HASHING

Content

1. Relative Sorting

Statement : Given two arrays **A1[]** and **A2[]** of size **N** and **M** respectively. The task is to sort A1 in such a way that the relative order among the elements will be same as those in A2. For the elements not present in A2, append them at last in sorted order. It is also given that the number of elements in A2[] are smaller than or equal to number of elements in A1[] and A2[] has all distinct elements.
Note: Expected time complexity is **O(N log(N))**.

```
public static void main (String[] args) {
        Scanner sc = new Scanner(System.in);
        int test = sc.nextInt();
        for(int t=0;t<test;t++){
                int n = sc.nextInt();
                int m = sc.nextInt();
                int array[] = new int[n];
                for(int i=0;i<n;i++){
                        array[i] = sc.nextInt();
                }
                int arr[] = new int[m];
                HashSet<Integer> sec = new HashSet<>();
                for(int i=0;i<m;i++){
                        arr[i] = sc.nextInt();
                        sec.add(arr[i]);
                }
                //Processing
                HashMap<Integer, Integer> map = new HashMap<>();
                ArrayList<Integer> list = new ArrayList<>();
                for(int i=0;i<n;i++){
                        int val = array[i];
                        if(sec.contains(val)){
                                Integer v = map.get(val);
                                if(v!=null){
                                        map.put(val, v+1);
                                }else{
                                        map.put(val, 1);
                                }
                        }else{
                                list.add(val);
                        }
                }
                //Printing
                Collections.sort(list);
```

```
                for(int i=0;i<m;i++){
                        if(map.containsKey(arr[i])){
                                int fr = map.get(arr[i]);
                                for(int j=0;j<fr;j++){
                                        System.out.print(arr[i] + " ");
                                }
                        }
                }
                for(int i=0;i<list.size();i++){
                        System.out.print(list.get(i)+" ");
                }
                System.out.println();
        }
}
```

2. Sorting Elements of an Array by Frequency

Statement : Given an array **A[]** of integers, **sort** the array according to **frequency** of elements. That is elements that have higher frequency come first. If frequencies of two elements are same, then smaller number comes first.

```
public class GFG {
        public static void main(String[] args) {
                Scanner sc = new Scanner(System.in);
                int test = sc.nextInt();
                for(int t=0;t<test;t++){
                        int n = sc.nextInt();
                        int arr[] = new int[n];
                        for(int i=0;i<n;i++){
                                arr[i] = sc.nextInt();
                        }
                        Map<Integer, Integer> map = new HashMap<>();
                        List<Integer> outputArray = new ArrayList<>();

                        // Assign elements and their count in the list and map
                        for (int current : array) {
                                int count = map.getOrDefault(current, 0);
                                map.put(current, count + 1);
                                outputArray.add(current);
                        }

                        // Compare the map by value
                        SortComparator comp = new SortComparator(map);
```

```
                // Sort the map using Collections CLass
                Collections.sort(outputArray, comp);

                // Final Output
                for (Integer i : outputArray) {
                        System.out.print(i + " ");
                }
        }
    }
}

// Implement Comparator Interface to sort the values
class SortComparator implements Comparator<Integer> {
    private final Map<Integer, Integer> freqMap;

    // Assign the specified map
    SortComparator(Map<Integer, Integer> tFreqMap) {
        this.freqMap = tFreqMap;
    }

    // Compare the values
    @Override
    public int compare(Integer k1, Integer k2){
        // Compare value by frequency
        int freqCompare = freqMap.get(k2).compareTo(freqMap.get(k1));

        // Compare value if frequency is equal
        int valueCompare = k1.compareTo(k2);

        // If frequency is equal, then just compare by value, otherwise -
        // compare by the frequency.
        if (freqCompare == 0)
            return valueCompare;
        else
            return freqCompare;
    }
}
```

3. Largest subarray with 0 sum

Statement : Given an array having both positive and negative integers. The task is to complete the function **maxLen()** which returns the length of maximum subarray with 0 sum. The function takes two arguments an array **A** and **n** where n is the size of the array A.

```
int maxLen(int arr[], int n)
  {
     HashMap<Integer,Integer> map = new HashMap<>();
     int sum = 0, max_len = 0;
     for(int i=0;i<n;i++){
        sum += arr[i];
        if(arr[i]==0 && max_len==0)
           max_len = 1;
        if(sum==0)
           max_len = i+1;
        Integer prev_x = map.get(sum);
        if(prev_x != null)
           max_len = Math.max(max_len, i-prev_x);
        else
           map.put(sum,i);
     }
     return max_len;
  }
```

4. Common elements

Statement : Given three increasingly sorted arrays **A, B, C** of sizes N_1, N_2, **and** N_3 respectively, you need to print all common elements in these arrays**.**

