Build Heap

Question Link: https://www.codingninjas.com/codestudio/problems/build-heap_975375?leftPanelTab=0

Reference Link: https://www.geeksforgeeks.org/building-heap-from-array/

```
import java.util.*;

public class Solution {

   public static void heapify(ArrayList<Integer> arr,int i,int n){

      int largest = i;

      int left = 2*i+1;

      int right = 2*i+2;

      if(left<n && arr.get(left)>arr.get(largest)){largest=left;}

      if(right<n && arr.get(right)>arr.get(largest)){largest=right;}

      if(largest!=i){

         Collections.swap(arr,largest,i);

         heapify(arr,largest,n);

      }
   }

   public static ArrayList<Integer> buildHeap(ArrayList<Integer> arr, int n) {

      for(int i=n/2-1;i>=0;i--){
         heapify(arr,i,n);
      }

      return arr;
   }
}
```

Heap Sort

Question Link: https://practice.geeksforgeeks.org/problems/heap-sort/1

Reference Link: https://www.geeksforgeeks.org/heap-sort/

```
//{ Driver Code Starts
import java.util.*;
class Heap_Sort
{
   void printArray(int arr[],int n)
   {
      //int n = arr.length;
      for (int i=0; i<n; ++i)
         System.out.print(arr[i]+" ");
      System.out.println();
   }
   public static void main(String args[])
   {
      Scanner sc  = new Scanner(System.in);
      Heap_Sort hs = new Heap_Sort();
      int arr[] = new int[1000000];
      int T = sc.nextInt();
      while(T>0)
      {
         int n = sc.nextInt();
         for(int i=0;i<n;i++)
            arr[i] = sc.nextInt();

         Solution ob=new Solution();
         ob.heapSort(arr,n);
         hs.printArray(arr,n);
         T--;
      }
   }

}

// } Driver Code Ends

class Solution
{
```

```
  //Function to build a Heap from array.
  static void buildHeap(int arr[], int n)
  {
     // Your code here
     for(int i=n/2-1;i>=0;i--){
        heapify(arr,n,i);
     }
  }

  //Heapify function to maintain heap property.
  static void heapify(int arr[], int n, int i)
  {
     // Your code here
     int largest = i;
     int left = 2*i+1;
     int right = 2*i+2;
     if(left<n && arr[left]>arr[largest]){largest=left;}
     if(right<n && arr[right]>arr[largest]){largest=right;}
     if(largest!=i){
        int temp = arr[i];
        arr[i] = arr[largest];
        arr[largest] = temp;
        heapify(arr,n,largest);
     }
  }

  //Function to sort an array using Heap Sort.
  static public void heapSort(int arr[], int n)
  {
     //code here
     buildHeap(arr,n);
     int size=n-1;
     while(size>0){
        int temp = arr[size];
        arr[size] = arr[0];
        arr[0] = temp;
        heapify(arr,size,0);
        size--;
     }
  }
}
```

Maximum of all subarrays of size K

Question Link: https://leetcode.com/problems/sliding-window-maximum/
Reference Link : https://www.geeksforgeeks.org/sliding-window-maximum-maximum-of-all-subarrays-of-size-k/

```
import java.util.*;
class Solution {
   public static class Node{
      int number;
      int index;
      Node(int number, int index){
         this.number = number;
         this.index  = index;
      }
   }

   static class The_Comparator implements Comparator<Node> {
   public int compare(Node n1, Node n2)
   {
      if(n1.number<n2.number) return 1;
      return -1;
   }
   }
   public int[] maxSlidingWindow(int[] nums, int k) {
      PriorityQueue<Node> max_heap = new PriorityQueue<Node>(new The_Comparator());
      ArrayList<Integer> ans = new ArrayList<Integer>();
      /*int j=0;
      for(int i=0;i<nums.length;i++){
         System.out.println(max_heap);
         if(max_heap.size()>=k){
            ans.add(max_heap.peek());
            max_heap.remove(nums[j]);
            j++;
         }
            max_heap.add(nums[i]);
      }
      ans.add(max_heap.peek());
      int[] arr = ans.stream().mapToInt(i -> i).toArray();
      return arr;
      */
      /*int i=0;
      for(;i<k;i++)
         max_heap.add(nums[i]);
      ans.add(max_heap.peek());
      max_heap.remove(nums[0]);
      for(;i<nums.length;i++){
         max_heap.add(nums[i]);
         ans.add(max_heap.peek());
         max_heap.remove(nums[i-k+1]);
```

```
    }*/

    for(int i=0;i<nums.length;i++){
       max_heap.add(new Node(nums[i],i));
       if(i>=k-1){
          while(!max_heap.isEmpty() && max_heap.peek().index<=i-k){max_heap.poll();}
          ans.add(max_heap.peek().number);
       }
    }
    int[] arr = ans.stream().mapToInt(j -> j).toArray();
    return arr;
  }
}
```

