

ASSIGNMENT -6 (ADVANCED PROGRAMMING) Shourya Gupta – 22BCS11276

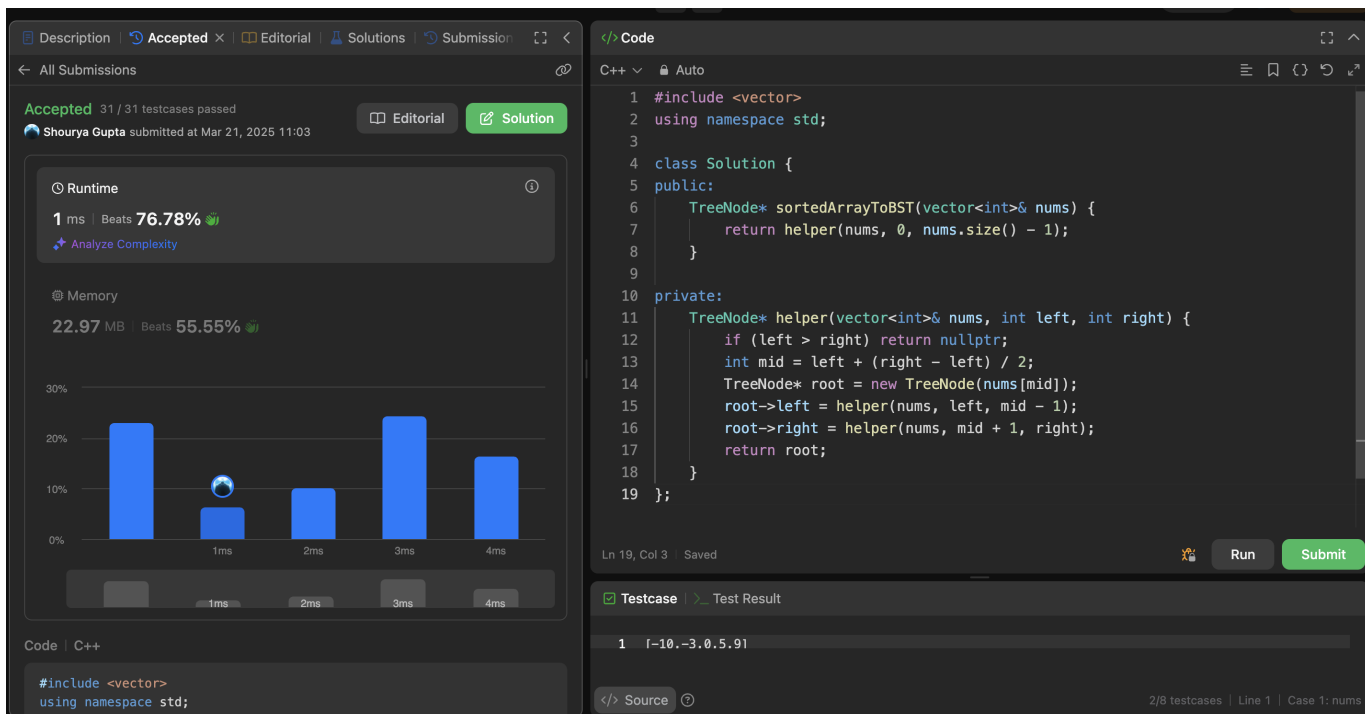
1. Problem 1: Convert Sorted Array to Binary Search Tree

2. Implementation/Code:

```
class Solution {
public:
    TreeNode sortedArrayToBST(int[] nums) {
        return constructBST(nums, 0, nums.length - 1);
    }
private:
    TreeNode constructBST(int[] nums, int left, int right) {
        if (left > right) {
            return null;
        }
        int mid = left + (right - left) / 2;
        TreeNode root = new TreeNode(nums[mid]);
        root.left = constructBST(nums, left, mid - 1);
        root.right = constructBST(nums, mid + 1, right);

        return root;
    }
}
```