Note: Please avoid printing the same common element more than once.

```
class GFG {
        public static void main (String[] args) {
                Scanner sc = new Scanner(System.in);
                int test = sc.nextInt();
                for(int t=0;t<test;t++){
                   int a = sc.nextInt();
                   int b = sc.nextInt();
                   int c = sc.nextInt();
                   int aa[] = new int[a];
```

```
                int bb[] = new int[b];
                int cc[] = new int[c];
                for(int i=0;i<a;i++){
                    aa[i] = sc.nextInt();
                }
                for(int i=0;i<b;i++){
                    bb[i] = sc.nextInt();
                }
                for(int i=0;i<c;i++){
                    cc[i] = sc.nextInt();
                }
                HashMap<Integer,Integer> map = new HashMap<>();
                for(int i=0;i<a;i++){
                    map.put(aa[i],1);
                }
                for(int i=0;i<b;i++){
                    Integer count = map.get(bb[i]);
                    if(count!=null){
                        map.put(bb[i],2);
                    }
                }
                ArrayList<Integer> list = new ArrayList<>();
                for(int i=0;i<c;i++){
                    Integer count = map.get(cc[i]);
                    if(count!=null && count==2){
                        if(!list.contains(cc[i]))
                            list.add(cc[i]);
                    }
                }
                if(list.size()==0){
                    System.out.println(-1);
                }else{
                for(int i=0;i<list.size();i++){
                    System.out.print(list.get(i)+" ");
                }
                    System.out.println();
                }
            }
        }
}
```

5. Find all four sum numbers

Statement : Given an array A of size N, find all combination of four elements in the array whose sum is equal to a given value K. For example, if the given array is {10, 2, 3, 4, 5, 9, 7, 8} and K = 23, one of the quadruple is “3 5 7 8” (3 + 5 + 7 + 8 = 23).

The output should contain only unique quadrples For example, if input array is {1, 1, 1, 1, 1, 1} and K = 4, then output should be only one quadrple {1, 1, 1, 1}.

Note: Print -1 if no such quadruple is found.

1. Hashing O(n^2(logn))

```
class GFG {
    class Pair{
        int first;
        int second;
        public Pair(int first,int second){
            this.first = first;
            this.second = second;
        }
    }
    StringBuilder findFourElement(int arr[], int n,int k){
        HashMap<Integer, Pair> map = new HashMap<>();
        HashSet<String> set = new HashSet<>();
        StringBuilder res = new StringBuilder();
        for(int i=0;i<n;i++){
            for(int j=i+1;j<n;j++){
                map.put(arr[i]+arr[j],new Pair(i,j));
            }
        }
        for(int i=0;i<n;i++){
            for(int j=i+1;j<n;j++){
                if(map.containsKey(k-(arr[i]+arr[j]))){
                    Pair pair = map.get(k-(arr[i]+arr[j]));
                    if(pair.first!=i && pair.first!=j && pair.second!=i && pair.second!=j){
                        ArrayList<Integer> A = new ArrayList<>();
                        A.add(arr[pair.first]);
                        A.add(arr[pair.second]);
                        A.add(arr[i]);
                        A.add(arr[j]);
                        Collections.sort(A);
                        String s = A.get(0)+" "+A.get(1)+" "+A.get(2)+" "+A.get(3)+" $";
```

```
                    if(!set.contains(s)){
                        set.add(s);
                        res.append(s);
                    }
                }
            }
        }
    }
    return res;
}
    public static void main (String[] args) {
            Scanner sc = new Scanner(System.in);
            int test = sc.nextInt();
            for(int i=0;i<test;i++){
                int n = sc.nextInt();
                int k = sc.nextInt();
                int arr[] = new int[n];
                for(int t=0;t<n;t++){
                    arr[t] = sc.nextInt();
                }
                Arrays.sort(arr);
                GFG gfg = new GFG();
                StringBuilder res = gfg.findFourElement(arr,n,k);
                if(res.length()>0){
                    System.out.println(res);
                }else{
                    System.out.println(-1);
                }
            }
    }
}
```

2. Simple Four For loop O(n^4)

```
class GFG {
    StringBuilder findFourElement(int arr[], int n,int kk){
        StringBuilder res = new StringBuilder();
        HashSet<String> set = new HashSet<>();
        for(int i=0;i<n-3;i++){
            for(int j=i+1;j<n-2;j++){
                for(int k=j+1;k<n-1;k++){
                    for(int l=k+1;l<n;l++){
                        if((arr[i]+arr[j]+arr[k]+arr[l])==kk){
```

```
                    String s = arr[i]+" "+arr[j]+" "+arr[k]+" "+arr[l]+" $";
                    if(!set.contains(s)){
                        set.add(s);
                        res.append(s);
                    }
                }
            }
        }
    }
    return res;
}
    public static void main (String[] args) {
            Scanner sc = new Scanner(System.in);
            int test = sc.nextInt();
            for(int i=0;i<test;i++){
                int n = sc.nextInt();
                int k = sc.nextInt();
                int arr[] = new int[n];
                for(int t=0;t<n;t++){
                    arr[t] = sc.nextInt();
                }
                Arrays.sort(arr);
                GFG gfg = new GFG();
                StringBuilder res = gfg.findFourElement(arr,n,k);
                if(res.length()>0){
                    System.out.println(res);
                }else{
                    System.out.println(-1);
                }
            }
    }
}
```

