169. Majority Element

Easy 9249 331 Add to List Share

Given an array `nums` of size `n`, return *the majority element*.

The majority element is the element that appears more than `⌊n / 2⌋` times. You may assume that the majority element always exists in the array.

Example 1:

```
Input: nums = [3,2,3]
Output: 3
```

Example 2:

```
Input: nums = [2,2,1,1,1,2,2]
Output: 2
```

Constraints:

- `n == nums.length`
- `1 <= n <= 5 * 10^4`
- `-10^9 <= nums[i] <= 10^9`

N:7

Majority Element is on n/2 index

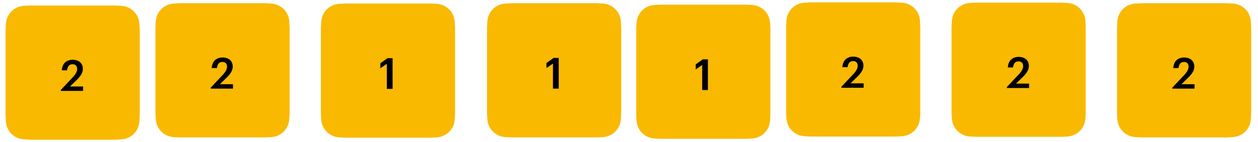
Time Complexity : O(nlogn)

Space Complexity : O(1)

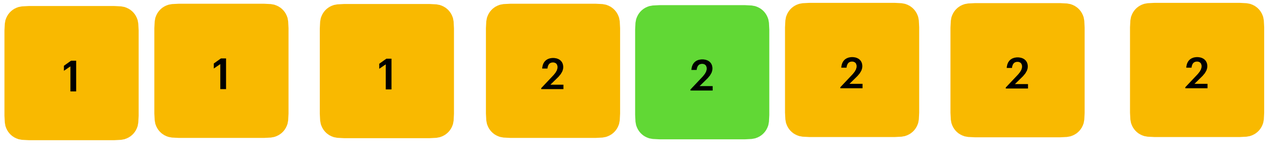
Sorting

Merge Sort O(n)

Quick Sort O(1)



N:8



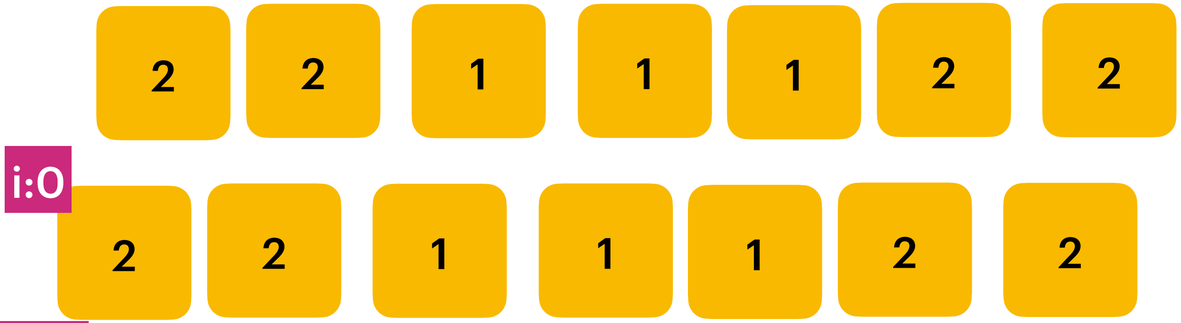
Majority Element is on n/2 index

Majority Element repeats > n/2 times

N:7

Base Check

Vote : 0
majorityElement :



i:0

Vote : 1
majorityElement : 2

i:1

Vote : 2
majorityElement : 2

i:2

Vote : 1
majorityElement : 2

Majority Element repeats > n/2 times

N:7

Base Check

Vote : 0
majorityElement :

Vote : 1
majorityElement : 2

i:3

Vote : 0
majorityElement : 2

Time Complexity : O(n)
Space Complexity : O(1)

i:4

Vote : 1
majorityElement : 1

i:5

Vote : 0
majorityElement : 1

i:6

i:7

Vote : 1
majorityElement : 2

4. Median of Two Sorted Arrays

Hard 15850 1951 Add to List Share

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))`.

