Double time

1.

Dpquestion

1.Lonest Repeating Replaceement

2.Longest Palindromic substring

3.Palindromic Substring

4.Longest Palindromic Subsequence

5. wildcard string matching

6. Max sum with no. two element are adjacent

Min& Max ELEMENT

Take two variable min and max using for loop get the result.

Reverse Array

Take first and last element of array as i&j. while (i<=j) arr i++ into temp and temp into arr j—then i++ j—and result

while(i<=j){

            int temp=a[i];

            a[i]=a[j];

            a[j]=temp;

            i++;

            j--;

        }

Max Subarray

Assign sum=0, maxsum=a[0]. Get for loop sum all array if sum<0 give sum=0.

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

            sum+= a[i];

            maxsum= Math.max(maxsum,sum);

            if(sum<0){

                sum=0;

            }

Contain Duplicates

Using haspmap if !conatain add value otherwise false.

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

        for(int i=0;i<a.length;i++){

            if(!map.containsKey(a[i])){

                map.put(a[i],1);

            }else{

               s=true;

            }

        }

Chocolate Distribution

Sort the array. Take index from 0to length-m and take diff i to i+m-1.

for(int i=0;i<=arr.length-m;i++){

            diff=arr[i+m-1]-arr[i];

            minDiff=Math.min(minDiff, diff);

        }

Search in Rotated Sorted Array

Firstly we check is left half is sorted or not, if sorted then if a[low]<=m<=a[mid] this yes then go left otherwise rightside.similarly for right side. If high<low then -1.

 while(low<=high){

            int mid=low+(high-low)/2;

            if(a[mid]==m){

                System.out.println(mid);

                return;

            }

            if(a[low]<=a[mid]){

                if(m>=a[low] && m<=a[mid]){

                    high=mid-1;

                }else{

                    low=mid+1;

                }

            }

            if(a[mid]<=a[high]){

                if(m>=a[mid] && m<=a[high]){

                    low=mid+1;

                }else{

                    high=mid-1;

                }

            }

        }

Next Permutation

Firstly we compare with two element and find lower element from right side. Then greater than that element and swap them and reverse i+1 to till last.

 int i=a.length-2;

        while(i>=0 && a[i]>=a[i+1])i--;

        if(i>=0){

            int j=a.length-1;

            while(a[j]<=a[i])j--;

            swap(a,i,j);

        }

        reverse(a, i+1, a.length-1);

Buy and sell stock

Firstly we find min element from starting diff at each step least min. then find max diff.

 int diff=0,mmin=a[0],maxdiff=Integer.MIN\_VALUE;

        for(int i=0;i<a.length;i++){

            mmin=Math.min(a[i],mmin);

            diff=a[i]-mmin;

            maxdiff=Math.max(maxdiff,diff);

        }

Repeating and Missing No.-

Using Hashmap integer and Boolean.

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

        for(Integer i : arr){

            if(map.get(i)==null){

                map.put(i,true);

            }else{

                System.out.println("Repeatng no"+ i);

            }

        }

        for(int i=1;i<arr.length;i++){

            if(map.get(i)==null){

                System.out.println("Missing no. "+i);

            }

        }

K largest Element

Using priority queue firstly k no. of item.Pq.peek() contain min no. of pq we will

Linear iterative if higher then we change it.

PriorityQueue<Integer> pq= new PriorityQueue<>();

        for(int i=0;i<k;i++){

            pq.add(arr[i]);

        }

        for(int i=k;i<arr.length;i++){

            if(pq.peek()<arr[i]){

                pq.poll();

                pq.add(arr[i]);

            }

        }

Rain trapping water

Assign left[0] and right[arr.length-1] then Math.max for left i-1,i and right i,i+1

Then math.min from left I and right I substract arr[i]. and add it.

 left[0]=arr[0];

        right[arr.length-1]=arr[arr.length-1];

        for(int i=1;i<arr.length;i++){

            left[i]=Math.max(left[i-1],arr[i]);

        }

        for(int i=arr.length-2;i>=0;i--){

            right[i]=Math.max(arr[i],right[i+1]);

        }

        int ans=0;

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

            ans += Math.min(left[i], right[i])-arr[i];

        }

Product Array except itself

From right we multiply and put into new array at each step.tae prod=1and multiply next step new array makers and get answer and last one see check.//

  int[] right=new int[a.length+1];

        int[] left=new int[a.length];

        int prod=1;

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

            right[i]=prod\*a[i];

            prod=prod\*a[i];

        }

        prod=1;

        right[a.length]=1;

        for(int i=0;i<a.length;i++){

            int lp=prod;

            int rp=right[i+1];

            left[i]=lp\*rp;

            prod=prod\*a[i];

        }

Max product Subarray

If a[i]<0 then swap otherwise multiply matn.max and min with a[i], a[i]\*max/min and assign answer at each step.

  int min=a[0],max=a[0],ans=a[0];

        for(int i=1;i<a.length;i++){

            if(a[i]<0){

                int temp=max;

                max=min;

                min=temp;

            }

            max=Math.max(a[i],max\*a[i]);

            min=Math.min(a[i], min\*a[i]);

            ans=Math.max(ans,max);

        }

Min in Rotated Array

It is same as simple binary search but one condition if(a[mid]-1>a[mid] then print the answer.

  while(low<=high){

            int mid=low+(high-low/2);

            m=Math.min(m,a[mid]);

            if(a[mid-1]>a[mid]){

                System.out.println(m);

            }

            if(a[mid]>a[high]){

                low=mid+1;

            }else{

                high=mid-1;

            }

        }

Find pair sum

*You can do it easily.*

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

        for(int i=0;i<a.length;i++){

            int s=sum-a[i];

            if(map.containsKey(s)){

                m=1;

            }else{

                map.put(a[i],1);

            }

        }

        if(m==0){

            System.out.println(false);

        }else{

            System.out.println(true);

        }

Contain with most water

We have to find area from min of left right and using while left—right++

 int left=0,right=he.length-1;

        int area=0;

        while(left<=right){

            int w=right-left;

            int h=Math.min(he[left],he[right]);

            area=Math.max(area, w\*h);

            if(he[right]<he[left]){

                right--;

            }else if(he[right]>he[left]){

                left++;

            }else{

                right--;

                left++;

            }

        }

3 Sum

A+b+c=0=> a=-(b+c) first sort the array. for loop finding- a while loop- b,c all will be distinctive we have to move further by using if and while loop.

Arrays.sort(a);

        for (int i = 0; i < a.length - 3; i++) {

            if (i == 0 || a[i] != a[i - 1]) {

                int b = -a[i];

                int l = i + 1, h = a.length - 1;

                while (l < h) {

                    int c = a[l] + a[h];

                    if (b == c) {

                        System.out.println(a[i] + "," + a[l] + "," + a[h]);

                        while (l < h && a[l] == a[l+ 1])

                            l++;

                        while (l < h && a[h] == a[h- 1])

                            h--;

                        l++;

                        h--;

                    } else if (c < b) {

                        l++;

                    } else {

                        h--;

                    }

                }

            }

        }

Kth Smallest No.

Similar as k largest element but we use collection reverse pq.peek() contain largest element of pq

 PriorityQueue<Integer> pq= new PriorityQueue<Integer>(Collections.reverseOrder());

        for(int i=0;i<k;i++){

            pq.add(a[i]);

        }

        for(int i=k;i<a.length;i++){

            if(pq.peek()>a[i]){

                pq.poll();

                pq.add(a[i]);

            }

        }

Min. count to make palindrome

Take left as a 0 and right as a.length-1. If both are equal and left++ right—else which is lower come forward with add count each step everypoint.

 int left=0,right=a.length-1;

        int count=0;

        while(left<=right){

            if(a[left]>a[right]){

                right--;

                a[right]+=a[right+1];

                count++;

            }else if(a[left]<a[right]){

                left++;

                a[left]+= a[left-1];

                count++;

            }else{

                left++;

                right--;

            }

        }

Arrange array to biggest no.

Firstly, we change integer to string array.sort the array using compare method.then string builder to change string array to string and get maximum no.condition of 0 will be check.

