# ASSIGNMENT 5

# NAME – ANSH KUMAR SINGH

# UID – 22BCS11057

# SECTION- 608/B

# Merge Sorted Array:

import java.util.Arrays;

class Solution {

public void merge(int[] nums1, int m, int[] nums2, int n) { int i = m - 1;

int j = n - 1;

int k = m + n - 1;

while (i >= 0 && j >= 0) {

if (nums1[i] > nums2[j]) {

nums1[k--] = nums1[i--];

} else {

nums1[k--] = nums2[j--];

}

}

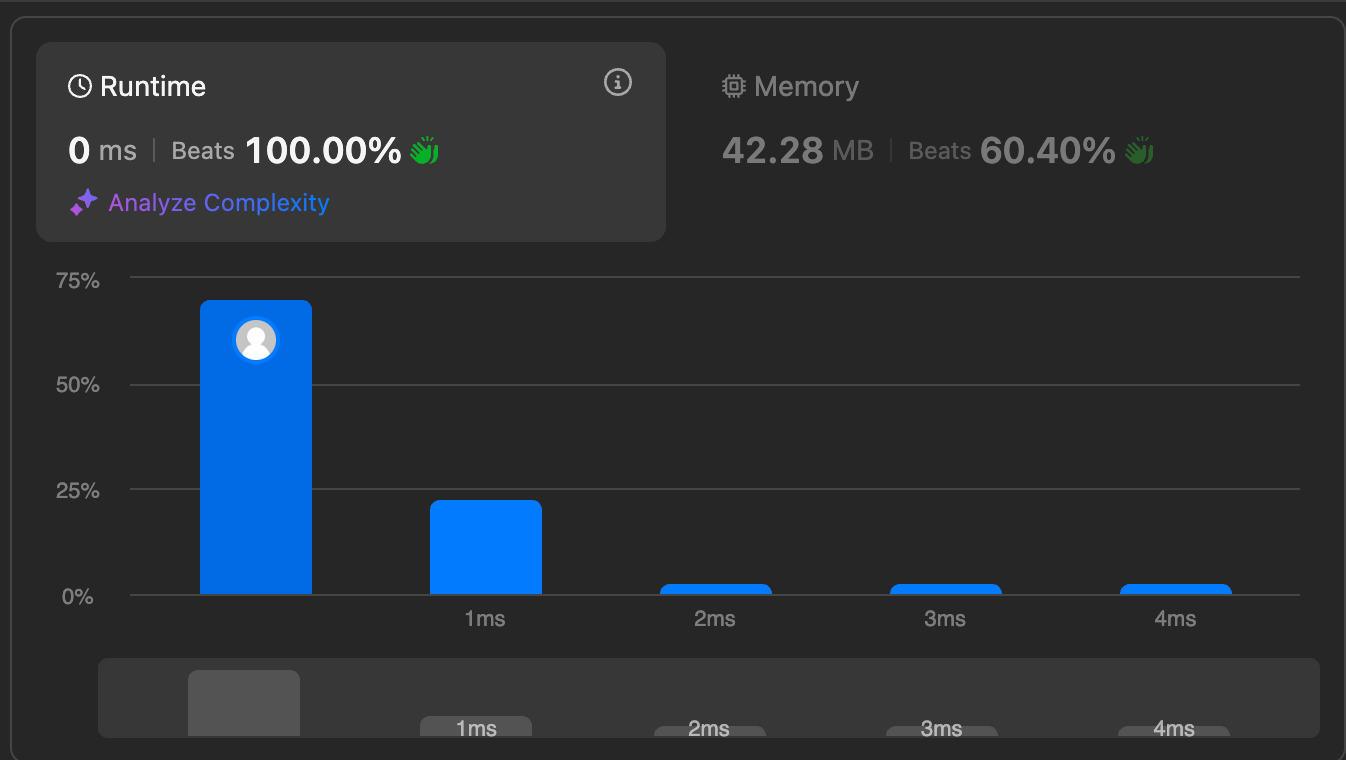
while (j >= 0) {

nums1[k--] = nums2[j--];

