grep -o "[0-9]*" data.txt| awk '{sum+=$1} END {print "Sum=" sum}'

1. 给一个 **binary tree**，问你是不是 **binary search tree**。然后让你写几个 **testcase**。**follow up** 问你时间和空间复杂度。

```java
public class Solution
{
    private void inorder(TreeNode root,ArrayList<Integer> result)
    {
        if(root==null)
        return;
        inorder(root.left,result);
        result.add(root.val);
        inorder(root.right,result);
    }
    public boolean isValidBST(TreeNode root)
    {
        ArrayList<Integer> result=new ArrayList<Integer>();
        inorder(root,result);
        int i;
        for(i=1;i<result.size();i++)//判断中序遍历是否有序
        {
            if(result.get(i)<=result.get(i-1))
            return false;
        }
        return true;
    }
}
```

test case: [2,1,3] true,
[4,2,5,1,3]true
[] true

时间复杂度：O(N), 空间复杂度：O(N)

```java
public boolean isValidBST (TreeNode root){
            Stack<TreeNode> stack = new Stack<TreeNode> ();
            TreeNode cur = root ;
            TreeNode pre = null ;
            while (!stack.isEmpty() || cur != null) {
                if (cur != null) {
                    stack.push(cur);
                    cur = cur.left ;
                } else {
                    TreeNode p = stack.pop() ;
                    if (pre != null && p.val <= pre.val) {
                        return false ;
                    }
                    pre = p ;
                    cur = p.right ;
                }
            }
            return true ;
        }
```

2. common elements in two lists，比如
   1 2 2 2 3
   2 1 2 3 5
   返回 2 2 3

```java
public class Solution {
    public int[] intersect(int[] nums1, int[] nums2) {
        HashMap<Integer, Integer> map = new HashMap<Integer, Integer>();
        ArrayList<Integer> result = new ArrayList<Integer>();
        for(int i = 0; i < nums1.length; i++)
        {
            if(map.containsKey(nums1[i])) map.put(nums1[i], map.get(nums1[i])+1);
            else map.put(nums1[i], 1);
        }

        for(int i = 0; i < nums2.length; i++)
        {
            if(map.containsKey(nums2[i]) && map.get(nums2[i]) > 0)
            {
                result.add(nums2[i]);
                map.put(nums2[i], map.get(nums2[i])-1);
            }
        }

       int[] r = new int[result.size()];
       for(int i = 0; i < result.size(); i++)
       {
           r[i] = result.get(i);
       }

       return r;
    }
}
```

input:
[1, 2, 2, 2, 3, 4]
[2, 1, 2, 5, 4]

outcome: [2,1,2,4]

如果已经排好序了

```java
public int[] intersection(int[] nums1, int[] nums2) {
        List<Integer> list=new ArrayList<>();
        int i = 0;
        int j = 0;
        while (i < nums1.length && j < nums2.length) {
            if (nums1[i] < nums2[j]) {
                i++;
            } else if (nums1[i] > nums2[j]) {
                j++;
            } else {
                list.add(nums1[i]);
                i++;
                j++;
            }
        }
        int[] result = new int[list.size()];
        int k = 0;
        for (i=0;i<list.size();i++) {
            result[k++] = list.get(i);
        }
        return result;
    }
```

3. sorted array to balanced tree

```java
public class Solution {
    public TreeNode helper(int[] nums,int start,int end){
        if(start>end)
        return null;
        int mid=(start+end)/2;
        TreeNode root=new TreeNode(nums[mid]);
        root.left=helper(nums,start,mid-1);
        root.right=helper(nums,mid+1,end);
        return root;
    }
    public TreeNode sortedArrayToBST(int[] nums) {
        if(nums.length==0)
        return null;
        TreeNode root=helper(nums,0,nums.length-1);
        return root;
    }
}
```

4. **2 sum**

```java
public class Solution
{
    public int[] twoSum(int[] nums, int target)
    {
        HashMap<Integer,Integer> map=new HashMap<>();
        int[] a=new int[2];
        int i;
        for(i=0;i<nums.length;i++)
        {
            if(map.containsKey(nums[i])==false)
            {
                map.put(target-nums[i],i);
            }
            else
            {
                a[0]=map.get(nums[i]);
                a[1]=i;
            }

        }
        return a;

    }
}
```

5. 给定一个**string** 找出**string**中第一个不重复的**character**。例如**input**：**amazon**，**return**：**m fist unique character**

```java
public class Solution
{
    public int firstUniqChar(String s) {
        int freq [] = new int[26];
        for(int i = 0; i < s.length(); i ++)
            freq [s.charAt(i) - 'a'] ++;
        for(int i = 0; i < s.length(); i ++)
            if(freq [s.charAt(i) - 'a'] == 1)
                return i;
        return -1;
    }
}
```

允许特殊字符，26 改成 256

如果只能遍历一遍数组的话，用栈配合

```java
public class Solution
{
    public int firstUniqChar(String s) {
        Stack<Integer> stack=new Stack<>();
        HashMap<Character,Integer> map=new HashMap<>();
        int i;
        for(i=s.length()-1;i>=0;i--){
            if(map.containsKey(s.charAt(i))==false){
                map.put(s.charAt(i),1);
                stack.push(i);
            }
            else{
                int val=map.get(s.charAt(i));
                val++;
                map.put(s.charAt(i),val);
            }
        }
        while(stack.isEmpty()==false){
            int res=stack.pop();
            char a=s.charAt(res);
            if(map.get(a)==1){
                return res;
            }
        }
        return -1;
    }
}
```