  String[] s=new String[a.length];

        for(int i=0;i<a.length;i++){

            s[i]=String.valueOf(a[i]);

        }

        Arrays.sort(s,(d,b)->(b+d).compareTo(d+b));   //sort array in descending order

        StringBuilder sb=new StringBuilder();

        for(int i=0;i<s.length;i++){

            sb.append(s[i]);

        }

        String result=sb.toString();

        System.out.println(result.startsWith("0")?"0":result);

Valid Palindrome

change to uppercase int A=65 AND Z=90. If it underrange i++ j—else if i++ and j—else false

 while (i <= j) {

            if (s.charAt(i) == s.charAt(j)) {

                i++;

                j--;

            } else if ((int) s.charAt(i) <= 65 &&(int) s.charAt(i) >= 90) {

                i++;

            } else if ((int) s.charAt(j) <= 65 && (int) s.charAt(j) >= 90) {

                j--;

            } else {

                System.out.println(false);

            }

        }

Valid Anagram

change to lowercase.assign to int array of 26 length give value acc. to no. of letter in them.

if both are equal then true else false;

  s.toLowerCase();

        s.toUpperCase();

        int[] s1=new int[26];

        int[] t1=new int[26];

        if(s.length()!=t.length()){

            System.out.println(false);

        }

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

            s1[s.charAt(i)-'a']+=1;

            t1[t.charAt(i)-'a']+=1;

        }

        int m=0;

        for(int i=0;i<s1.length;i++){

            if(s1[i]!=t1[i]){

                m=1;

            }

        }

Valid Paranthesis

Take char array, l==-1 and interate for loop it is equal to that parenthesis give to char[++i] else close parenthesis i—- till i>=0.

 char[] m= new char[s.length()];

        int l=-1,b=0;

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

            if(s.charAt(i)=='{' ||s.charAt(i)=='(' || s.charAt(i)==']'){

                m[++l]=s.charAt(i);

            }else{

                if(l>-1){

                    if(m[l]=='{' && s.charAt(i)=='}' || m[l]=='(' && s.charAt(i)==')' || (m[l]=='[' && s.charAt(i)==']')){

                        l--;

                    }else{

                        b=1;

                        break;

                    }

                }else{

                    b=1;

                    break;

                }

            }

        }

Remove Consecutive Character

Take one empty string m and add m+=given.chart(0).

 String s="aaabb",m="";

        m+= s.charAt(0);

        int j=0;

        for(int i=1;i<s.length();i++){

            if(s.charAt(i)!=m.charAt(j)){

                j++;

                m+= s.charAt(i);

            }

        }

Print Duplicates

Using Hashmap count each no. of character and using map.entry and entry we woll show result please remember last step.

 String str = "test string";

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

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

            if(map.containsKey(str.charAt(i))){

                map.put(str.charAt(i),map.get(str.charAt(i))+1);

            }else{

                map.put(str.charAt(i), 1);

            }

        }

        for (Map.Entry<Character,Integer> mapElement : map.entrySet()) {   //iterating through the unordered map

            if (mapElement.getValue() > 1)   //if the count of characters is greater than 1 then duplicate found

              System.out.println(mapElement.getKey() + ", count = " + mapElement.getValue());

          }

Longest Sbstring Without repeating character

We use Hashset store each character and increase size. If it already exist we remove char from indexing 0.

HashSet<Character> s=new HashSet<>();

        int b=0,c=0,size=0;

        while(c<a.length()){

            if(!s.contains(a.charAt(c))){

                s.add(a.charAt(c));

                c++;

                size=Math.max(size,s.size());

            }else{

                s.remove(a.charAt(b));

                b++;

            }

        }

Longest Substring Replacement

Using hashmap find the element which comes maximum no. end-begin+1-count>k then we r

Remove one element from hasmap and moves one step.

 int start=0,end=0,maxlength=0,count=0;

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

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

            map.put(s.charAt(end), map.getOrDefault(s.charAt(end),0)+1);

            count=Math.max(count,map.get(s.charAt(end)));

            if(end-start+1 - count>k){

                map.put(s.charAt(start),map.get(s.charAt(start))-1);

                start++;

            }

            maxlength=Math.max(maxlength,end-start+1);

        }

Group Anagaram

Mzedaar h dekho

 HashMap<HashMap<Character,Integer>,ArrayList<String>> bmap= new HashMap<>();

         for(String str:s){     // for(int i=0;i<s.length;i++)

            HashMap<Character,Integer> fmap= new HashMap<>();

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

                fmap.put(str.charAt(i), fmap.getOrDefault(str.charAt(i), 0)+1);

            }

            if(!bmap.containsKey(fmap)){

                ArrayList<String> list= new ArrayList<>();

                list.add(str);

                bmap.put(fmap,list);

            }else{

                ArrayList<String> list= bmap.get(fmap);

                list.add(str);

            }

        }

ZigZAG traversal

Firstly we traverse rowwise and coloumn wise

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

            System.out.print(arr[k][0] + " ");

            // set row index for next

            // point in diagonal

            int i = k - 1;

            // set column index for

            // next point in diagonal

            int j = 1;

            /\* Print Diagonally upward \*/

            while (isValid(i, j))

            {

                System.out.print(arr[i][j] + " ");

                i--;

                // move in upright direction

                j++;

            }

            System.out.println("");

        }

        for (int k = 1; k < C; k++) {

            System.out.print(arr[R - 1][k] + " ");

            // set row index for next

            // point in diagonal

            int i = R - 2;

            // set column index for

            // next point in diagonal

            int j = k + 1;

            /\* Print Diagonally upward \*/

            while (isValid(i, j))

            {

                System.out.print(arr[i][j] + " ");

                // move in upright direction

                i--;

                j++;

            }

            System.out.println("");

        }

Set Matrix Zeros

two loop both are same condition change to value and vice-versa.

We used first row and first col to store value and then iterate with condition.

 for(int i=0;i<r;i++){

            if(arr[i][0]==0){

                col0=0;

            }

            for(int j=1;j<c;j++){

            if(arr[i][j]==0){

                arr[i][0]=arr[0][j]=0;

            }

        }

    }

        for(int i=r-1;i>=0;i--){

            if(col0==0){

                arr[i][0]=0;

            }

            for(int j=c-1;j>=0;j--){

                if(arr[i][0]==0 || arr[0][j]==0){

                    arr[i][j]=0;

                }

            }

        }

Spiral Matrix

While and for loop use . while loop has basic condition left right top bottom use and you know everything.

  int left = 0, top = 0, bottom = matrix.length-1, right = matrix[0].length-1;

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

        while(top <= bottom && left <= right){

            for(int i=left; i<=right; i++)

               list.add(matrix[top][i]);

            top++;

            for(int i=top; i<=bottom; i++)

                list.add(matrix[i][right]);

            right--;

            if (top <= bottom){

            for(int i=right; i>=left; i--)

                list.add(matrix[bottom][i]);

            bottom--;

            }

            if (left <= right){

            for(int i=bottom; i>=top; i--)

                list.add(matrix[i][left]);

            left++;

            }

        }

Word Search

Backtracking done in it. First chec basic condtion of row &col then if char==board[i][j]. if length==k then true else if all four condition equal then true Otherwise false;

 for(int row=0;row<s.length;row++){

            for(int col=0;col<s[0].length;col++){

                if(s[row][col]== word.charAt(0) && nif(s,word,row,col,0)){

                    b=1;

                }

            }

        }

        if(b==0){

            System.out.println(false);

        }else{

            System.out.println(true);

        }

    }

    public static boolean nif(char[][] board,String word, int i, int j, int k){

        if(i<0 || i>=board.length|| j<0 || j>= board[0].length){

            return false;

        }

        if(board[i][j] == word.charAt(k))

        {

            char temp = board[i][j];

            board[i][j]='#';

            if(k==word.length()-1)

            {

                return true;

            }

            else if(nif(board, word, i-1, j, k+1) ||nif(board, word, i+1, j, k+1) ||nif(board, word, i, j-1, k+1) ||nif(board, word, i, j+1, k+1))

            {

                return true;

            }

            board[i][j]=temp;

        }

        return false;

    }

Count No. Of ISLAND

Two row and col for loop if ==1 then count++ check everycondition by backtrackingof void class.

In backtracking basic condition is same and m[i][j]==0 then return back.

