Harsh Pandey
22BCS17030
IOT-609/B

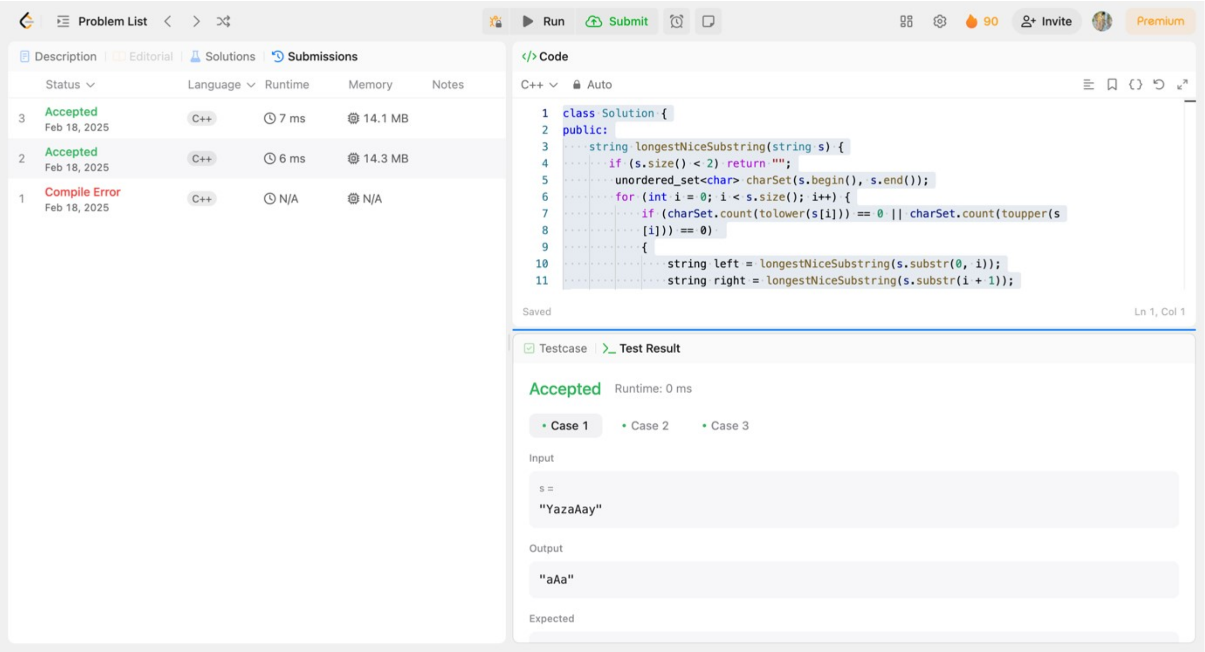
Advance Programming

Assignment 7

1. Longest Nice Substring:

CODE:

```
class Solution { public:    string
longestNiceSubstring(string s) {      if
(s.size() < 2) return "";
      unordered_set<char>
charSet(s.begin(), s.end());
      for (int i = 0; i < s.size(); i++) {
         if (charSet.count(tolower(s[i])) == 0 || charSet.count(toupper(s
         [i])) == 0)
         {
            string left = longestNiceSubstring(s.substr(0, i));
string right = longestNiceSubstring(s.substr(i + 1));
return left.size() >= right.size() ? left : right;
         }
      }
      return s;
   }
};
```



	Status	Language	Runtime	Memory	Notes
3	Accepted Feb 18, 2025	C++	7 ms	14.1 MB	
2	Accepted Feb 18, 2025	C++	6 ms	14.3 MB	
1	Compile Error Feb 18, 2025	C++	N/A	N/A	

```
class Solution {
public:
    string longestNiceSubstring(string s) {
        if (s.size() < 2) return "";
         unordered_set<char> charSet(s.begin(), s.end());
         for (int i = 0; i < s.size(); i++) {
             if (charSet.count(tolower(s[i])) == 0 || charSet.count(toupper(s
             [i])) == 0)
             {
                 string left = longestNiceSubstring(s.substr(0, i));
                 string right = longestNiceSubstring(s.substr(i + 1));
```

