1-Merge two sorted Arrays of integers into Single Array in O(n) time.

int i=m-1,j=n-1,k=n+m-1;

while(i>=0 && j>=0){

if(nums1[i]>nums2[j]){

nums1[k--]=nums1[i--];

}

else{

nums1[k--]=nums2[j--];

}

}

while(j>=0){

nums1[k--]=nums2[j--];

}

Extra-Second largest element in array

int firstMax=-1;

int secondMax=-1;

for(int  i=0;i<input.**length**;i++){

  if(input[i]>firstMax){

    secondMax=firstMax;

    firstMax=input[i];

  }

  else if(input[i]>secondMax){

  secondMax=input[i];

}

}

2-Kth Largest number

  int input[]={3,5,6,7,8,2,43,12};

    int k=2;

    Queue<**Integer**> heap= new PriorityQueue<**Integer**>(Collections.*reverseOrder*());

    for( int i :input){

      heap.add(i);

    }

    while(k>1){

     heap.poll();

     k--;

    }

3-A reverse a string

  int size=s.length;

        int j=size-1;

        for(int i=0;i<j;i++){

            char temp=s[i];

            s[i]=s[j];

            s[j]=temp;

            j--;

        }

4-B Reverse words in String

String stringArray[] = s.split(" ");

String output="";

for(int i=stringArray.length-1;i>=0;i--){

if(stringArray[i].trim()==""){

continue;

}

output = output+stringArray[i]+" ";

}

return output.trim();

5-Check if String and list are same like: list = ["I","Like","Apple"] and str = "I Like Apple" are same in O(n) time

**String** strg = "I Like Apple";

**String** str[] =strg.split(" ");

List<**String**> stringInput =new ArrayList<**String**>();

stringInput.add("I");

stringInput.add("Like");

stringInput.add("Apple");

boolean ans=true;

if(stringInput.size()==str.**length**){

 for(int i=0;i<str.**length**;i++){

   if(stringInput.get(i).equals(str[i])){

    ans=true;

   }

   else

   {

    ans=false;

    break;

   }

}

}

else{

 ans=false;

}

**System**.**out**.println(ans);

}

6-Right view of binary tree

 List<Integer> outputList= new LinkedList<Integer>();

        Queue<TreeNode> q= new LinkedList<TreeNode>();

        q.add(root);

        while(q.size()>0){

            if(root==null){

                break;

            }

            int n=q.size();

            for(int i=0;i<n;i++){

                TreeNode currNode=q.poll();

                if(i==n-1){

                    outputList.add(currNode.val);

                }

                if(currNode.left!=null){

                    q.add(currNode.left);

                }

                if(currNode.right!=null){

                    q.add(currNode.right);

                }

            }

        }

        return outputList;

    }

7-Find duplicates in an array in O(n) time.

List<Integer> outputArray = new LinkedList<Integer>();

Map<Integer,Integer> trackMap= new HashMap<Integer,Integer>();

int c=0;

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

if(trackMap.containsKey(nums[i])){

trackMap.put(nums[i],trackMap.get(nums[i])+1);

}

else{

trackMap.put(nums[i],1);

}

}

for(Map.Entry<Integer, Integer> entry:trackMap.entrySet()){

if(entry.getValue()>1){

outputArray.add(entry.getKey());

}

}

return outputArray;

8-Write a program to sort the given array in nlog(n) time.

void merge(int arr[], int l, int m, int r)

{

// Find sizes of two subarrays to be merged

int n1 = m - l + 1;

int n2 = r - m;

/\* Create temp arrays \*/

int L[] = new int[n1];

int R[] = new int[n2];

/\*Copy data to temp arrays\*/

for (int i = 0; i < n1; ++i)

L[i] = arr[l + i];

for (int j = 0; j < n2; ++j)

R[j] = arr[m + 1 + j];

/\* Merge the temp arrays \*/

// Initial indexes of first and second subarrays

int i = 0, j = 0;

// Initial index of merged subarray array

int k = l;

while (i < n1 && j < n2) {

if (L[i] <= R[j]) {

arr[k] = L[i];

i++;

}

else {

arr[k] = R[j];

j++;

}

k++;

}

/\* Copy remaining elements of L[] if any \*/

while (i < n1) {

arr[k] = L[i];

i++;

k++;

}

/\* Copy remaining elements of R[] if any \*/

while (j < n2) {

arr[k] = R[j];

j++;

k++;

}

}

// Main function that sorts arr[l..r] using

// merge()

void sort(int arr[], int l, int r)