3. Output:

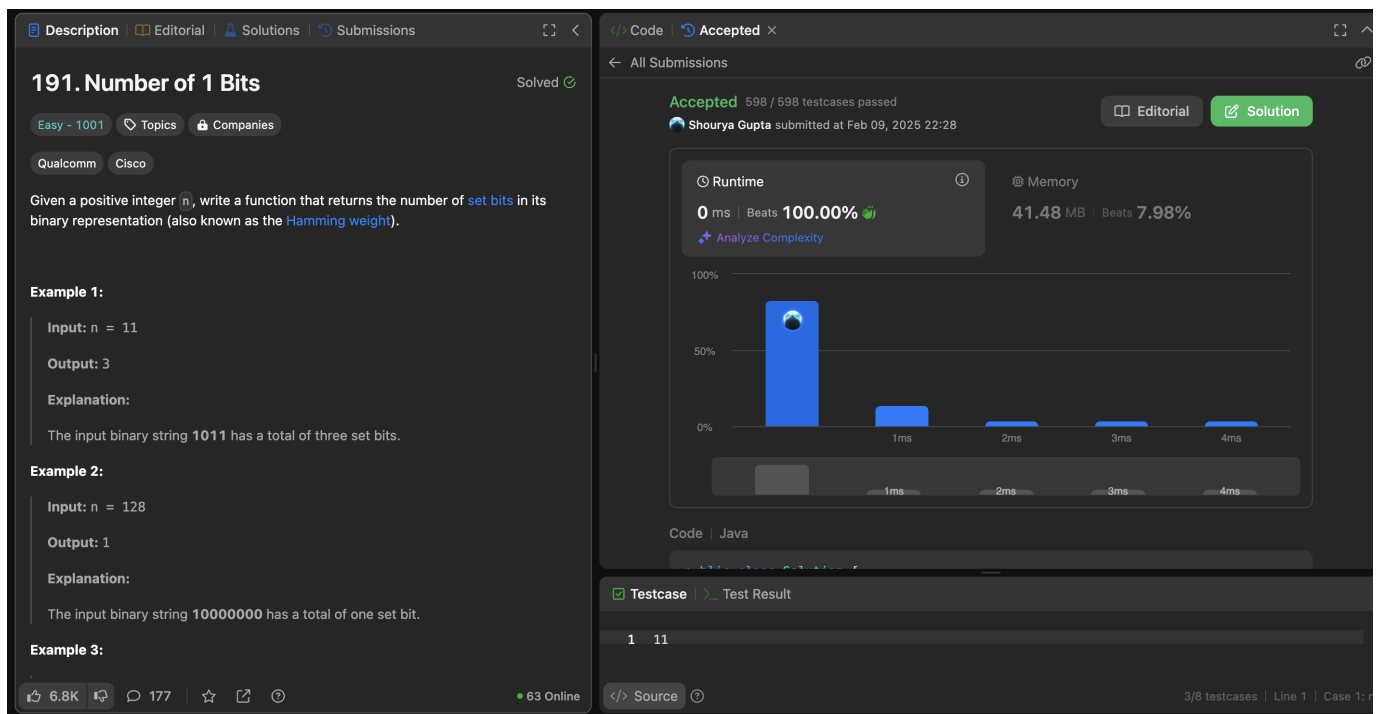


1. Problem 2: Number of 1 bits

2. Implementation/code:

```
public class Solution {  
    public int hammingWeight(int n) {  
        int count = 0;  
        while (n != 0) {  
            count += (n & 1);  
            n >>= 1;  
        }  
        return count;  
    }  
}
```

3. Output:



The screenshot displays a coding platform interface for the problem "191. Number of 1 Bits". The left panel shows the problem description, examples, and company tags (Qualcomm, Cisco). The right panel shows the submission details, including the runtime (0 ms), memory (41.48 MB), and a bar chart comparing the solution's performance to other submissions. The bottom panel shows the test case input and output.

191. Number of 1 Bits Solved ✓

Easy - 1001 Topics Companies

Qualcomm Cisco

Given a positive integer `n`, write a function that returns the number of **set bits** in its binary representation (also known as the **Hamming weight**).

Example 1:

Input: `n = 11`

Output: 3

Explanation:

The input binary string `1011` has a total of three set bits.