**Matrix-o-x-replace-o-x-**

Replace all O to – then in all corner row and coloumn backtracking in all direction make – to O.THEN AT LAST MAKE – to x

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

            for (int j = 0; j < mat[0].length; j++) {

                if (mat[i][j] == 'O') {

                    mat[i][j] = '-';

                }

            }

        }

        for (int j = 0; j < mat[0].length; j++) {

            if (mat[0][j] == '-') {

                fill(mat, 0, j, '-', 'O');

            }

        }

        for (int j = 0; j < mat[0].length; j++) {

            if (mat[j][0] == '-') {

                fill(mat, j, 0, '-', 'O');

            }

        }

        for (int j = 0; j < mat[0].length; j++) {

            if (mat[mat.length - 1][j] == '-') {

                fill(mat, mat.length - 1, j, '-', 'O');

            }

        }

        for (int j = 0; j < mat[0].length; j++) {

            if (mat[j][mat[0].length - 1] == '-') {

                fill(mat, j, mat[0].length - 1, '-', 'O');

            }

        }

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

        for (int j = 0; j < mat[0].length; j++){

            if (mat[i][j] == '-')

                mat[i][j] = 'X';

    }}

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

        {

            for (int j = 0; j < mat[0].length; j++)

                System.out.print(mat[i][j] + " ");

            System.out.println("");

        }

    }

    static void fill(char mat[][], int x,

            int y, char prevV,

            char newV) {

        // Base cases

        if (x < 0 || x >= mat.length ||

                y < 0 || y >= mat[0].length)

            return;

        if (mat[x][y] != prevV)

            return;

        // Replace the color at (x, y)

        mat[x][y] = newV;

        // Recur for north,

        // east, south and west

        fill(mat, x + 1, y,

                prevV, newV);

        fill(mat, x - 1, y,

                prevV, newV);

        fill(mat, x, y + 1,

                prevV, newV);

        fill(mat, x, y - 1,

                prevV, newV);

    }

‘Common Row wise element

First take element of frst row in hashmap assign value 1.thn use element which is in map has element equal to previous row size so it will be common in each element and add 1 value 1 them. For printing if element equal to row size then print it.

int r = mat.length, c = mat[0].length;

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

        for(int i=0;i<c;i++){

            map.put(mat[0][i],1);

        }

        for (int i = 1; i < r; i++) {

            for (int j = 0; j < c; j++) {

                if (map.containsKey(mat[i][j]) && map.get(mat[i][j])==i) {

                        map.put(mat[i][j], map.get(mat[i][j]) + 1);

                }

                if(i==r-1 && map.containsKey(mat[i][j])&& map.get(mat[i][j])==r){

                    System.out.println(mat[i][j]);

                }

            }

        }

Rectangular ox

Like spiral matrix, we take all four direction put value x then using condition change x to 0 then again whole loop assign 0

int m=5, n=5,r=5,c=5;

        int i,k = 0, l = 0;

        char[][] a=new char[m][n];

        char x='X';

        while (k < m && l < n)

        {

            /\* Fill the first row from the remaining rows \*/

            for (i = l; i < n; ++i)

                a[k][i] = x;

            k++;

            /\* Fill the last column from the remaining columns \*/

            for (i = k; i < m; ++i)

                a[i][n-1] = x;

            n--;

            /\* Fill the last row from the remaining rows \*/

            if (k < m)

            {

                for (i = n-1; i >= l; --i)

                    a[m-1][i] = x;

                m--;

            }

            /\* Print the first column

                    // from the remaining columns \*/

            if (l < n)

            {

                for (i = m-1; i >= k; --i)

                    a[i][l] = x;

                l++;

            }

            x = (x == '0')? 'X': '0';

        }

Count Sort

In this sort, range is already given first we take how many time no. come then we use it accordingly.

  int n=a.length;

        int[] b=new int[8];

        for(int i=0;i<n;i++){

            int s=a[i];

            b[s]++;

        }

        int i=0,j=0;

        while(i<n){

            if(b[j]>0){

                int s=b[j];

                for(int l=0;l<s;l++){

                    a[i]=j;

                    i++;

                }

            }

            j++;

        }

        for(int k=0;k<n;k++){

            System.out.print(a[k]);

        }

Common Three Sorted Array

If all three equal i++ j++ k++ else if ar1[i]<ar2[j] i++ else if ar2[j]<ar3[k] j++ else k++;

int i=0,j=0,k=0;

        while(i<ar1.length && j<ar2.length && k<ar3.length){

            if(ar1[i]==ar2[j]&& ar2[j]==ar3[k]){

                System.out.println(ar1[i]);

                i++;

                j++;

                k++;

            }else if(ar1[i]<ar2[j]){

                i++;

            }else if(ar2[j]<ar3[k]){

                j++;

            }else{

                k++;

            }

        }

Search In Adjacent K

Generally we have serach element in array. For optimized way, we do

i= i+Math.max(1, Math.ans(x-arr[i])/k)

int arr[] = { 2, 4, 5, 7, 7, 6 };

        int x = 6;

        int k = 2;

        int n = arr.length;

        int i=0;

        while(i<n){

            if(arr[i]==x){

                System.out.println(i);

            }

            i= i+ Math.max(1,Math.abs((x-arr[i])/k));

        }

Ceil Search

It is same as binary search just last condition for no is not in arr then return arr[low]

 if(x==0){

            System.out.println(-1);

        }

        int low=0,high=arr.length-1;

        while(low<=high){

            int mid= low + (high-low)/2;

            if(arr[mid]==x){

                System.out.println(arr[mid]);

            }else if(arr[mid]>x){

                high=mid-1;

            }else{

                low=mid+1;

            }

        }

        System.out.println(arr[low]);

Majority Element

if element is greater than half of size of array print element else no. we do it use hashmap

  for(int i=0;i<a.length;i++){

            if(map.containsKey(a[i])){

                map.put(a[i],map.get(a[i])+1);

            }else{

                map.put(a[i],1);

            }

            if(map.get(a[i])>a.length/2){

                System.out.println(a[i]);

            }

            if(map.get(a[i])<(a.length/2) && i==a.length-1 ){

                System.out.println("no");

            }

        }

Triplet sum smaller than value

First sort array for loop from I to n-2 then j=i+1, k=n-1.then while j<k if a[I,j,k]>= sum them k-- else ans+= k-j and j++.

 Arrays.sort(a);

        for (int i = 0; i < a.length - 2; i++) {

            int j = i + 1, k = a.length - 1;

            while (j < k) {

                if (a[i] + a[j] + a[k] >= sum) {

                    k--;

                } else {

                    count += (k - j);

                    j++;

                }

            }

        }

Merge sorted Array with 0 space

while(arr1’s last element>arr2’s first elemnt we run loop if(arr1[i]>arr[0] we samp and sort arr2 and do i++

int i=0;

        while(arr1[m-1]>arr2[0]){

            if(arr1[i]>arr2[0]){

                int temp=arr1[i];

                arr1[i]=arr2[0];

                arr2[0]=temp;

                Arrays.sort(arr2);

            }

            i++;

        }

Inversion Count

Question good

import java.util.Arrays;

public class InversionCount {

    public static void main(String[] args) {

        int[] a={8,4,2,1};

        System.out.println(mergesort(a,0,a.length-1));

    }

    public static int mergesort(int[] a,int low, int high){

        int count =0;

        if(low<high){

            int mid= low+(high-low)/2;

            count +=mergesort(a, low, mid);

            count +=mergesort(a, mid+1, high);

            count += merge(a,low,mid,high);

        }

        return count;

    }

    public static int merge(int[] a, int l,int m, int h){

        int[] left=Arrays.copyOfRange(a, l, m+1);

        int[] right=Arrays.copyOfRange(a, m+1, h+1);

        int i=0,j=0,k=l,count=0;

        while (i < left.length && j < right.length){

            if(left[i]<=right[j]){

                a[k++]=left[i++];

            }else{

                a[k++]=right[j++];

                count +=(m + 1) - (l + i);;

            }

        }

        while (i < left.length)

            a[k++] = left[i++];

        while (j < right.length)

            a[k++] = right[j++];

        return count;

    }

}

Duplicates with no space

if(a[math.abs(a[i])>=0) then change it negative a[math.abs(a[i])=- a[math.abs(a[i]) otherwise print Math.abs(a[i])

Equal Min Cost

First we find value of y if array length is odd then =a[n/2] else y=(a[n/2]+a[n-2/2])/2 after finding value of y we total a[i]-y

if(n%2==1){

            y=a[n/2];

        }else{

            y=(a[n/2]+a[(n-2)/2])/2;

        }

        int s=0;

        for(int i=0;i<n;i++){

            s+= Math.abs(a[i]-y);

        }

Check reverse sub array make sorted array

We take new temp array and give all a array value to temp then sort a array.