{

if (l < r) {

// Find the middle point

int m = l + (r - l) / 2;

// Sort first and second halves

sort(arr, l, m);

sort(arr, m + 1, r);

// Merge the sorted halves

merge(arr, l, m, r);

}

}

8-B

 ListNode temp1=l1,temp2=l2;

        int carr=0,sum;

        ListNode currNode=new ListNode(0);

        ListNode head=currNode;

        while((temp1 !=null) || (temp2 != null) || carr==1 ){

            sum=0;

            if(temp1 !=null){

                sum+=temp1.val;

                temp1=temp1.next;

            }

            if(temp2 != null){

                sum+=temp2.val;

                temp2=temp2.next;

            }

             sum+=carr;

            carr=sum/10;

            currNode.next = new ListNode(sum%10);

            currNode=currNode.next;

        }

        return head.next;

    }

9-Detectd loop in linkedlist

        boolean result=false;

        while(head!=null){

            if(head.val==7){

                return true;

            }

            head.val=7;

            head=head.next;

        }

        return false;

11-Height of a binary tree Tree

 public int maxDepth(TreeNode root) {

        return maxHeight(root);

    }

    int maxHeight(TreeNode root){

        if(root==null){

            return 0;

        }

        int lMaxTree=maxHeight(root.left);

        int rMaxTree=maxHeight(root.right);

        if(lMaxTree>rMaxTree){

            return 1+lMaxTree;

        }

        else{

            return 1+rMaxTree;

        }

    }

Check perfect BST

13- boolean result=false;

Queue<TreeNode> q= new LinkedList<TreeNode>();

q.add(root);

while(!q.isEmpty()){

TreeNode temp=q.poll();

if(temp==null) {

result=true;

}

else{

if(result==true){

return false;

}

q.add(temp.left);

q.add(temp.right);

}

}

return true;

14-**Print the no. occurrences of alphabets in a string. Using arrays and time complexity should be O(n).**

Map<**Character**,**Integer**> trackMap= new HashMap<**Character**,**Integer**>();

  for(int i=0;i<input.length();i++){

    if(trackMap.containsKey(input.charAt(i))){

       trackMap.put(input.charAt(i),trackMap.get(input.charAt(i))+1);

    }

    else{

      trackMap.put(input.charAt(i),1);

    }

  }

  for(Map.*Entry*<**Character**,**Integer**> entry:trackMap.entrySet()){

**System**.**out**.println(entry.getKey() + " is the key and value is "+ entry.getValue());

  }

15-Reverse Nodes in k-Group

if(k == 1) return head;

ListNode currNode = head;

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

if(head != null) {

head = head.next;

}else {

return currNode;

}

}

ListNode nextGroup = reverseKGroup(head,k);

ListNode currStart = currNode;

ListNode tempPre = currNode;

ListNode tempNext = null;

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

tempNext = currNode.next;

currNode.next = tempPre;

tempPre = currNode;

currNode = tempNext;

}

// connect curr to next group

currStart.next = nextGroup;

return tempPre;;

}

16-

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

if(s.indexOf(s.charAt(i))==s.lastIndexOf(s.charAt(i))){

return i;

}

}

return -1;

Medium-

1-Given a list of lists that contains arrival and departure of N number of trains e.g.: {{A1,D1},{A2,D2}...{An,Dn}},`

int count=0,ans=0,i=0,j=0;

Arrays.sort(arr);

Arrays.sort(dep);

while(i<n){

if(arr[i]<=dep[j]){

count++;i++;

ans=Math.max(count,ans);

}

else{

count--;

j++;

}

}

return ans;

}

2-public int countNodes(TreeNode root) {

        if(root==null){

            return 0;

        }

        else{

            return countNodes(root.left)+countNodes(root.right) +1;

        }

    }

3-Find the missing number in an integer array of N-1 numbers such that it contains only distinct integers from 1 to N.

int sum=(nums.length \*( nums.length+1 ))/2;

int s=0;

for(int i:nums){

s+=i;

}

return sum-s;

4-Given an array of integers where each element represents the max number of steps that can be made forward from that element. Write a function to return the minimum number of jumps to reach the end of the array (starting from the first element). If an element is 0, they cannot move through that element. If the end isn’t reachable, return -1.  
Input: arr[] = {1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9}  
Output: 3 (1-> 3 -> 9 -> 9)

int currStop=0,currReach=0;

int step=0;

for(int i=0;i<nums.length-1;i++){

currReach=Math.max(i+nums[i],currReach);

if(i==currStop){

step++;

if(currReach>=(nums.length)-1){

return step;

}

currStop=currReach;

}

}

return step;

5-Given an array of N integers, and an integer K, find the number of pairs of elements in the array whose sum is equal to K.