Example 2:

Input: `n = 128`

Output: 1

Explanation:

The input binary string `10000000` has a total of one set bit.

Example 3:

6.8K 177 63 Online

Accepted 598 / 598 testcases passed

Shourya Gupta submitted at Feb 09, 2025 22:28

Editorial Solution

Runtime 0 ms | Beats 100.00% Analyze Complexity

Memory 41.48 MB | Beats 7.98%

Code | Java

Testcase Test Result

1 11

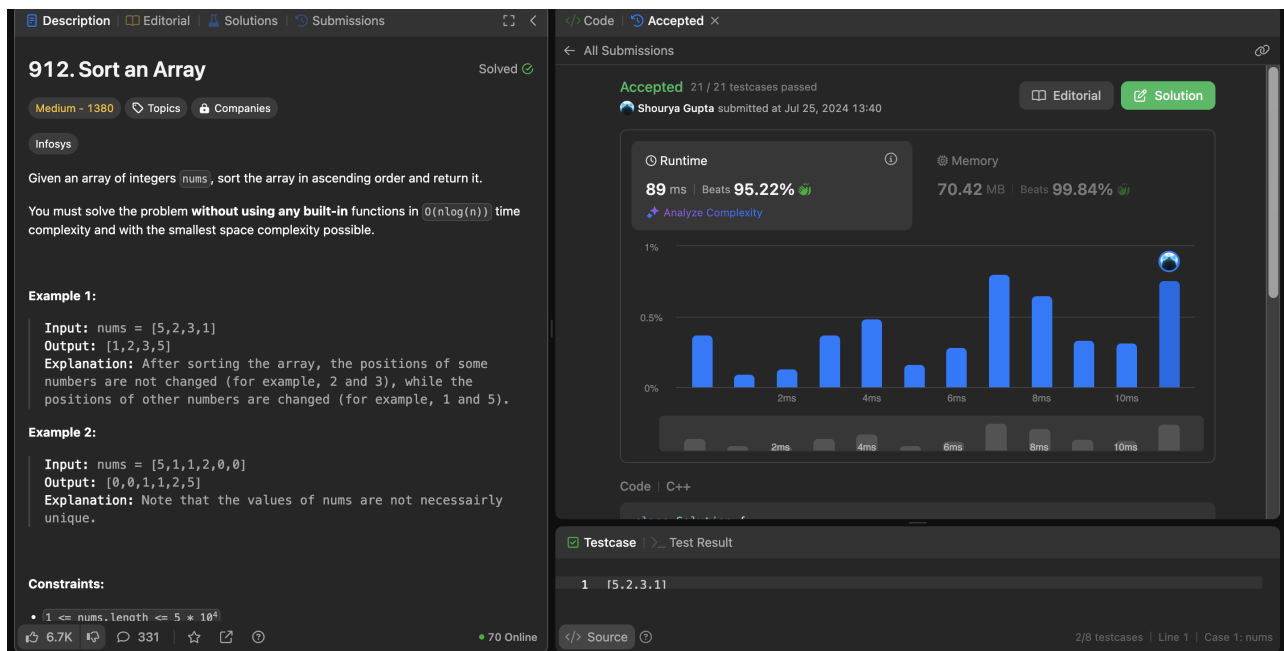
3/8 testcases | Line 1 | Case 1: r

1. Problem 3: Sort an Array

2. Implementation/Code:

```
class Solution {
    public int[] sortArray(int[] nums) {
        mergeSort(nums, 0, nums.length - 1);
        return nums;
    }
    public static void mergeFun(int[] arr, int l, int m, int r) {
        int n1 = m + 1 - 1; int n2 = r - m; int[] left = new int[n1];
        for (int i = 0; i < n1; i++) { left[i] = arr[l + i]; }
        int[] right = new int[n2];
        for (int i = 0; i < n2; i++) { right[i] = arr[m + 1 + i]; }
        int i = 0, j = 0, k = l;
        while (i < n1 || j < n2) {
            if (j == n2 || i < n1 && left[i] < right[j]) arr[k++] = left[i++];
            else arr[k++] = right[j++];
        }
    }
    public static void mergeSort(int[] arr, int low, int high) {
        if (low < high) { int middle = (high - low) / 2 + low;
            mergeSort(arr, low, middle);
            mergeSort(arr, middle + 1, high);
            mergeFun(arr, low, middle, high);
        }
    }
}
```

3. Output:



The screenshot displays a coding platform interface for the '912. Sort an Array' problem. The left panel shows the problem description, examples, and constraints. The right panel shows the solution code in C++ and a performance graph.