Example 1:

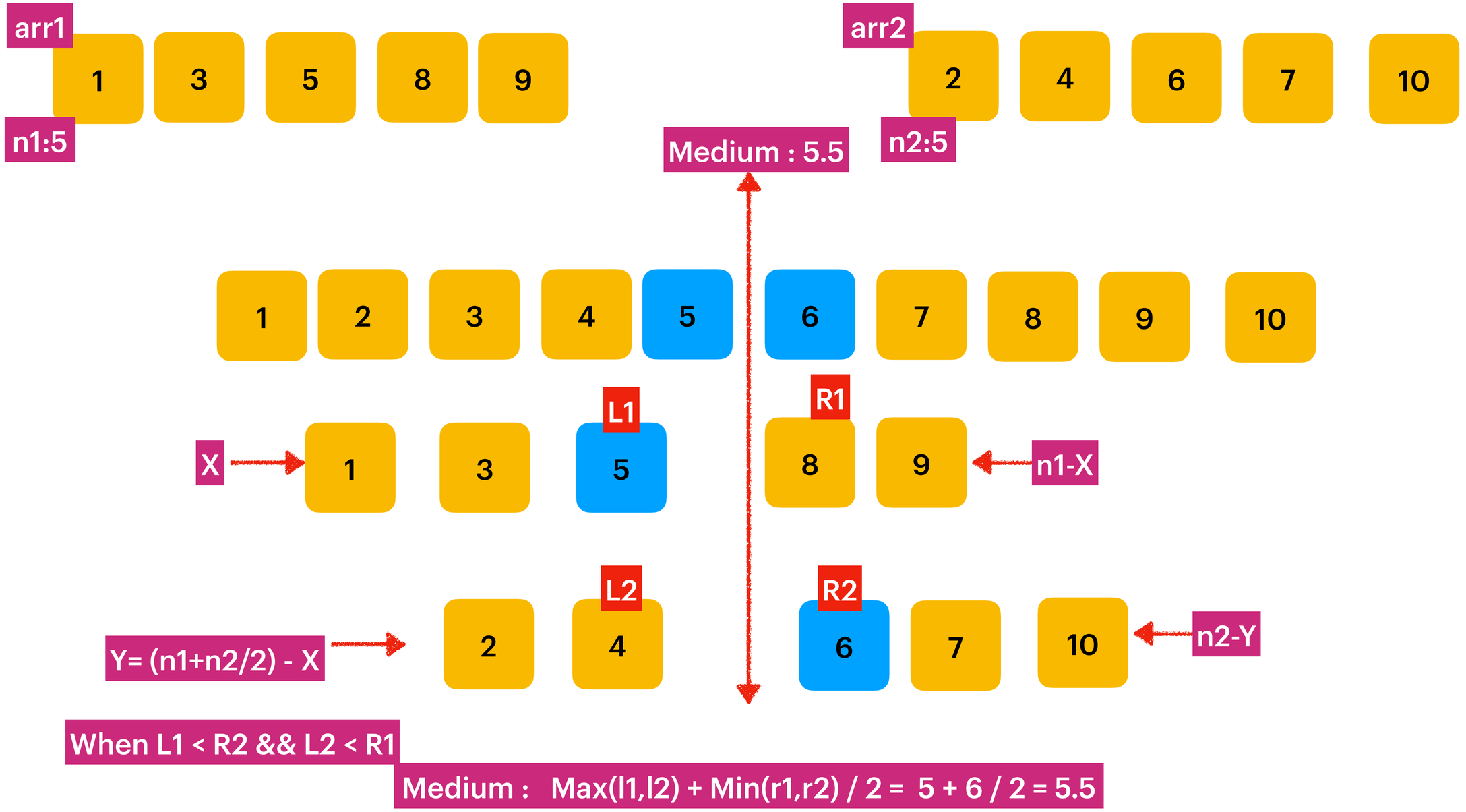
```
Input: nums1 = [1,3], nums2 = [2]
Output: 2.00000
Explanation: merged array = [1,2,3] and median is 2.
```