}

}

}



# First Bad Version:

public class Solution extends VersionControl { public int firstBadVersion(int n) {

int left = 1, right = n;

while (left < right) {

int mid = left + (right - left) / 2;

if (isBadVersion(mid)) { right = mid;

} else {

left = mid + 1;

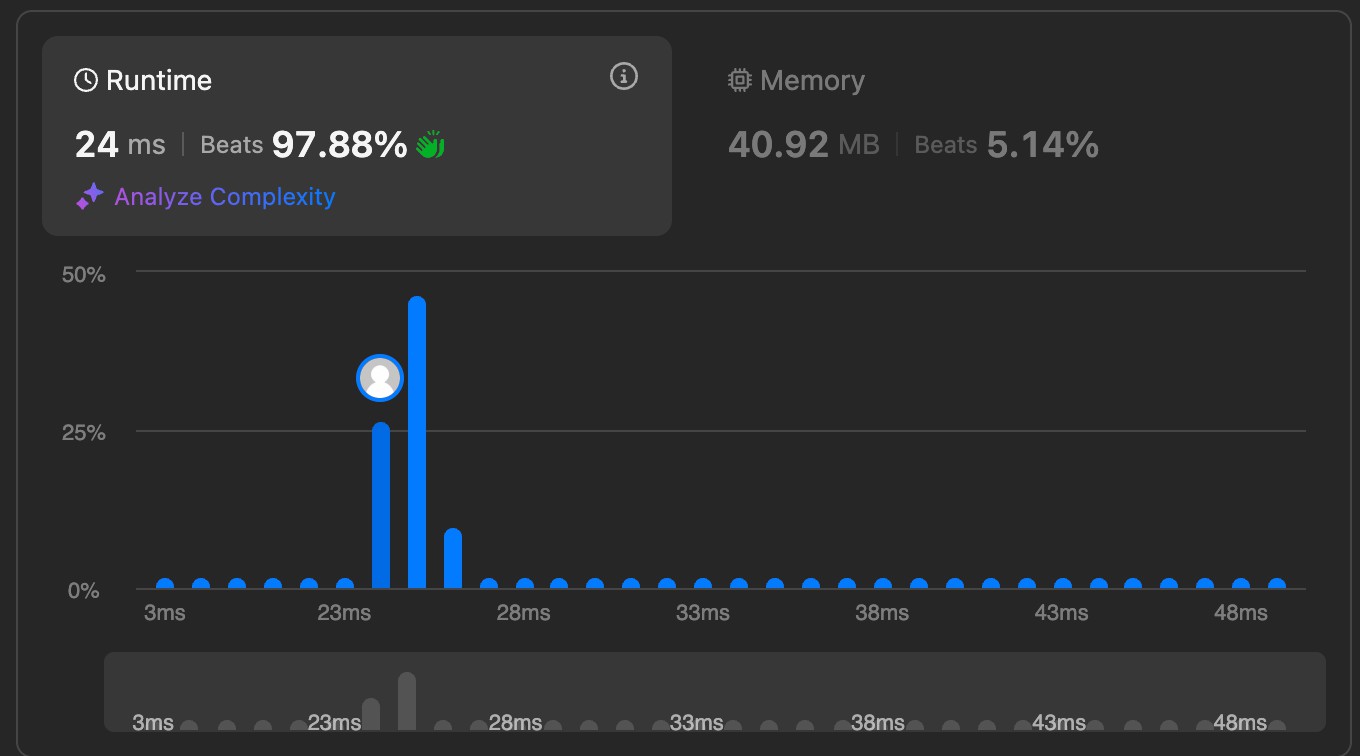
}

}

return left;

}

}



# Sort Colors:

class Solution {

public void sortColors(int[] nums) {

int low = 0, mid = 0, high = nums.length - 1;

while (mid <= high) {

if (nums[mid] == 0) { swap(nums, low, mid); low++;

mid++;

} else if (nums[mid] == 1) { mid++;

} else {

swap(nums, mid, high); high--;