Problem Description: 912. Sort an Array. Medium - 1380. Topics: Arrays, Sorting. Companies: Infosys.

Given: An array of integers `nums`, sort the array in ascending order and return it.

You must solve the problem without using any built-in functions in $O(n \log(n))$ time complexity and with the smallest space complexity possible.

Example 1:
Input: `nums = [5,2,3,1]`
Output: `[1,2,3,5]`
Explanation: After sorting the array, the positions of some numbers are not changed (for example, 2 and 3), while the positions of other numbers are changed (for example, 1 and 5).

Example 2:
Input: `nums = [5,1,1,2,0,0]`
Output: `[0,0,1,1,2,5]`
Explanation: Note that the values of `nums` are not necessarily unique.

Constraints:
• $1 \leq \text{nums.length} \leq 5 \times 10^4$

Solution: The solution code is in C++ and uses a merge sort algorithm. It is marked as 'Accepted' with 21/21 testcases passed. The runtime is 89 ms (Beats 95.22%) and the memory is 70.42 MB (Beats 99.84%).

Performance Graph: The graph shows the runtime performance of the solution compared to other submissions. The x-axis represents runtime in milliseconds (2ms, 4ms, 6ms, 8ms, 10ms) and the y-axis represents the percentage of submissions beaten (0%, 0.5%, 1%). The solution is shown as a blue bar at the top of the graph, indicating it is among the fastest.

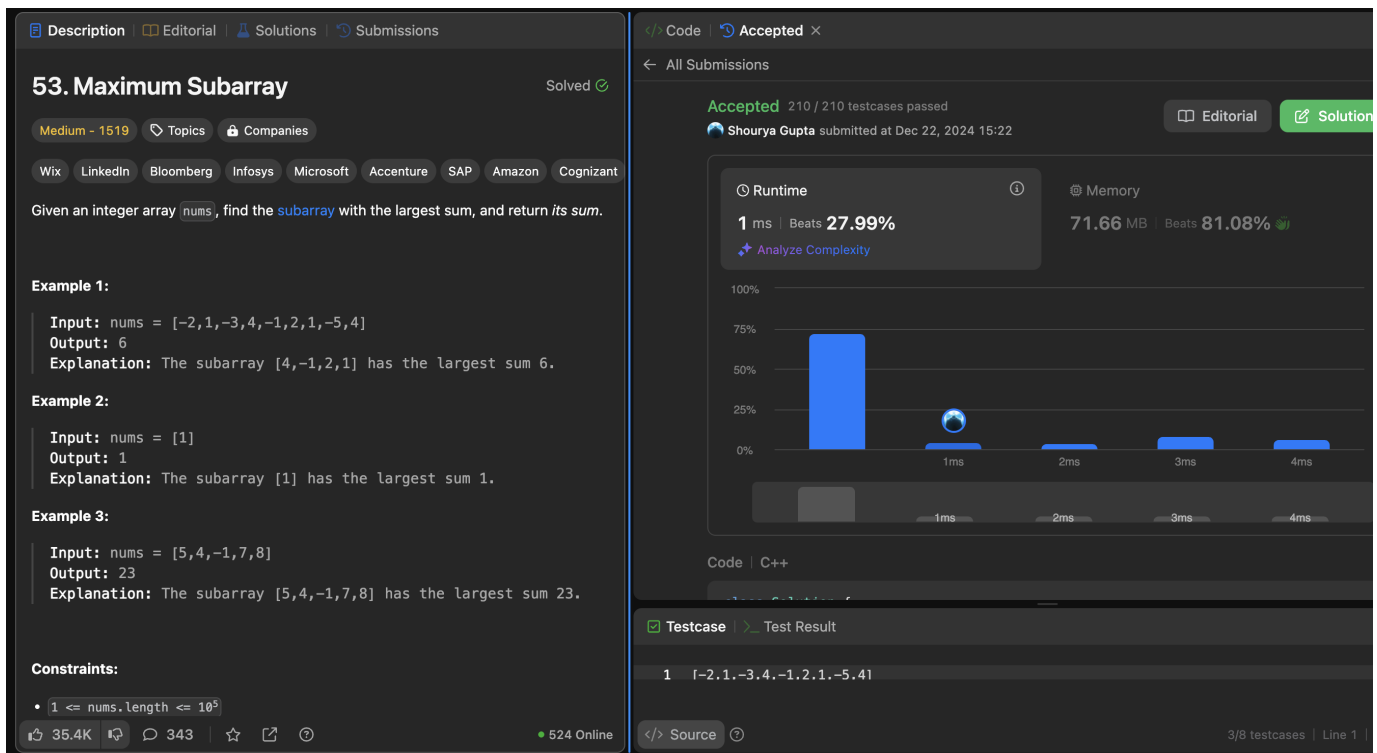
Testcase: The test result for the first testcase is shown as '1 (5.2.3.1)'.

