Advanced Programming LAB II

ASSIGNMENT - 6

Submitted by,

Jiya | 22BCS14856

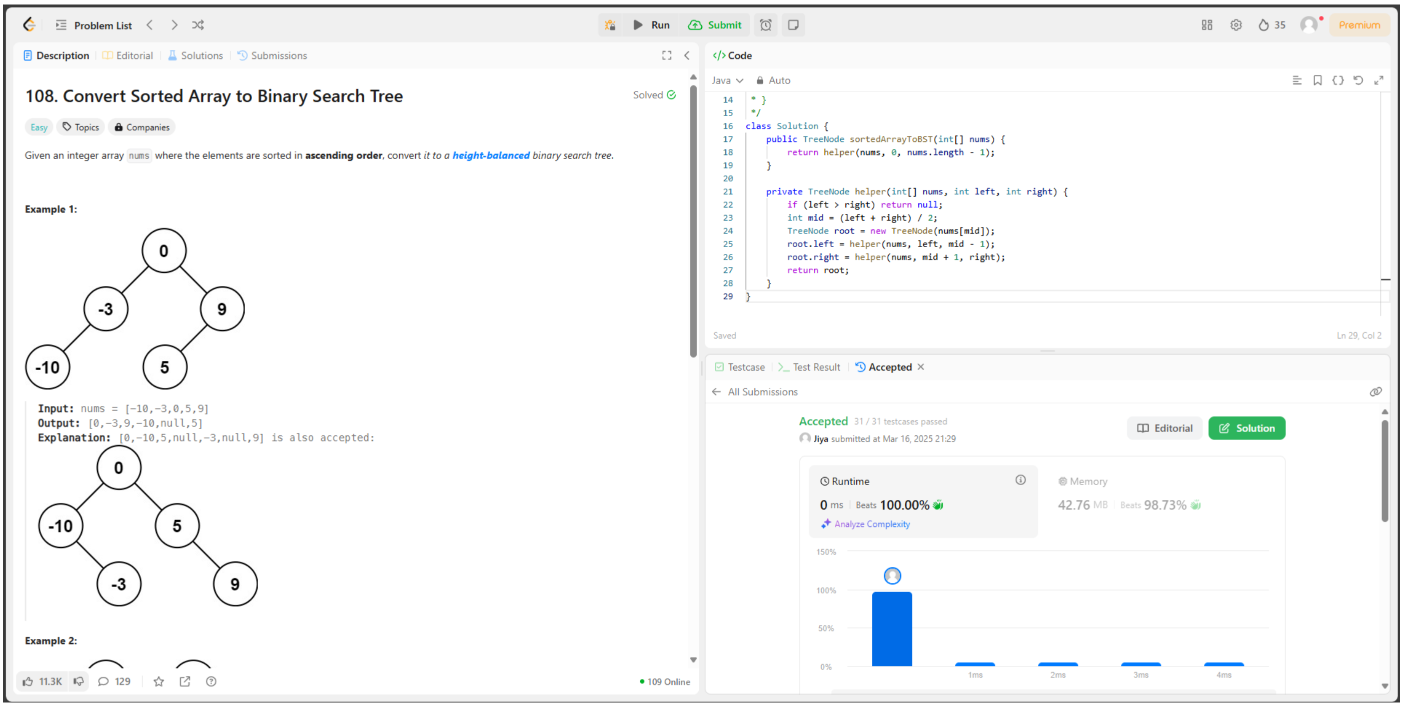
22BCS_FL_IOT-601 (A)

108. Convert Sorted Array to Binary Search Tree

https://leetcode.com/problems/convert-sorted-array-to-binary-search-tree/description/

```
class Solution {
    public TreeNode sortedArrayToBST(int[] nums) {
        return helper(nums, 0, nums.length - 1);
    }

    private TreeNode helper(int[] nums, int left, int right) {
        if (left > right) return null;
        int mid = (left + right) / 2;
        TreeNode root = new TreeNode(nums[mid]);
        root.left = helper(nums, left, mid - 1);
        root.right = helper(nums, mid + 1, right);
        return root;
    }
}
```



