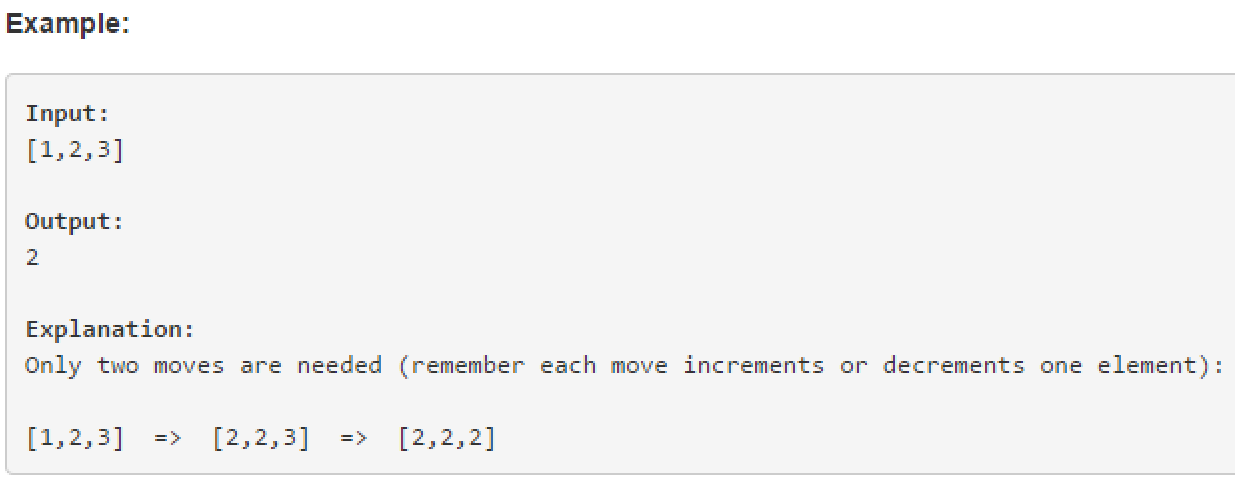
462. Minimum Moves to Equal Array Elements II

Add to List
QuestionEditorial Solution
 My Submissions

- Total Accepted: **4317**
- Total Submissions: **8589**
- Difficulty: **Medium**
- Contributors: **andrew56**

Given a **non-empty** integer array, find the minimum number of moves required to make all array elements equal, where a move is incrementing a selected element by 1 or decrementing a selected element by 1.

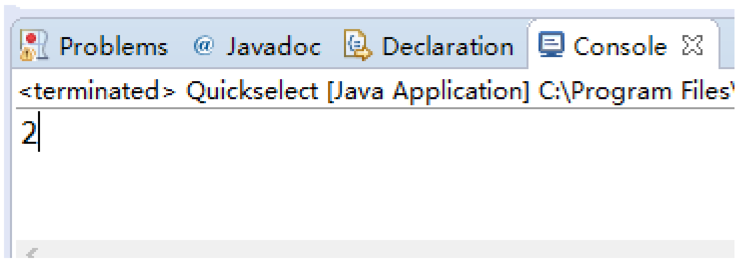
You may assume the array's length is at most 10,000.



Example:

```
Input:
[1,2,3]

Output:
2

Explanation:
Only two moves are needed (remember each move increments or decrements one element):

[1,2,3]  =>  [2,2,3]  =>  [2,2,2]
```

```
public class Quickselect {

        private static int partition(int[] nums,int l,int r)
        {
                int len = r-l+1;
                Random rd = new Random();
                int index = rd.nextInt(len) + l;
                swap(nums,index,r);
                int pivot = nums[r];

                int less = l;
                for(int i=l;i<r;i++)
                {
                        if(nums[i]<=pivot)
                        {
                                swap(nums,i,less++);
                        }
        }
        swap(nums,less,r);
        return less;
}

private static int partition2(int[] nums,int l,int r)
{
        int x = nums[l];
        int j=r+1;
        int i=l;
        while(true)
        {
                while(nums[++i]<x && i<r);
                while(nums[--j]>x && j>l);
                if(i>=j) break;
                swap(nums, i, j);
        }
        nums[r] = nums[j];
        nums[j] = x;
        return j;
}

private static int selectK(int[] nums,int k,int l,int r)
{
        if(l>=r) return l;
        int pivot = partition2(nums, l, r);
        if(pivot-l+1==k) return pivot;
        if(pivot-l+1<k)
                return selectK(nums, k-(pivot-l+1), pivot+1, r);
        else
                return selectK(nums, k, l, pivot-1);
}

private static int minMoves2(int[] nums) {

        if(nums.length==0) return 0;
        int Count=0;
        int median = nums.length/2;
        int mid = nums[selectK(nums,median+1,0,nums.length-1)];
        for (int a:nums)
        {
                if(a>mid)
                {
                        Count += a-mid;
                }else{
                        Count += mid-a;
                }
        }
        return Count;
}
private static void swap(int[] nums, int a,int b)
{
        int t = nums[a];
        nums[a] = nums[b];
        nums[b] = t;
}

public static void main(String[] args) {
		// TODO Auto-generated method stub
		int[] nums = {1,3,2};
		int ans = minMoves2(nums);
		//swap(nums,1,2);
		System.out.print(ans);
	}
}
```



```
<terminated> Quickselect [Java Application] C:\Program Files
2
```