6. 判断两个二叉树的结构是否相同。the same binary tree

```java
public class Solution {
    public boolean isSameTree(TreeNode p, TreeNode q) {
        if(p==null&&q==null)
        return true;
        if(p==null||q==null)
        return false;
        if(p.val==q.val){
        boolean a=isSameTree(p.left,q.left);
        boolean b=isSameTree(p.right,q.right);
        if(a==true && b==true)
        return true;
        else
        return false;
        }
        else
        return false;
    }
}
```

7. 算出小于某个数所有质数和

We know all multiples of 2 must not be primes, so we mark them off as non-primes. Then we look at the next number, 3. Similarly, all multiples of 3 such as 3 × 2 = 6, 3 × 3 = 9, ... must not be primes, so we mark them off as well.

```
public class Solution
{
    public int countPrimes(int n)
    {
        if(n==0||n==1||n==2)
        return 0;
        int[] book=new int[n];
        int i;
        for(i=2;i<n;i++)
        book[i]=1;
        int k=2;
        int s=0;
        while(k<n)
        {
            if(book[k]==1)
            {
                for(i=2;i*k<n;i++)
                {
                    book[i*k]=0;
                }
            }
            k++;
        }
        for(i=2;i<n;i++)
        {
            if(book[i]==1)
            s++;
        }
        return s;
    }
}
```

8. 给 start 点和 end 点，在 matrix 里找最短路径

```
Public int minPath(char[][] grid,int sx,int sy,int dx,int dy)
{
    Queue<List<Integer>> queue=new LinkedList<>();
    int[][] book=new int[grid.length][grid[0].length];
    int[][] next={{1,0},{0,1},{0,-1},{-1,0}};
    ArrayList<Integer> a=new ArrayList<>();
    a.add(sx);
    a.add(sy);
    a.add(0);
    queue.offer(a);
    book[sx][sy]=true;
    int flag=0;
    while(queue.isEmpty()==false)
    {
        ArrayList<Integer> cur=queue.poll();
        int x=cur.get(0);
        int y=cur.get(1);
        int sum=cur.get(2);
        for(int i=0;i<4;i++)
        {
            int xx=x+next[i][0];
            int yy=y+next[i][1];
            if(xx<0||xx>=grid.length||yy<0||yy>=grid[0].length)
            continue;
            if(book[xx][yy]==false)
            {
                book[xx][yy]=true;
                sum++;
                queue.offer(Arrays.asList(xx,yy,sum));
            }
            if(xx==dx&&yy==dy)
            {
                flag=1;
                return sum;
            }
        }
    }
}
```

如果要输出路径的话：

```java
Public List<List<Integer>> minPath(char[][] grid,int sx,int sy,int dx,int dy)
{
    Queue<List<Integer>> queue=new LinkedList<>();
    int[][] book=new int[grid.length][grid[0].length];
    int[][] next={{1,0},{0,1},{0,-1},{-1,0}};
    int[][] prex=new int[grid.length][grid[0].length];
    int[][] prey=new int[grid.length][grid[0].length];
    ArrayList<Integer> a=new ArrayList<>();
    a.add(sx);
    a.add(sy);
    a.add(0);
    queue.offer(a);
    book[sx][sy]=true;
    int flag=0;
    while(queue.isEmpty()==false)
    {
        ArrayList<Integer> cur=queue.poll();
        int x=cur.get(0);
        int y=cur.get(1);
        int sum=cur.get(2);
        for(int i=0;i<4;i++)
        {
            int xx=x+next[i][0];
            int yy=y+next[i][1];
            if(xx<0||xx>=grid.length||yy<0||yy>=grid[0].length)
            continue;
            if(book[xx][yy]==false)
            {
                book[xx][yy]=true;
                sum++;
                prex[xx][yy]=x;
                prey[xx][yy]=y;
                queue.offer(Arrays.asList(xx,yy,sum));
            }
            if(xx==dx&&yy==dy)
            {
                flag=1;
                break;
            }
        }
        if(flag==1)
        break;
    }
    int p=dx,q=dy;
    List<List<Integer>> res=new ArrayList<>();
    res.add(Arrays.asList(p,q));
    while(p!=sx&&q!=sy)
    {
        int curx=p;
        int cury=q;
        p=prex[curx][cury];
        q=prey[curx][cury];
        res.add(Arrays.asList(p,q));
    }
    List<List<Integer>> result=new ArrayList<>();
    for(int i=0;i<res.size();i++)
    {
        result.add=res.get(res.size()-1-i);
    }
    return result;
}
```

9. Word break

```java
public class Solution {
    public boolean wordBreak(String s, List<String> dict) {
        boolean [] breakable = new boolean[s.length()+1];
        breakable[0] = true;

        for(int i=1;i<=s.length();i++){
            for(int j=0;j<i;j++){
                if(breakable[j]&&dict.contains(s.substring(j,i))){
                    breakable[i] = true;
                    break;
                }
            }
        }
        return breakable[s.length()];
    }
}
```

10. Serialize and Deserialize BST

```java
    public String serialize(TreeNode root) {
        if(root==null)
        return "";
        StringBuilder sb=new StringBuilder();
        Queue<TreeNode> queue=new LinkedList<>();
        queue.offer(root);
        while(queue.isEmpty()==false){
            TreeNode node=queue.poll();
            if(node==null){
                sb.append("X ");
                continue;
            }
            sb.append(node.val+" ");
            queue.offer(node.left);
            queue.offer(node.right);
        }
        return sb.toString();
    }
```

```
33      public TreeNode deserialize(String data) {
34         if(data=="")
35          return null;
36          String[] res=data.split(" ");
37          Queue<TreeNode> queue=new LinkedList<>();
38          TreeNode root=new TreeNode(Integer.parseInt(res[0]));
39          queue.add(root);
40          for(int i=1;i<res.length;i++){
41              TreeNode node=queue.poll();
42              if(res[i].equals("X")==false){
43                  TreeNode left=new TreeNode(Integer.parseInt(res[i]));
44                  node.left=left;
45                  queue.offer(left);
46              }
47              i++;
48              if(res[i].equals("X")==false){
49                  TreeNode right=new TreeNode(Integer.parseInt(res[i]));
50                  node.right=right;
51                  queue.offer(right);
52              }
53          }
54          return root;
55      }
56  }
```