K largest elements

Question Link: https://practice.geeksforgeeks.org/problems/k-largest-elements4206/1

```
class Solution {
   int[] kLargest(int[] arr, int n, int k) {
      // code here
      int[] ans = new int[k];
      PriorityQueue<Integer> min_heap = new PriorityQueue<Integer>();
      for(int i=0;i<n;i++){
         if(min_heap.size()>=k){
            if(min_heap.peek()<arr[i])
            {
               min_heap.poll();
               min_heap.add(arr[i]);
            }
         }
         else{min_heap.add(arr[i]);}
      }
      int j=k-1;
      while(!min_heap.isEmpty()){
         ans[j--] = min_heap.poll();
      }
      return ans;
   }
}
```

Merge k Sorted Arrays

Question Link : https://practice.geeksforgeeks.org/problems/merge-k-sorted-arrays/1

```
//User function Template for Java
class Solution
{
   static class Node{
      int data;
      int row;
      int column;
      Node(int data, int row, int column){
         this.data = data;
         this.row = row;
         this.column = column;
      }
   }

   static class The_Comparator implements Comparator<Node>{
      public int compare(Node n1, Node n2){
         if(n1.data>n2.data) return 1;
         return -1;
      }
   }
   //Function to merge k sorted arrays.
   public static ArrayList<Integer> mergeKArrays(int[][] arr,int K)
   {
      // Write your code here.
      ArrayList<Integer> ans = new ArrayList<Integer>();
      PriorityQueue<Node> min_heap = new PriorityQueue<Node>(new The_Comparator());
      for(int i=0;i<K;i++){
         min_heap.add(new Node(arr[i][0],i,0));
      }
      while(min_heap.size()>0){
         Node temp = min_heap.poll();
         ans.add(temp.data);
         if(temp.column+1<arr[temp.row].length)
         {
            min_heap.add(new Node(arr[temp.row][temp.column+1],temp.row,temp.column+1));
         }
      }
```

```
        return ans;
    }
}
```

Merge two binary Max heaps

Question Link: https://practice.geeksforgeeks.org/problems/merge-two-binary-max-heap0144/1

```
class Solution{
   public static void heapify(int[] arr, int n,int i){
      int largest=i;
      int left = (2*i)+1;
      int right = (2*i)+2;
      if(left<n && arr[left]>arr[largest]){largest=left;}
      if(right<n && arr[right]>arr[largest]){largest=right;}
      if(largest!=i){
         //Collections.swap(arr,largest,i);
         int temp = arr[largest];
         arr[largest] = arr[i];
         arr[i]  = temp;
         heapify(arr,n,largest);
      }
   }

   public int[] mergeHeaps(int[] a, int[] b, int n, int m) {
      // your code here
      int[] sarr=new int[n+m];
      System.arraycopy(a,0,sarr,0,n);
      System.arraycopy(b,0,sarr,n,m);
      int size = n+m;
      for(int i=(size/2)-1;i>=0;i--){
         heapify(sarr,size,i);
      }
      return sarr;
   }
}
```

K-th Largest Sum Contiguous Subarray

Question Link: https://practice.geeksforgeeks.org/problems/k-th-largest-sum-contiguous-subarray/1

```
class Solution {
   public static int kthLargest(int N, int K, int[] arr) {
      // code here
      PriorityQueue<Integer> min_heap = new PriorityQueue<Integer>();
      for(int i=0;i<N;i++){
         int sum = 0;
         for(int j=i;j<N;j++){
            sum+=arr[j];
            if(min_heap.size()<K)
               min_heap.add(sum);
            else if(min_heap.peek()<sum)
            {
               min_heap.poll();
               min_heap.add(sum);
            }
         }
      }
      return min_heap.peek();
   }
}
```

