Assignment 4

Name: Abhigyan	**Uid: 22BCS10097**
Branch: BE_CSE	**Semester: 6th**
Section: IOT_637-B	**Subject: AP Lab II**

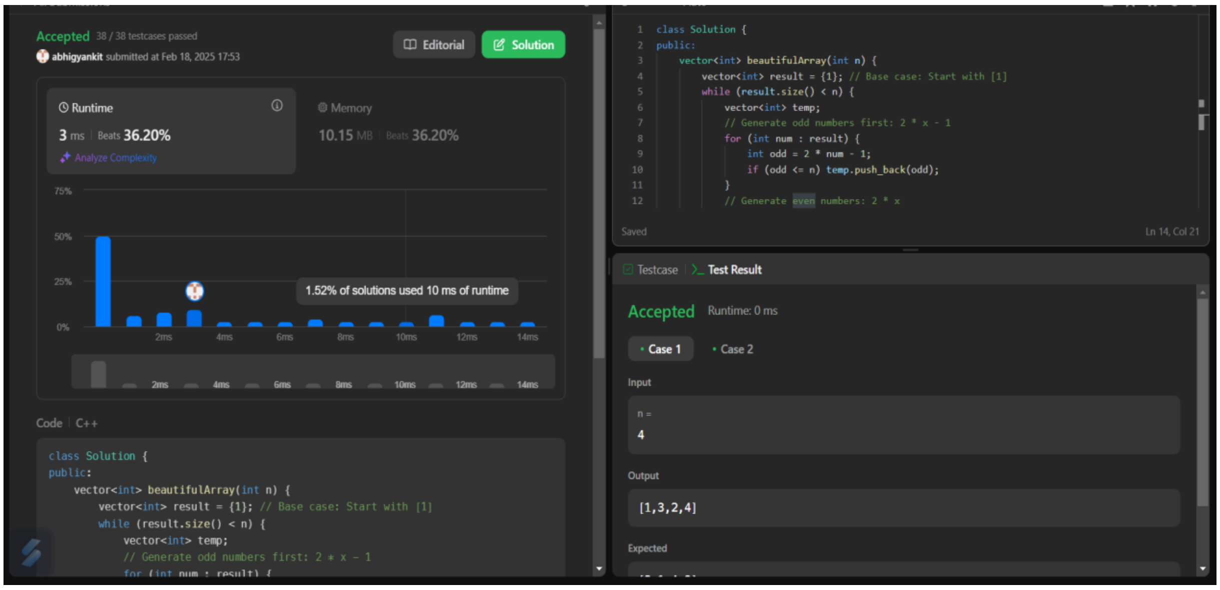
932. Beautiful Array

Aim: An array nums of length n is beautiful if: nums is a permutation of the integers in the range [1, n]. For every 0 <= i < j < n, there is no index k with i < k < j where 2 * nums[k] == nums[i] + nums[j]. Given the integer n, return any beautiful array nums of length n. There will be at least one valid answer for the given n.

Code:

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> result = {1}; // Base case: Start with [1]
        while (result.size() < n) {
            vector<int> temp;
            // Generate odd numbers first: 2 * x - 1
            for (int num : result) {
                int odd = 2 * num - 1;
                if (odd <= n) temp.push_back(odd);
            }
            // Generate even numbers: 2 * x
            for (int num : result) {
                int even = 2 * num;
                if (even <= n) temp.push_back(even);
            }
            result = temp; // Update the result
        }
        return result;
    }
};
```

Output:



Accepted 38 / 38 testcases passed

abhigyankit submitted at Feb 18, 2025 17:53

Runtime

3 ms | Beats 36.20%

Memory

10.15 MB | Beats 36.20%

1.52% of solutions used 10 ms of runtime

Code | C++

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> result = {1}; // Base case: Start with [1]
        while (result.size() < n) {
            vector<int> temp;
            // Generate odd numbers first: 2 * x - 1
```

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        vector<int> result = {1}; // Base case: Start with [1]
        while (result.size() < n) {
            vector<int> temp;
            // Generate odd numbers first: 2 * x - 1
            for (int num : result) {
                int odd = 2 * num - 1;
                if (odd <= n) temp.push_back(odd);
            }
            // Generate even numbers: 2 * x
```

Saved   Ln 14, Col 21

Testcase | Test Result

Accepted Runtime: 0 ms

Case 1 · Case 2

Input