Then for loop from front break if not equal similarly from back.then while loop tillfront != back {front++ if a[front-1]<a[front] then false} return true

 int[] temp= new int[n];

    for(int i=0;i<n;i++){

        temp[i]=a[i];

    }

    int front,back;

    Arrays.sort(a);

    for(front=0;front<n;front++){

        if(temp[front]!= a[front]){

            break;

        }

    }

    for(back=n-1;back>=0;back--){

        if(temp[back]!= a[back]){

            break;

        }

    }

    if(front>=back){

        return true;

    }

    while(front!=back){

        front++;

        if(temp[front-1]<temp[front]){

            return false;

        }

    }

    return true;

Reverse Linked List

4 line must be learn.it running two time samething. node prev=null, node curr=head, while(curr!=null){

Node temp=curr.next, curr.next=prev, prev=curr, curr=temp} return prev

Node reverse(Node head)

    {

        Node prev= null;

        Node curr=head;

        while(curr != null){

            Node temp= curr.next;

            curr.next=prev;

            prev=curr;

            curr=temp;

        }

        return prev;

    }

Loop Cycle

Using slow and fast u get answer

static boolean detectLoop(Node head)

    {

        Node slow=head;

        Node fast=head;

        while(slow!=null &&fast!=null){

            if(fast==slow){

                return true;

            }

            slow=slow.next;

            fast=fast.next.next;

        }

        return true;

    }

Merge Two Sorted LinkList

Take one empty node. Using while loop we find l1.val is less then add this make next if not do samething for l2. After that if loop for l1 and another if loop for l2.

 public ListNode mergeTwoLists(ListNode list1, ListNode list2) {

       ListNode cur1= new ListNode(-1);

       ListNode curr= cur1;

       while(list1!=null && list2!=null){

           if(list1.val<=list2.val){

               curr.next=list1;

               list1=list1.next;

           }else{

               curr.next=list2;

               list2=list2.next;

           }

           curr=curr.next;

       }

       if(list1!=null){

           curr.next=list1;

       }

       if(list2!=null){

            curr.next=list2;

        }

        return cur1.next;

    }

Delete without head node

public void deleteNode(Node head)

    {

        Node temp=head.next;

        head.data=temp.data;

        head.next=temp.next;

        temp=null;

    }

Remove Duplicates

take curr as head then put it in hashmap then store surr in prev move one step ahead

curr=curr.next then use hashmap if contain prev.next to curr.next otherwise prev to curr cur to prev.next

static void removeDuplicate(node head)

    {

        node curr =head;

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

        map.put(curr.val,1);

        node prev=curr;

        curr=curr.next;

        while(curr!=null){

            if(map.containsKey(curr.val)){

                prev.next=curr.next;

            }else{

                map.put(curr.val, 1);

                prev=curr;

            }

            curr=prev.next;

        }

    }

Sorted Linkedlist 0 ,1,2

Using int array find no. of 0,1,2.take ptr= head. While(ptr!=null) if a[i]==0 i++else

Ptr=i ptr=ptr.next a[i]—

static void sorting(node head)

    {

        int[] a=new int[3];

        node curr=head;

        while(curr!=null){

            a[curr.val]++;

            curr=curr.next;

        }

        int i=0;

        node ptr=head;

        while(ptr!=null){

            if(a[i]==0){

                i++;

            }else{

                ptr.val=i;

                a[i]--;

                ptr=ptr.next;

            }

        }

    }

Multiply Two Numbers

Take int a=b=0; we give value to a and multiply prev value by 10 for both the node then multiply them.

static long multiplyTwoLists(Node first, Node second)

    {

        Node curr1=first,curr2=second;

        int a=0,b=0;

        while(curr1!=null){

           a= (a\*10)+curr1.data;

           curr1=curr1.next;

        }

         while(curr2!=null){

           b= (b\*10)+curr2.data;

           curr2=curr2.next;

        }

        return (a\*b);

    }

Remove Nth element from last

First we count total element in node. Then for loop till count-n. after loop if curr1!=null cprev.net=cur.next

static void Removing(node head,int n)

    {

        node curr=head;

        int count=0;

        while(curr!=null){

            count++;

            curr=curr.next;

        }

        node temp=null,curr1=head;

        for(int i=0;i<count-n;i++){

            temp=curr1;

            curr1=curr1.next;

        }

        if(curr1!=null){

            temp.next=curr1.next;

        }

    }

Reorder List

First find middle value reverse from middle to last. By using while loop we take from first side and then from second side. But ek baar del lena.

    static void Reordering(node head)

    {

        node slow=head,fast=head.next;

        while(fast!=null && fast.next!=null){

            slow=slow.next;

            fast=fast.next.next;

        }

        node node1=head;

        node node2=slow.next;

        slow.next=null;

        node2= reverse(node2);

        node m= new node(0);

        node s=m;

        while(node1!=null || node2!=null){

           if(node1 !=null){

                s.next=node1;

                s=s.next;

                node1=node1.next;

           }

           if(node2 !=null){

                s.next=node2;

                s=s.next;

                node2=node2.next;

           }

        }

    }

Detect And Remove Loop

Frstly, we detect loopas befor take it slow,node to next function where we do while dekh lena.

int detectAndRemoveLoop(Node node)

    {

        Node slow = node, fast = node;

        while (slow != null && fast != null

            && fast.next != null) {

            slow = slow.next;

            fast = fast.next.next;

            if (slow == fast) {

                removeLoop(slow, node);

                return 1;

            }

        }

        return 0;

    }

    void removeLoop(Node loop, Node head)

    {

        Node ptr1=loop;

        Node ptr2=loop;

        Node prevNode=ptr1;

        while(ptr1.next!=ptr2){

            prevNode=ptr1;

            ptr1=ptr1.next;

        }

        prevNode.next=null;

    }

Intersection Points

We use hashset from head1 we store all valuesas hs.add(node1) then another while if hs.contains(node2) return node.data and node2=node2.next

    static int getIntersectNode(Node head1, Node head2){

        HashSet<Node> hs= new HashSet<>();

        Node curr1=head1;

        Node curr2=head2;

        while(curr1!=null){

            hs.add(curr1);

            curr1=curr1.next;

        }

        while(curr2!=null){

            if(hs.contains(curr2)){

                return curr2.data;

            }

            curr2=curr2.next;

        }

        return -1;

    }

Flattening Multilevel

Method 1 using queue n space

void flattenList(Node node) {

            Node h=node;

            Queue<Node> q=new ArrayDeque<>();

            while(h!=null){

                if(h.next==null){

                    h.next=q.poll();

                }

                if(h.child!=null){

                    q.add(h.child);

                }

                h=head.next;

            }

        }

Method 2 -padthe rhna acha h

void flattenList(Node node) {

            Node temp=null;

            Node tail=node;

            while(tail.next!=null){

                tail=tail.next;

            }

            Node curr=node;

            while(tail!= curr){

                if(curr.child!=null){

                    tail.next=curr.child;

                    temp=curr.child;

                    while(temp.next!=null){

                        temp=temp.next;

                    }

                    tail=temp;

                }

                curr=curr.next;

            }

        }

Zigzag

A<b>c<d>e…. Easy hai

Take flag, if flag==0 check first condition a<b if not then swap similarly flag==1 condition b>c if not then swap.

    // Swap both nodes

    public void swap(Node a, Node b)

    {

        if (a == null || b == null)

            return;

        int temp = a.data;

        a.data = b.data;

        b.data = temp;

    }

    // Rearrange the linked list

    // in zig zag way

    public Node zigZag(Node node, int flag)

    {

        if (node == null || node.next == null) {

            return node;

        }

        if (flag == 0) {

            if (node.data > node.next.data) {

                swap(node, node.next);

            }

            return zigZag(node.next, 1);

        }

        else {

            if (node.data < node.next.data) {

                swap(node, node.next);

            }

            return zigZag(node.next, 0);

        }

    }

Reverse Doubly linkedlist

First we go to last point then swap for last then for middle using while loop then firstone

void reverse()

    {

        Node curr = head;

        while(curr.next !=null){

            curr=curr.next;

        }

        head=curr;

        curr.next=curr.prev;

        curr.prev=null;

        curr=curr.next;

        while(curr.prev!=null){

            Node temp=curr.next;

            curr.next=curr.prev;

            curr.prev=temp;

            curr=curr.next;

        }

        curr.prev=curr.next;

        curr.next=null;

    }

Segregate Even odd

Take all four evend odd (start end). While(curr!=null){if eventhe even start else oddstart if even start oddstart is null at that time else evenend.next=currnode

void segregateEvenOdd()

    {

        Node evenstart= null,evenend=null;

        Node oddstart=null,oddend=null;

        Node curr=head;

        while(curr!=null){

            int s=curr.data;

            if(s%2==0){

                if(evenstart==null){

                    evenstart=curr;

                    evenend=evenstart;

                }else{

                    evenend.next=curr;

                    evenend=evenend.next;

                }

            }else{

                 if(oddstart==null){

                    oddstart=curr;

                    oddend=oddstart;

                }else{

                    oddend.next=curr;

                    oddend=oddend.next;

                }

            }

            curr=curr.next;

        }

        if(oddstart==null||evenstart==null){

            return;

        }

        evenend.next=oddstart;

        oddend.next=null;

        head=evenstart;

    }

Point to next higher value node

There is two pointer next and arbitrary pointer.firstly we make arb pointer same as next pointer. Then we do merge sorting in arb pointer so that everyone connected to nect higher value node.merge sort is same but we use arb in place of next

private Node populateArbit(Node start)

    {

        Node temp = start;

        // Copy next pointers to arbit pointers

        while (temp != null)

        {

            temp.arbit = temp.next;

            temp = temp.next;

        }

        // Do merge sort for arbitrary pointers and

        // return head of arbitrary pointer linked list

        return MergeSort(start);

    }

Sort Biotonic Linkedlist(1,5,7,12,9,6,3)

First we find from which element curr.next<curr then curr.prev.next=null, curr.prev=null then we reverse array from there then merge that two array.