}

}

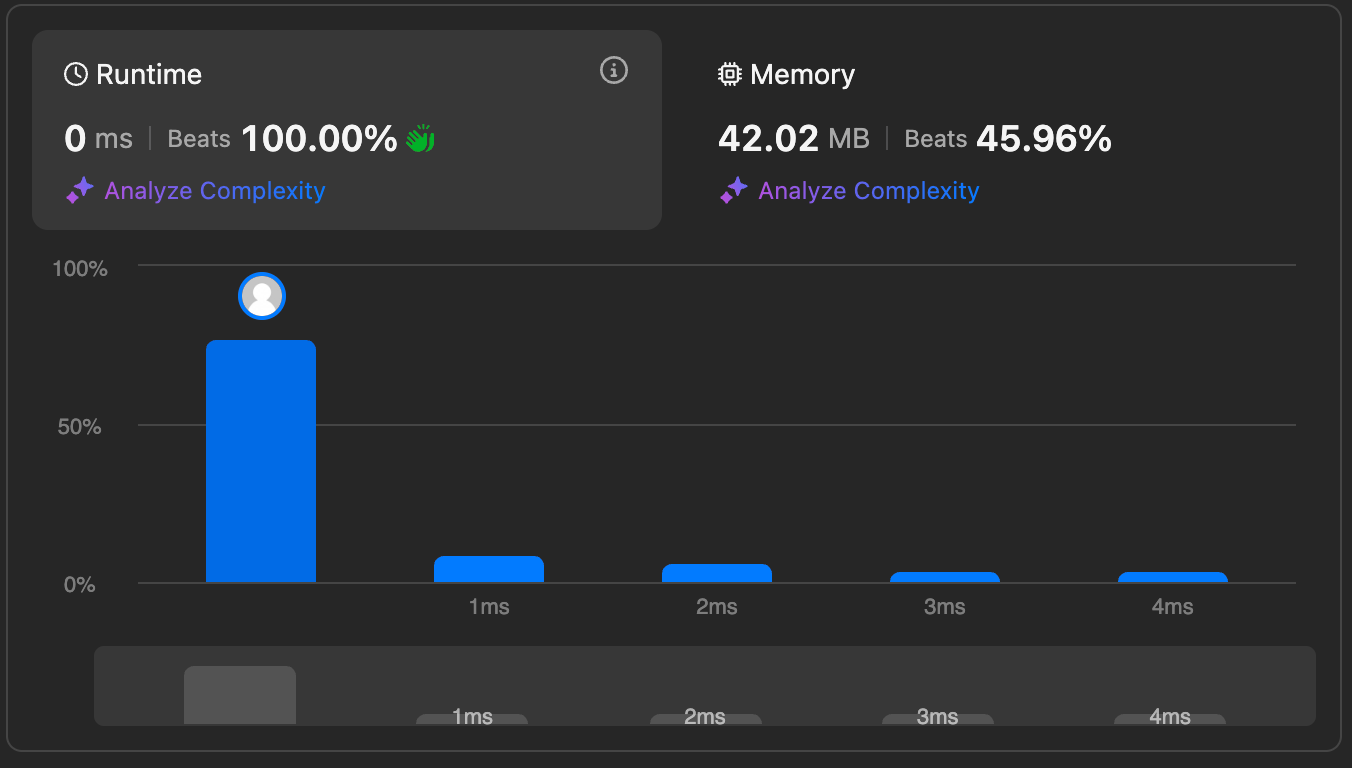
}

private void swap(int[] nums, int i, int j) { int temp = nums[i];

nums[i] = nums[j]; nums[j] = temp;

}

}



# Find Peak Element:

class Solution {

public int findPeakElement(int[] nums) { int left = 0, right = nums.length - 1;

while (left < right) {

int mid = left + (right - left) / 2;

if (nums[mid] > nums[mid + 1]) { right = mid;

} else {

left = mid + 1;

}

}

return left;

}

}



# Median of Two Sorted Arrays:

import java.util.Arrays;

class Solution {

public double findMedianSortedArrays(int[] nums1, int[] nums2) { int m = nums1.length;

int n = nums2.length;

int[] result = new int[m + n];

System.arraycopy(nums1, 0, result, 0, m); System.arraycopy(nums2, 0, result, m, n);

Arrays.sort(result);

int mid = result.length; if (mid % 2 == 0) {

return ((double)result[mid / 2 - 1] + result[mid / 2]) / 2;

} else {

return result[mid / 2];

}

}

}