```
n =
4
```

Output

```
[1,3,2,4]
```

Expected

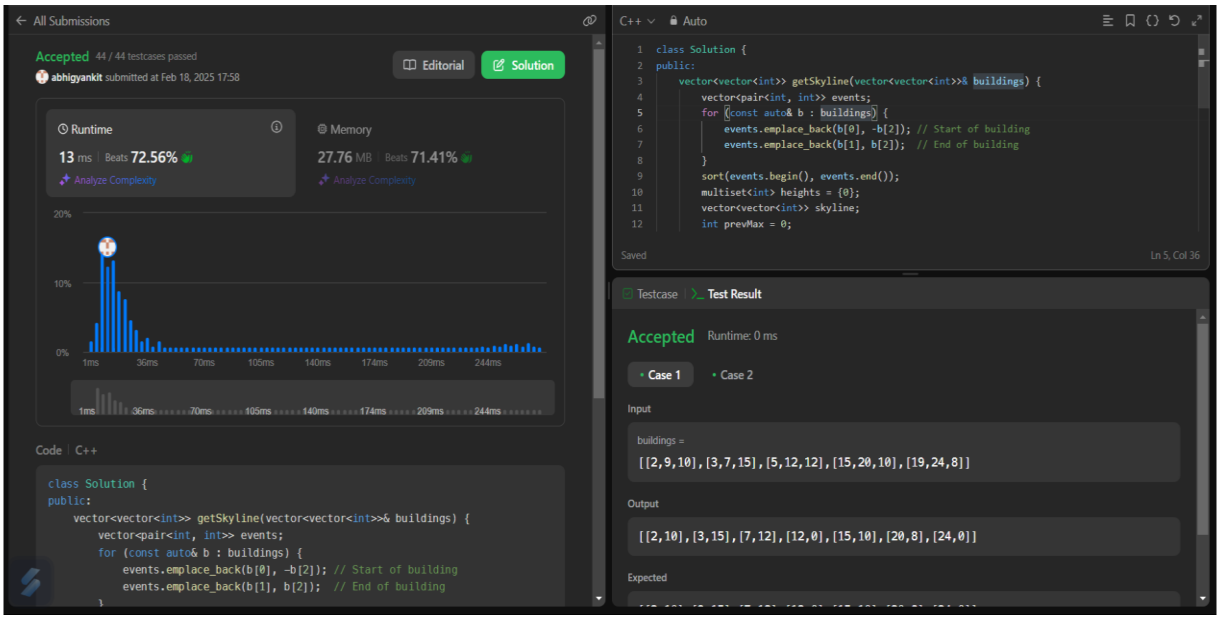
218. The Skyline Problem

Aim: A city's skyline is the outer contour of the silhouette formed by all the buildings in that city when viewed from a distance. Given the locations and heights of all the buildings, return the skyline formed by these buildings collectively. The geometric information of each building is given in the array buildings where buildings[i] = [lefti, righti, heighti]: lefti is the x coordinate of the left edge of the ith building. righti is the x coordinate of the right edge of the ith building. heighti is the height of the ith building. You may assume all buildings are perfect rectangles grounded on an absolutely flat surface at height 0.

Code:

```
class Solution {
public:
   vector<vector<int>> getSkyline(vector<vector<int>>& buildings)
{
        vector<pair<int, int>> events;
        for (const auto& b : buildings) {
            events.emplace_back(b[0], -b[2]); // Start of building
            events.emplace_back(b[1], b[2]);  // End of building
        }
        sort(events.begin(), events.end());
        multiset<int> heights = {0};
        vector<vector<int>> skyline;
        int prevMax = 0;
        // Process each event
        for (const auto& [x, h] : events) {
            if (h < 0) {
                heights.insert(-h); // Add building height
            } else {
                heights.erase(heights.find(h)); // Remove building height
            }
            int currMax = *heights.rbegin(); // Get current max height
            if (currMax != prevMax) { // If height changed, add key point
                skyline.push_back({x, currMax});
                prevMax = currMax;
            }
        }
        return skyline;
    }
};
```

Output



Accepted 44 / 44 testcases passed
abhigyankit submitted at Feb 18, 2025 17:58

Runtime
13 ms | Beats 72.56%

Memory
27.76 MB | Beats 71.41%

Code | C++

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<pair<int, int>> events;
        for (const auto& b : buildings) {
            events.emplace_back(b[0], -b[2]); // Start of building
            events.emplace_back(b[1], b[2]);  // End of building
        }
        sort(events.begin(), events.end());
        multiset<int> heights = {0};
        vector<vector<int>> skyline;
        int prevMax = 0;
```

Accepted Runtime: 0 ms

Input

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

Output
```
[[2,10],[3,15],[7,12],[12,0],[15,10],[20,8],[24,0]]
```

Expected