11. 给一个 matrix 和 position，返回 matrix 值和这个 position 所在的 matrix 值相同而且连通的所有 position。
比输入一个二维数组[[1,2,3,4], [2,2,1,2], [2,3,1,4]]和一个位置(0,1)，输出应该是[[0,1], [1,1], [1,0], [2,0]]。

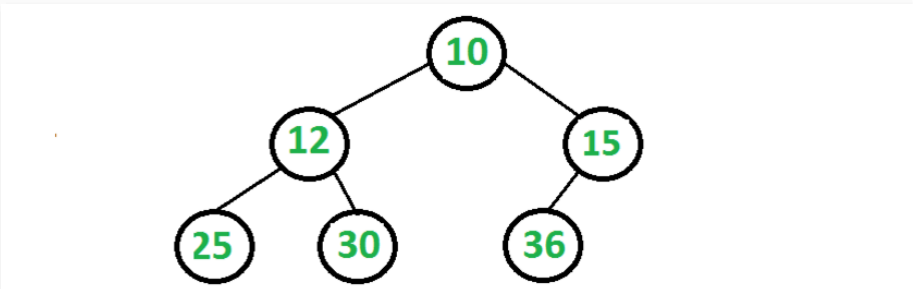
DFS 版本

```
List<List<Integer>> res=new ArrayList<>();
boolean[][] book;
int[][] next={{1,0},{0,1},{-1,0},{0,-1}};
public void helper(int[][] matrix,int val,int x,int y)
{
    for(int k=0;k<4;k++)
    {
        int tx=x+next[k][0];
        int ty=y+next[k][1];
        if(tx<0||tx>=matrix.length||ty<0||ty>=matrix[0].length)
        continue;
        if(book[tx][ty]==false && matrix[tx][ty]==val)
        {
            book[tx][ty]=true;
            res.add(Arrays.asList(tx,ty));
            helper(matrix,val,tx,ty);
        }
    }
}
public List<List<Integer>> sulution(int[][]matrix, int a,int b)
{
    int val=matrix[a][b];
    book=new boolean[matrix.length][matrix[0].length];
    book[a][b]=true;
    res.add(Arrays.asList(a,b));
    helper(matrix,val,a,b);
    return res;
}
```

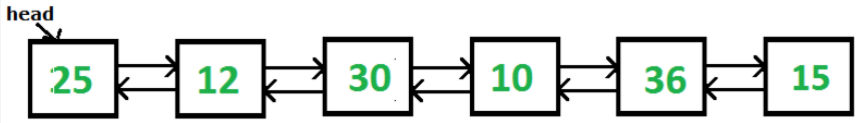
BFS 版本

```
List<List<Integer>> res=new ArrayList<>();
boolean[][] book;
int[][] next={{1,0},{0,1},{-1,0},{0,-1}};
public List<List<Integer>> sulution(int[][]matrix, int a,int b)
{
    int val=matrix[a][b];
    book=new boolean[matrix.length][matrix[0].length];
    Queue<List<Integer>> queue=new LinkedList<>();
    book[a][b]=true;
    res.add(Arrays.asList(a,b));
    queue.offer(Arrays.asList(a,b));
    while(queue.isEmpty()==false)
    {
        ArrayList a=queue.poll();
        int x=a.get(0);
        int y=a.get(1);
        for(int k=0;k<4;k++)
        {
            int tx=x+next[k][0];
            int ty=y+next[k][1];
            if(tx<0||tx>=matrix.length||ty<0||ty>=matrix[0].length)
            continue;
            if(book[tx][ty]==false&&matrix[tx][ty]==val)
            {
                book[tx][ty]=true;
                res.add(Arrays.asList(tx,ty));
                queue.offer(Arrays.asList(tx,ty));
            }
        }
    }
    return res;
```

12. BST to Doubly LinkedList



The above tree should be in-place converted to following Doubly Linked List(DLL).



The idea is to do inorder traversal of the binary tree. While doing inorder traversal, keep track of the previously visited node in a variable say *prev*. For every visited node, make it next of *prev* and previous of this node as *prev*.

```
class Node
{
    int data;
    Node left, right;
    public Node(int data)
    {
        this.data = data;
        left = right = null;
    }
}
class BinaryTree
{
    Node root;
    Node head;// head --> Pointer to head node of created doubly linked list
    static Node prev = null;
    void BinaryTree2DoubleLinkedList(Node root)
    {
        if (root == null)
            return;
        BinaryTree2DoubleLinkedList(root.left);
        if (prev == null)
            head = root;
        else
        {
            root.left = prev;
            prev.right = root;
        }
        prev = root;
        BinaryTree2DoubleLinkedList(root.right);
    }
}
```

13. K nearest point 最大堆

```java
public class Problem1
{
    public class Point
    {
        public double x;
        public double y;
    }

    public class Elem
    {
        public Point p;
        public double dis;
        public Elem(Point x,double y)
        {
            this.p=x;
            this.dis=y;
        }
    }
    public List<Point> KNN(Point [] a,Point target,int k)
    {
        PriorityQueue<Elem> heap=new PriorityQueue<>(new Comparator<Elem>() {
                  public int compare(Elem e1, Elem e2) {
                    return (e2.dis - e1.dis);
                  }
                });
        int num=0;
         for (int i=0;i<a.length;++i)
         {
             double dist,dx,dy;
             dx=(a[i].x-target.x);
             dy=(a[i].y-target.y);
             dist=Math.sqrt(dx*dx+dy*dy);
             if (num<k)
             {
                 heap.offer(new Elem(a[i],dist));
                 ++num;
             }
             else
             {
                 Elem u=heap.peek();
                 if (u.dis>dist)
                 {
                     head.poll();
                     heap.offer(new Elem(a[i],dist));
                 }
             }
         }
         LinkedList<Point> res=new LinkedList<>();
         for (Elem v:heap)
         {
             res.add(v.p);
         }
         return res;
     }
}
```