Accepted Runtime: 0 ms

• Case 1 • Case 2 • Case 3

Input

s =

```
"YazaAay"
```

Output

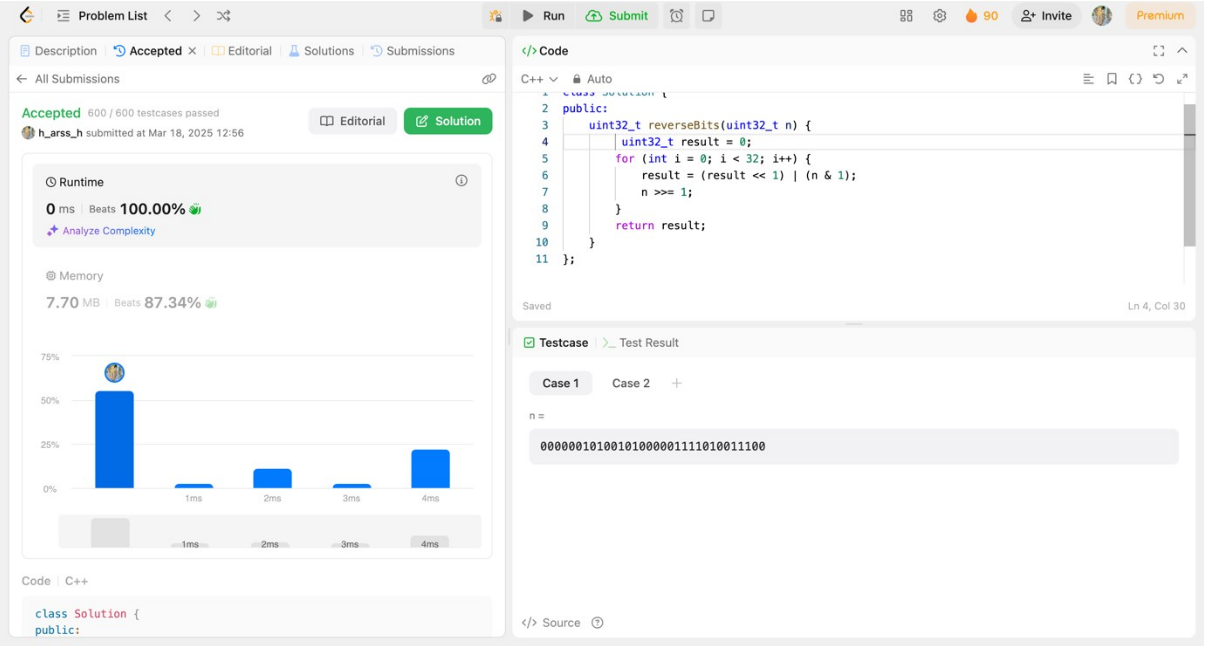
```
"aAa"
```

Expected

2. Reverse Bits:

CODE: class Solution { public:

```
uint32_t reverseBits(uint32_t n) {
       uint32_t result = 0;       for (int
i = 0; i < 32; i++) {          result =
(result << 1) | (n & 1);          n >>=
1;
      }
      return result;
   }
};
```



Accepted 600 / 600 testcases passed

h_arss_h submitted at Mar 18, 2025 12:56

Runtime

0 ms | Beats 100.00%

Memory

7.70 MB | Beats 87.34%

Code | C++

```
class Solution {
public:
```

```
class Solution {
public:
    uint32_t reverseBits(uint32_t n) {
        uint32_t result = 0;
        for (int i = 0; i < 32; i++) {
            result = (result << 1) | (n & 1);
            n >>= 1;
        }
        return result;
    }
};
```