108. Convert Sorted Array to Binary Search Tree

Solved

Easy · Topics · Companies

Given an integer array `nums` where the elements are sorted in **ascending order**, convert *it to a* ***height-balanced*** *binary search tree.*

Example 1:

```
Input: nums = [-10,-3,0,5,9]
Output: [0,-3,9,-10,null,5]
Explanation: [0,-10,5,null,-3,null,9] is also accepted:
```

Example 2:

```
 * }
 */
class Solution {
    public TreeNode sortedArrayToBST(int[] nums) {
        return helper(nums, 0, nums.length - 1);
    }

    private TreeNode helper(int[] nums, int left, int right) {
        if (left > right) return null;
        int mid = (left + right) / 2;
        TreeNode root = new TreeNode(nums[mid]);
        root.left = helper(nums, left, mid - 1);
        root.right = helper(nums, mid + 1, right);
        return root;
    }
}
```

Saved

Ln 29, Col 2

Accepted 31 / 31 testcases passed

Jiya submitted at Mar 16, 2025 21:29

Runtime

0 ms | Beats 100.00%

Analyze Complexity

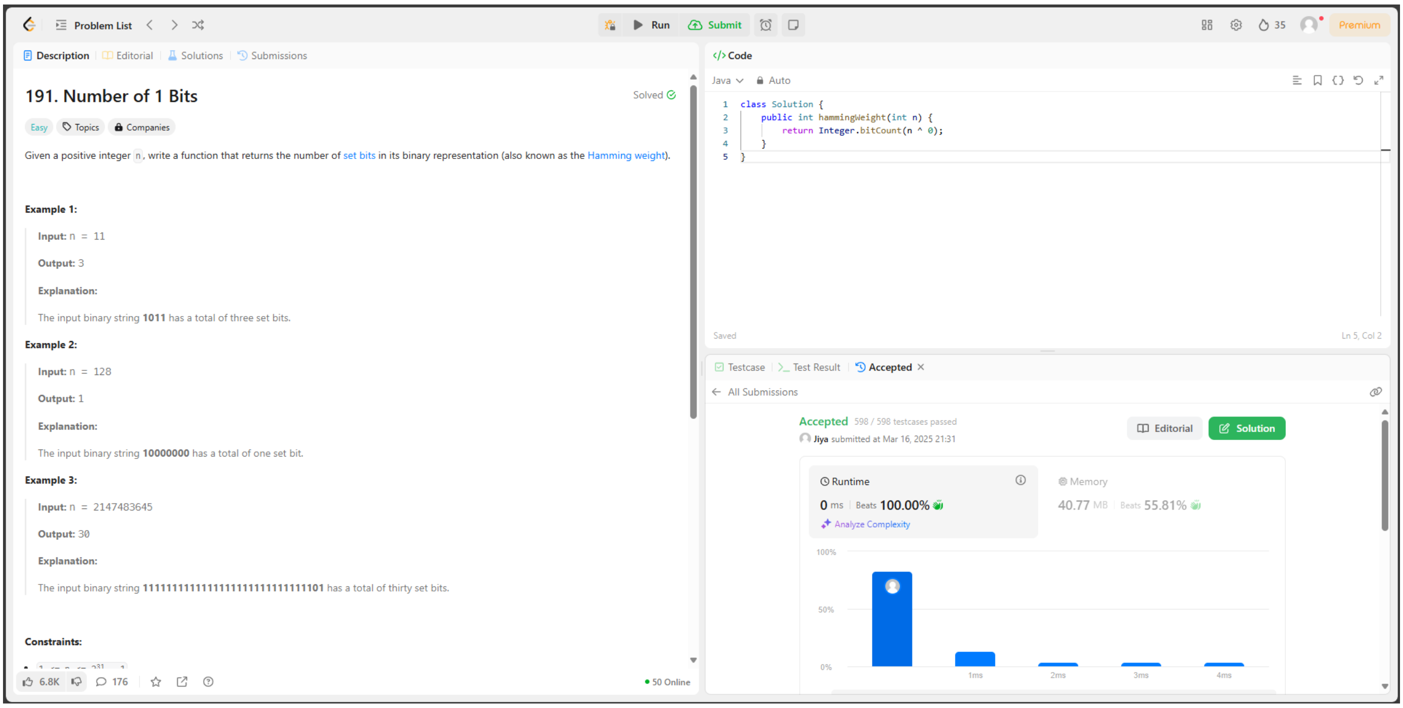
Memory

42.76 MB | Beats 98.73%

191. Number of 1 Bits

https://leetcode.com/problems/number-of-1-bits/description/

```
class Solution {
    public int hammingWeight(int n) {
        return Integer.bitCount(n ^ 0);
    }
}
```