Merge K Sorted List

Firstly,we do pointer at starting and end. We join it at the first and merge it.similarly this loop again and again till last!=0

public ListNode mergeKLists(ListNode[] arr) {

        int k=arr[0].length;

        int i=0,last=k-1,j;

        while(last!=0){

            i=0;

            j=last;

            while(i<j){

                arr[i]=merge(arr[i],arr[j]);

                i++;

                j--;

                if(i>=j){

                    last=j;

                }

            }

        }

        return arr[0];

    }

Merge Sorted linkedlist

First we find middle element then separate them each time then merge it

node mergeSort(node h)

    {

         if (h == null || h.next == null) {

            return h;

        }

        node middle = getMiddle(h);

        node nextofmiddle = middle.next;

        middle.next = null;

        node left = mergeSort(h);

        node right = mergeSort(nextofmiddle);

        node sortedlist = sortedMerge(left, right);

        return sortedlist;

    }

public static node getMiddle(node head)

    {

        if(head==null){

            return head;

        }

        node slow=head;

        node fast=head;

        while(fast.next!=null&&fast.next.next!=null){

            slow=slow.next;

            fast=fast.next.next;

        }

        return slow;

    }

node sortedMerge(node a, node b)

    {

        node answer =null;

        if(a==null){

            return b;

        }

        if(b==null){

            return a;

        }

            if(a.val<=b.val){

                answer=a;

                answer.next= sortedMerge(a.next, b);

            }else{

                answer=b;

                answer.next=sortedMerge(a, b.next);

            }

          return answer;

    }

Sum of two linked list

We tae new node ans assign some int initial.them we do some from both of them.totla will be sum+carry. We add totl%10to curr.next and carry=total/10. It give sum in reverse way. So we reverse the array.

 int carry=0,sum=0;

    Node dumy= new Node(0);

    Node curr=dumy;

    while(l1!=null ||l2!=null||carry==1){

        int sum=0;

        if(l1!=null){

            sum+=l1.data;

            l1=l1.next;

        }

        if(l2!=null){

            sum+=l2.data;

            l2=l2.next;

        }

        int total=sum+carry;

        carry=total/10;

        Node s= new Node(total%10);

        curr.next=s;

        curr=curr.next;                       // till taht we get in reverse {9,7,5}

    }

     return dumy.next;

Flattening and sorting linkedlist

We return merge(root,flatten(root.right). root right helps me to reach till null.Merging done down inplace of next.

Node flatten(Node root)

    {

        if(root==null ||root.right==null){

            return root;

        }

        return merge(root,flatten(root.right));

    }

Two stack in an array

void push1(int x)

    {

        if(top1<top2-1){

            top1++;

            arr[top1]=x;

        }else{

            System.out.println("overflow");

        }

    }

    void push2(int x)

    {

        if(top1<top2-1){

            top2--;

            arr[top2]=x;

        }else{

            System.out.println("overflow");

        }

    }

    int pop1()

    {

        if(top1>=0){

            int x=arr[top1];

            top1--;

            return x;

        }else{

            System.out.println("unerflow");

            return 0;

        }

    }

    int pop2()

    {

        if(top2<size){

            int x=arr[top2];

            top2++;

            return x;

        }else{

            System.out.println("underflow");

            return 0;

    }

    }

PostFix Expression

Stack<Integer> stack = new Stack<>();

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

            char c = exp.charAt(i);

            if (c == ' ')

                continue;

            else if (Character.isDigit(c)) {

                int n = 0;

                while (Character.isDigit(c)) {

                    n = n \* 10 + (int)(c - '0');

                    i++;

                    c = exp.charAt(i);

                }

                stack.push(n);

            }

            else {

                int val1 = stack.pop();

                int val2 = stack.pop();

                switch (c) {

                case '+':

                    stack.push(val2 + val1);

                    break;

                case '-':

                    stack.push(val2 - val1);

                    break;

                case '/':

                    stack.push(val2 / val1);

                    break;

                case '\*':

                    stack.push(val2 \* val1);

                    break;

                }

            }

        }

        return stack.pop();

stack function-push,pop,isEpmty,peek

Queue function-add,remove,isEmpty,peek

Dequeue function-addfirst,addlast,popfirst,poplast

Queue Reversal

We use stack for reversing. First we put all element of queue in stack then stack to queue

static void reversequeue()

    {

        Stack<Integer> st= new Stack<Integer>();

        while(!queue.isEmpty()){

            st.push(queue.peek());

            queue.remove();

        }

        while(!st.isEmpty()){

            queue.add(st.peek());

            st.pop();

        }

    }

Reverse first k element of queue

We do same step as we do reverse whole queue but in this for k element but last step we do queue for remaining element 0 to q.size()-k q.add(q.peek()) and q.remove()

Stack<Integer> st= new Stack<Integer>();

    int s=k;

    while(k!=0 && !q.isEmpty()){

        st.push(q.peek());

        q.remove();

        k--;

    }

    while(!st.isEmpty()){

        q.add(st.peek());

        st.pop();

    }

    for(int i=0;i<q.size()-s;i++){

        q.add(q.peek());

        q.remove();

    }

    return q;

Stack with Middle operation

We do it using dequeue and stack for push operation we add element to deque last till d.size>s.size+1 if that happen remove element from deque first to stack in pop operaion we remove element from dequeu last. If stack size >deueue size we do spop put it in deque first. For getting, middle element do deueue.Getfirst. for deleting middle element deueue.deletefirst if stacksize>qequeue s.pop put it in deque.add first.

void add(int data)

    {

        dq.addLast(data);

        if(dq.size()>s.size()+1){

            int temp=dq.pollFirst();

            s.push(temp);

        }

    }

    void pop()

    {

        dq.pollLast();

        if(dq.size()<s.size()){

            int temp=s.pop();

            dq.addFirst(temp);

        }

    }

    int getMiddleElement() {

        return dq.getFirst();

     }

    void deleteMiddleElement()

    {

        dq.pollFirst();

        if(dq.size()<s.size()){

            int temp=s.pop();

            dq.addFirst(temp);

        }

    }

Infix to postfix

Take empty string. To for loop f character come add to string if this ‘(‘ add to stack if this’)’ come we remove element from stack till’(‘ if opersator vome then check one condition higher prec come then remove lower prec then add higher prec. Aane wla chota h nikal do bda wla prec(c)=>prec(stack.peek()).

static int Prec(char ch)

    {

        switch (ch) {

        case '+':

        case '-':

            return 1;

        case '\*':

        case '/':

            return 2;

        case '^':

            return 3;

        }

        return -1;

    }

String result = new String("");

        Deque<Character> stack

            = new ArrayDeque<Character>();

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

            char c = exp.charAt(i);

            if (Character.isLetterOrDigit(c))

                result += c;

            else if (c == '(')

                stack.push(c);

            else if (c == ')') {

                while (!stack.isEmpty()

                    && stack.peek() != '(') {

                    result += stack.peek();

                    stack.pop();

                }

                stack.pop();

            }

            else{

                while (!stack.isEmpty()

                    && Prec(c) <= Prec(stack.peek())) {

                    result += stack.peek();

                    stack.pop();

                }

                stack.push(c);

            }

Implement Special stack(getmin)

Make two Stack-super & min. While push, if stack empty then add element in both of them.if not then aane wla element agr chota h dono me daal do super me hi dalo sirf.