Saved

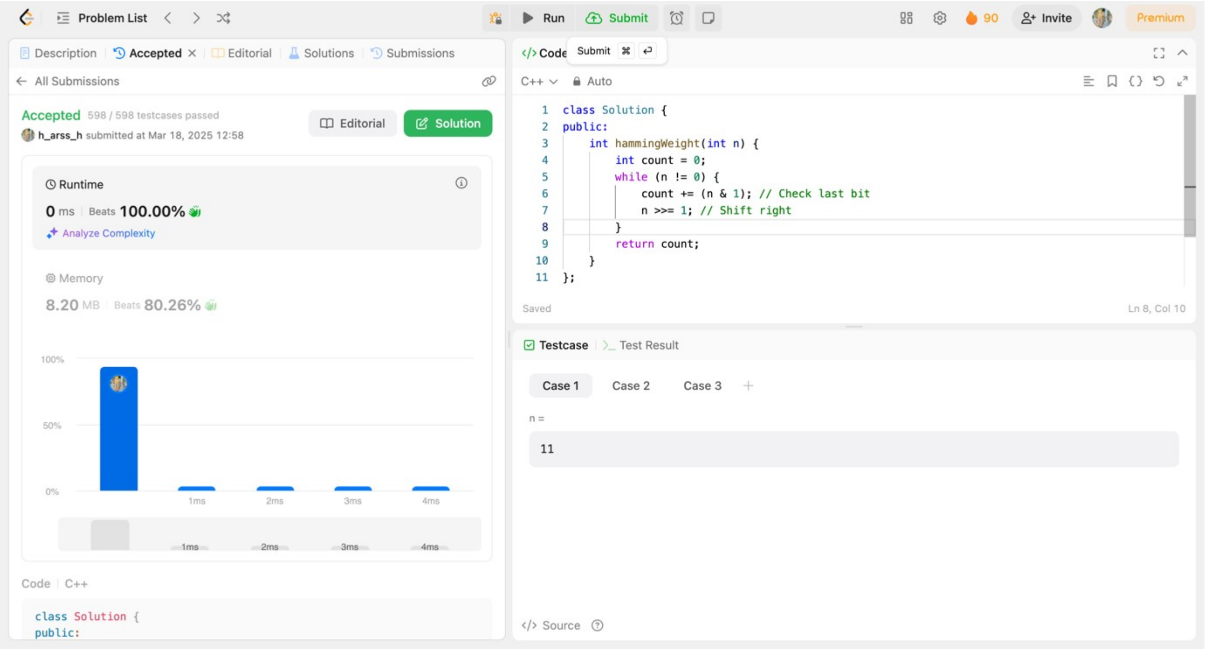
Ln 4, Col 30

n =

```
00000010100101000001111010011100
```

3. **Number of 1 Bits: CODE:** class Solution {

```
public:
  int hammingWeight(int n) {
int count = 0;       while (n !=
0) {
          count += (n & 1); // Check last bit
          n >>= 1; // Shift right
      }
      return count;
   }
};
```



Accepted 598 / 598 testcases passed

h_arss_h submitted at Mar 18, 2025 12:58

Runtime

0 ms | Beats 100.00%

Memory

8.20 MB | Beats 80.26%

Code | C++

```
class Solution {
public:
```

```
class Solution {
public:
    int hammingWeight(int n) {
        int count = 0;
        while (n != 0) {
            count += (n & 1); // Check last bit
            n >>= 1; // Shift right
        }
        return count;
    }
};
```

Saved

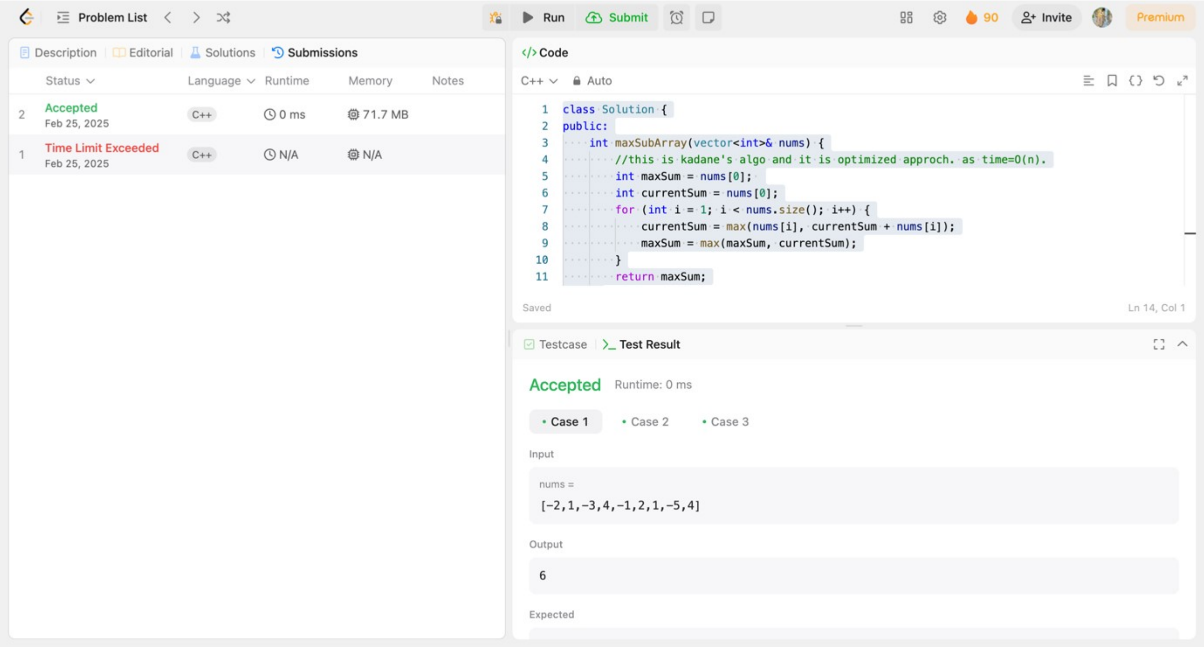
Ln 8, Col 10

n =

11

4. **Maximum Subarray:**

```
CODE: class Solution { public:     int
maxSubArray(vector<int>& nums) {
        int maxSum = nums[0];         int
currentSum = nums[0];        for (int i
= 1; i < nums.size(); i++) {
            currentSum = max(nums[i], currentSum + nums[i]);
maxSum = max(maxSum, currentSum);
        }
        return maxSum;
    }
};
```



	Status	Language	Runtime	Memory	Notes
2	Accepted Feb 25, 2025	C++	0 ms	71.7 MB	
1	Time Limit Exceeded Feb 25, 2025	C++	N/A	N/A	

```
class Solution {
public:
    int maxSubArray(vector<int>& nums) {
        //this is kadane's algo and it is optimized approch. as time=O(n).
        int maxSum = nums[0];
        int currentSum = nums[0];
        for (int i = 1; i < nums.size(); i++) {
            currentSum = max(nums[i], currentSum + nums[i]);
            maxSum = max(maxSum, currentSum);
        }
        return maxSum;
```