191. Number of 1 Bits

Solved

Easy Topics Companies

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:

Input: n = 2147483645

Output: 30

Explanation:

The input binary string **1111111111111111111111111111101** has a total of thirty set bits.

Constraints:

```
class Solution {
    public int hammingWeight(int n) {
        return Integer.bitCount(n ^ 0);
    }
}
```

Accepted 598 / 598 testcases passed

Jiya submitted at Mar 16, 2025 21:31

Runtime

0 ms | Beats 100.00%

Analyze Complexity

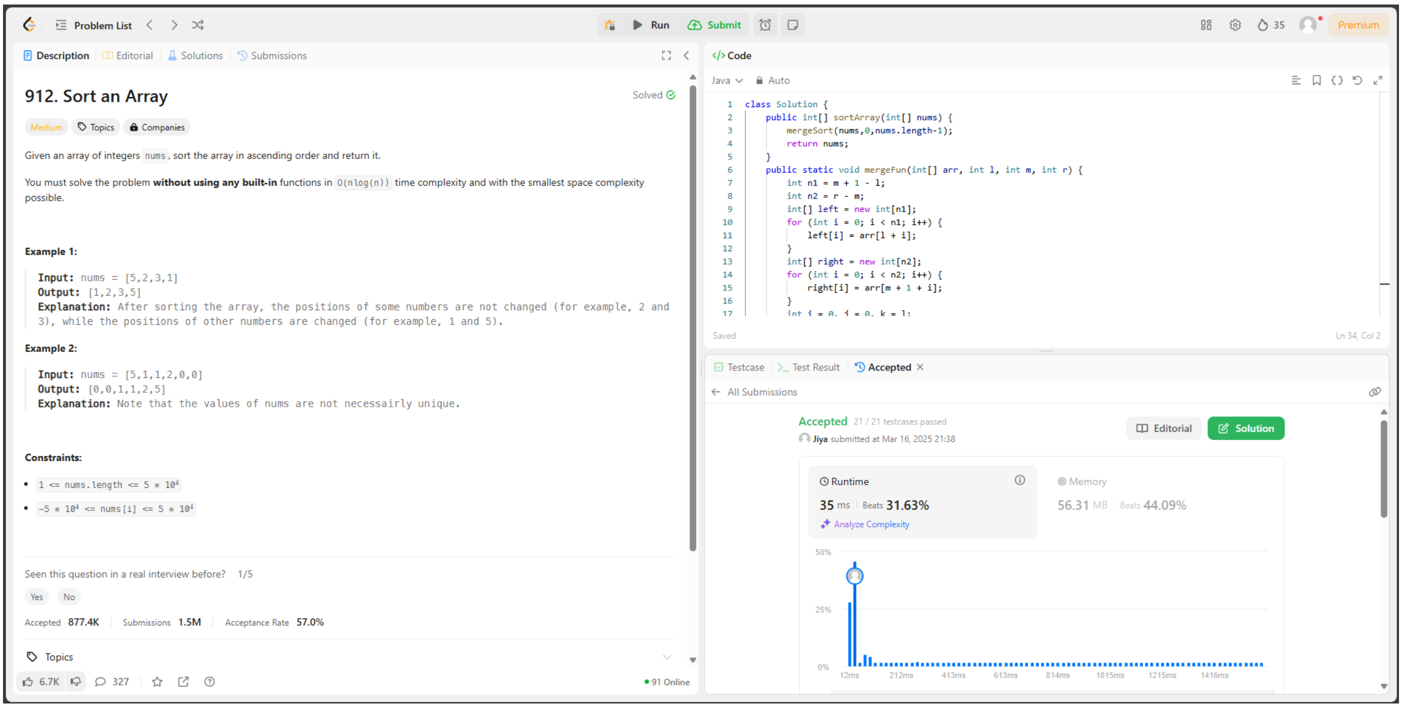
Memory

40.77 MB | Beats 55.81%

912. Sort an Array

https://leetcode.com/problems/third-maximum-number/description/

```
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 - l;
        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);
        }
    }
}
```