6. Swapping pairs make sum equal

Statement : Given two arrays of integers, write a program to check if a pair of values (one value from each array) exists such that swapping the elements of the pair will make the sum of two arrays equal.

```
class GFG {
    public static void main (String[] args) {
```

```
Scanner sc = new Scanner(System.in);
int test = sc.nextInt();
for(int t=0;t<test;t++){
	int n = sc.nextInt();
	int m = sc.nextInt();
	int sum1 = 0, sum2 = 0;
	int arr1[] = new int[n];
	int arr2[] = new int[m];
	for(int i=0;i<n;i++){
		arr1[i] = sc.nextInt();
		sum1 += arr1[i];
	}
	for(int i=0;i<m;i++){
		arr2[i] = sc.nextInt();
		sum2 += arr2[i];
	}
	int ans = -1;
	HashSet<Integer> set = new HashSet<>();
	//first array should be small or equal then second
	if(n>=m){
		for(int i=0;i<m;i++){
			set.add(arr2[i]);
		}
		int diff = ( sum2 - sum1 ) /2;
		for(int i=0;i<n;i++){
			int temp = diff + arr1[i];
			if(set.contains(temp)){
				ans = 1;
				break;
			}
		}
	}else{
		for(int i=0;i<n;i++){
			set.add(arr1[i]);
		}
		int diff = ( sum1 - sum2 ) /2;
		for(int i=0;i<m;i++){
			int temp = diff + arr2[i];
			if(set.contains(temp)){
				ans = 1;
				break;
```

```
                        }
                    }
                }
                System.out.println(ans);
            }
        }
}
```

7. Count distinct elements in every window

Statement : Given an array of integers and another number K. Find the count of distinct elements in every window of size K in the array.

```
void countDistinct(int A[], int k, int n)
   {
       for(int i=0;i<=n-k;i++){
          HashSet<Integer> set = new HashSet<>();
          for(int j=i;j<i+k;j++){
             set.add(A[j]);
          }
          System.out.print(set.size()+" ");
       }
   }
```

8. Array Pair Sum Divisibility Problem

Statement : Given an array of integers and a number k, write a function that returns true if given array can be divided into pairs such that sum of every pair is divisible by k.

```
class GFG {
   static boolean findPairs(int arr[], int k){
      if(arr.length%2==1)  //odd no of element
         return false;

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

     // count frequency of remainders
      for(int i=0;i<arr.length;i++){
         int rem = arr[i]%k;
         if(!map.containsKey(rem)){
            map.put(rem,0);
         }
```

```
            map.put(rem,map.get(rem)+1);
        }

        for(int i=0;i<arr.length;i++){
            int rem = arr[i]%k;
            if(rem*2==k){
                if(map.get(rem)%2==1)
                    return false;
            }else if(rem==0){
                if(map.get(rem)%2==1)
                    return false;
            }else{
                if(map.get(rem)!=map.get(k-rem))
                    return false;
            }
        }

        return true;
    }
      public static void main (String[] args) {
              Scanner sc = new Scanner(System.in);
              int test = sc.nextInt();
              for(int t=0;t<test;t++){
                  int n = sc.nextInt();
                  int arr[] = new int[n];
                  for(int i=0;i<n;i++){
                      arr[i] = sc.nextInt();
                  }
                  int k = sc.nextInt();
                  boolean res = findPairs(arr,k);
                  if(res)
                      System.out.println("True");
                  else
                      System.out.println("False");
              }
      }
}
*+
```

9. Longest consecutive subsequence

Statement : Given an array of positive integers. Find the length of the longest sub-sequence such that elements in the subsequence are consecutive integers,

the **consecutive numbers can be in any order.**

```
static int findLongestConseqSubseq(int arr[], int n)
 {
        Arrays.sort(arr);
        int res = 1;
        int len = 1;
        int prev = arr[0];
        for(int i=1;i<n;i++){
                if(prev+1==arr[i]){
                        len++;
                }else if((prev+1)<arr[i]){
                        len = 1;
                }
                prev = arr[i];
                if(res<len)
                        res = len;
        }
        return res;

}
```

10. Array Subset of another array

Statement : Given two arrays: arr1[0..m-1] of size m and arr2[0..n-1] of size n. Task is to check whether arr2[] is a subset of arr1[] or not. Both the arrays can be both unsorted or sorted. It may be assumed that elements in both array are distinct.

```
class GFG {
        public static void main (String[] args) {
           Scanner sc = new Scanner(System.in);
           int test = sc.nextInt();
           for(int t=0;t<test;t++){
              int n = sc.nextInt();
              int m = sc.nextInt();
              int first[] = new int[n];
              for(int i=0;i<n;i++){
                 first[i] = sc.nextInt();
              }
              int sec[] = new int[m];
              for(int i=0;i<m;i++){
                 sec[i] = sc.nextInt();
              }
```

```
            String ans = "Yes";
            HashSet<Integer> set = new HashSet<>();
            for(int i=0;i<n;i++){
               set.add(first[i]);
            }
            for(int i=0;i<m;i++){
               if(!set.contains(sec[i])){
                  ans = "No";
                  break;
               }
            }
               System.out.println(ans);
         }
      }
}
```