Accepted Runtime: 0 ms

Input

```
nums =
[-2,1,-3,4,-1,2,1,-5,4]
```

Output

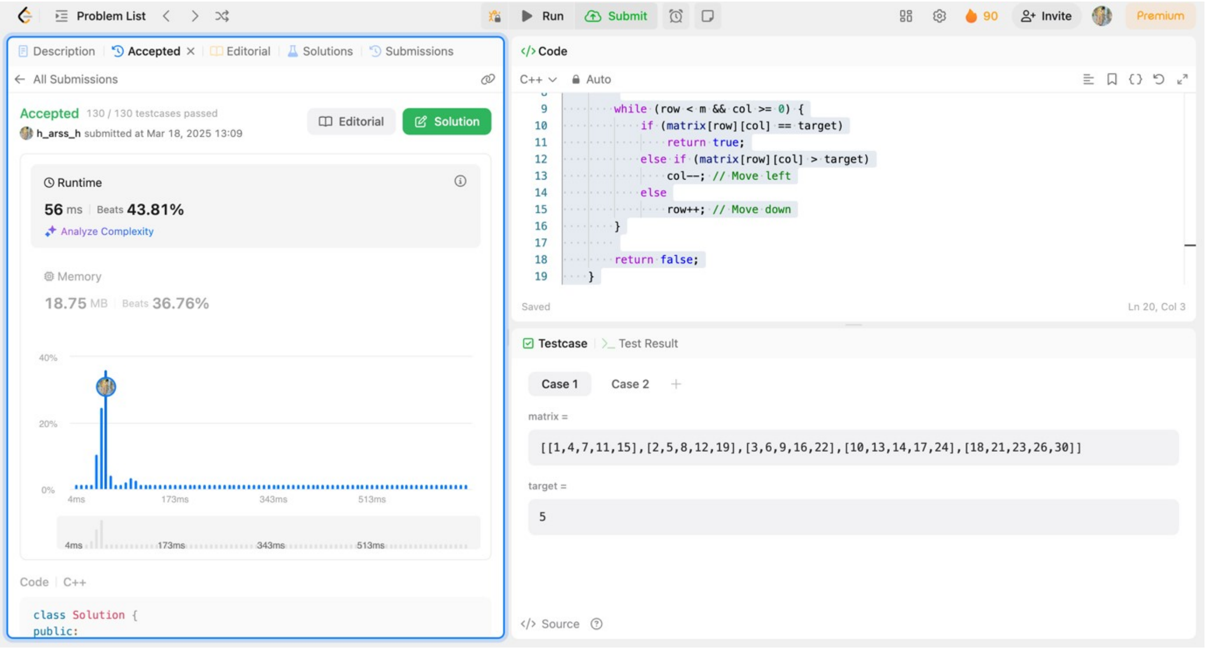
```
6
```

Expected

5. Search a 2D Matrix II:

CODE:

```
class Solution { public:     bool
searchMatrix(vector<vector<int>>& matrix, int target) {        int
m = matrix.size();
    int n = matrix[0].size();

    int row = 0, col = n - 1; // Start from top-right

    while (row < m && col >= 0) {
if (matrix[row][col] == target)
        return true;
      else if (matrix[row][col] > target)
        col--; // Move left
      else
        row++; // Move down
    }

    return false;
  }
};
```



Accepted 130 / 130 testcases passed

h_arss_h submitted at Mar 18, 2025 13:09

Runtime

56 ms | Beats 43.81%

Memory

18.75 MB | Beats 36.76%

Code | C++

```
class Solution {
public:
```

```
        while (row < m && col >= 0) {
            if (matrix[row][col] == target)
                return true;
            else if (matrix[row][col] > target)
                col--; // Move left
            else
                row++; // Move down
        }

        return false;
    }
```

Saved

Ln 20, Col 3

matrix =

```
[[1,4,7,11,15],[2,5,8,12,19],[3,6,9,16,22],[10,13,14,17,24],[18,21,23,26,30]]
```

