



Experiment 4

Student Name: Bhaskar Semwal

Branch: BE/CSE

Semester: 6th

Subject Name: AP_LAB-II

UID: 22BCS10717

Section/Group: 22BCS_IOT-638/A

Date of Performance: 19/02/25

Subject Code: 22CSP-351

- 1. Problem 01: Beautiful Array:** An array `nums` of length `n` is beautiful if: `nums` is a permutation of the integers in the range `[1, n]`. For every $0 \leq i < j < n$, there is no index `k` with $i < k < j$ where $2 * \text{nums}[k] == \text{nums}[i] + \text{nums}[j]$. Given the integer `n`, return any beautiful array `nums` of length `n`. There will be at least one valid answer for the given `n`.

2. CODE:-

```
class Solution {
public:
    vector<int> beautifulArray(int n) {
        if (n == 1) return {1};

        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;
    }
};
```

3. Output:

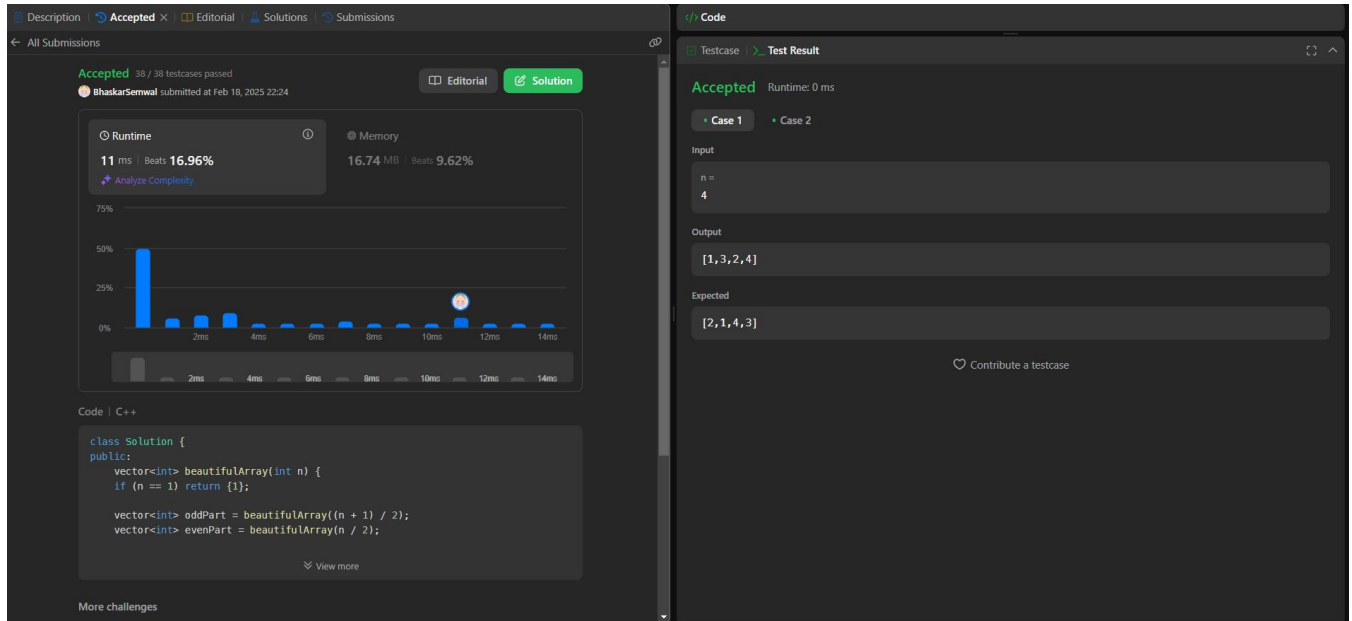


Fig 1: Test Case Accepted and Submission Screenshot

4. Problem 02: The Skyline Problem:-

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. The skyline should be represented as a list of "key points" sorted by their x-coordinate in the form [[x1,y1],[x2,y2],...]. Each key point is the left endpoint of some horizontal segment in the skyline except the last point in the list, which always has a y-coordinate 0 and is used to mark the skyline's termination where the rightmost building ends. Any ground between the leftmost and rightmost buildings should be part of the skyline's contour.