Find K Nearest Point

```java
import java.util.PriorityQueue;
import java.util.Comparator;

public class kNearestPoint {
    public Point[] Solution(Point[] array, Point origin, int k) {
        Point[] rvalue = new Point[k];
        int index = 0;
        PriorityQueue<Point> pq = new PriorityQueue<Point> (k, new Comparator<Point> () {
            @Override
            public int compare(Point a, Point b) {
                return (int) (getDistance(a, origin) - getDistance(b, origin));
            }
        });

        for (int i = 0; i < array.length; i++) {
            pq.offer(array[i]);
            if (pq.size() > k)
                pq.poll();
        }
        while (!pq.isEmpty())
            rvalue[index++] = pq.poll();
        return rvalue;
    }
    private double getDistance(Point a, Point b) {
        return Math.sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
    }
}
```

14. Lowest common ancestor

```java
public class Solution
{
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q)
    {
        if(root==null||root==p||root==q)
        return root;
        TreeNode left=lowestCommonAncestor(root.left,p,q);
        TreeNode right=lowestCommonAncestor(root.right,p,q);
        if(left!=null&&right!=null)
        return root;
        if(left!=null)
        return left;
        if(right!=null)
        return right;
        return null;
    }
}
```

parent node

```
class Node
{
    int key;
    Node left, right, parent;

    Node(int key)
    {
        this.key = key;
        left = right = parent = null;
    }
}
public int depth(Node node)
    {
        int d = -1;
        while (node != null)
        {
            d++;
            node = node.parent;
        }
        return d;
    }
public Node LCA(Node n1,Node n2)
{
    int d1=depth(n1);
    int d2=depth(n2);
    if(d2>d1){
        Node temp=n1;
        n1=n2;
        ne=temp;
    }
    int diff=d1-d2;
    while(diff>0){
        n1=n1.parent;
        diff--;
    }
    while(n1!=null&&n2!=null){
    if(n1==n2)
    return n1;
    n1=n1.parent;
    n2=n2.parent;
    }
    return null;
}
```

15. Lowest common ancestor in BST

```
public class Solution
{
    public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q)
    {
        if(root==null||root.val==p.val||root.val==q.val)
        return root;
        if(p.val<root.val&&q.val>root.val)
        return root;
        if(p.val<root.val&&q.val<root.val)
        return lowestCommonAncestor(root.left,p,q);
        if(p.val>root.val&&q.val>root.val)
        return lowestCommonAncestor(root.right,p,q);

        return root;
    }
}
```

16. 给一个 array，找出出现频率最高的一个。

```
public int solution(int[] array)
{
    HashMap<Integer,Integer> map=new HashMap<>();
    for(int i=0;i<array.length;i++){
        if(map.containsKey(array[i])==false){
            map.put(array[i],1);
        }
        else{
            int val=map.get(array[i]);
            val++;
            map.put(array[i],val);
        }
    }
    int maxfreq=-1;
    int max=Integer.MIN_VALUE;
    for(Map.Entry<Integer,Integer> entry:map.entrySet()){
        int temp=entrySet.getValue();
        if(temp>max){
            max=temp;
            maxfreq=entrySet.getKey();
        }
    }
    return maxfreq;
}
```

17. two sum

```java
public class Solution
{
    public int[] twoSum(int[] nums, int target)
    {
        HashMap<Integer,Integer> map=new HashMap<>();
        int[] a=new int[2];
        int i;
        for(i=0;i<nums.length;i++)
        {
            if(map.containsKey(nums[i])==false)
            {
                map.put(target-nums[i],i);
            }
            else
            {
                a[0]=map.get(nums[i]);
                a[1]=i;
            }

        }
        return a;

    }
}
```

18. Leetcode 237, 给一个 node，删了他

```java
public void deleteNode(ListNode node) {
    node.val = node.next.val;
    node.next = node.next.next;
}
```

19. Leetcode Closest Binary Search Tree Value

```
10 public class Solution {
11     public int min;
12     public double mindouble;
13     public void helper(TreeNode root, double target)
14     {
15         if(root==null)
16         return;
17         if(Math.abs(target-root.val)<mindouble){
18             mindouble=Math.abs(target-root.val);
19             min=root.val;
20         }
21         if(Math.abs(target-root.val)<0.00001)
22         return;
23         if(target-root.val<0)
24         helper(root.left,target);
25         if(target-root.val>0)
26         helper(root.right,target);
27     }
28     public int closestValue(TreeNode root, double target) {
29         min=root.val;
30         mindouble=Math.abs(root.val-target);
31         helper(root,target);
32         return min;
33     }
34 }
```

20. Leetcode 49 Group Anagrams

```java
public class Solution
{

    public List<List<String>> groupAnagrams(String[] strs)
    {
        List<List<String>> result=new ArrayList<>();
        HashMap<String,ArrayList<String>> map=new HashMap<>();
        int i;
        for(i=0;i<strs.length;i++)
        {
            String s=strs[i];
            char[] c=s.toCharArray();
            Arrays.sort(c);
            String t=String.copyValueOf(c);
            if(map.containsKey(t)==true)
            {
                ArrayList<String> list=map.get(t);
                list.add(s);
                map.put(t,list);
            }
           else
            {
                ArrayList<String> list1=new ArrayList<>();
                list1.add(s);
                map.put(t,list1);
            }
        }
        for(Map.Entry<String,ArrayList<String>> p:map.entrySet())
        {
            result.add(p.getValue());
        }
        return result;
    }
}
```

