Sort

1. Find Intersection

class Solution {

public int[] intersection(int[] nums1, int[] nums2) {

HashSet<Integer> set1 = new HashSet<Integer>();

HashSet<Integer> set2 = new HashSet<>();

for (int i = 0; i < nums1.length; i++) {

set1.add(nums1[i]);

}

for (int i = 0; i < nums2.length; i++) {

set2.add(nums2[i]);

}

set1.retainAll(set2);

int index = 0;

int[] res = new int[set1.size()];

for (int i : set1) {

res[index] = i;

index++;

}

return res;

}

}

1. Find nth Smallest of an unsorted array

class Solution {

public int findKthLargest(int[] nums, int k) {

PriorityQueue<Integer> heap = new PriorityQueue<>((a,b) -> b - a);

for (int i = 0; i < nums.length; i++) {

heap.add(nums[i]);

if (heap.size() > k) {

heap.poll();

}

}

return heap.poll();

}

}

1. Sort colors

class Solution {

public void sortColors(int[] nums) {

if (nums == null || nums.length < 2) return;

int zero = 0;

int two = nums.length - 1;

int cur = 0;

while (cur <= two) {

if (nums[cur] == 0) {

int temp = nums[zero];

nums[zero++] = nums[cur];

nums[cur++] = temp;

} else if (nums[cur] == 2) {

int temp = nums[cur];

nums[cur] = nums[two];

nums[two--] = temp;

} else cur++;

}

}

}

class Solution {

public List<Integer> pancakeSort(int[] A) {

List<Integer> out = new ArrayList<>();

boolean active = false;

int target = A.length;

int len = A.length-1;

int left = 0;

int right = 0;

while (len > 0) {

int i = 0;

while (I < A.length) {

if (A[I] == target && I < len) {

target--;

active = true;

break;

} else if (A[i] == target && I == len){

target--;

len--;

active = false;

break;

}else {

active = false;

i++;

}

}

if(active&&len>1) {

out.add(i+1);

left = 0;

right = i;

while(left<right) {

int temp = A[left];

A[left] = A[right];

A[right] = temp;

left++;

right--;

}

left = 0;

right = len;

out.add(len+1);

while(left <right) {

int temp = A[left];

A[left] = A[right];

A[right] = temp;

left++;

right--;

}

len--;

}else if(active&&len<=1){

left = 0;

right = len;

out.add(len+1);

while(left <right) {

int temp = A[left];

A[left] = A[right];

A[right] = temp;

left++;

right--;

}

len--;

}

}

return out;

}

}