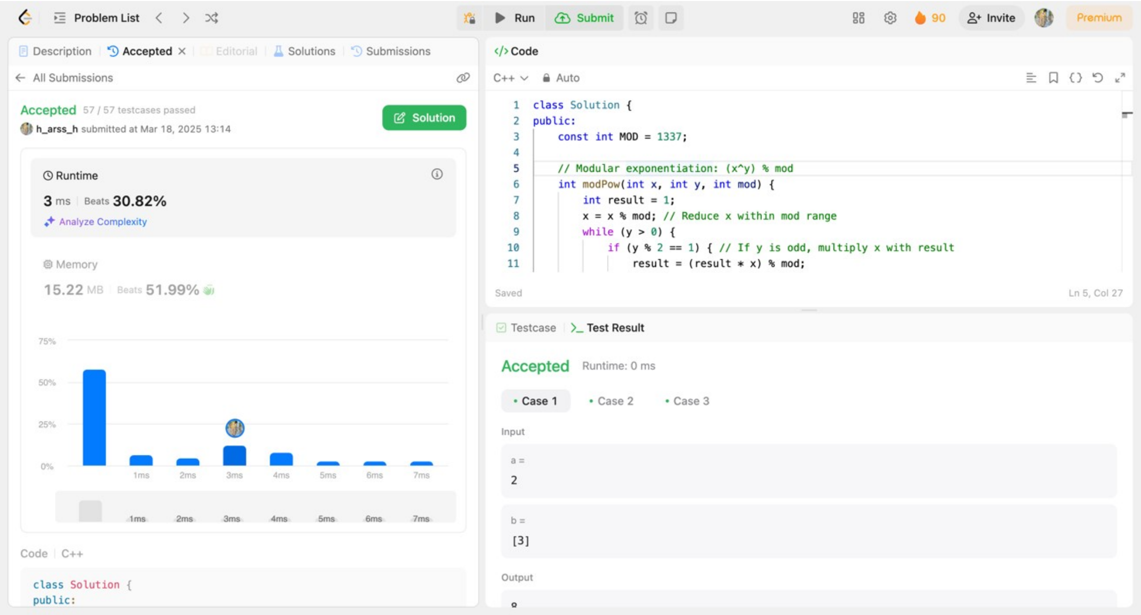
target =

```
5
```

6. Super Pow:

CODE: class Solution { public:

```
const int MOD = 1337;     int
modPow(int x, int y, int mod) {
int result = 1;        x = x % mod;
while (y > 0) {           if (y % 2 == 1) {
                result = (result * x) % mod;
            }
x = (x * x) % mod; // Square x
y /= 2; // Reduce y
    }
    return result;
  }
  int superPow(int a, vector<int>& b) {
if (b.empty()) return 1; // Base case
int lastDigit = b.back();
    b.pop_back(); // Remove last digit       int part1
= modPow(superPow(a, b), 10, MOD);       int part2
= modPow(a, lastDigit, MOD);       return (part1 *
part2) % MOD;
  }
};
```



Accepted 57 / 57 testcases passed

h_arss_h submitted at Mar 18, 2025 13:14

3 ms | Beats 30.82%

15.22 MB | Beats 51.99%

Code | C++

```
class Solution {
public:
```

```
class Solution {
public:
    const int MOD = 1337;

    // Modular exponentiation: (x^y) % mod
    int modPow(int x, int y, int mod) {
        int result = 1;
        x = x % mod; // Reduce x within mod range
        while (y > 0) {
            if (y % 2 == 1) { // If y is odd, multiply x with result
                result = (result * x) % mod;
```

Saved

Ln 5, Col 27

Accepted Runtime: 0 ms

Input

a =
2

b =
[3]

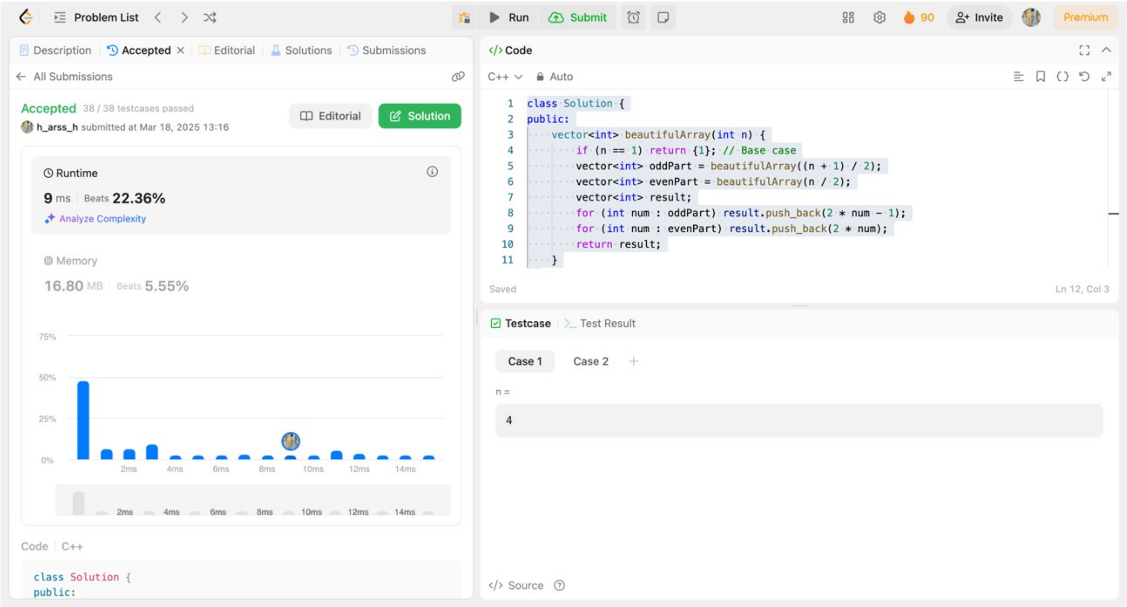
Output

8

7. Beautiful Array:

CODE:

```
class Solution { public:
  vector<int> beautifulArray(int n) {       if (n == 1) return
{1}; // Base case       vector<int> oddPart =
beautifulArray((n + 1) / 2);       vector<int> evenPart =
beautifulArray(n / 2);       vector<int> result;       for (int
num : oddPart) result.push_back(2 * num - 1);
    for (int num : evenPart) result.push_back(2 * num);
return result;
  }
};
```