21. BFS 遍历二叉树

```java
public List<Integer> BFSofTree(TreeNode root)
{
    List<Integer> res=new ArrayList<>();
    if(root==null)
    return res;
    Queue<TreeNode> queue=new LinkedList<>();
    queue.offer(root);
    while(queue.isEmpty()==false)
    {
        TreeNode node=queue.poll();
        res.add(node.val);
        if(node.left!=null)
        queue.offer(root.left);
        if(node.right!=null)
        queue.offer(root.right);
    }
    return res;
}

```

22. 找 median 最大堆 r

```
class MedianFinder {
    // max queue is always larger or equal to min queue
    PriorityQueue<Integer> min = new PriorityQueue();
    PriorityQueue<Integer> max = new PriorityQueue(1000, Collections.reverseOrder());
    // Adds a number into the data structure.
    public void addNum(int num) {
        max.offer(num);
        min.offer(max.poll());
        if (max.size() < min.size()){
            max.offer(min.poll());
        }
    }

    // Returns the median of current data stream
    public double findMedian() {
        if (max.size() == min.size()) return (max.peek() + min.peek()) /  2.0;
        else return max.peek();
    }
};
```

23. Wiggle Sort

```
public class Solution {
    public void wiggleSort(int[] nums) {
        for(int i=1;i<nums.length;i++){
            if(i%2==1){
                if(nums[i-1]>nums[i])
                swap(nums,i);
            }
            if(i%2==0){
                if(nums[i-1]<nums[i])
                swap(nums,i);
            }
        }
    }
    public void swap(int[] nums,int i){
        int temp=nums[i];
        nums[i]=nums[i-1];
        nums[i-1]=temp;
    }
}
```

严格小于

```java
public class Solution {
    public void wiggleSort(int[] nums) {
        Arrays.sort(nums);
        int x=nums.length;
        int mid=(x-1)/2;
        int median=nums[mid];
        int s=1;
        int e=0;
        if (x%2==0)
            e=x-2;
        else
            e=x-1;
        while (s<e)
        {
            int temp=nums[s];
            nums[s]=nums[e];
            nums[e]=temp;
            s+=2;
            e-=2;
        }
         e=x-1-x%2;
         for (int i=2;i<x;i+=2)
         {
             if (nums[i]!=median||i==s) continue;
             nums[i]=nums[s];
             nums[s]=median;
             s+=2;
         }
         for (int i=e-2;i>0;i-=2)
         {
             if (nums[i]!=median||i==e) continue;
             nums[i]=nums[e];
             nums[e]=median;
             e-=2;
         }
     }
}
```

24. symmetric tree 回文树

```java
public class Solution {
    private boolean helper(TreeNode left,TreeNode right){
        if(left==null&&right==null)
        return true;
        if(left==null)
        return false;
        if(right==null)
        return false;
        if(left.val!=right.val)
        return false;
        boolean a=helper(left.left,right.right);
        boolean b=helper(left.right,right.left);
        if(a==true&&b==true)
        return true;
        else
        return false;
    }
    public boolean isSymmetric(TreeNode root) {
        if(root==null)
        return true;
        return(helper(root.left,root.right));
    }
}
```

25. Remove Duplicates from Sorted Array

```java
public class Solution
{
    public int removeDuplicates(int[] nums)
    {
        if(nums==null||nums.length==0)
        return 0;
        int k=0;
        int i;
        for(i=1;i<nums.length;i++)
        {
            if(nums[k]!=nums[i])
            {
                k++;
                nums[k]=nums[i];
            }
        }
        return k+1;
    }

}
```

变形

```
public class Solution
{
    public int removeDuplicates(int[] nums)
    {
        int p=0;
        int k=0;
        int i;
        if(nums==null||nums.length==0)
        return 0;
        for(i=1;i<nums.length;i++)
        {
            if(nums[i]==nums[k])
            {
                p++;
                if(p<2)
                {
                    k++;
                    nums[k]=nums[i];
                }
                else
                continue;
            }
            else
            {
                p=0;
                k++;
                nums[k]=nums[i];
            }
        }
        return k+1;
    }
}
```

26. merge sort in sorted list

```java
public class Solution
{
    public ListNode deleteDuplicates(ListNode head)
    {
       if(head==null)
       return null;
       ListNode post=head.next;
       ListNode t=head;
       while(post!=null&&t!=null)
       {
           if(post.val==t.val)
           {
               t.next=post.next;
               post=post.next;
           }
           else
           {
               post=post.next;
               t=t.next;
           }
       }
        return head;
    }
}
```

变形,全部删了

```java
public class Solution
{
    public ListNode deleteDuplicates(ListNode head)
    {
        if(head==null)
        return null;
        ListNode dummy=new ListNode(0);
        dummy.next=head;
        ListNode node;
        node=dummy;
        while(node.next!=null&&node.next.next!=null)
        {
            if(node.next.val==node.next.next.val)
            {
                int val=node.next.val;
                while(node.next!=null&&node.next.val==val)
                node.next=node.next.next;
            }
            else
            node=node.next;

        }
        return dummy.next;

    }
}
```

27, merge two sorted array

```java
public class Solution {
    public void merge(int[] nums1, int m, int[] nums2, int n) {
       int i=m-1;
       int j=n-1;
       int index=m+n-1;
       while(i>=0&&j>=0){
           if(nums1[i]>=nums2[j]){
               nums1[index]=nums1[i];
               i--;
               index--;
           }
           else{
               nums1[index]=nums2[j];
               j--;
               index--;
           }
       }
       while(i>=0){
           nums1[index]=nums1[i];
           i--;
           index--;
       }
       while(j>=0){
           nums1[index]=nums2[j];
           j--;
           index--;
       }
    }
}
```

