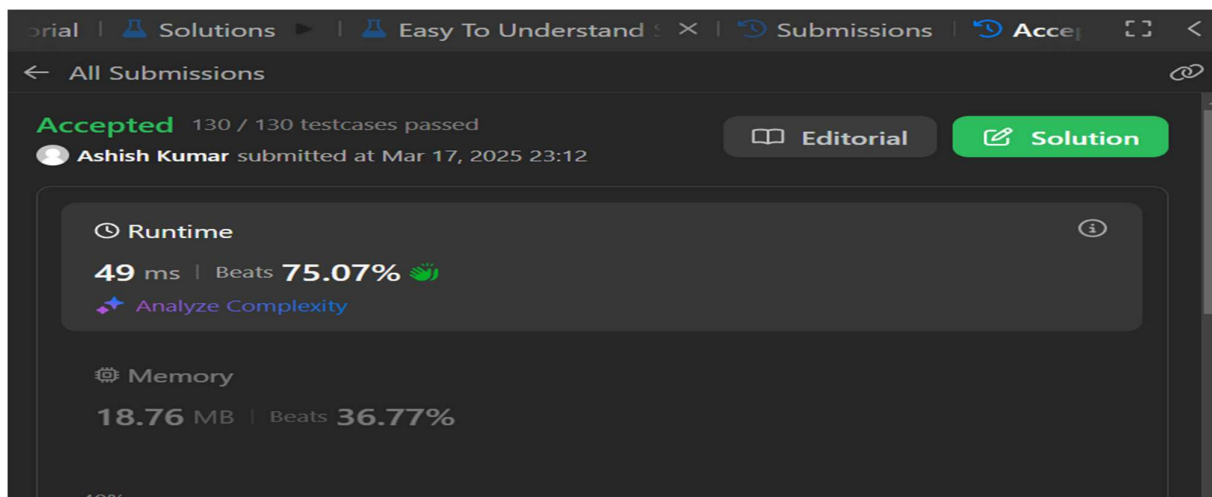
}

return false;

}

};

```



[372. Super Pow](#)

```

class Solution {

    const int base = 1337;

    int powmod(int a, int k) //a^k mod 1337 where 0 <= k <= 10

    {

        a %= base;

```

```

    int result = 1;

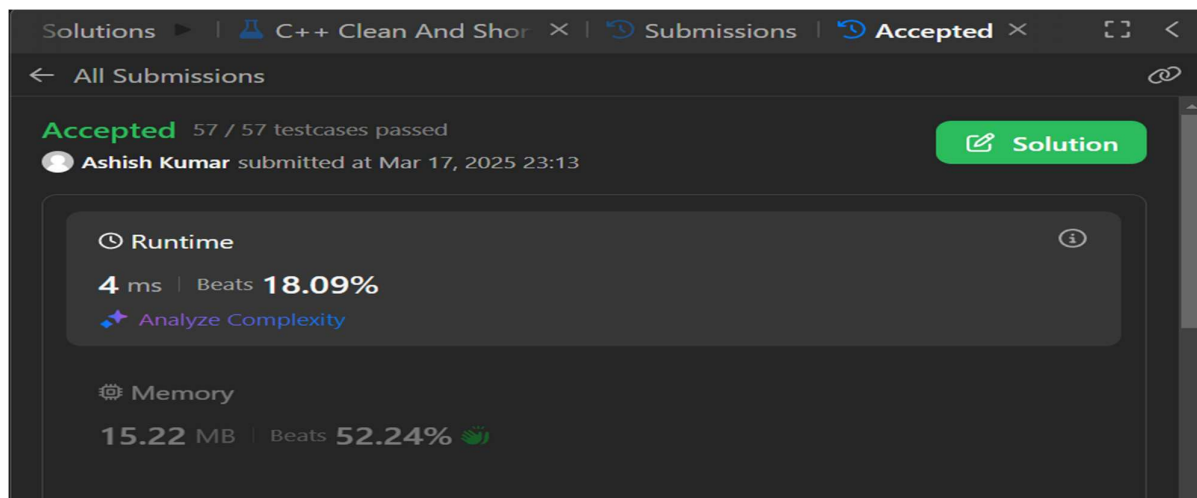
    for (int i = 0; i < k; ++i)
        result = (result * a) % base;

    return result;
}

public:

    int superPow(int a, vector<int>& b) {
        if (b.empty()) return 1;
        int last_digit = b.back();
        b.pop_back();
        return powmod(superPow(a, b), 10) * powmod(a, last_digit) % base;
    }
};

```



932. Beautiful Array