Accepted 38 / 38 testcases passed

h_arss_h submitted at Mar 18, 2025 13:16

Runtime: 9 ms | Beats 22.36%

Memory: 16.80 MB | Beats 5.55%

Code | C++

```
class Solution {
public:
```

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        if (n == 1) return {1}; // Base case
        vector<int> oddPart = beautifulArray((n + 1) / 2);
        vector<int> evenPart = beautifulArray(n / 2);
        vector<int> result;
        for (int num : oddPart) result.push_back(2 * num - 1);
        for (int num : evenPart) result.push_back(2 * num);
        return result;
    }
```

Saved

Ln 12, Col 3

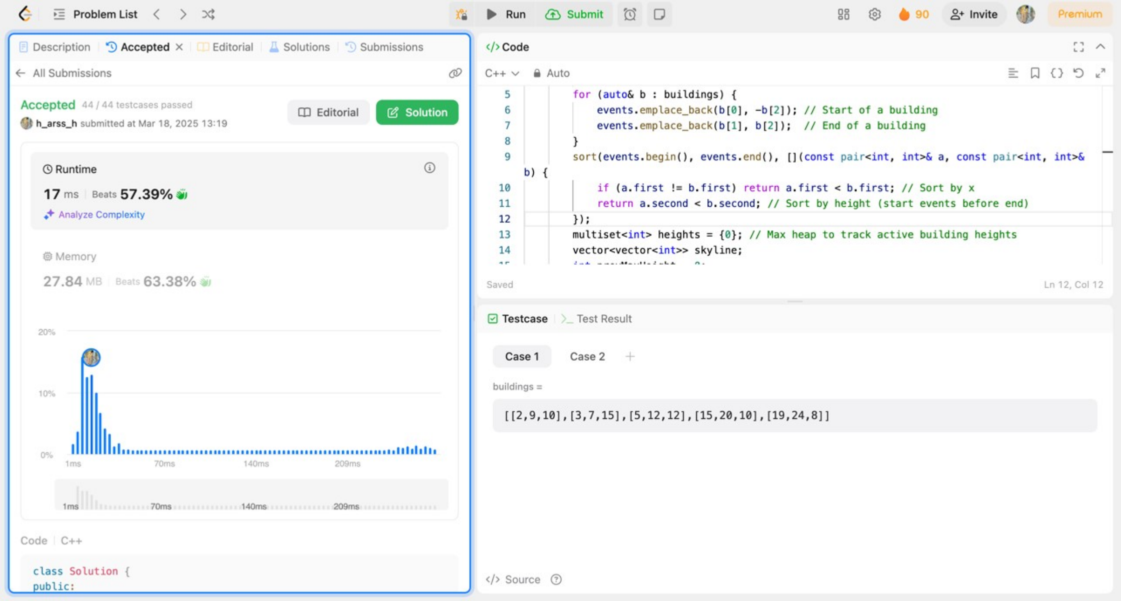
n =

4

8. The Skyline Problem:

CODE:

```
class Solution { public:
  vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
     vector<pair<int, int>> events; // Store (x, height)
    for (auto& b : buildings) {
events.emplace_back(b[0], -b[2]); // Start of a building
events.emplace_back(b[1], b[2]);  // End of a building
    }
    sort(events.begin(), events.end(), [](const pair<int, int>& a, const pair<int, int>&
b) {
      if (a.first != b.first) return a.first < b.first; // Sort by x
      return a.second < b.second; // Sort by height (start events before end)
    });
    multiset<int> heights = {0}; // Max heap to track active building heights
vector<vector<int>> skyline;     int prevMaxHeight = 0;     for (auto& e :
events) {       int x = e.first, h = e.second;
      if (h < 0) heights.insert(-h); // Building starts: add height        else
heights.erase(heights.find(h)); // Building ends: remove height        int
currMaxHeight = *heights.rbegin(); // Get max height
      if (currMaxHeight != prevMaxHeight) { // If height changes, record key point
skyline.push_back({x, currMaxHeight});          prevMaxHeight =
currMaxHeight;
      }
    }
    return skyline;
    }
};
```



Accepted 44 / 44 testcases passed

h_arss_h submitted at Mar 18, 2025 13:19

Runtime

17 ms | Beats 57.39%

Memory

27.84 MB | Beats 63.38%

Code | C++

