

ASSIGNMENT-6

Name: Urvi Jindal

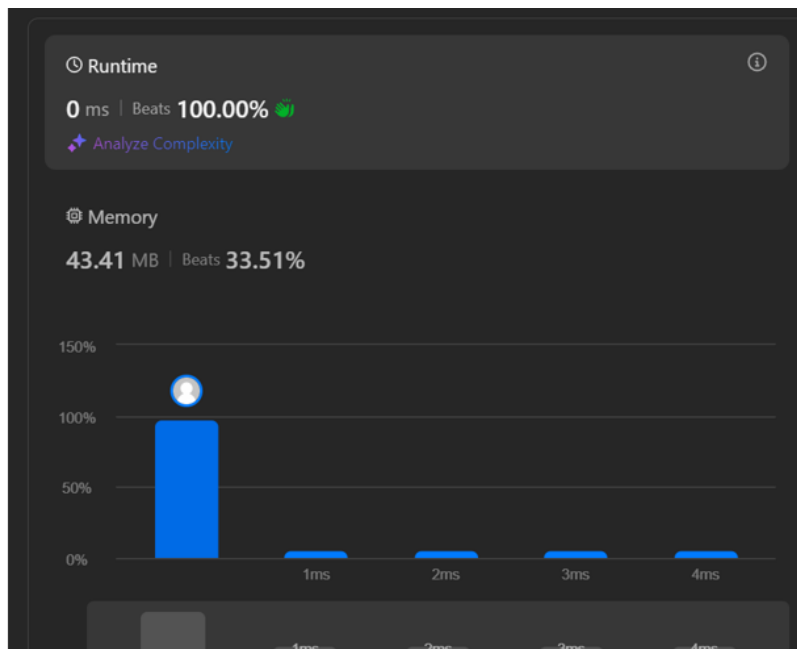
Section: FL_IOT-603/A

UID: 22BCS14860

108. Convert Sorted Array to Binary Search Tree

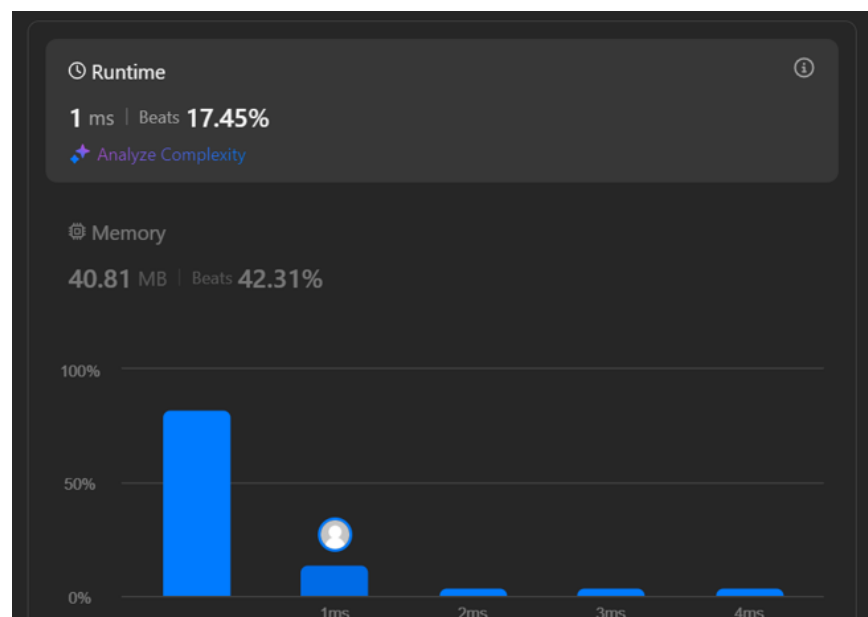
```
class Solution {
    public TreeNode sortedArrayToBST(int[] nums) {
        return buildBST(nums, 0, nums.length - 1);
    }

    private TreeNode buildBST(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 = buildBST(nums, left, mid - 1);
        root.right = buildBST(nums, mid + 1, right);
        return root;
    }
}
```



191. [Number of 1 Bits](#)

```
class Solution {  
    public int hammingWeight(int n) {  
        ArrayList<Integer> arr=new ArrayList<>();  
        int i=0;  
        while(n >0){  
            int x=n%2;  
            arr.add(x);  
            if(x==1){  
                i++;  
            }  
            n=n/2;  
        }  
        return i;  
    }  
}
```