While pop, agr super ka peek element min ke peek element ke equal to dono se nikalo wrna super se nikalo sirf.

oid push(int x)

    {

        if(isEmpty()){

            super.push(x);

            min.push(x);

        }else{

        int y=min.pop();

        min.push(y);

        super.push(x);

        if(x<=y){

            min.push(x);

        }

    }

    }

    public Integer pop()

    {

        int y=min.pop();

        int x=super.pop();

        if(x!=y){

            min.push(y);

        }

        return x;

    }

    int getMin()

    {

        int x = min.pop();

        min.push(x);

        return x;

    }

Longest Valid String

We use stack if ‘(‘ add to stack s.push(i) if this ‘)’ come remove s.pop. Result=Math.MAX(RESULT,i-s.peek())

Stack<Integer> st = new Stack<>();;

        st.push(-1);

        int result = 0;

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

            if(str.charAt(i)=='('){

                st.push(i);

            }else{

                if(!st.isEmpty()){

                    st.pop();

                }

                if(!st.isEmpty()){

                    result=Math.max(result,i-st.peek());

                }

                else{

                    st.push(i);

                }

            }

        }

        return result;

Duplicates paratheis or not

If ‘)’ we do work else st.push whtever it is. Now what operation we have to do it for ‘)’ if st.peek==’(’ then true uplicate parathesis. While(st.peek==’(‘)) yh na aa jaye tb tak nikalte rho stackse .

static boolean findDuplicateparenthesis(String s) {

        Stack<Character> st = new Stack<>();

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

            char ch=s.charAt(i);

            if(ch==')'){

                if(st.peek()=='('){

                    return true;

                }else{

                    while(st.peek()!='('){

                        st.pop();

                    }

                    st.pop();

                }

            }else{

                 st.push(ch);

            }

        }

        return false;

    }

Check if an array is stack permutation

Given two array j=0 loop for i we do arr1[i] to st.push till st.peek() is equal to arr2[j] if that happen we pop element tillst.peek()==arr2[j]

int[] arr1={1,2,3},arr2={2,1,3};

        Stack<Integer> st= new Stack<>();

        int j=0;

        for(int i=0;i<arr1.length;i++){

            st.push(arr1[i]);

            if(st.peek()==arr2[j]){

                while(!st.isEmpty() && arr2[j]==st.peek()){

                    st.pop();

                    j++;

                }

            }

        }

Sort a stack using recursion

Ratt lo asaaan hai

 static void sort(Stack<Integer> st){

        if(!st.isEmpty()){

            int x=st.pop();

            sort(st);

            sorting(st,x);

        }

    }

    static void sorting(Stack<Integer> st,int x){

        if(st.isEmpty()|| st.peek()<x){

            st.push(x);

            return;

        }

        int temp=st.pop();

        sorting(st, x);

        st.push(temp);

    }

Queue first non.repeating character

First we add element to hashmap. If it map.get[arr[i]== 1) then add to queue.ab kro jab tak queue hali na ho jaye.We see queue peek element phir map se kitne elemnt aae h agr 1 h to whi se break ho jao agr zyada h q.remove. print krne ke liye agr queue khali ho -1 agr ho to q.peek() element daal do.

char[] ch={'a','a','b','c','c','d'};

        Queue<Character> q= new LinkedList();

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

        for(int i=0;i<ch.length;i++){

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

            if(map.get(ch[i])==1){

                q.add(ch[i]);

            }

            while(!q.isEmpty()){

                if(map.get(q.peek())==1){

                    break;

                }else{

                    q.remove();

                }

            }

        }

        if(q.isEmpty()){

            System.out.println(-1);

        }else{

            System.out.println(q.peek());

        }

The celebrity Problem

First we put no. of rows in stack.while stack size is greater equal to 2 Then we pop two element from stack as I,j if for a[i][j]==1 st.push(j) else st.push(i). now only one element left in stack int pop=st.pop() then for loop for j if(pop!=j)then a[pop][j]==1 || a[j][pop]==0 return false else true

  Stack<Integer> st= new Stack<>();

        for(int i=0;i<m.length;i++){

            st.push(i);

        }

        while(st.size()>=2){

            int i=st.pop();

            int j=st.pop();

            if(m[i][j]==1){

                st.push(j);

            }else{

                st.push(i);

            }

        }

        int pop=st.pop();

        for(int i=0;i<m.length;i++){

            if(i!=pop){

                if(m[i][pop]==0 || m[pop][i]==1){

                    System.out.println(-1);

                    return;

                }

            }

        }

        System.out.println(pop);

Next largest Element

We make one array for nle. We add element index to stack if(st.isempty) or st.peek element is larger then curr elemt if element is smaller thn we pop element and put at that index the curr value.

 int[] arr={4,5,2,25};

        int[] ans=new int[arr.length];

        Stack<Integer> st= new Stack<>();

        st.push(0);

        for(int i=1;i<arr.length;i++){

            if(arr[st.peek()]<arr[i]){

                while(!st.isEmpty() && arr[st.peek()]<arr[i]){

                    ans[st.peek()]=arr[i];

                    st.pop();

                }

            }

            st.push(i);

        }

        while(!st.isEmpty()){

            ans[st.peek()]=-1;

            st.pop();

        }

Similarly Next Smaleer element

Distance of nearest cell having 1

First we add index to queue who hav value 1.for direction make array for saving make 2-d array. then two while queue is not empty int[] idx =q.pop() check in all dirs.

 int n=grid.length,m=grid[0].length;

        Queue<int[]> q=new LinkedList<>();

        boolean[][] vis= new boolean[n][m];

        for(int i=0;i<n;i++){

            for(int j=0;j<m;j++){

                if(grid[i][j]==1){

                    q.add(new int[]{i,j});

                    vis[i][j]=true;

                }

            }

        }

        int[][] result= new int[n][m];

        int[][] dirs={{1,0},{0,1},{0,-1},{-1,0}};

        int dis=0;

        while(!q.isEmpty()){

            int sz=q.size();

            while(sz-->0){

                int[] idx=q.remove();

                int x=idx[0];

                int y=idx[1];

                result[x][y]=dis;

                for(int[] dir:dirs){

                    int i=x+dir[0];

                    int j=y+dir[1];

                    if(i>=0 && j>=0 && i<n && j<m &&vis[i][j]!=true){

                        q.add(new int[]{i,j});

                        vis[i][j]=true;

                    }

                }

            }

            dis++;

        }

Rotten Oranges

First we index value in queue who value equal to 2 and calculate total oranges in one loop. Then while loop q is not empty. Then for loop in all four direction check add value to queue.

 int n=grid.length,m=grid[0].length;

        int count\_frsh=0;

        Queue<int[]> q=new LinkedList<>();

        for(int i=0;i<n;i++){

            for(int j=0;j<m;j++){

                if(grid[i][j]==2){

                    q.add(new int[]{i,j});

                }

                if(grid[i][j]!=0){

                    count\_frsh++;

                }

            }

        }

        int[][] dirs={{1,0},{0,1},{0,-1},{-1,0}};

        int dis=0,cnt=0;

        while(!q.isEmpty()){

            int sz=q.size();

            cnt+=sz;

            for(int r=0;r<sz;r++){

                int[] idx=q.remove();

                for(int[] dir:dirs){

                    int i=idx[0]+dir[0];

                    int j=idx[1]+dir[1];

                    if(i>=n || j>=m || i<0 || j<0 ||grid[i][j]!=1){

                        continue;

                    }

                    grid[i][j]=2;

                    q.add(new int[]{i,j});

                }

            }

            if(!q.isEmpty()){

                dis++;

            }

        }

        if(count\_frsh==cnt){

            System.out.println(dis);

        }else{

            System.out.println(-1);

        }

CircuLar TOUR

We inliatize deficit=0, curr=0,start=0. Add value ineach if curr<0 then add vlue to deficit and make curr=0,start =i+1 after that if curr+deficit>=0 then start else false.

int[] p={6,3,7},d={4,6,3};

        int capacity=0,deficit=0,start=0;

        for(int i=0;i<p.length;i++){

            capacity+= p[i]-d[i];

            if(capacity<0){

                deficit+=capacity;

                capacity=0;

                start=i+1;

            }

        }

        if(capacity+deficit>=0){

            System.out.println(start);

        }else{

            System.out.println(-1);

        }

Activity Selection problem

we have to array st fn we print(0) int j=0; for loop for i 1 to n if(fn[j]<=st[i]) then print(i) and j=i

int[] s={1,3,0,5,8}, f={2,4,6,7,9,9};

        System.out.println(0);

        int j=0;

        for(int i=1;i<s.length;i++){

            if(s[i]>=f[j]){

                System.out.println(i);

                j=i;

            }

        }

Minimum sum of two numbers formed from digits of an array

we add value to pq while(pq is not empty) add to s1 string if(pq.is not empty) then add to s2 string then add both value.