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.
```

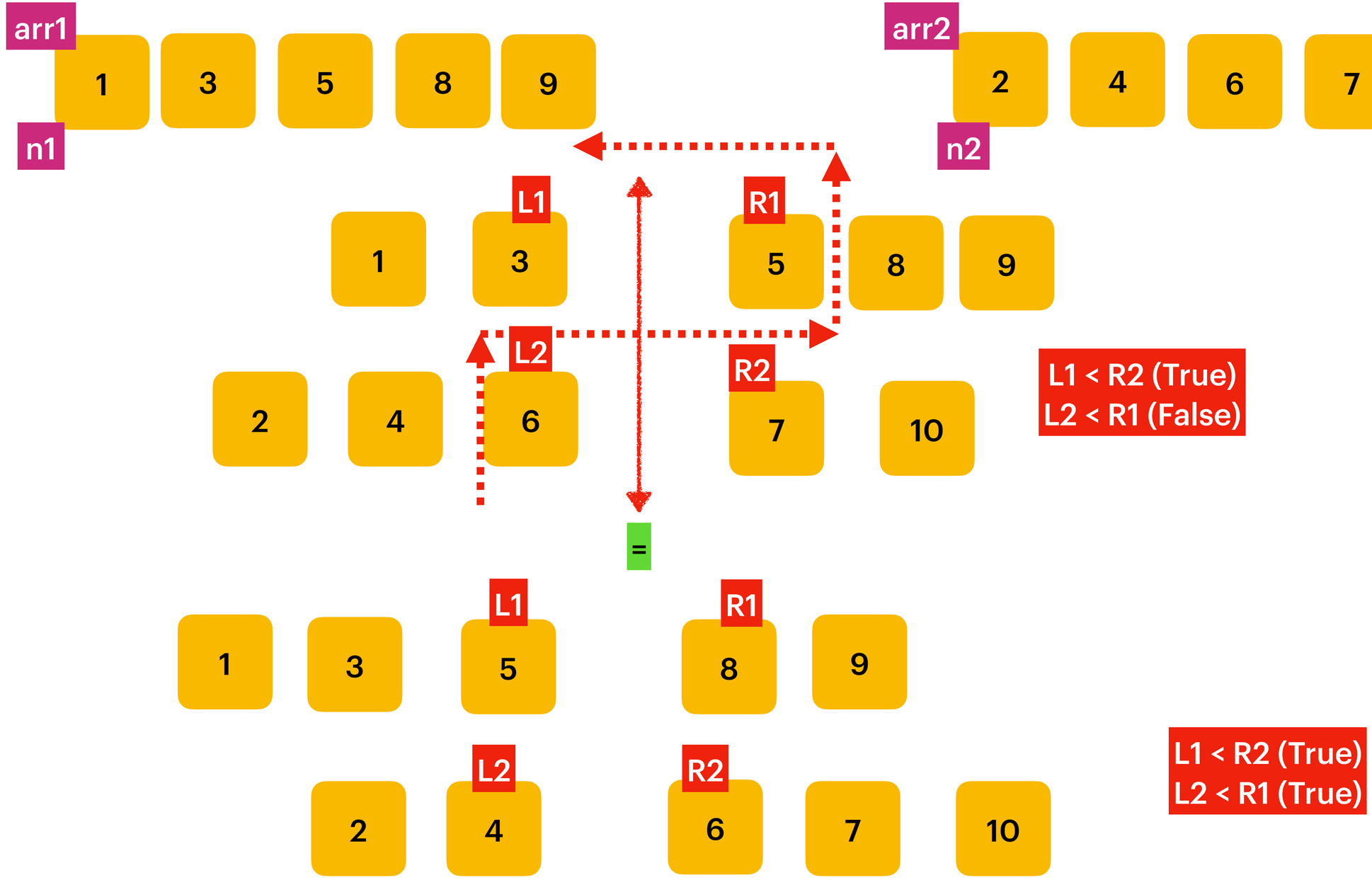
Constraints:

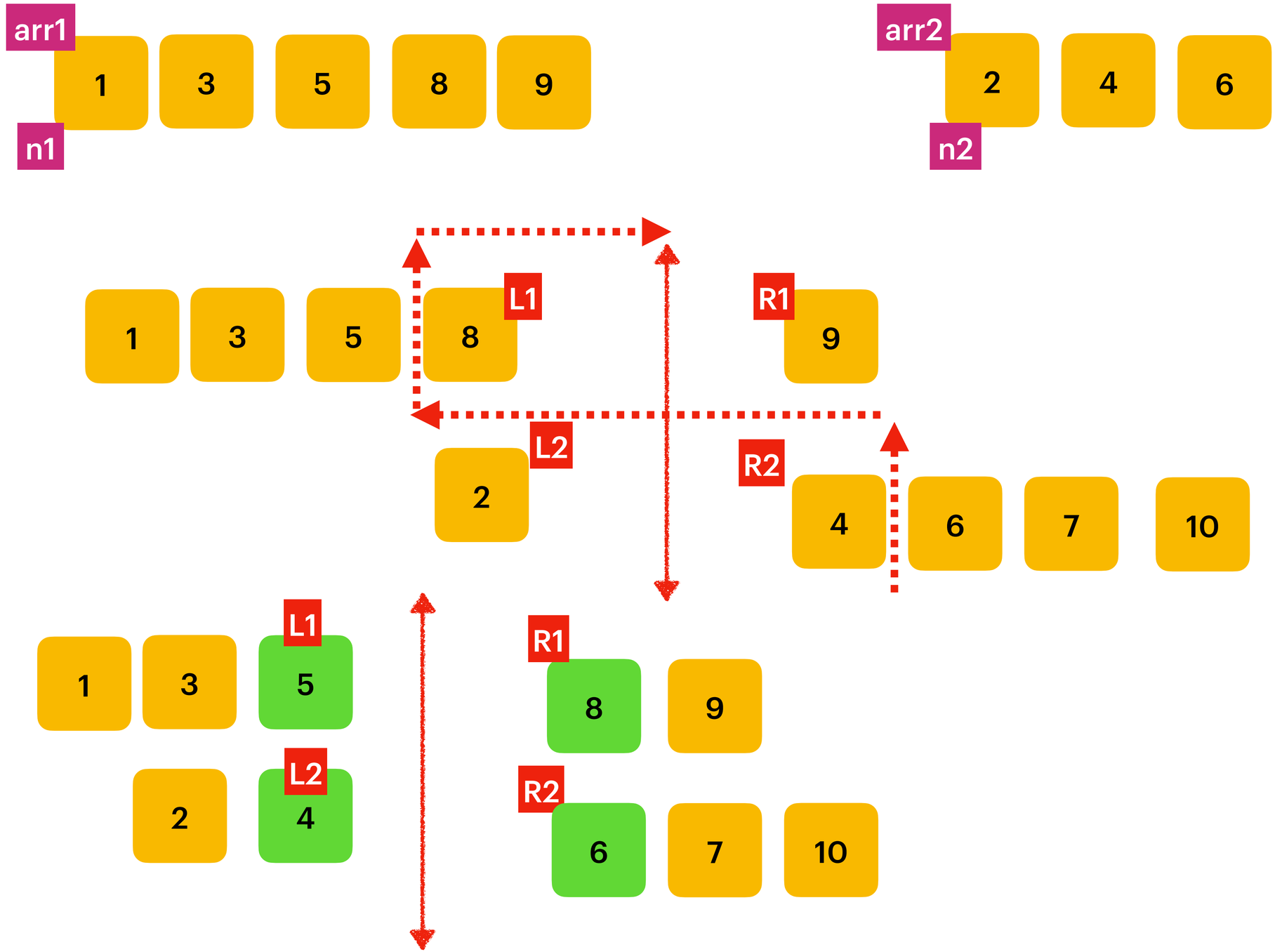
- `nums1.length == m`
- `nums2.length == n`
- `0 <= m <= 1000`
- `0 <= n <= 1000`
- `1 <= m + n <= 2000`
- `-10^6 <= nums1[i], nums2[i] <= 10^6`



When L1 < R2 && L2 < R1

Medium : Max(l1,l2) + Min(r1,r2) / 2 = 5 + 6 / 2 = 5.5


arr1
1
3
5
8
9
n1
arr2
2
4
6
7
10
n2
L1
1
3
R1
5
8
9
L2
2
4
6
R2
7
10
L1 < R2 (True)
L2 < R1 (False)
=
L1
1
3
5
R1
8
9
L2
2
4
R2
6
7
10
L1 < R2 (True)
L2 < R1 (True)


arr1
1
3
5
8
9
n1
arr2
2
4
6
7
10
n2
L1
R1
L2
R2
L1
R1
L2
R2

Practice

Quick Sort

Selection Sort

Insertion Sort