912. Sort an Array

Solved

Medium

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(nlog(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 necessairly unique.
```

Constraints:

- `1 <= nums.length <= 5 * 10`^{4}
- `-5 * 10`^{4} `<= nums[i] <= 5 * 10`^{4}

Seen this question in a real interview before? 1/5

Accepted 877.4K | Submissions 1.5M | Acceptance Rate 57.0%

```
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 - l;
        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;
```

Saved

Ln 34, Col 2

Accepted 21 / 21 testcases passed

Jiya submitted at Mar 16, 2025 21:38

Runtime

35 ms | Beats 31.63%

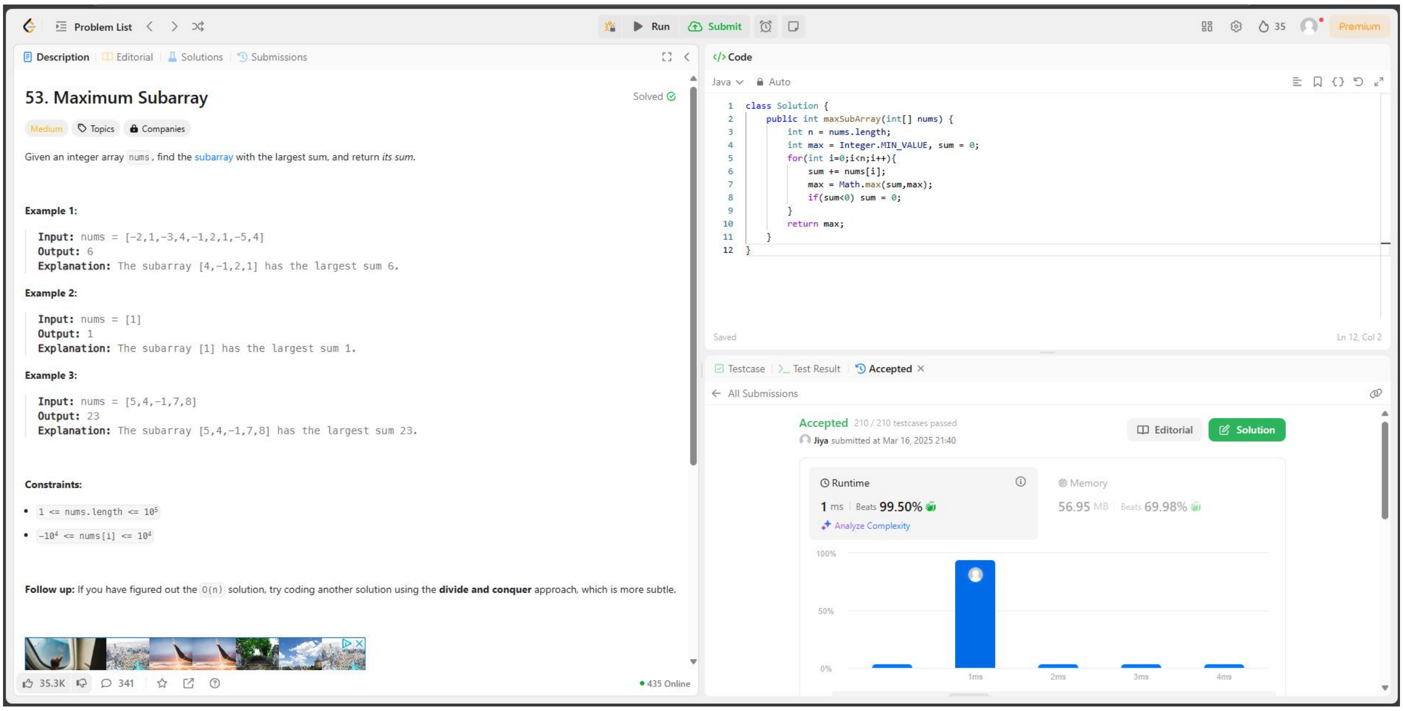
Memory

56.31 MB | Beats 44.09%

53. Maximum Subarray

https://leetcode.com/problems/maximum-subarray/description/