PriorityQueue<Integer> pq=new PriorityQueue<>();

        StringBuilder s1=new StringBuilder();

        StringBuilder s2=new StringBuilder();

        for(int i=0;i<a.length;i++){

            pq.add(a[i]);

        }

        while(!pq.isEmpty()){

            s1.append(pq.poll()+"");

            if(!pq.isEmpty()){

                s2.append(pq.poll()+"");

            }

        }

        Long sum=Long.parseLong(s1.toString())+Long.parseLong(s2.toString());

        System.out.println(sum);

Min Absoute diff pair

We sort both the array then do diff+= easily.

Arrays.sort(a);

        Arrays.sort(b);

        int diff=0;

        for(int i=0;i<a.length;i++){

            diff+= Math.abs(a[i]-b[i]);

        }

        System.out.println(diff);

Max hight Pyramid

Sort the Array.we take pw=a[o], pc=0 ,ans=1 then for loop add cc+1 cw+=a[1] if cc>pc && cw>pw we do pc=cc, pw=cw, ans++ Then print ans.

 Arrays.sort(a);   //20,30,40,100

        int prev\_count=1,prev\_width=a[0];

        int curr\_count=0,curr\_width=0;

        int ans=1;

        for(int i=1;i<a.length;i++){

            curr\_count+=1;

            curr\_width+=a[i];

            if(prev\_count<curr\_count && prev\_width<curr\_width){

                prev\_count=curr\_count;

                prev\_width=curr\_width;

                curr\_count=0;

                curr\_width=0;

                ans++;

            }

        }

        System.out.println(ans);

Min cost for acquire k coins.

We sort the array.Int f=Math.cel(n/k+1) for loop i=0 to i<=0 sum it a[i]

 int[] coin={100,20,50,10,2,5};

        int k=3;

        Arrays.sort(coin);

        int s=(int)Math.ceil(coin.length/(k+1));

        int total=0;

        for(int i=0;i<=s;i++){

            total += coin[i];

        }

        System.out.println(total);

Equal sum three stack

We sum all three stack separately. While(true) if sum of all three qual then print sum else whivh is high the remove peek of that.

int n1=stack1.length,n2=stack2.length,n3=stack3.length;

        int a1=0,a2=0,a3=0;

        for(int i=0;i<n1;i++){

            a1+= stack1[i];

        }

        for(int i=0;i<n2;i++){

            a2+= stack2[i];

        }

        for(int i=0;i<n3;i++){

            a3+= stack3[i];

        }

        int b1=0,b2=0,b3=0;

        while(true){

            if(b1==n1 || b2==n2 || b3==n3){

                System.out.println(0);

                return;

            }

            if(a1==a2 && a1==a3){

                System.out.println(a1);

                return;

            }

            if(a1>=a2 && a1>= a3){

                a1-=stack1[b1++];

            }else if(a2>=a1 && a2>=a3){

                a2-= stack2[b2++];

            }else if(a3>=a2 && a3>=a1){

                a3-= stack3[b3++];

            }

        }

Job Sequencing problem

We sort the array in desceding order according to profit.we take one empty array of max size of deadline.we can put atmost at that place.we take Boolean array of same size if it not use. We use it,make it true.

void printJobScheduling(ArrayList<Job> arr, int t)

    {

        int n=arr.size();

        char[] job= new char[t];

        boolean[] f= new boolean[n];

        Collections.sort(arr,(a,b)->b.profit-a.profit);

        for(int i=0;i<n;i++){

            for(int j=Math.min(n, arr.get(i).deadline-1);j>=0;j--){

                if(f[j]==false){

                    f[j]=true;

                    job[j]= arr.get(i).id;

                    break;

                }

            }

        }

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

            System.out.println(job[i]);

        }

    }

Egyptain Fraction

some basic we can do it main it if nr>dr then print nr/dr Egyptian(nr%dr,dr) then we do for nr<dr int n=dr/nr+1 print “1/”+n eqyptian(nr\*n-dr,dr\*n)

static void printEgyptian(int nr, int dr)

    {

        if (dr == 0 || nr == 0) {

            return;

        }

        if (dr % nr == 0) {

            System.out.print("1/" + dr / nr);

            return;

        }

        if (nr % dr == 0) {

            System.out.print(nr / dr);

            return;

        }

        if (nr > dr) {

            System.out.print(nr / dr + " + ");

            printEgyptian(nr % dr, dr);

            return;

        }

        int n = dr / nr + 1;

        System.out.print("1/" + n + " + ");

        printEgyptian(nr \* n - dr, dr \* n);

    }

Fractional knapsack Problem

By using comparator we find ratios of eah of them. Sort them according to decreasing order.if capacity-curwt >=0 then we use it substratct it. If not then we find fractional value and multiply with curval.

private static double getMaxValue(ItemValue[] arr,

                                    int capacity)

    {

        Arrays.sort(arr, new Comparator<ItemValue>(){

            public int compare(ItemValue item1,ItemValue item2){

                double cpr1= new Double((double)item1.profit/(double)item1.weight);

                double cpr2= new Double((double)item2.profit/(double)item2.weight);

                if(cpr1<cpr2){

                    return 1;

                }else{

                    return -1;

                }

            }

        });

        Double total =0d;

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

            int curwt= (int)arr[i].weight;

            int curval=(int)arr[i].profit;

            if(capacity-curwt>=0){

                capacity=capacity- curwt;

                total+=curval;

            }else{

                double fraction=((double)capacity/(double)curwt);

                total+= curval\*fraction;

                capacity=(int)(capacity-(curwt\*fraction));

                break;

            }

        }

        return total;

    }

Smallest no. with given no. of digit and sum of digits

First we -1 from sum. Do loop from n-1to 1.if(s>9) res[i]=9 s-=9 else res[i]=s s=0

Res[0]=s+1;

 int s=20,d=3;

        if(d\*9<s){

            System.out.println("not possible");

            return;

        }

        int[] m=new int[d];

        s=s-1;

        for(int i=d-1;i>0;i--){

            if(s>9){

                m[i]=9;

                s-=9;

            }else{

                m[i]=9;

                s=0;

            }

        }

        m[0]=s+1;

Max sum of consecutive diff in circular array

We sort the array. Int total=0; loop from 0 to n/2. Total-=2\*a[i] total+=2\*a[i-m-1]

Print total,

 public static void main(String[] args) {

        int[] arr={4,2,1,8};

        Arrays.sort(arr);

        int sum=0,n=arr.length;

        for(int i=0;i<n/2;i++){

            sum-=2\*arr[i];

            sum+=2\*arr[n-i-1];

        }

        System.out.println(sum);

    }

Paper Cut

We make first one is greater than second if not swap it.int total =0 total +=a/b, rem=a%b,a=b,b=rem return total.

static int paper(int a,int b){

        int sum=0,rem=0;

        if(b>a){

            swap(a,b);

        }

        while(b>0){

            sum+= a/b;

            rem= a%b;

            a=b;

            b=rem;

        }

        return sum;

    }

Chocola &cut board into square

We sort row cut and col cut into descending order int sum=0,default row1=1,col1=1 if we do colwise then sum=col1\*row[i] row1++ i++ else sum=row1\*col[i] col1++ j++ after that alg alg print sum

 int m=r.length,n=c.length;

        Arrays.sort(c,Collections.reverseOrder());

        Arrays.sort(r,Collections.reverseOrder());

        int r1=1,c1=1;

        int i=0,j=0;

        int total=0;

        while(i<m && j<n){

            if(c[j]<r[i]){

                total+=c1\*r[i];

                r1++;

                i++;

            }else{

                total+= r1\*c[j];

                c1++;

                j++;

            }

        }

        while(i<m){

            total+=c1\*r[i];

            i++;

        }

        while(j<n){

            total+= r1\*c[j];

            j++;

        }

        System.out.println(total);

Min time to finish all job

Recursive manner.

static int min=Integer.MAX\_VALUE;

    public static void main(String[] args) {

        int k=2;

        int[] job={4,5,10};

        int[] res= new int[k];

        jobs(job,0,res);

        System.out.println(min);

    }

    static void jobs(int[] job,int j,int[] sum){

        int max=getmax(sum);

        if(max>=min){

            return;

        }

        if(j==job.length){

            min=Math.min(min,max);

            return;

        }

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

            if(i>0 && sum[i]==sum[i-1])

                continue;

            sum[i]+=job[j];

            jobs(job,j+1,sum);

            sum[i]-=job[j];