5. CODE:-

```
class Solution {
public:
    vector<vector<int>> getSkyline(vector<vector<int>>& buildings) {
        vector<vector<int>> ans;
        multiset<int> pq{0};

        vector<pair<int, int>> points;
```

```
for(auto b: buildings){
    points.push_back({b[0], -b[2]});
    points.push_back({b[1], b[2]});
}

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

int ongoingHeight = 0;

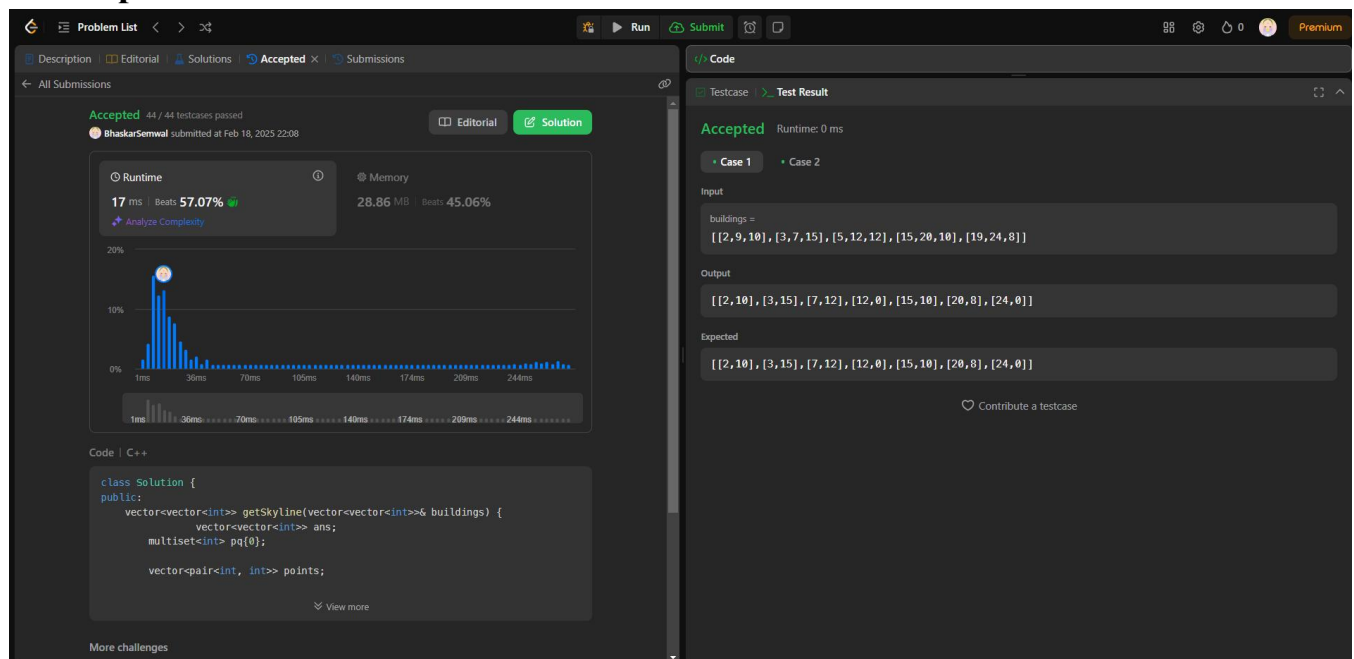
for(int i = 0; i < points.size(); i++){
    int currentPoint = points[i].first;
    int heightAtCurrentPoint = points[i].second;

    if(heightAtCurrentPoint < 0){
        pq.insert(-heightAtCurrentPoint);
    } else {
        pq.erase(pq.find(heightAtCurrentPoint));
    }

    auto pqTop = *pq.rbegin();
    if(ongoingHeight != pqTop){
        ongoingHeight = pqTop;
        ans.push_back({currentPoint, ongoingHeight});
    }
}

return ans;
};
```

6. Output:-



The screenshot displays a coding platform interface with the following components:

- Problem List:** Shows the current problem and navigation options.
- Description:** Contains the problem statement and a link to the solution.
- Runtime:** Displays a graph showing the execution time of the solution. The graph indicates a peak in runtime around 10ms, with a total runtime of 17ms. The solution is marked as "Accepted" with 44/44 testcases passed.
- Memory:** Shows the memory usage of the solution, which is 28.86 MB, with a peak of 45.06%.
- Code:** Displays the C++ code for the solution, which uses a priority queue to maintain the current skyline.
- Test Result:** Shows the input and output for the solution. The input is a list of buildings, and the output is a list of points representing the skyline. The solution is marked as "Accepted" with a runtime of 0 ms.