1. Problem 4: Maximum Sub array


2. Implementation/code:

```
public class Solution {  
    public int maxSubArray(int[] nums) {  
        int maxSum = nums[0], currentSum = 0;  
        for (int num : nums) {  
            currentSum = Math.max(num, currentSum + num);  
            maxSum = Math.max(maxSum, currentSum);  
        }  
        return maxSum;  
    }  
}
```

3. Output:



The screenshot displays the LeetCode interface for problem 53, "Maximum Subarray". The left sidebar contains the problem description, examples, and constraints. The main area shows the solution status as "Accepted", the runtime performance (1 ms, 27.99% beats), and a bar chart comparing the solution's runtime to other submissions. The bottom section shows the test cases and the source code.

53. Maximum Subarray Solved 

Medium - 1519 Topics Companies

Wix LinkedIn Bloomberg Infosys Microsoft Accenture SAP Amazon Cognizant

Given an integer array `nums`, find the **subarray** with the largest sum, and return *its sum*.


Example 1:
Input: `nums = [-2,1,-3,4,-1,2,1,-5,4]`
Output: 6
Explanation: The subarray `[4,-1,2,1]` has the largest sum 6.

Example 2:
Input: `nums = [1]`
Output: 1
Explanation: The subarray `[1]` has the largest sum 1.

Example 3:
Input: `nums = [5,4,-1,7,8]`
Output: 23
Explanation: The subarray `[5,4,-1,7,8]` has the largest sum 23.

Constraints:
• `1 <= nums.length <= 105`


35.4K 343 524 Online


Code Accepted 

All Submissions

Accepted 210 / 210 testcases passed
Shourya Gupta submitted at Dec 22, 2024 15:22

Editorial Solution

Runtime  @ Memory

1 ms Beats 27.99% 71.66 MB Beats 81.08% 

Analyze Complexity

100% 75% 50% 25% 0%

1ms 2ms 3ms 4ms

Code C++

Testcase Test Result

1 [-2,1,-3,4,-1,2,1,-5,4]