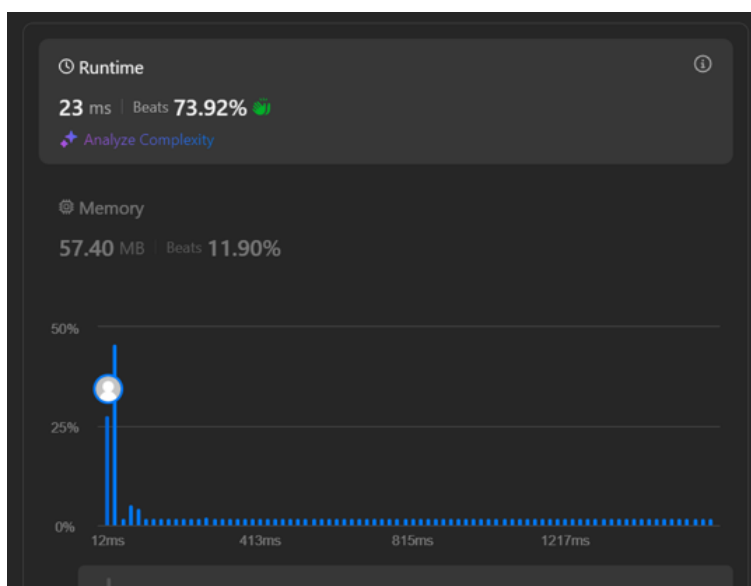


912.Sort an Array

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

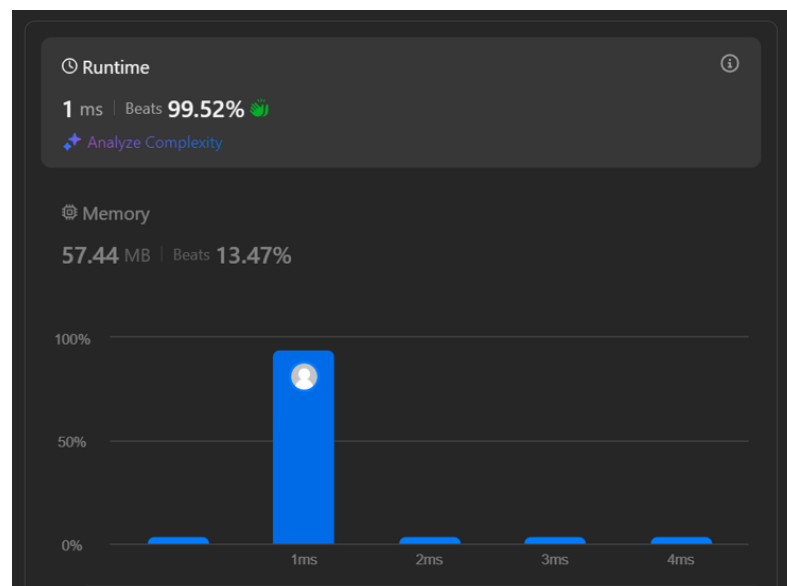
    private void mergeSort(int[] nums, int left, int right) {
        if (left >= right) return;
        int mid = left + (right - left) / 2;
        mergeSort(nums, left, mid);
        mergeSort(nums, mid + 1, right);
        merge(nums, left, mid, right);
    }

    private void merge(int[] nums, int left, int mid, int right) {
        int[] temp = new int[right - left + 1];
        int i = left, j = mid + 1, k = 0;
        while (i <= mid && j <= right) {
            if (nums[i] <= nums[j]) temp[k++] = nums[i++];
            else temp[k++] = nums[j++];
        }
        while (i <= mid) temp[k++] = nums[i++];
        while (j <= right) temp[k++] = nums[j++];
        System.arraycopy(temp, 0, nums, left, temp.length);
    }
}
```



53. Maximum Subarray

```
class Solution {  
    public int maxSubArray(int[] nums) {  
        int maxSum = nums[0], currentSum = nums[0];  
        for (int i = 1; i < nums.length; i++) {  
            currentSum = Math.max(nums[i], currentSum + nums[i]);  
            maxSum = Math.max(maxSum, currentSum);  
        }  
        return maxSum;  
    }  
}
```



932.Beautiful Array

```
class Solution {
public int[] beautifulArray(int n) {
    int[] arr = new int[n];
    for (int i = 0; i < n; ++i)
        arr[i] = i + 1;
    divide(arr, 0, n - 1, 1);
    return arr;
}

private void divide(int[] arr, int l, int r, int mask) {
    if (l >= r)
        return;
    final int m = partition(arr, l, r, mask);
    divide(arr, l, m, mask << 1);
    divide(arr, m + 1, r, mask << 1);
}

private int partition(int[] arr, int l, int r, int mask) {
    int nextSwapped = l;
    for (int i = l; i <= r; ++i)
        if ((arr[i] & mask) > 0)
            swap(arr, i, nextSwapped++);
    return nextSwapped - 1;
}

private void swap(int[] arr, int i, int j) {
    final int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}
```