 Map<Integer,Integer> trackMap= new HashMap<Integer,Integer>();

        int count=0;

        for(int i:nums){

            if(trackMap.containsKey(i) && trackMap.get(i)>0 ){

                count++;

                trackMap.put(i,trackMap.get(i)-1);

            }

            else{

                trackMap.put(k-i,trackMap.getOrDefault(k-i,0)+1);

            }

        }

        return count;

6-Check if Binary tree is Balanced Binary Tree or not.

public boolean isBalanced(TreeNode root) {

return dfsHeight(root) !=-1;

}

int dfsHeight(TreeNode root){

if(root==null){

return 0;

}

int leftHeight=dfsHeight(root.left);

if(leftHeight==-1) return -1;

int rightHeight=dfsHeight(root.right);

if(rightHeight==-1) return -1;

if(Math.abs(leftHeight-rightHeight)>1) return -1;

return Math.max(leftHeight,rightHeight)+1;

}

1. 7-Given a string s, find the length of the longest substring without repeating characters.

Example 1:

Input: s = "abcabcbb"

Output: 3

String LongestSubstring = null;

int LongestSubstringLength = 0;

LinkedHashMap<Character, Integer> map = new LinkedHashMap<>();

char ch[] = s.toCharArray();

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

char c = ch[i];

if(!map.containsKey(c)){

map.put(c,i);

}

else{

i=map.get(c);

map.clear();

}

if(map.size()>LongestSubstringLength)

LongestSubstringLength = map.size();

}

return LongestSubstringLength;

8-Find the maximum amount of water that can be trapped within a given set of bars where each bar’s width is 1 unit.

 int n=height.length;

       int leftMax[]= new int[n];

       int rightMax[]= new int[n];

       leftMax[0]=height[0];

       rightMax[n-1]=height[n-1];

       int i=1;

       int j=n-2;

       while((i<n) &&(j>=0)){

           leftMax[i] =Math.max(leftMax[i-1],height[i]);

           rightMax[j] =Math.max(rightMax[j+1],height[j]);

             i++;j--;

       }

       int sum=0;

       for(i=0;i<n;i++){

           sum+=Math.min(leftMax[i],rightMax[i])-height[i];

       }

    return sum;

Algo

1. 1-Given an input string, write a function that returns the Run Length Encoded string for the input string.

For example,

if the input string is “wwwwaaadexxxxxx”,

then the function should

return “w4a3d1e1x6”

 int n = str.length();

    for (int i = 0; i < n; i++) {

      // Count occurrences of current character

      int count = 1;

      while (i < n - 1 &&

        str.charAt(i) == str.charAt(i + 1)) {

        count++;

        i++;

      }

      if (count == 1) {

        System.out.print(str.charAt(i) + "1");

      }

      else {

        System.out.print(str.charAt(i));

        System.out.print(count);

      }

    }

1. 2-**Let 1 represent ‘A’, 2 represents ‘B’, and so on. Given a digit sequence, count the number of possible decoding’s of the given digit sequence.**

**Input:**

**digits[] = "121"**

**Output: 3**

int n= s.length();

if(s==null || n==0)

return 0;

int dp[]= new int[n+1];

dp[0] = 1;

dp[1] = s.charAt(0)!='0'?1:0;

for(int i=2;i<=n;i++){

int first = Integer.valueOf(s.substring(i-1,i));

int second = Integer.valueOf(s.substring(i-2,i));

if(first>=1 && first<=9){

dp[i]+=dp[i-1];

}

if(second>=10 && second<=26){

dp[i]+=dp[i-2];

}

}

return dp[n];

1. Given an array of strings strings, group the anagrams together. You can return the answer in any order.

An Anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once.

Input:

strs = ["eat","tea","tan","ate",

"nat","bat"]

Output:

[["bat"],["nat","tan"],

["ate","eat","tea"]]

Expected Time Complexity: O(n)

List<List<String>>s=new ArrayList<>();

Map<String,List<String>>map=new LinkedHashMap<>();

for(String e:strs){

char ch[]=e.toCharArray();

Arrays.sort(ch);

String str=new String(ch);

if(!map.containsKey(str)){

map.put(str,new ArrayList<>());

}

map.get(str).add(e);

}

s.addAll(map.values());

return s;

1. Given a file containing data of student name and marks scored by him/her in 3 subjects. The task is to find the list of students having the maximum average score.