28. LRU cache

```java
public class LRUCache {

    class Node{
        int key;
        int value;
        Node pre;
        Node next;
        public Node(int key,int value){
            this.key=key;
            this.value=value;
        }
    }
    int size;
    int count;
    HashMap<Integer,Node> map;
    Node head;
    Node tail;
```

```
18      public LRUCache(int capacity) {
19          this.size=capacity;
20          count=0;
21          map=new HashMap<>();
22          head=new Node(0,0);
23          tail=new Node(0,0);
24          head.pre=null;
25          head.next=tail;
26          tail.pre=head;
27          tail.next=null;
28      }
29      public void deleteNode(Node node){
30          node.pre.next=node.next;
31          node.next.pre=node.pre;
32      }
33      public void addtoHead(Node node){
34          node.next=head.next;
35          node.next.pre=node;
36          node.pre=head;
37          head.next=node;
38      }
39      public int get(int key) {
40          if(map.get(key)!=null){
41              Node node=map.get(key);
42              int res=node.value;
43              deleteNode(node);
44              addtoHead(node);
45              return res;
46          }
47          else
48          return -1;
49      }
50       public void put(int key, int value) {
51           if(map.get(key)!=null){
52               Node node=map.get(key);
53               node.value=value;
54               deleteNode(node);
55               addtoHead(node);
56           }
57           else{
58               Node node=new Node(key,value);
59               map.put(key,node);
60               if(count<size){
61                   count++;
62                   addtoHead(node);
63               }
64               else{
65                   map.remove(tail.pre.key);
66                   deleteNode(tail.pre);
67                   addtoHead(node);
68               }
69           }
70       }
71  }
```

29. 一个 tree 是不是另一个 tree 的 subtree

```
public void inorder(TreeNode node, List<Character> list){
    if(node == null)
    return;
    inorder(node.left);
    list.add(node.val);
    inorder(node.right);
}
public void preorder(TreeNode node, List<Character> list){
    if(node == null)
    return;
    list.add(node.val);
    preorder(node.left);
    preorder(node.right);
}
public boolean isSublist(ArrayList<Character> list1, ArrayList<Character> list2){
    for(int i=0;i<list1.size();i++){
        int k=0;
        for(int j=0;j<list2.size();j++){
            if(list1.get(i)!=list2.get(j))
            break;
            k++;
        }
        if(k==list2.size())
        return true;
    }
    return false;
}
public boolean isSubtree(TreeNode node1,TreeNode node2){
    List<Integer> preorderNode1 = new ArrayList<>();
    List<Integer> inorderNode1 = new ArrayList<>();
    List<Integer> preorderNode2 = new ArrayList<>();
    LIst<Integer> inorderNode2 = new ArrayList<>();
    inorder(node1,inorderNode1);
    inorder(node1,preorderNode1);
    preorder(node2,inorderNode1);
    preorder(node2,preorderNode2);
    if(isSublist(preorderNode1,preorderNode2)==true&&isSublist(inorderNode1,inorderNode2)==true)
    return true;
    else
    return false;
}
```

30. Letter Combinations of a Phone Numbe

```java
public class Solution
{
    List<String> result=new ArrayList<>();
    HashMap<Character,char[]> map=new HashMap<>();
    private void helper(String digits,StringBuilder sb)
    {
        if(sb.length()==digits.length())
        {
            result.add(sb.toString());
            return;
        }
        for(char c:map.get(digits.charAt(sb.length())))
        {
            sb.append(c);
            helper(digits,sb);
            sb.deleteCharAt(sb.length()-1);
        }
    }
     public List<String> letterCombinations(String digits)
     {
         if(digits.length()==0)
         return result;
         map.put('1',new char[]{});
         map.put('2',new char[]{'a','b','c'});
         map.put('3',new char[]{'d','e','f'});
         map.put('3', new char[] { 'd', 'e', 'f' });
         map.put('4', new char[] { 'g', 'h', 'i' });
         map.put('5', new char[] { 'j', 'k', 'l' });
         map.put('6', new char[] { 'm', 'n', 'o' });
         map.put('7', new char[] { 'p', 'q', 'r', 's' });
         map.put('8', new char[] { 't', 'u', 'v'});
         map.put('9', new char[] { 'w', 'x', 'y', 'z' });
         StringBuilder sb=new StringBuilder();
         helper(digits,sb);
         return result;

     }
}
```

31. 3Sum

```java
public class Solution {
    public List<List<Integer>> threeSum(int[] nums)
    {
        Arrays.sort(nums);
        int i,j;
        List<List<Integer>> result=new ArrayList<>();
        HashSet<ArrayList<Integer>> set=new HashSet<>();
        for(i=0;i<=nums.length-3;i++)
        {
            int b=-nums[i];
            int head=i+1;
            int tail=nums.length-1;
            while(head<tail)
            {
                if(nums[head]+nums[tail]<b)
                head++;
                if(nums[head]+nums[tail]>b)
                tail--;
                if(head==tail)
                break;
                if(nums[head]+nums[tail]==b)
                {
                    ArrayList<Integer> a=new ArrayList<>();
                    a.add(nums[i]);
                    a.add(nums[head]);
                    a.add(nums[tail]);
                    if(set.contains(a)==false)
                    {
                        result.add(a);
                        set.add(a);
                    }
                    head++;
                }
            }

        }
        return result;

    }
}
```