```

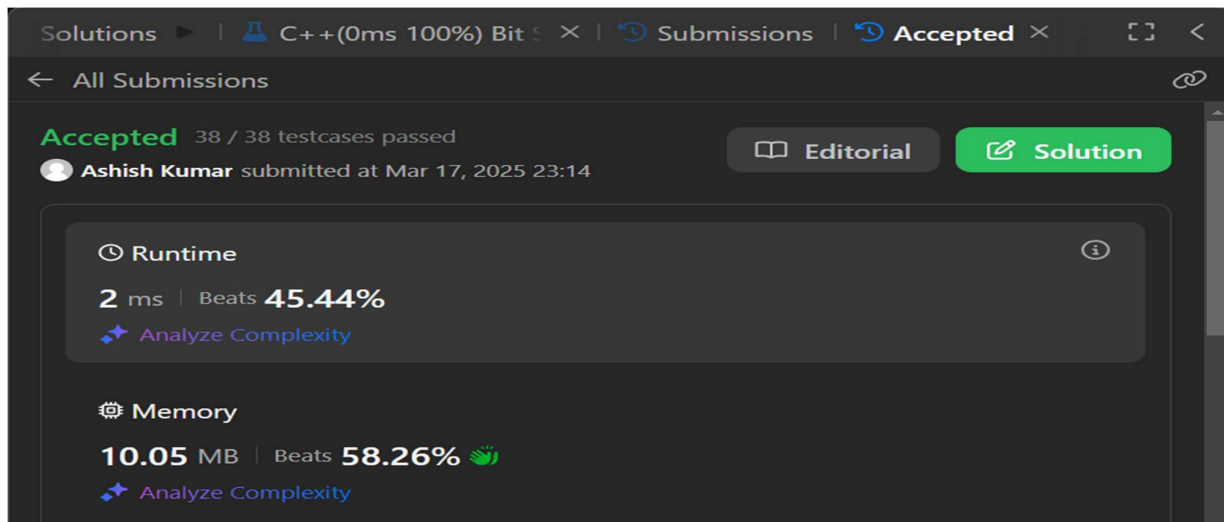
class Solution {
public:

```

```

static bool comp(const int &a, const int &b){
    int mask = 1;
    while(true)
        if((a&mask) == (b&mask)) mask = mask<<1;
        else return (a&mask) > (b&mask);
    }
vector<int> beautifulArray(int n) {
    vector<int> answer;
    while(n) answer.push_back(n--);
    sort(answer.begin(), answer.end(), comp);
    return answer;
}
};

```



218. 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});

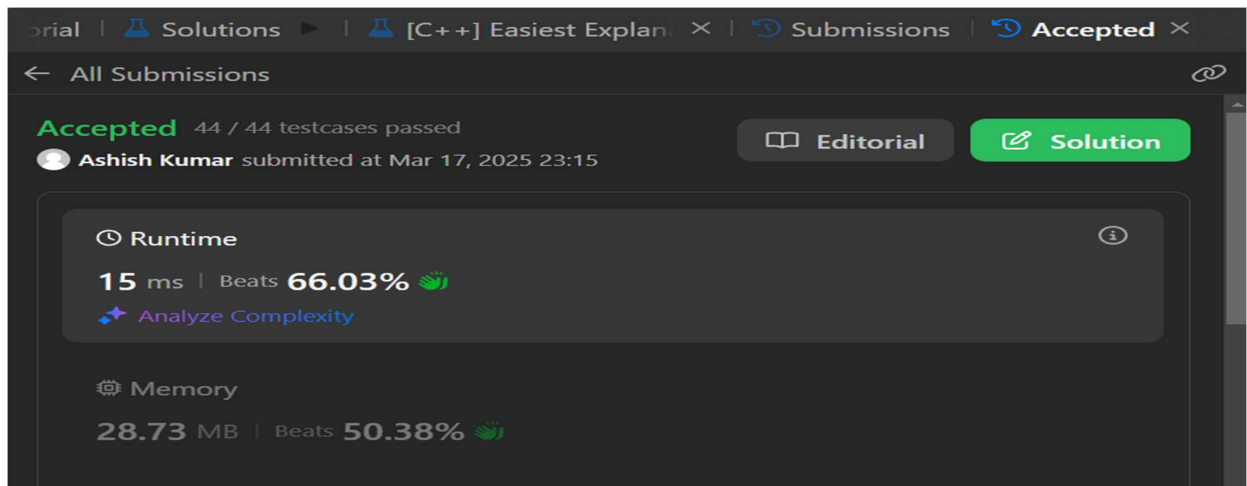
    }

    return skyline;

}

};