Merge K sorted linked lists

Question Link: https://practice.geeksforgeeks.org/problems/merge-k-sorted-linked-lists/1

```
///*class Node
{
   int data;
   Node next;

   Node(int key)
   {
      data = key;
      next = null;
   }
}
*/

// a is an array of Nodes of the heads of linked lists
// and N is size of array a


class Solution
{

   //Function to merge K sorted linked list.
   static class The_comparator implements Comparator<Node>
   {
      public int compare(Node n1, Node n2){
         if(n1.data>n2.data) return 1;
         return -1;
      }
   }

   Node mergeKList(Node[]arr,int K)
   {
      PriorityQueue<Node> min_heap = new PriorityQueue<Node>(new
The_comparator());
      for(int i=0;i<K;i++){
         if(arr[i]!=null)
         min_heap.add(arr[i]);
      }
```

```
        Node head=null;
        Node tail=null;
        while(min_heap.size()>0){
            Node temp = min_heap.poll();
            if(temp.next!=null)
            min_heap.add(temp.next);
            if(head==null)
            {
                head = temp;
                tail = head;
            }
            else{
                tail.next = temp;
                tail = tail.next;
            }
        }
        return head;
    }
}
```

Smallest range in K lists

Question Link : https://practice.geeksforgeeks.org/problems/find-smallest-range-containing-elements-from-k-lists/1

```
class Solution
{
   static class Node{
      int data;
      int column;
      int row;
      Node(int data, int row, int column){
         this.data = data;
         this.row = row;
         this.column = column;
      }
   }
   static class The_comparator implements Comparator<Node>{
      public int compare(Node n1, Node n2)
      {
         if(n1.data>n2.data) return 1;
         return -1;
      }
   }
       static int[] findSmallestRange(int[][] arr,int n,int k)
       {
          PriorityQueue<Node> min_heap = new PriorityQueue<Node>(new The_comparator());
          int max = Integer.MIN_VALUE;
          int min = Integer.MAX_VALUE;
          for(int i=0;i<k;i++)
          {
             min_heap.add(new Node(arr[i][0],i,0));
             max = Math.max(max,arr[i][0]);
             min = Math.min(min,arr[i][0]);
          }
          int start = min, end = max;
          while(min_heap.size()>0)
          {
             Node temp = min_heap.poll();
             min = temp.data;
             if(max-min<end-start)
             {
                start=min;
                end=max;
             }
             if(temp.column+1<n){
                max = Math.max(max,arr[temp.row][temp.column+1]);
                min_heap.add(new Node(arr[temp.row][temp.column+1],temp.row,temp.column+1));
             }
             else{break;}
          }
          return new int[]{start,end};
```

```
        }
    }
```

Is Binary Tree Heap

Question Link : https://practice.geeksforgeeks.org/problems/is-binary-tree-heap/1

```
// User Function template for JAVA

/*
Node defined as
class Node{
   int data;
   Node left,right;
   Node(int d){
      data=d;
      left=right=null;
   }
}
*/

class Solution {
   public static int count(Node tree)
   {
      if(tree==null) return 0;
      int result = 1+count(tree.left)+count(tree.right);
      return result;
   }

   public static boolean isCBT(Node tree,int i ,int count)
   {
      if(tree==null) return true;
      if(i>count) return false;
      return isCBT(tree.left,i*2+1,count)&&isCBT(tree.right,i*2+2,count);
   }
   public static boolean maxorder(Node tree)
   {
      if(tree==null) return true;
      if(tree.right==null&&tree.left==null) return true;
      if(tree.right==null) return tree.data>tree.left.data;
      return
maxorder(tree.left)&&maxorder(tree.right)&&tree.left.data<tree.data&&tree.rig
```

```
ht.data<tree.data;
    }
    boolean isHeap(Node tree) {
        // code here
        int count = count(tree);
        return  isCBT(tree,0,count)&&maxorder(tree);
    }
}
```

Convert BST to Min Heap

Question link: https://www.codingninjas.com/codestudio/problems/convert-bst-to-min-heap_920498

```
/************************************************************

    Following is the Binary Tree node structure:

    class BinaryTreeNode {

        int data;
        BinaryTreeNode left;
        BinaryTreeNode right;

        BinaryTreeNode(int data) {
            this.data = data;
            left = null;
            right = null;
          }
    }

************************************************************/
import java.util.*;
public class Solution {
    public static ArrayList<Integer> inorder(BinaryTreeNode root,
ArrayList<Integer> ls)
    {
        if(root==null) return null;
        inorder(root.left,ls);
        ls.add(root.data);
        inorder(root.right,ls);
        return ls;
    }

    public static void fillinorder(BinaryTreeNode root, ArrayList<Integer> ls, int[] i)
    {
        if(root==null) return;
        i[0]+=1;
        root.data = ls.get(i[0]);
```

```
        fillinorder(root.left,ls,i);
        fillinorder(root.right,ls,i);
    }

    public static void preorder(BinaryTreeNode root)
    {
        if(root==null) return ;
        System.out.print(root.data);
        preorder(root.left);
        preorder(root.right);
    }

        public static BinaryTreeNode convertBST(BinaryTreeNode root) {
                // Write your code here.
        //BinaryTreeNode temp = root;
        ArrayList<Integer> inorder = inorder(root, new ArrayList<Integer>());
        fillinorder(root,inorder,new int[]{-1});
        //preorder(root);
        return root;
        }
}
```

