(/) Explore(/explore/) Problems(/problemset/all/) Interview New (/contest/) Contest Discuss(/discuss/) Store ☆ Premium (/subscribe?ref=nb_npl) 0radius-subarray-averages/)

Majority Element (/problems/majority-element/)

Submission Detail

45 / 45 test cases passed.

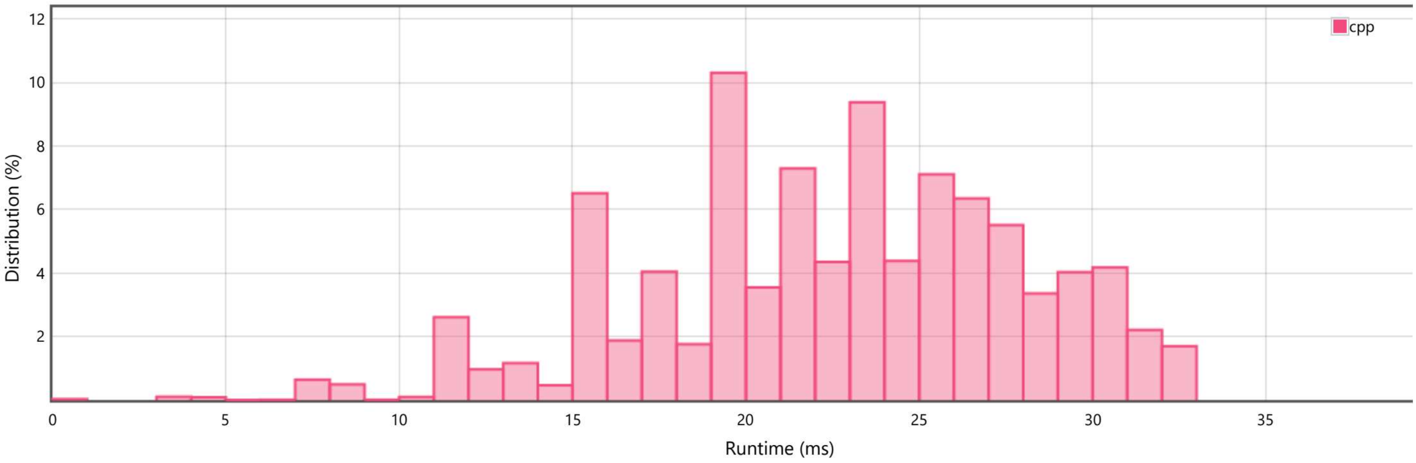
Status: Accepted

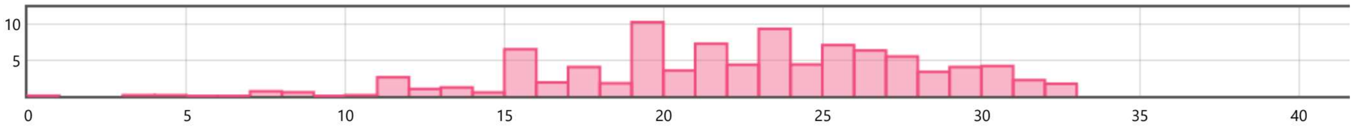
Runtime: **211 ms**

Memory Usage: **82.2 MB**

Submitted: **8 hours, 13 minutes ago**

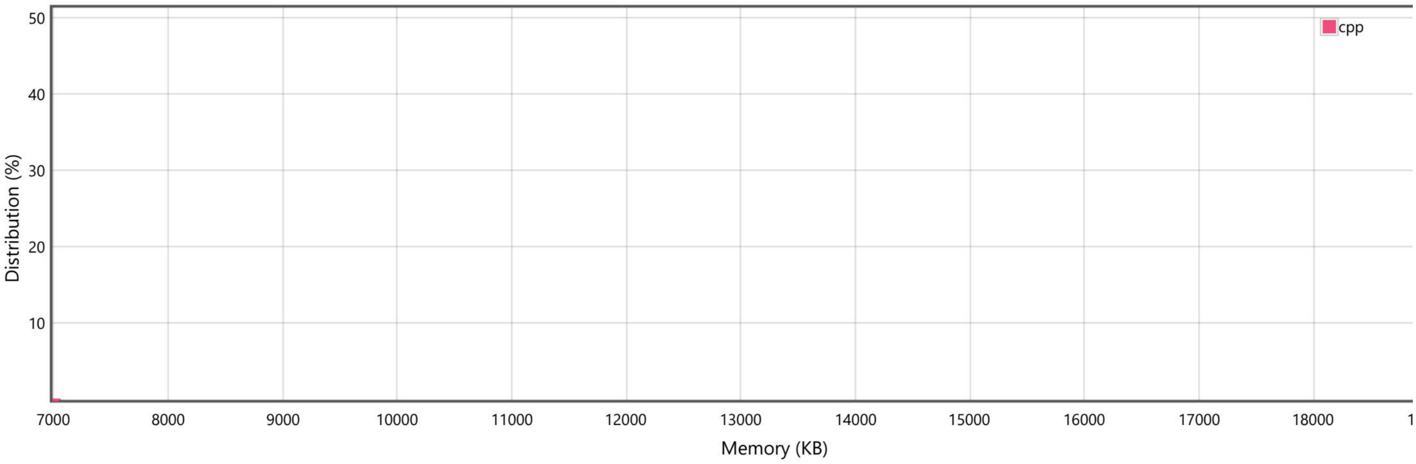
Accepted Solutions Runtime Distribution

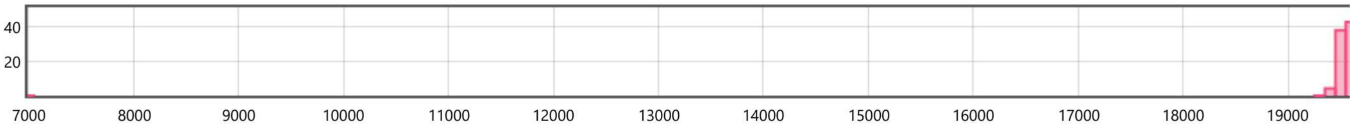




Zoom area by dragging across this chart

Accepted Solutions Memory Distribution





Zoom area by dragging across this chart

Invite friends to challenge **Majority Element**

Submitted Code: 8 hours, 13 minutes ago

Language: cpp

Edit Code

```
class Solution {
public:

    void merge(vector<int>& nums, int l, int mid, int r){
        int p1 = mid - l + 1;
        int p2 = r - mid;

        vector<int> A1(p1);
        vector<int> A2(p2);

        for (int i = 0; i < p1; i++)
            A1[i] = nums[l + i];

        for (int j = 0; j < p2; j++)
            A2[j] = nums[mid + 1 + j];

        int iterA1, iterA2, iterNums;
        iterA1 = 0;
        iterA2 = 0;
        iterNums = l;

        while (iterA1 < p1 && iterA2 < p2) {
            if (A1[iterA1] <= A2[iterA2]) {
                nums[iterNums] = A1[iterA1];
                iterA1++;
            } else {