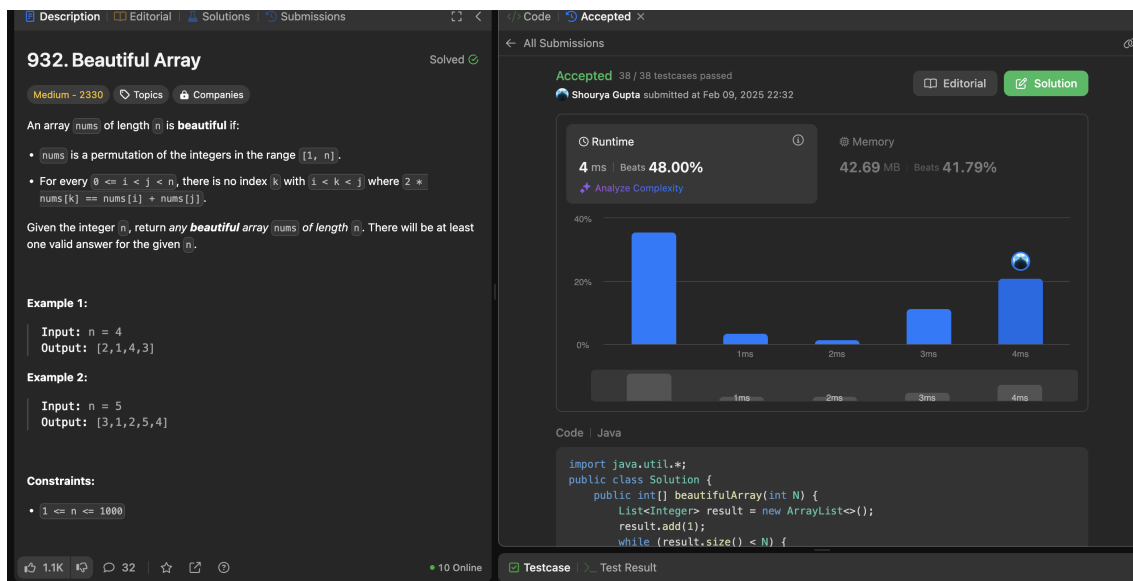
3/8 testcases | Line 1

1. Problem 5: Beautiful Array

2. Implementation/Code:

```
import java.util.*;
public class Solution {
    public int[] beautifulArray(int N) {
        List<Integer> result = new ArrayList<>();
        result.add(1);
        while (result.size() < N) {
            List<Integer> temp = new ArrayList<>();
            for (int num : result) {
                if (num * 2 - 1 <= N) temp.add(num * 2 - 1);
            }
            for (int num : result) {
                if (num * 2 <= N) temp.add(num * 2);
            }
            result = temp;
        }
        int[] arr = new int[result.size()];
        for (int i = 0; i < result.size(); i++) {
            arr[i] = result.get(i);
        }
        return arr;
    }
}
```

3. Output:



932. Beautiful Array Solved

Medium - 2330 Topics Companies

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`.

Example 1:
Input: `n = 4`
Output: `[2,1,4,3]`

Example 2:
Input: `n = 5`
Output: `[3,1,2,5,4]`

Constraints:

- $1 \leq n \leq 1000$

Accepted 38 / 38 testcases passed
Shourya Gupta submitted at Feb 09, 2025 22:32

Runtime 4 ms | Beats 48.00%
Memory 42.69 MB | Beats 41.79%

Code | Java

```
import java.util.*;
public class Solution {
    public int[] beautifulArray(int N) {
        List<Integer> result = new ArrayList<>();
        result.add(1);
        while (result.size() < N) {
```

Testcase | **Test Result**

1. Problem 6: Super Pow

2. Implementation/Code:

```
public class Solution {
    private static final int MOD = 1337;
    private int pow(int a, int b) {
        int res = 1;
        a %= MOD;
        for (int i = 0; i < b; i++) {
            res = (res * a) % MOD;
        }
        return res;
    }
    public int superPow(int a, int[] b) {
        int res = 1;
        for (int i = b.length - 1; i >= 0; i--) {
            res = (res * pow(a, b[i])) % MOD;
            a = pow(a, 10);
        }
        return res;
    }
}
```

3. Output:

372. Super Pow

Solved

Medium - 1688

Your task is to calculate $a^b \text{ mod } 1337$ where a is a positive integer and b is an extremely large positive integer given in the form of an array.