If more than one student has the maximum average score, print them as per the order in the file.

Input:

file[] = {“Shrikanth”, “20”, “30”,

“10”, “Ram”, “100”, “50”, “10”}

Output:

Ram 53

Average scores of Shrikanth, Ram are 20 and 53 respectively. So Ram has the maximum average score of 53.

Or

Input: [{"Bob","87"}, {"Mike", "35"},{"Bob", "52"}, {"Jason","35"}, {"Mike", "55"}, {"Jessica", "99"}]

String[][] students= {{"Bob","87"}, {"Mike", "35"},{"Bob", "52"}, {"Jason","35"}, {"Mike", "55"}, {"Jessica", "99"}};

int high=Integer.MIN\_VALUE;

Map<String,Integer> sum = new HashMap<String,Integer>();

Map<String,Integer> count = new HashMap<String,Integer>();

for(String[] student:students) {

int avg=0;

if(count.containsKey(student[0])) {

int currCount=count.get(student[0]);

currCount++;

count.put(student[0], currCount);

int currSum=sum.get(student[0]);

currSum+=Integer.valueOf(student[1]);

sum.put(student[0], currSum);

avg=currSum/currCount;

}else {

count.put(student[0], 1);

sum.put(student[0], Integer.valueOf(student[1]));

avg=Integer.valueOf(student[1]);

}

high=Math.max(high, avg);

}

System.out.println(high);

6 Given a Binary Search Tree (BST) and a positive integer k, find the k’th largest element in the Binary Search Tree.

Hint: Get inorder(aka sorted list since BST) and only traverse tree till you find kth node

TC: O(k), SC: O(k)

class Solution {

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

public int kthSmallest(TreeNode root, int k) {

printInorder(root, k);

return inorder.get(k-1);

}

public void printInorder(TreeNode root, int k) {

if(root != null && inorder.size() < k) {

printInorder(root.left, k);

inorder.add(root.val);

printInorder(root.right, k);

}

}

}

7-

 int max=nums[0];

        int beg=-2,end=-1;

        int n=nums.length;

        int min=nums[n-1];

        for(int i=1;i<n;i++){

            max=Math.max(nums[i],max);

            min=Math.min(nums[n-i-1],min);

            if(nums[i]<max){

                beg=i;

            }

            if(nums[n-i-1]>min){

                end=n-i-1;

            }

        }

        return beg-end +1;

    }

1. Given an unsorted array arr[0..n-1] of size n, find the minimum length subarray arr[s..e] such that sorting this subarray makes the whole array sorted.

Example: If the input array is [10, 12, 20, 30, 25, 40, 32, 31, 35, 50, 60], your program should be able to find that the subarray lies between indexes 3 and 8.

int n = A.length, beg = -1, end = -2, min = A[n-1], max = A[0];

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

max = Math.max(max, A[i]);

min = Math.min(min, A[n-1-i]);

if (A[i] < max) end = i;

if (A[n-1-i] > min) beg = n-1-i;

}

return end - beg + 1;

10-

Given an array of random numbers, push all the zeroes of a given array to the end of the array. For example, if the given array is {1, 9, 8, 4, 0, 0, 2, 7, 0, 6, 0}, it should be changed to {1, 9, 8, 4, 2, 7, 6, 0, 0, 0, 0}. The order of all other elements should be same.

Expected time complexity is O(n) and extra space is O(1).

public static void pushZero(int arr[], int n){

int count = 0;

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

if(arr[i]!=0){

arr[count++] = arr[i];

}

}

while(count<n)

arr[count++] = 0;

}

11-Joseph Problem

public int findTheWinner(int n, int k) {

if (n==1)

return 1;

else

return (findTheWinner(n-1,k) + k - 1) % n + 1;

}

12- Given an array of integers, sort the array into a wave like array and return it,  
In other words, arrange the elements into a sequence such that a1 >= a2 <= a3 >= a4 <= a5.....

**void** sortInWave(**int** arr[], **int** n)

    {

        // Traverse all even elements

**for**(**int** i = 0; i < n-1; i+=2){

            //swap odd and even positions

**if**(i > 0 && arr[i - 1] > arr[i])

              swap(arr, i, i-1);

**if**(i < n-1 && arr[i + 1] > arr[i])

              swap(arr, i, i+1);

        }

    }