```
class Solution {
    public int maxSubArray(int[] nums) {
        int n = nums.length;
        int max = Integer.MIN_VALUE, sum = 0;
        for(int i=0;i<n;i++){
            sum += nums[i];
            max = Math.max(sum,max);
            if(sum<0) sum = 0;
        }
        return max;
    }
}
```



53. Maximum Subarray

Solved ✓

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:

- $1 <= nums.length <= 10^5$
- $-10^4 <= nums[i] <= 10^4$

Follow up: If you have figured out the `O(n)` solution, try coding another solution using the **divide and conquer** approach, which is more subtle.

```
class Solution {
    public int maxSubArray(int[] nums) {
        int n = nums.length;
        int max = Integer.MIN_VALUE, sum = 0;
        for(int i=0;i<n;i++){
            sum += nums[i];
            max = Math.max(sum,max);
            if(sum<0) sum = 0;
        }
        return max;
    }
}
```

Accepted 210 / 210 testcases passed

Jiya submitted at Mar 16, 2025 21:40

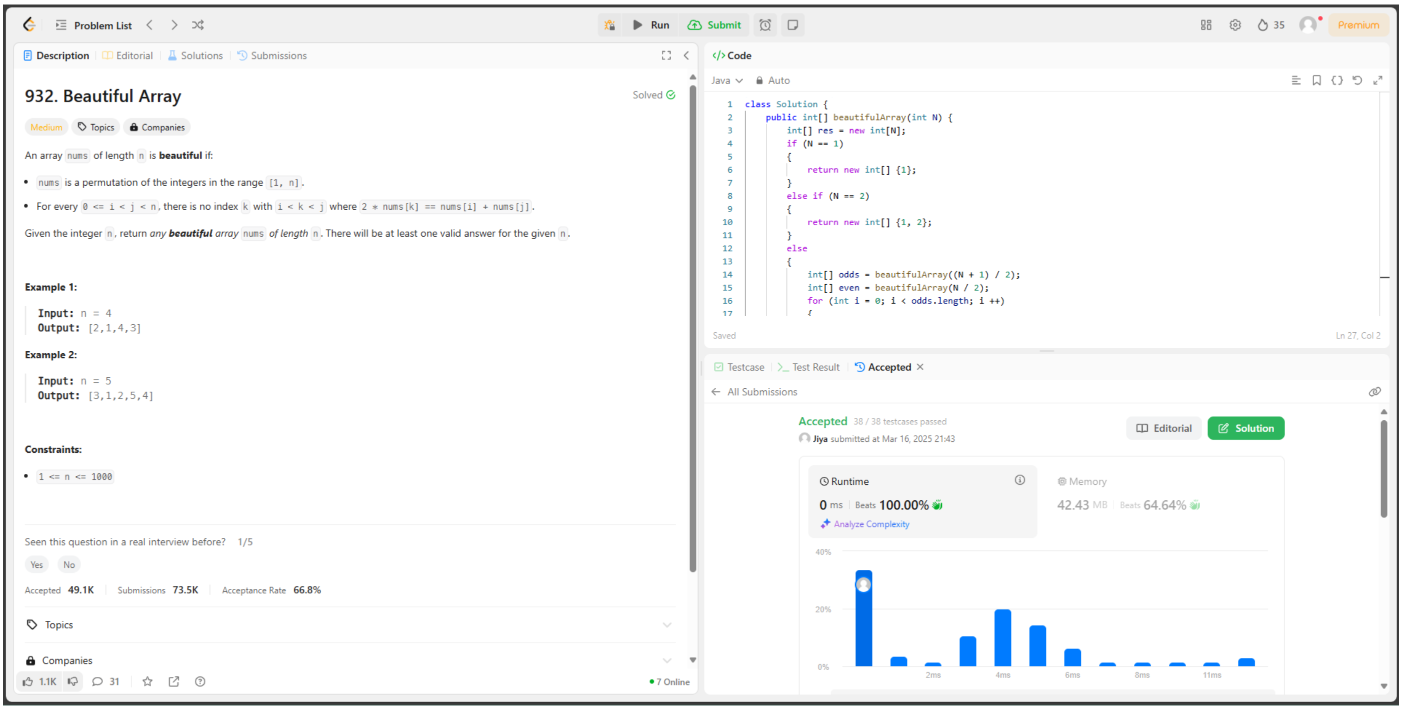
Runtime: 1 ms | Beats 99.50%

Memory: 56.95 MB | Beats 69.98%

932. Beautiful Array

https://leetcode.com/problems/beautiful-array/description/