Example 1:
Input: $a = 2, b = [3]$
Output: 8

Example 2:
Input: $a = 2, b = [1,0]$
Output: 1024

Example 3:
Input: $a = 1, b = [4,3,3,8,5,2]$
Output: 1

Constraints:

- $1 \leq a \leq 2^{31} - 1$
- $1 \leq b.length \leq 2000$
- $0 \leq b[i] \leq 9$

5 Online

All Submissions

Accepted 57 / 57 testcases passed

Shourya Gupta submitted at Feb 09, 2025 22:31

Solution

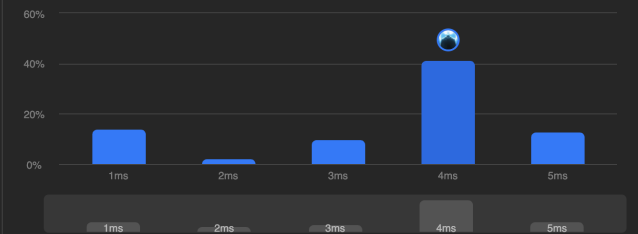
Runtime

4 ms | Beats 74.40%

Analyze Complexity

Memory

44.31 MB | Beats 58.91%



Code | Java

Testcase | Test Result

1 2

3/8 testcases | Line 1 | Case 1: a

1. Problem 7: The Skyline Problem

2. Implementation/Code:

```
import java.util.*;
class Solution {
    public List<List<Integer>> getSkyline(int[][] buildings) {
        return divideAndConquer(buildings, 0, buildings.length - 1);
    }
    private List<List<Integer>> divideAndConquer(int[][] buildings, int left,
int right) {
        if (left > right) return new ArrayList<>();
        if (left == right) {
            List<List<Integer>> result = new ArrayList<>();
            result.add(Arrays.asList(buildings[left][0], buildings[left][2]));
            result.add(Arrays.asList(buildings[left][1], 0));
            return result;
        }

        int mid = left + (right - left) / 2;
        List<List<Integer>> leftSkyline = divideAndConquer(buildings, left,
mid);
        List<List<Integer>> rightSkyline = divideAndConquer(buildings, mid
+ 1, right);

        return mergeSkylines(leftSkyline, rightSkyline);
    }
    private List<List<Integer>> mergeSkylines(List<List<Integer>> left,
List<List<Integer>> right) {
        List<List<Integer>> result = new ArrayList<>();
        int h1 = 0, h2 = 0, i = 0, j = 0;
        while (i < left.size() && j < right.size()) {
            List<Integer> point1 = left.get(i);
            List<Integer> point2 = right.get(j);

            int x;
            if (point1.get(0) < point2.get(0)) {
```

```

        x = point1.get(0);
        h1 = point1.get(1);
        i++;
    } else if (point1.get(0) > point2.get(0)) {
        x = point2.get(0);
        h2 = point2.get(1);
        j++;
    } else {
        x = point1.get(0);
        h1 = point1.get(1);
        h2 = point2.get(1);
        i++;
        j++;
    }
    int maxHeight = Math.max(h1, h2);
    if (result.isEmpty() || result.get(result.size() - 1).get(1) != maxHeight)
    {
        result.add(Arrays.asList(x, maxHeight));
    }
    while (i < left.size()) result.add(left.get(i++));
    while (j < right.size()) result.add(right.get(j++));

    return result;
}

```

3. Output:

Description
Editorial
Solutions
Submissions

218. The Skyline Problem

Hard - 1929

Yelp X Salesforce Citadel Google Goldman Sachs

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 *i*th building.
- `righti` is the x coordinate of the right edge of the *i*th building.
- `heighti` is the height of the *i*th 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.

Note: There must be no consecutive horizontal lines of equal height in the output skyline. For instance, `[..., [2 3], [4 5], [7 5], [11 5], [12 7], ...]` is not acceptable; the three lines of height 5 should be merged into one in the final output as such: `[..., [2 5], [11 5], [12 7], ...]`.

Solved


Code
Accepted

All Submissions

Accepted 44 / 44 testcases passed
Shourya Gupta submitted at Feb 09, 2025 22:33

Runtime
14 ms | Beats 91.17%

Memory
50.07 MB | Beats 98.27%



Code | Java

Testcase | Test Result

1 [2,9,10], [3,7,15], [5,12,12], [15,20,10], [19,24,8]

2/8 testcases | Line 1 | Case 1: buildings