Data Structures and Algo in Java - Day 14

Basically I did a revision today . I have covered all the tough concepts from the start and understood the problem thoroughly . I even written the pseudo code in paper without referring the actual code . here are the questions and the answers are provided below

import java.util.\*;

/\*

Going to Revise This Weeks Completed Sums..

1. Merge Sort

2. Quick Sort

3. Hashing - Find the Most Frequency of an Element in an array

4. Second Largest Element in an Array

5. Remove Duplicates

6. Left rotate by 1

7. Right rotate by D

8. Move Zeros to the end

9. Find union

10. Find Intersection

11. Missing Number

12. Consecutive ones

13. Find which number appear once where other numbers appear twice

14. Find the Longest Subarray with sum K

15. Find the Longest Subarray with sum K (both positives and negatives)

\*/

public class week1{

public static void main(String[] args)

{

//1-merge sort

int arr1 [] = {5,4,2,8,9,10,7,1,6,3};

int arr2[] = {0,1,1,2,0,0,3,2,1,1,0,0};

int arr3[] = {7,4,23,16,1,0,2};

int arr4[] = {0,0,1,1,1,1,2,2,3,3,3,3,3,3,4,4,4,4,4};

int arr5[] = {1,1,1,1,2,2,3,3,3,0,0,0,0,0,0,3,3,3,4,4,4,4,4};

int arr6[] = {1,2,3,4,5,7,8};

int arr7[] = {1,0,1,1,0,0,0,1,1,0,0,1,1,1,1,1,1,0,0};

int arr8[] = {1,1,2,2,3,3,4,4,5,6,6,7,7};

int arr9[] = {1, 2, 3, 1, 1, 1, 1,1, 4, 2, 3};

int k=7;

// int low = 0;

// int high = n-1;

// quicksort(arr1,low,high);

// System.out.println(Arrays.toString(arr1));

// freq(arr2,n);

// seclar(arr3);

// duplic(arr4);

// leftby1(arr3);

// rightbydopt(arr3);

// leftbydopt(arr3);

// moveZeros(arr5);

// unionop(arr1,arr2);

// interop(arr1,arr2);

// misnumopt(arr6);

// consecones(arr7);

// singleOpt(arr8);

// longestsubarraykopt(arr9,k);

}

public static void mergesort(int arr [], int low, int high)

{

if(low==high)

{

return;

}

int middleIndex = (low+high)/2;

mergesort(arr,low,middleIndex);//5

mergesort(arr,middleIndex+1,high);//4

merge(arr,low,middleIndex,high);//0,0,1

}

public static void merge(int arr[],int low, int mid, int high)//0,0,1

{

ArrayList<Integer> temp = new ArrayList<>();

int left = low;

int right = mid+1;

while(left<=mid && right<=high)

{

if(arr[left]<=arr[right])

{

temp.add(arr[left]);

left++;

}

else{

temp.add(arr[right]);

right++;

}

}

while(left<=mid)

{

temp.add(arr[left]);

left++;

}

while(right<=high)

{

temp.add(arr[right]);

right++;

}

for(int i=low;i<=high;i++)

{

arr[i]=temp.get(i-low);

}

}

public static int index(int arr [] , int low, int high)

{

int pivot = arr[low];

int i = low;

int j = high;

while(i<j)

{

while(i<=high-1 && arr[i]<=pivot)

{

i++;

}

while(j>=low+1 && arr[j]>pivot)

{

j--;

}

if(i<j)

{

int temp = arr[i];

arr[i] = arr[j];

arr[j] = temp;

}

}

// swap pivot and j

int temp = arr[low];

arr[low] = arr[j];

arr[j] = temp;

return j;

}

public static void quicksort(int arr[], int low,int high)

{

if(low>=high)

{

return;

}

int partitionIndex = index(arr,low,high);

quicksort(arr,low,partitionIndex-1);

quicksort(arr,partitionIndex+1,high);

}

public static void freq(int arr[],int n)

{

HashMap<Integer,Integer> map = new HashMap<>();

for(int i=0;i<n;i++)

{

map.put(arr[i],map.getOrDefault(arr[i],0)+1);

}

int maxele=-1;

int maxfreq=-1;

for(Map.Entry<Integer,Integer> variable : map.entrySet())

{

if(variable.getValue()>maxfreq)

{

maxfreq=variable.getValue();

maxele=variable.getKey();

}

}

System.out.println("The maximum frequency is: "+maxfreq+" for the element :"+maxele);

}

public static void seclar(int arr[])

{

int lar = -1;

int sec = -1;

int n = arr.length;

for(int i=0;i<n;i++)

{

if(arr[i]>lar)

{

sec=lar;//7

lar=arr[i];//7//23

}

else if(arr[i]<lar && arr[i]>sec)

{

sec = arr[i];