                nums[iterNums] = A2[iterA2];
                iterA2++;
            }
            iterNums++;
        }

        while (iterA1 < p1) {
            nums[iterNums] = A1[iterA1];
            iterA1++;
            iterNums++;
        }

        while (iterA2 < p2) {
            nums[iterNums] = A2[iterA2];
            iterA2++;
            iterNums++;
        }

    }

    void mergeSort(vector<int>& nums, int l, int r){
        if(l<r){
            int mid = (l + (r-1)) / 2;

            mergeSort(nums, l, mid);
            mergeSort(nums, mid + 1, r);

            merge(nums, l, mid, r);
        }

    }

    int majorityElement(vector<int>& nums) {
        int count = 0;
        int max;
        int max_count = 0;

        mergeSort(nums, 0, nums.size() - 1);

        max = nums[0];
        for (int i = 0; i < nums.size(); ++i) {
            count++;
            if(i!= nums.size()-1){
                if (nums[i] != nums[i + 1]) {
                    if (count > max_count) {
                        max_count = count;
                        max = nums[i];
                        count = 0;
                    }
                }
            }else if(i == nums.size()-1){
                if (count > max_count) {
                    max_count = count;
                    max = nums[i];
                    count = 0;
                }
            }

        }

        return max;
    }
```

Back to problem (/problems/majority-element/)