32. minstack

```java
public class MinStack {

    /** initialize your data structure here. */
    Stack<Integer> min;
    Stack<Integer> s;
    public MinStack() {
        min=new Stack<Integer>();
        s=new Stack<Integer>();

    }

    public void push(int x) {
        if(x<=getMin())
        min.push(x);
        s.push(x);
    }

    public void pop() {
        int value=s.pop();
        if(value==getMin())
        min.pop();
    }

    public int top() {
        return (int)s.peek();
    }

    public int getMin() {
        if(min.empty()==true)
        return Integer.MAX_VALUE;
        else
        return min.peek();
    }
}
```

或者用 linkedlist

```java
public class MinStack {

    /** initialize your data structure here. */
    LinkedList<Integer> stack;
    LinkedList<Integer> minStack;
    public MinStack() {
        stack = new LinkedList<>();
        minStack = new LinkedList<>();
    }

    public void push(int x) {
        stack.addLast(x);
        if(minStack.size()==0||x<=minStack.get(0)){
            minStack.addFirst(x);
        }
    }

    public void pop() {
        int temp = stack.removeLast();
        if(temp==minStack.get(0)){
            minStack.removeFirst();
        }
    }

    public int top() {
      return stack.get(stack.size()-1);
    }

    public int getMin() {
        return minStack.get(0);
    }
}
```

33. reverse words in string

```java
public class Solution {
    public String reverseWords(String s) {
        String res=new String();
        if(s.length()==0||s==null)
        return res;
        List<String> list = new ArrayList<>();
        StringBuilder temp=new StringBuilder();
        for(int i=0;i<s.length();i++){
            if(s.charAt(i)==' ')
            {
                if(temp.length()!=0)
                list.add(temp.toString());
                temp.delete(0,temp.length());
            }
            else{
                temp.append(s.charAt(i));
            }
            if(i==s.length()-1&&s.charAt(i)!=' '){
                list.add(temp.toString());
            }
        }
        if(list.size()!=0)
        res=list.get(list.size()-1);
        for(int i=list.size()-2;i>=0;i--)
        res=res+" "+list.get(i);
        return res;
    }
}
```

The input string does not contain leading or trailing spaces and the words are always separated by a single space.

```java
public class Solution
{
    public void reverseWords(char[] s)
    {
        helper(s,0,s.length-1);
        int k=0;
        for(int i=0;i<s.length;i++){
            if(s[i]==' '){
                helper(s,k,i-1);
                k=i+1;
            }
        }
        helper(s,k,s.length-1);
    }
    public void helper(char[] s,int a,int b){
        int start=a;
        int end=b;
        while(start<end){
            char temp=s[start];
            s[start]=s[end];
            s[end]=temp;
            start++;
            end--;
        }
    }
}
```

34. Integer to English words

```
private final String[] LESS_THAN_20 = {"", "One", "Two", "Three", "Four", "Five", "Six", "Seven", "Eight", "Nine", "Ten", "Eleven", "Twelve", "Thirteen", "Fourteen", "Fifteen", "Sixteen", "Seventeen", "Eighteen", "Nineteen"};
private final String[] TENS = {"", "Ten", "Twenty", "Thirty", "Forty", "Fifty", "Sixty", "Seventy", "Eighty", "Ninety"};
```

```
public static String helper(int num){
    if(num==0)
        return "";
    if(num<20)
        return LESS_THAN_20[num];
    if(num<100)
        return TENS[num/10]+" "+helper(num%10);
    else
        return LESS_THAN_20[num/100]+" Hundred "+ helper(num%100);
}
```

35. binary tree order level traversal

```
10  public class Solution
11  {
12      public ArrayList<List<Integer>> levelOrder(TreeNode root)
13      {
14          ArrayList result=new ArrayList();
15          if(root==null)
16          return result;
17          Queue<TreeNode> queue=new LinkedList<TreeNode>();
18          queue.offer(root);
19          while(!queue.isEmpty())
20          {
21              int size=queue.size();
22              int i;
23              ArrayList<Integer> level=new ArrayList<Integer>();//用level来保存在这一层里面的所有元素
24              for(i=0;i<size;i++)
25              {
26                  TreeNode head=queue.poll();//poll用来返回头元素并删除头元素,注意这个是写在循环里面
27                  level.add(head.val);
28                  if(head.left!=null)
29                  queue.offer(head.left);
30                  if(head.right!=null)
31                  queue.offer(head.right);
32              }
33              result.add(level);//将这一层里面的元素添加到总的数组里面去
34          }
35          return result;
36      }
37  }
```

36. implement queue using array

```
class Myqueue{
    int [] queue;
    int head;
    int tail;
    int size;
    public Myqueue(int x){
        queue=new int[x];
        size=x;
        head=0;
        tail=0;
    }
    public void enqueue(int x){
        int pos=tail%size;
        queue[pos]=x;
        tail++;
    }
    public int dequeue(){
        int pos=head%size;
        head++;
        return queue[pos];
    }

}
```

37. Find K Pairs with Smallest Sums

```
public class Solution {
    public List<int[]> kSmallestPairs(int[] nums1, int[] nums2, int k) {
        PriorityQueue<List<Integer>> maxHeap = new PriorityQueue<>(new Comparator<List<Integer>>(){
            public int compare(List<Integer> a,List<Integer> b){
                return b.get(0)+b.get(1)-(a.get(0)+a.get(1));
            }
        });
        int num=0;
        for(int i=0;i<nums1.length;i++){
            for(int j=0;j<nums2.length;j++){
                List<Integer> res=new ArrayList<>();
                if(num<k){
                    res.add(nums1[i]);
                    res.add(nums2[j]);
                    maxHeap.add(res);
                    num++;
                }
                else{
                    List<Integer> temp=new ArrayList<>();
                    temp=maxHeap.peek();
                    if(nums1[i]+nums2[j]<temp.get(0)+temp.get(1))
                    {
                        maxHeap.poll();
                        res.add(nums1[i]);
                        res.add(nums2[j]);
                        maxHeap.add(res);
                    }
                }
            }
        }
```

```
        List<int[]> list=new ArrayList<>();
        if(maxHeap.size()==0)
        return list;
        while(maxHeap.size()!=0){
            List<Integer> a=new ArrayList<>();
            int[] result=new int[2];
            a=maxHeap.poll();
            result[0]=a.get(0);
            result[1]=a.get(1);
            list.add(result);
        }
        return list;
    }
}
```