}

}

System.out.println(sec);

}

public static void duplic(int arr[])

{

int n = arr.length;

int i =0;

for(int j=1;j<n;j++)

{

if(arr[j]!=arr[i])

{

i++;

arr[i] = arr[j];

}

}

System.out.println("Number of unique elements: "+(i+1));

System.out.println("new Array is: "+Arrays.toString(Arrays.copyOfRange(arr, 0, i+1)));

System.out.print("original array is "+Arrays.toString(arr));

}

public static void leftby1(int arr [] )

{

int n = arr.length;

int temp = arr[0];

for(int i=1;i<n;i++)

{

arr[i-1] = arr[i];

}

arr[n-1] = temp;

System.out.println(Arrays.toString(arr));

}

public static void rightbyd(int arr[])

{

int d = 3;

int n = arr.length;

for(int i=0;i<d;i++)

{

int temp = arr[n-1];

for(int j=n-1;j>0;j--)

{

arr[j] = arr[j-1];

}

arr[0]=temp;

}

System.out.println(Arrays.toString(arr));

}

public static void rightbydopt(int arr[])

{

int n = arr.length;

int d = 3;

reverse(arr,0,n-1);

reverse(arr,0,d-1);

reverse(arr,d,n-1);

System.out.println(Arrays.toString(arr));

}

public static void reverse(int arr[],int start,int end)

{

while(start<end)

{

int temp = arr[start];

arr[start] = arr[end];

arr[end] = temp;

start++;

end--;

}

}

public static void leftbydopt(int arr[])

{

int n = arr.length;

int d = 3;

reverse(arr,0,d-1);

reverse(arr,d,n-1);

reverse(arr,0,n-1);

System.out.println(Arrays.toString(arr));

}

public static void moveZeros(int arr[])

{

int n = arr.length;

int j = -1;

for(int i=0;i<n;i++)

{

if(arr[i]==0)

{

j=i;

break;

}

}

for(int i=j+1;i<n;i++)

{

if(arr[i]!=0)

{

int temp = arr[i];

arr[i]=arr[j];

arr[j]=temp;

j++;

}

}

System.out.println(Arrays.toString(arr));

}

public static void unionop(int arr1[] , int arr2[])

{

Set<Integer> set = new HashSet<>();

for(int nums:arr1)

{

set.add(nums);

}

for(int nums:arr2)

{

set.add(nums);

}

System.out.println(set);

}

public static void interop(int arr1[],int arr2[])

{

Set<Integer> set = new HashSet<>();

for(int nums:arr1)

{

set.add(nums);

}

Set<Integer> intersection = new HashSet<>();

for(int nums:arr2)

{

if(set.contains(nums))

{

intersection.add(nums);

}

}

System.out.println(intersection);

}

public static void misnumopt(int arr[])

{

int n = arr.length;

int xor1=0;

int xor2=0;

for(int i=0;i<n;i++)

{

xor1=xor1^(i+1);

xor2=xor2^arr[i];

}

xor1=xor1^(n+1);

int miss = xor1^xor2;

System.out.println(miss);

}

public static void consecones(int arr[])

{

int n = arr.length;

int maxi = 0;

int count = 0;

for(int i=0;i<n;i++)

{

if(arr[i]==1)

{

count++;

maxi=Math.max(maxi,count);

}

else{

count=0;

}

}

System.out.println("1's has appeared :"+maxi+" times");

}

public static void singleOpt(int arr[])

{

int n = arr.length;

int xor = 0;

for(int i=0;i<n;i++){

xor=xor^arr[i];

}

System.out.println(xor);

}

public static void longestsubarrayk(int arr[],int k)

{

// 1,2,3,1,1,1,1,4,2,3 and k = 6

int n = arr.length;

int sum =0;

int maxleng=0;

HashMap<Integer,Integer> map = new HashMap<>();

for(int i=0;i<n;i++)

{

sum=sum+arr[i];

if(sum==k)

{

maxleng=i+1;

}

if(map.containsKey(sum-k))

{

int previousIndex = map.get(sum-k);

maxleng=Math.max(maxleng,i-previousIndex);

}

if(!map.containsKey(sum))

{

map.put(sum,i);

}

}

System.out.println(maxleng);

}

public static void longestsubarraykopt(int arr[], int k)

{

//sliding window!!!

//1,2,3,1,1,1,1,4,2,3

int left = 0;

int right = 0;

int maxlengt =0;

int sum = arr[0];

int n = arr.length;

while(right<n)

{

while(left<=right && sum>k)

{

sum = sum - arr[left];

left++;

}

if(sum==k)

{

maxlengt=Math.max(maxlengt,right-left+1);

}

right++;

if(right<n)

{

sum=sum+arr[right];

}

}

System.out.println(maxlengt);

}

}