```
class Solution {
public:
```

Code

C++ Auto

```
        for (auto& b : buildings) {
            events.emplace_back(b[0], -b[2]); // Start of a building
            events.emplace_back(b[1], b[2]);  // End of a building
        }
        sort(events.begin(), events.end(), [](const pair<int, int>& a, const pair<int, int>&
b) {
            if (a.first != b.first) return a.first < b.first; // Sort by x
            return a.second < b.second; // Sort by height (start events before end)
        });
        multiset<int> heights = {0}; // Max heap to track active building heights
        vector<vector<int>> skyline;
```

Saved

Ln 12, Col 12

buildings =

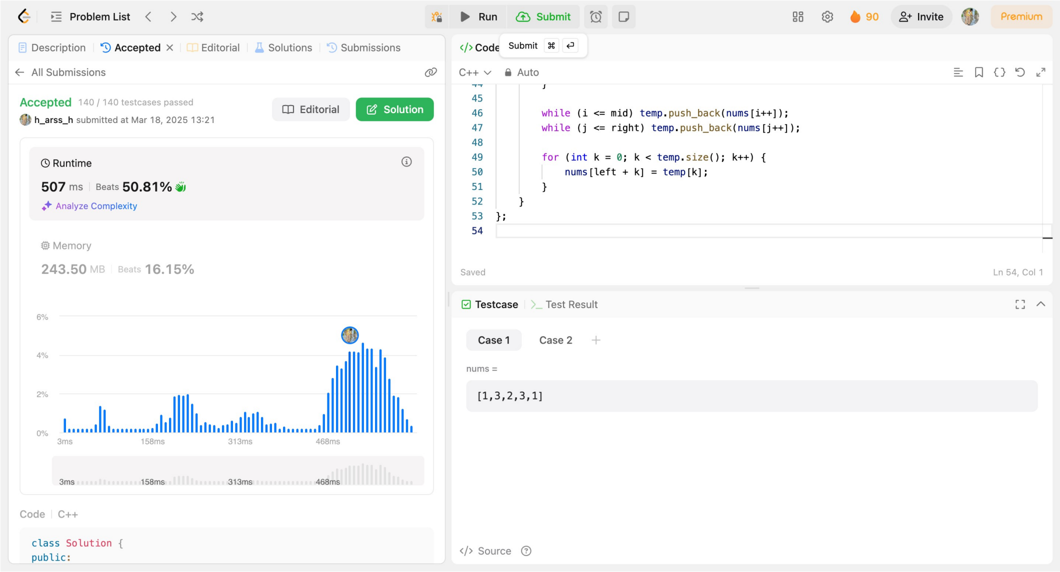
```
[[2,9,10],[3,7,15],[5,12,12],[15,20,10],[19,24,8]]
```

9. Reverse Pairs:

CODE:

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

private:     int mergeSort(vector<int>& nums, int left,
int right) {        if (left >= right) return 0;

        int mid = left + (right - left) / 2;
        int count = mergeSort(nums, left, mid) + mergeSort(nums, mid + 1, right);

        // Count valid reverse pairs
        count += countPairs(nums, left, mid, right);

        // Merge the sorted halves
        merge(nums, left, mid, right);

        return count;
    }

    int countPairs(vector<int>& nums, int left, int mid, int right) {
int count = 0;        int j = mid + 1;
```

```
        for (int i = left; i <= mid; i++) {
            while (j <= right && nums[i] > 2LL * nums[j]) {
j++;
            }
            count += (j - (mid + 1));
        }

        return count;
    }

    void merge(vector<int>& nums, int left, int mid, int right) {
vector<int> temp;        int i = left, j = mid + 1;

        while (i <= mid && j <= right) {
            if (nums[i] <= nums[j]) temp.push_back(nums[i++]);
else temp.push_back(nums[j++]);
        }

        while (i <= mid) temp.push_back(nums[i++]);
        while (j <= right) temp.push_back(nums[j++]);

        for (int k = 0; k < temp.size(); k++) {
            nums[left + k] = temp[k];
        }
    }
};
```



Accepted 140 / 140 testcases passed

h_arss_h submitted at Mar 18, 2025 13:21

Runtime

507 ms | Beats 50.81%

Analyze Complexity

Memory

243.50 MB | Beats 16.15%

Code | C++

```
class Solution {
public:
```

```
        }

        while (i <= mid) temp.push_back(nums[i++]);
        while (j <= right) temp.push_back(nums[j++]);

        for (int k = 0; k < temp.size(); k++) {
            nums[left + k] = temp[k];
        }
    }
};