```



493. Reverse Pairs

```

class Solution {
private:
    void merge(vector<int>& nums, int low, int mid, int high, int&
reversePairsCount){
        int j = mid+1;
        for(int i=low; i<=mid; i++){
            while(j<=high && nums[i] > 2*(long long)nums[j]){
                j++;
            }
        }
    }
}

```

```

        reversePairsCount += j-(mid+1);
    }

    int size = high-low+1;
    vector<int> temp(size, 0);
    int left = low, right = mid+1, k=0;
    while(left<=mid && right<=high){
        if(nums[left] < nums[right]){
            temp[k++] = nums[left++];
        }
        else{
            temp[k++] = nums[right++];
        }
    }

    while(left<=mid){
        temp[k++] = nums[left++];
    }

    while(right<=high){
        temp[k++] = nums[right++];
    }

    int m=0;
    for(int i=low; i<=high; i++){
        nums[i] = temp[m++];
    }
}

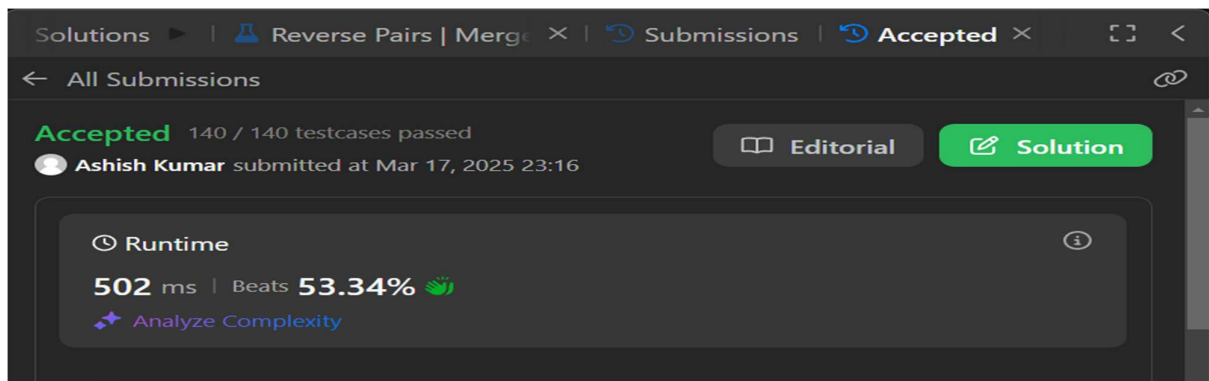
```

```

void mergeSort(vector<int>& nums, int low, int high, int&
reversePairsCount){
    if(low >= high){
        return;
    }
    int mid = (low + high) >> 1;
    mergeSort(nums, low, mid, reversePairsCount);
    mergeSort(nums, mid+1, high, reversePairsCount);
    merge(nums, low, mid, high, reversePairsCount);
}

public:
    int reversePairs(vector<int>& nums) {
        int reversePairsCount = 0;
        mergeSort(nums, 0, nums.size()-1, reversePairsCount);
        return reversePairsCount;
    }
};

```



[2407. 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];
}
};


```


Solutions | C++ Bottom-Up Seg | Submissions | **Accepted** ×

← All Submissions


Accepted 84 / 84 testcases passed


 **Ashish Kumar** submitted at Mar 17, 2025 23:17

 **Solution**


 **Runtime** ⓘ

30 ms | Beats **93.58%** 🍃

 [Analyze Complexity](#)

 **Memory**

46.26 MB | Beats **98.76%** 🍃

 [Analyze Complexity](#)