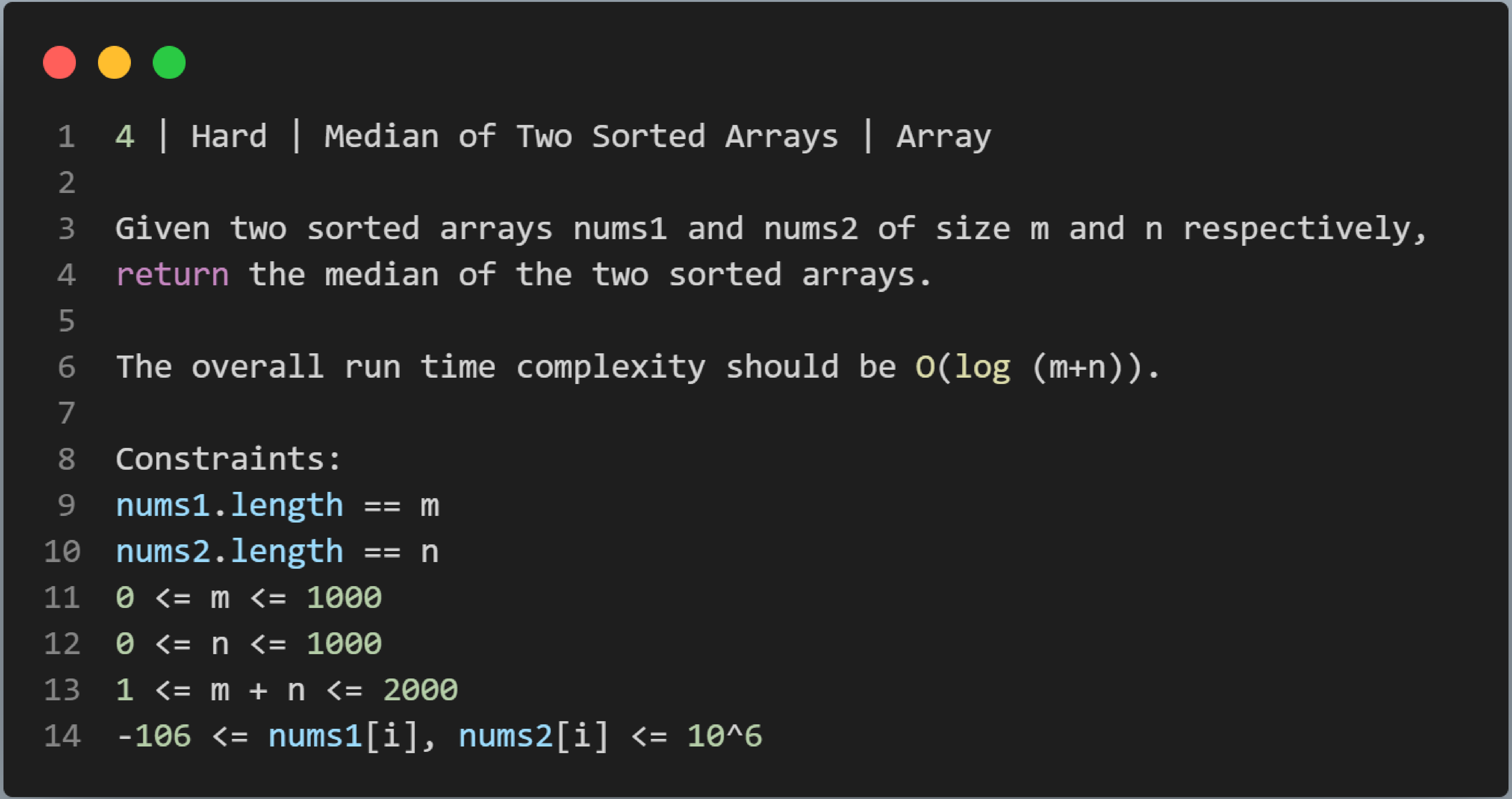


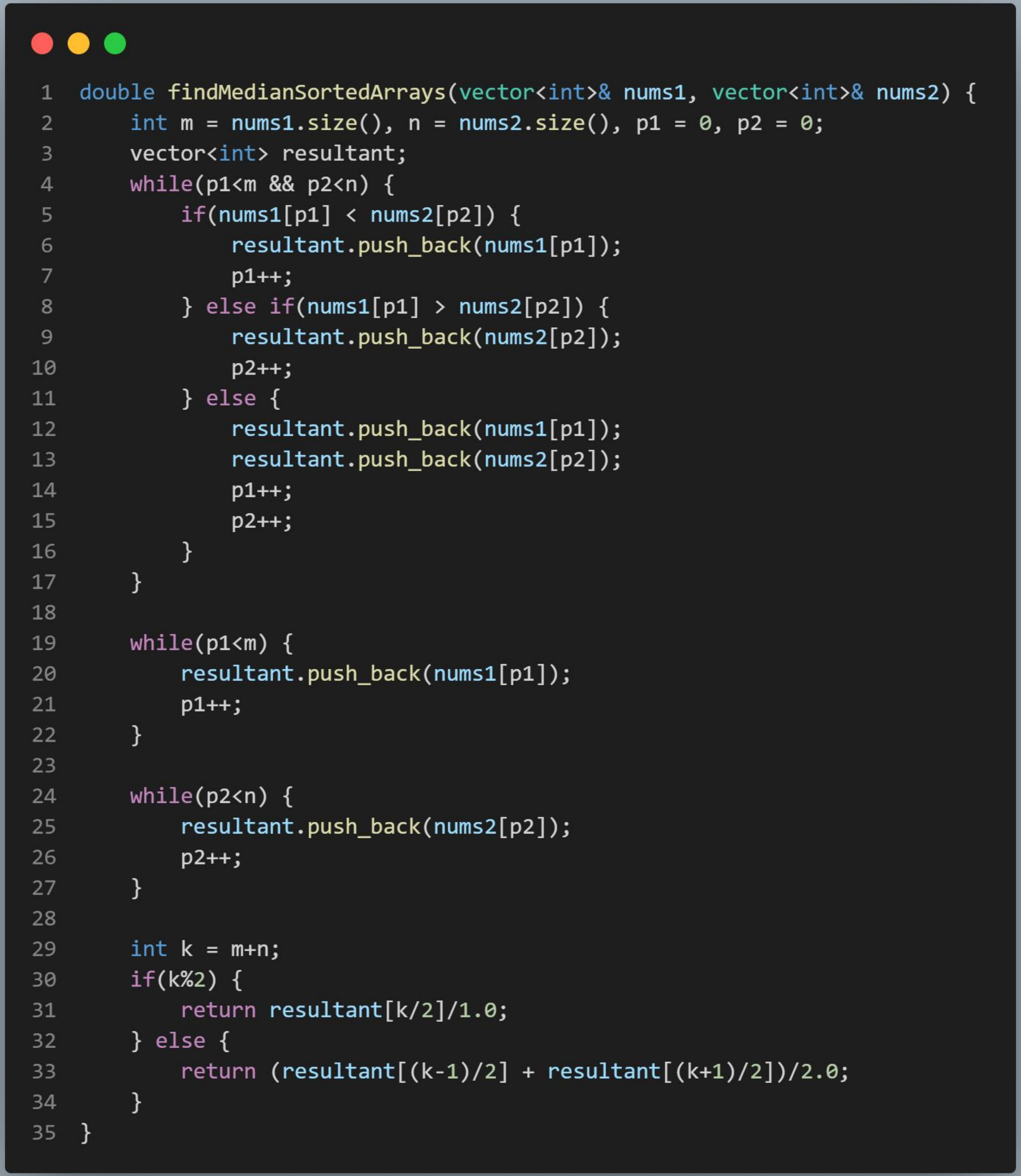
```
4 | Hard | Median of Two Sorted Arrays | Array

Given two sorted arrays nums1 and nums2 of size m and n respectively,
return the median of the two sorted arrays.

The overall run time complexity should be O(log (m+n)).

Constraints:
nums1.length == m
nums2.length == n
0 <= m <= 1000
0 <= n <= 1000
1 <= m + n <= 2000
-106 <= nums1[i], nums2[i] <= 10^6
```

Example 1:

```
Input: nums1 = [1,3], nums2 = [2]
Output: 2.00000
Explanation: merged array = [1,2,3] and median is 2.
```

Example 2:

```
Input: nums1 = [1,2], nums2 = [3,4]
Output: 2.50000
Explanation: merged array = [1,2,3,4] and median is (2 + 3) / 2 = 2.5.
```



```cpp
double findMedianSortedArrays(vector<int>& nums1, vector<int>& nums2) {
    int m = nums1.size(), n = nums2.size(), p1 = 0, p2 = 0;
    vector<int> resultant;
    while(p1<m && p2<n) {
        if(nums1[p1] < nums2[p2]) {
            resultant.push_back(nums1[p1]);
            p1++;
        } else if(nums1[p1] > nums2[p2]) {
            resultant.push_back(nums2[p2]);
            p2++;
        } else {
            resultant.push_back(nums1[p1]);
            resultant.push_back(nums2[p2]);
            p1++;
            p2++;
        }
    }

    while(p1<m) {
        resultant.push_back(nums1[p1]);
        p1++;
    }

    while(p2<n) {
        resultant.push_back(nums2[p2]);
        p2++;
    }

    int k = m+n;
    if(k%2) {
        return resultant[k/2]/1.0;
    } else {
        return (resultant[(k-1)/2] + resultant[(k+1)/2])/2.0;
    }
}
```

#100daysofDSA

/rvislive

Rakesh Vishwakarma