```

Saved

Ln 54, Col 1

nums =

```
[1,3,2,3,1]
```

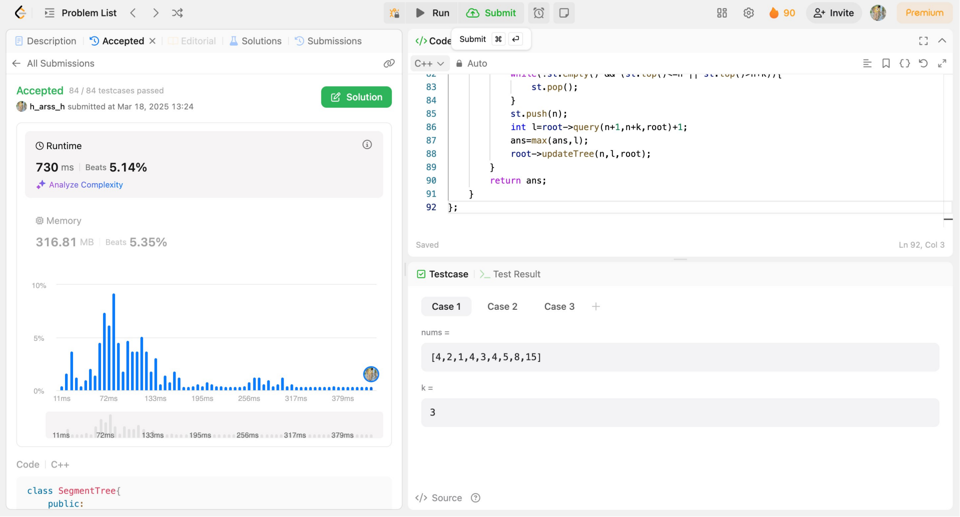
10. **Longest Increasing Subsequence II:**

CODE:

```
class SegmentTree{     public:
  int leftIndex;
int rightIndex;
  SegmentTree* left;
SegmentTree* right;     int
maxNum;
    SegmentTree(int leftI,int rightI,int val){
      leftIndex=leftI;
rightIndex=rightI;
maxNum=val;
left=NULL;           right=NULL;
    }

    void updateTree(int index,int val,SegmentTree* root){
      if(root->leftIndex==root->rightIndex){              root-
>maxNum=val;
        return;
      }

      int midIndex=(root->leftIndex+root->rightIndex)/2;
if(midIndex>=index){
        updateTree(index,val,root->left);
      } else {
        updateTree(index,val,root->right);
      }
      root->maxNum=max(root->left->maxNum,root->right->maxNum);
return;
    }


    int query(int leftI,int rightI,SegmentTree* root){
if(root->rightIndex==rightI && root->leftIndex==leftI)
return root->maxNum;

      if(leftI>rightI){
return -1;
      }

      int midIndex=(root->leftIndex+root->rightIndex)/2;
int ans=0;
```

```
      if(leftI<=midIndex && midIndex<=rightI){
ans=max(ans,max(query(leftI,midIndex,root>left),query(
midIndex+1,rightI,root->right)));
      } else if(midIndex<leftI) {
        ans=max(ans,query(leftI,rightI,root->right));
      } else {
        ans=max(ans,query(leftI,rightI,root->left));
      }
      return ans;
    }
};

SegmentTree* construct(int leftI,int rightI){
if(leftI==rightI){
    return new SegmentTree(leftI,rightI,-1);
  }

  int midIndex=(leftI+rightI)/2;
  SegmentTree* root=new SegmentTree(leftI,rightI,0);
  SegmentTree* leftTree=construct(leftI,midIndex);
SegmentTree* rightTree=construct(midIndex+1,rightI);
root->left=leftTree;   root->right=rightTree;
  root->maxNum=max(leftTree->maxNum,rightTree->maxNum);
  return root;
}

class Solution { public:   int
lengthOfLIS(vector<int>& nums, int k) {
    int maxN=-1;
for(auto n:nums){
      maxN=max(maxN,n);
    }
    stack<int> st;
SegmentTree* root;
root=construct(0,maxN+k);
int ans=1;
    for(int i=nums.size()-1;i>=0;i--){
      int n=nums[i];
      while(!st.empty() && (st.top()<=n || st.top()>n+k)){
st.pop();
      }
```

```
            st.push(n);
            int l=root->query(n+1,n+k,root)+1;
            ans=max(ans,l);
            root->updateTree(n,l,root);
        }
        return ans;
    }
};
```



Accepted 84 / 84 testcases passed

h_arss_h submitted at Mar 18, 2025 13:24

Runtime

730 ms | Beats 5.14%

Analyze Complexity

Memory

316.81 MB | Beats 5.35%

Code | C++

```
class SegmentTree{
    public:
```

```
            while(!st.empty() && (st.top()<=n || st.top()>n+k)){
                st.pop();
            }
            st.push(n);
            int l=root->query(n+1,n+k,root)+1;
            ans=max(ans,l);
            root->updateTree(n,l,root);
        }
        return ans;
    }
};
```

Saved

Ln 92, Col 3

nums =

```
[4,2,1,4,3,4,5,8,15]
```

k =

```
3
```