Median in a stream

Question Link: https://www.codingninjas.com/codestudio/problems/median-in-a-stream_975268

```
import java.util.*;
public class Solution {
   public static int signum(int a, int b)
   {
      if(a==b) return 0;
      if(a>b) return 1;
      if(a<b) return -1;
      return -5;
   }
   public static void callmedian(int ele, PriorityQueue<Integer>
max_heap,PriorityQueue<Integer> min_heap, int[] median){
      switch(signum(max_heap.size(),min_heap.size()))
      {
         case 0:
            if(ele>median[0])
            {
               min_heap.add(ele);
               median[0] = min_heap.peek();
            }
            else
            {
               max_heap.add(ele);
               median[0] = max_heap.peek();
            }
            break;
         case 1:
            if(ele>median[0])
            {
               min_heap.add(ele);
               median[0] = (max_heap.peek()+min_heap.peek())/2;
            }
            else
            {
               min_heap.add(max_heap.poll());
               max_heap.add(ele);
               median[0] = (max_heap.peek()+min_heap.peek())/2;
```

```
            }
            break;
         case -1:
            if(ele>median[0])
            {
               max_heap.add(min_heap.poll());
               min_heap.add(ele);
               median[0] = (max_heap.peek()+min_heap.peek())/2;
            }
            else
            {
               max_heap.add(ele);
               median[0] = (max_heap.peek()+min_heap.peek())/2;
            }
            break;
      }
      return ;
   }
   public static int[] findMedian(int[] arr, int n) {
      // Write your code here.
      PriorityQueue<Integer> min_heap = new PriorityQueue<Integer>();
      PriorityQueue<Integer> max_heap = new PriorityQueue<Integer>
(Collections.reverseOrder());
      int[] ans = new int[n];
      int[] median = new int[]{-1};
      for(int i=0;i<n;i++){
         callmedian(arr[i],max_heap,min_heap,median);
         ans[i] = median[0];
      }
      return ans;
   }
}
```

Minimum sum

Question Link: https://practice.geeksforgeeks.org/problems/minimum-sum4058/1

```
#User function Template for python3
from heapq import *;
class Solution:
    def solve(self, arr, n):
        # code here
        heapify(arr)
        num1,num2=0,0
        flag=0
        while(len(arr)>0):
            if(flag):
                num2 = num2*10 + heappop(arr);
                flag=0
            else:
                num1 = num1*10 + heappop(arr);
                flag=1
        return num1+num2
```

Reorganize String

Question Link: https://leetcode.com/problems/reorganize-string/

```
import java.util.*;
class Solution {

   public static class Node{
      String chr;
      Integer count;
      Node(String chr, Integer count)
      {
         this.chr = chr;
         this.count = count;
      }
   }

   public static class My_comparator implements Comparator<Node>{
      public int compare(Node n1, Node n2)
      {
         if(n1.count<n2.count) return 1;
         return -1;
      }
   }

   public String reorganizeString(String s) {
      HashMap<String,Integer> map = new HashMap<String,Integer>();

      for(String i:s.split("")){

         if(map.containsKey(i))
            map.put(i,map.get(i)+1);

         else
            map.put(i,1);
      }

      PriorityQueue<Node> max_heap = new PriorityQueue<Node>(new
My_comparator());
```

```
        for(Map.Entry<String,Integer> e: map.entrySet())
            max_heap.add(new Node(e.getKey(),e.getValue()));

        Node prev = null;
        String ans = "";
        while(max_heap.size()>0 || prev!=null)
        {
            if(prev!=null && max_heap.size()==0)
                return "";
            Node n = max_heap.poll();
            //System.out.println(n.chr);
            ans+=n.chr;
            n.count-=1;

            if(prev!=null)
            {
                max_heap.add(prev);
                prev=null;
            }
            if(n.count>0)
                prev = n;
        }
        return ans;
    }
}
```