38.postfix notation

```
    }
    public static int postfix(char[] input){
        Stack<Integer> stack = new Stack();
        for(int i=0;i<input.length;i++){
            if(input[i]=='+' || input[i]=='-'||input[i]=='*'){
                if(stack.size()>=2){
                    int s1 = stack.pop();
                    int s2 = stack.pop();
                    if(input[i]=='+')
                        stack.push(s1+s2);
                    if(input[i]=='-')
                        stack.push(s1-s2);
                    if(input[i]=='*')
                        stack.push(s1*s2);
                }
                else
                    return -1;

            }
            else if(input[i]>='0'&&input[i]<='9')
                stack.push(Character.getNumericValue(input[i]));
            else
                return -1;
        }
        if(stack.size()>1)
            return -1;
        return stack.pop();
    }
```

简单版

```java
public class Solution {
    public int evalRPN(String[] tokens) {
        Stack<Integer> stack = new Stack<>();
        for(String t: tokens){
            if(t.equals("+")||t.equals("-")||t.equals("*")||t.equals("/")){
            int a = stack.pop();
            int b = stack.pop();
            if(t.equals("+"))
            stack.push(a+b);
            if(t.equals("-"))
            stack.push(b-a);
            if(t.equals("*"))
            stack.push(a*b);
            if(t.equals("/"))
            stack.push(b/a);
        }
        else
            stack.push(Integer.parseInt(t));
        }
        return stack.pop();
    }
}
```

39. swap kth and the last kth

可以直接交换值：

```
public ListNode swap(ListNode head, int k) {
    ListNode itr = head;
    for(int i = 0; i < k - 1; i++) {
        itr = itr.next;
        if(itr = null) //longer
            return head;
    }
    ListNode temp = itr;
    ListNode slow = head;
    while(itr != null) {
        slow = slow.next;
        itr = itr.next;
    }
    int value = slow.val;
    slow.val = temp.val;
    temp.val = value;

    return head;
```

如果不可以换的话

```
13 public class swap(ListNode head,int k){
14     ListNode dummy=new ListNode(0);
15     dummy.next=head;
16     ListNode fast=dummy;
17     for(int i=0;i<k-1;i++)
18     fast=fast.next;     //fast=1
19     ListNode l1=fast;     //l1=1
20     fast=fast.next;     //fast=2
21     ListNode slow=dummy;
22     while(fast.next!=null){
23         slow=slow.next;     //slow=5
24         fast=fast.next;
25     }
26     ListNode l2=slow;     //l2=5
27     slow=slow.next;     //slow=6
28     //swap 2 and 6
29     fast=l1.next;
30     ListNode temp=slow.next;    //temp=7
31     l1.next=slow;     // 1->6
32     if (fast.next!=slow)
33         slow.next=fast.next;
34     else
35         slow.next=fast;     //6->3
36     fast.next=temp;     //2->7
37     l2.next=fast; //5->2
38
39     return dummy.next;
40
41 }
```

40. Populating Next Right Pointers in Each Node

```
 9 public class Solution {
10     public void connect(TreeLinkNode root) {
11         if(root==null)
12         return;
13         while(root.left!=null){
14             TreeLinkNode cur = root;
15             while(cur!=null){
16                 cur.left.next = cur.right;
17                 if(cur.next!=null){
18                 cur.right.next = cur.next.left;
19                 }
20                 cur = cur.next;
21             }
22             root = root.left;
23         }
24     }
25 }
```

41. Subarray Sum Equals k (LC325)

```
public class Solution
{
    public int maxSubArrayLen(int[] nums, int k)
    {
        HashMap<Integer,Integer> map=new HashMap<>();
        int i;
        int sum=0;
        int max=0;
        for(i=0;i<nums.length;i++)
        {
            sum=sum+nums[i];
            if(sum==k)
            max=i+1;
            else if(map.containsKey(sum-k)==true) //上面那个肯定比下面这个长
            max=Math.max(max,i-map.get(sum-k));
            if(map.containsKey(sum)==false)
            map.put(sum,i);

        }
        return max;
    }
}
```

42. Merge k Sorted Lists’

复杂度：nlogk

```
public class Solution {
    public ListNode mergeKLists(ListNode[] lists) {
        PriorityQueue<ListNode> queue = new PriorityQueue<ListNode>(new Comparator<ListNode>(){
             public int compare(ListNode l1,ListNode l2){
                 return l1.val-l2.val;
             }
            });
            for(int i=0;i<lists.length;i++){
                if(lists[i]!=null)
                queue.add(lists[i]);
            }
            ListNode dummy = new ListNode(0);
            ListNode tail = dummy;
            while(queue.isEmpty()==false){
                ListNode node = queue.poll();
                tail.next = node;
                tail = tail.next;
                if(node.next!=null)
                queue.offer(node.next);
            }
            return dummy.next;
    }
}
```

43. gcd

```
public class Solution {
    int [][]mat;
    public initialize(int n)
    {
        mat=new int[n][m];
        for (int i=0;i<n;++i)
            mat[i][0]=i;
        for (int i=1;i<n;++i)
        {
            for (int j=1;j<=i;++j)
            {
                int x=i%j;
                int y=j;
                if (x<y)
                {
                    int temp=y;
                    y=x;
                    x=temp;
                }
                mat[i][j]=mat[x][y];
            }
        }
    }
    public gcd(int x, int y)
    {
        return mat[x][y];
    }
}
```