```
class Solution {
    public int[] beautifulArray(int N) {
        int[] res = new int[N];
        if (N == 1)
        {
            return new int[] {1};
        }
        else if (N == 2)
        {
            return new int[] {1, 2};
        }
        else
        {
            int[] odds = beautifulArray((N + 1) / 2);
            int[] even = beautifulArray(N / 2);
            for (int i = 0; i < odds.length; i ++)
            {
                res[i] = odds[i] * 2 - 1;
            }
            for (int j = 0; j < even.length; j ++)
            {
                res[odds.length + j] = even[j] * 2;
            }
        }
        return res;
    }
}
```



932. 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 <= 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`.

Example 1:

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

Example 2:

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

Constraints:

- `1 <= n <= 1000`

Seen this question in a real interview before? 1/5

Yes No

Accepted 49.1K | Submissions 73.5K | Acceptance Rate 66.8%

```
class Solution {
    public int[] beautifulArray(int N) {
        int[] res = new int[N];
        if (N == 1)
        {
            return new int[] {1};
        }
        else if (N == 2)
        {
            return new int[] {1, 2};
        }
        else
        {
            int[] odds = beautifulArray((N + 1) / 2);
            int[] even = beautifulArray(N / 2);
            for (int i = 0; i < odds.length; i ++)
            {
```

Accepted 38 / 38 testcases passed

Jiya submitted at Mar 16, 2025 21:43

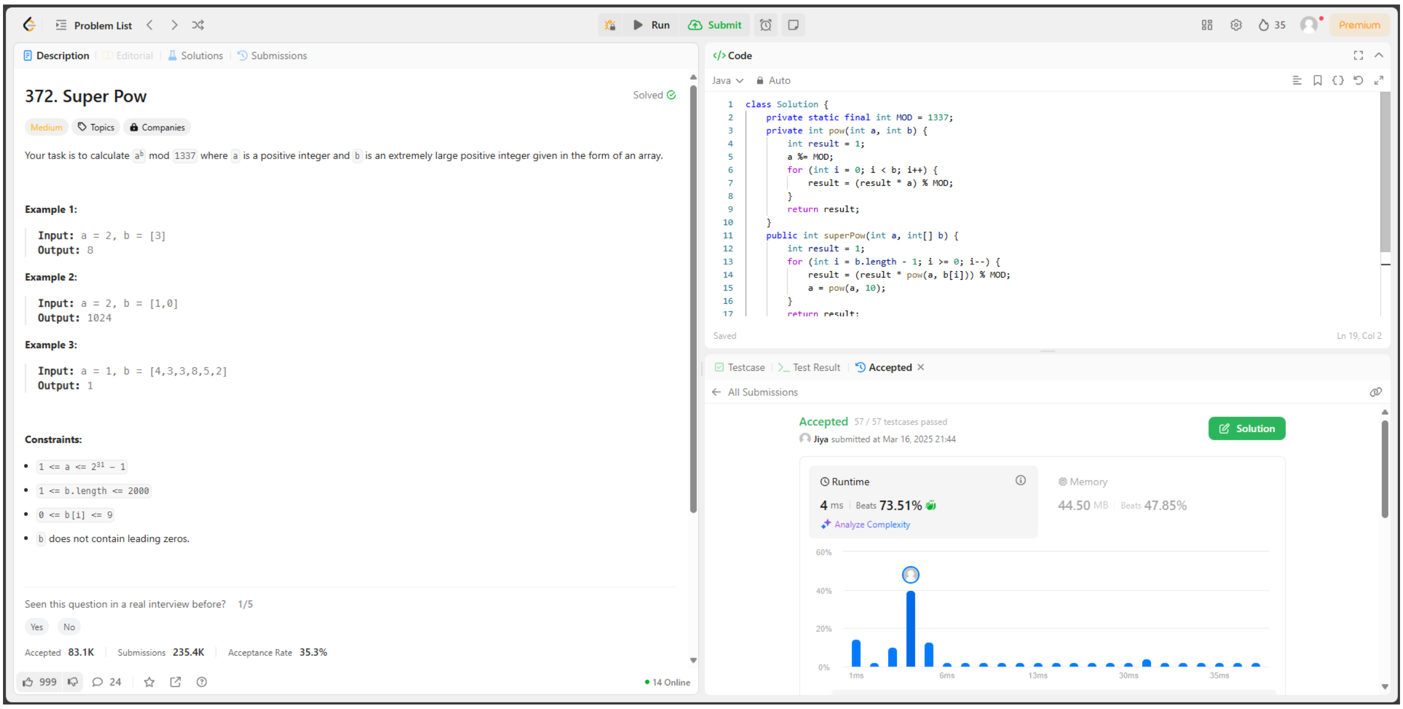
Runtime: 0 ms | Beats 100.00%

Memory: 42.43 MB | Beats 64.64%

372.Super Pow

https://leetcode.com/problems/super-pow/description/

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



372. Super Pow

Solved

Medium · Topics · Companies

Your task is to calculate a^b 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:

- $1 <= a <= 2^{31} - 1$
- `1 <= b.length <= 2000`
- `0 <= b[i] <= 9`
- b does not contain leading zeros.

Seen this question in a real interview before? 1/5

Yes No

Accepted 83.1K | Submissions 235.4K | Acceptance Rate 35.3%

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

Accepted 57 / 57 testcases passed

Jiya submitted at Mar 16, 2025 21:44

Runtime
4 ms | Beats 73.51%

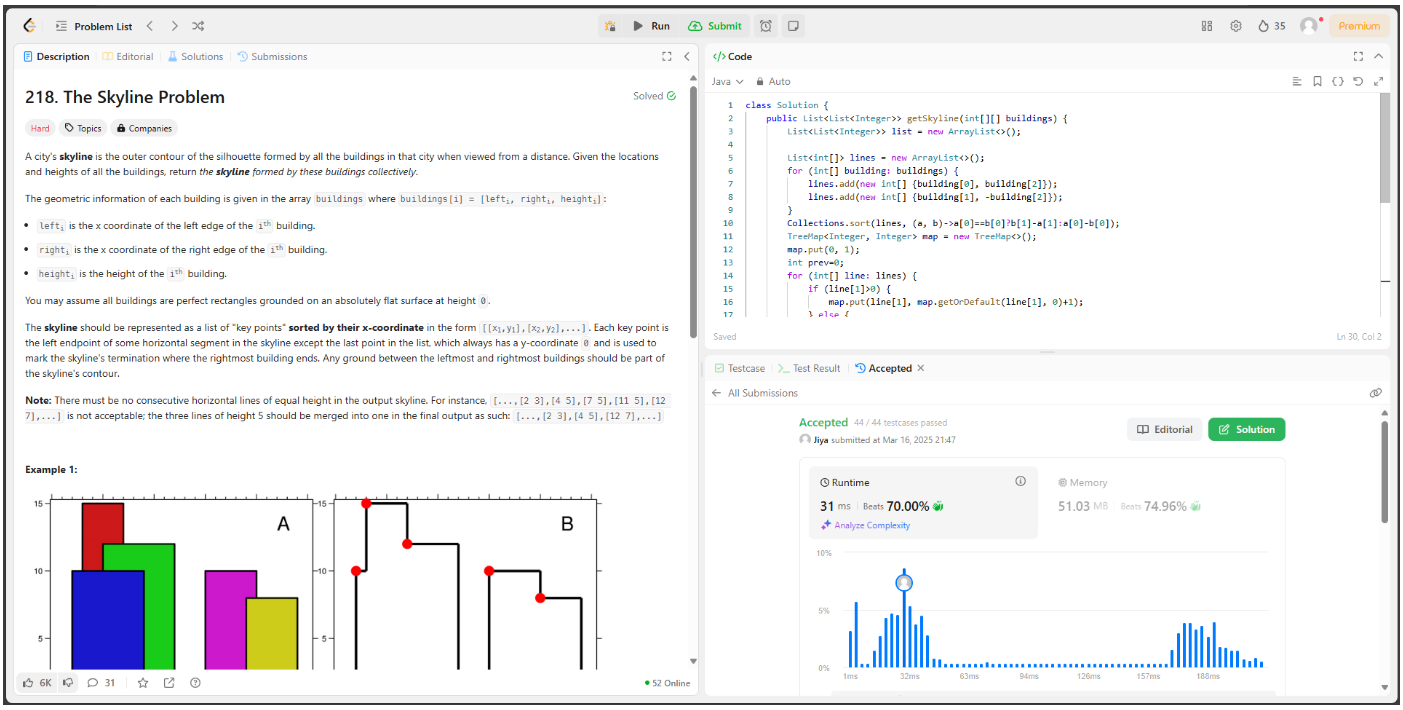
Memory
44.50 MB | Beats 47.85%

218. The Skyline Problem

https://leetcode.com/problems/the-skyline-problem/description/

```java
class Solution {
    public List<List<Integer>> getSkyline(int[][] buildings) {
        List<List<Integer>> list = new ArrayList<>();

        List<int[]> lines = new ArrayList<>();
        for (int[] building: buildings) {
            lines.add(new int[] {building[0], building[2]});
            lines.add(new int[] {building[1], -building[2]});
        }
        Collections.sort(lines, (a, b)->a[0]==b[0]?b[1]-a[1]:a[0]-b[0]);
        TreeMap<Integer, Integer> map = new TreeMap<>();
        map.put(0, 1);
        int prev=0;
        for (int[] line: lines) {
            if (line[1]>0) {
                map.put(line[1], map.getOrDefault(line[1], 0)+1);
            } else {
                int f = map.get(-line[1]);
                if (f==1) map.remove(-line[1]);
                else map.put(-line[1], f-1);
            }
            int curr = map.lastKey();
            if (curr!=prev) {
                list.add(Arrays.asList(line[0], curr));
                prev=curr;
            }
        }
        return list;
    }
}
```



218. The Skyline Problem

Solved

Hard · Topics · Companies

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] = [left_i, right_i, height_i]`:

- `left_i` is the x coordinate of the left edge of the `i^th` building.
- `right_i` is the x coordinate of the right edge of the `i^th` building.
- `height_i` 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 3],[4 5],[12 7],...]`

Example 1:

```java
class Solution {
    public List<List<Integer>> getSkyline(int[][] buildings) {
        List<List<Integer>> list = new ArrayList<>();

        List<int[]> lines = new ArrayList<>();
        for (int[] building: buildings) {
            lines.add(new int[] {building[0], building[2]});
            lines.add(new int[] {building[1], -building[2]});
        }
        Collections.sort(lines, (a, b)->a[0]==b[0]?b[1]-a[1]:a[0]-b[0]);
        TreeMap<Integer, Integer> map = new TreeMap<>();
        map.put(0, 1);
        int prev=0;
        for (int[] line: lines) {
            if (line[1]>0) {
                map.put(line[1], map.getOrDefault(line[1], 0)+1);
            } else {
```

Saved

Ln 30, Col 2

Accepted 44 / 44 testcases passed

Jiya submitted at Mar 16, 2025 21:47

Runtime

31 ms | Beats 70.00%

Analyze Complexity

Memory

51.03 MB | Beats 74.96%