        }

    }

Rearrange char to string no two adjacent char are same

First we make array of 26 letter and add value find max char and its frequency. Then take one string s=”” and do one loop s+=’ ‘. Then add value similarly

 int[] ch=new int[26];

        int max=Integer.MIN\_VALUE;

        char c=s.charAt(0);

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

            int v=s.charAt(i)-'a';

            ch[v]++;

            if(ch[v]>max){

                max=ch[v];

                c=s.charAt(i);

            }

        }

       int ind=0;

       String m="";

       for(int i=0;i<s.length();i++){

            m+= ' ';

       }

       int n=s.length();

       while(max>0){

            m=m.substring(0, ind)+c+m.substring(ind+1);

            ind=ind+2;

            max--;

       }

       ch[c-'a']=0;

       for(int i=0;i<26;i++){

            while(ch[i]>0){

                ind=(ind>n)? 1:ind;

                m=m.substring(0, ind)+(char)('a'+i)+m.substring(ind+1);

                ind+=2;

                ch[i]--;

            }

       }

Isbalanced Tree

First we check left height then right height if bitg lh-rh<=1 then check root.left and root.right

public static boolean Isbalanced(Node root){

        if(root==null){

            return true;

        }

        int lh=height(root.left);

        int rh=height(root.right);

        if(Math.abs(lh-rh)<=1 && Isbalanced(root.left) && Isbalanced(root.right)){

            return true;

        }

        return false;

   }

Bottom View

If we move left than -1 if move right sde than +1 we queue and hashmap. Hashmap stare the the last value mean bottom value

private void TopView(Node root)

    {

        class QueueObj {

            Node node;

            int hd;

            QueueObj(Node node, int hd)

            {

                this.node = node;

                this.hd = hd;

            }

        }

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

        Map<Integer, Node> map

            = new TreeMap<Integer, Node>();

        if (root == null) {

            return;

        }

        else {

            q.add(new QueueObj(root, 0));

        }

        System.out.println(

            "The top view of the tree is : ");

        while (!q.isEmpty()) {

            QueueObj tmp = q.poll();

            int hd=tmp.hd;

            Node h=tmp.node;

            map.put(hd,h);

            if (h.left != null) {

                q.add(new QueueObj(h.left,hd - 1));

            }

            if (h.right != null) {

                q.add(new QueueObj(h.right,hd + 1));

            }

        }

        for (Map.Entry<Integer, Node> entry :

            map.entrySet()) {

            System.out.print(entry.getValue().key + " ");

        }

Boundary Traversal

First we left traversal except leaf then leaf traversal then right traversal in reverse order using stack.

ArrayList<Integer> boundary(Node root)

    {

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

        if(!isLeaf(root)){

            ans.add(root.data);

        }

        addLeftBound(root, ans);

        addLeaves(root, ans);

        addRightBound(root, ans);

        return ans;

    }

private void addLeftBound(Node root,

                            ArrayList<Integer> ans)

    {

        Node curr=root.left;

        while(curr!=null){

            if(isLeaf(curr)==false){

                ans.add(curr.data);

            }

            if(curr.left!=null){

                curr=curr.left;

            }else{

                curr=curr.right;

            }

        }

    }

private void addLeaves(Node root,

                        ArrayList<Integer> ans)

    {

        if(isLeaf(root)){

            ans.add(root.data);

            return;

        }

        if(root.left!=null){

            addLeaves(root.left, ans);

        }

        if(root.right!=null){

            addLeaves(root.right, ans);

        }

    }

private void addLeftBound(Node root,

                            ArrayList<Integer> ans)

    {

        Node curr=root.left;

        while(curr!=null){

            if(isLeaf(curr)==false){

                ans.add(curr.data);

            }

            if(curr.left!=null){

                curr=curr.left;

            }else{

                curr=curr.right;

            }

        }

    }

    private void addRightBound(Node root,

                            ArrayList<Integer> ans)

    {

        Stack<Integer> st=new Stack<>();

        Node curr=root.right;

        while(curr!=null){

            if(isLeaf(curr)==false){

                st.push(curr.data);

            }

            if(curr.right!=null){

                curr=curr.right;

            }else{

                curr=curr.left;

            }

        }

        while(!st.isEmpty()){

            ans.add(st.pop());

        }

    }

Check Duplicate Subtree

public static String dupSubUtil(Node root, HashSet<String> subtrees)

    {

        String s = "";

        if (root == null)

            return s + "$";

        String lStr = dupSubUtil(root.left,subtrees);

        if (lStr.equals(s))

            return s;

        String rStr = dupSubUtil(root.right,subtrees);

        if (rStr.equals(s))

            return s;

        s = s + root.data + "%" + lStr+ "%" + rStr;

        if (s.length() > 7 && subtrees.contains(s))

            return "";

        subtrees.add(s);

        return s;

    }

    public static String dupSub(Node root)

    {

        HashSet<String> subtrees=new HashSet<>();

        return dupSubUtil(root,subtrees);

    }

Dimater of tree

public static int height(Node root)

    {

        int[] dia=new int[1];

        diameter(root,dia);

        return dia[0];

    }

    public static int diameter(Node root, int[] dia){

        if(root==null){

            return 0;

        }

        int lh=diameter(root.left,dia);

        int rh=diameter(root.right, dia);

        dia[0]=Math.max(dia[0],lh+rh);

        return 1+Math.max(lh,rh);

    }

Duplicate Subtrees

static String inorder(Node node)

    {

        if(node==null){

            return "";

        }

        String s="";

        s += inorder(node.left);

        s+= Integer.toString(node.data);

        s+= inorder(node.right);

        if(m.get(s)!=null && m.get(s)==1){

            System.out.println(node.data);

        }

        if(m.containsKey(s)){

            m.put(s, m.get(s)+1);

        }else{

            m.put(s, 1);

        }

        return s;

    }

    static void printAllDups(Node root)

    {

        m = new HashMap<>();

        inorder(root);

    }

Invert subtree/ mirror

static void mirror(Node root)

    {

        if (root == null)

            return;

        Queue<Node> q = new java.util.LinkedList<>();

        q.add(root);

        while(!q.isEmpty()){

            Node curr=q.poll();

            Node temp=curr.left;

            curr.left=curr.right;

            curr.right=temp;

            if(curr.left!=null){

                q.add(curr.left);

            }

            if(curr.right!=null){

                q.add(curr.right);

            }

        }

    }

Isomorphic Tree

boolean isIsomorphic(Node n1, Node n2)

    {

        if(n1==null || n2==null){

            return n1==n2;

        }

        if(n1.data!=n2.data){

            return false;

        }

        return (isIsomorphic(n1.left, n2.left) && isIsomorphic(n1.right, n2.right) ||

        isIsomorphic(n1.left, n2.right) && isIsomorphic(n1.right,n2.left));

    }

Valid Subtree

 boolean isSubtree(Node T, Node S)

    {

    if(S==null){

      return true;

    }

        if(T==null){

      return false;

    }

    if(areIdentical(T, S)){

      return true;

    }

    return isSubtree(T.left, S)||isSubtree(T.right, S);

    }

 boolean areIdentical(Node root1, Node root2)

    {

    if(root1 ==null && root2 ==null){

      return root1==root2;

    }

    if(root1.data!=root2.data){

      return false;

    }

    return areIdentical(root1.left, root2.left)&& areIdentical(root1.right, root2.right);

    }

Kth ancestor

static Node kthAncestorDFS(Node root, int node )

{

    if(root==null){

        return root;

    }

    if(root.data==node || temp==kthAncestorDFS(root.left, node) || temp==kthAncestorDFS(root.right, node)){

        if(k==0){

            System.out.println(root.data);

            return null;

        }

        k--;

        return root;

    }

    return null;

}

IS Same level leaf

Find min height and max height if both are equal then true else false

public static boolean leaf(Node root){

        if(root==null){

            return true;

        }

        int max\_Depth=heightmax(root);

        int min\_Depth=heightmin(root);

        if(max\_Depth==min\_Depth){

            return true;

        }

        return false;

    }

    public static int heightmax(Node root)

    {

        if(root==null){

            return 0;

        }

        int lh=heightmax(root.left);

        int rh=heightmax(root.right);

        return 1+Math.max(lh,rh);

    }

    public static int heightmin(Node root)

    {

        if(root==null){

            return 0;

        }

        int lh=heightmin(root.left);

        int rh=heightmin(root.right);

        return 1+Math.min(lh,rh);

    }

Left view

public static ArrayList<Integer> leftView(Node root)

    {

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

        flaa(root,0,ans);

        return ans;

    }

    static void flaa(Node root, int level,ArrayList ans){

        if(root==null){

            return;

        }

        if(ans.size()==level){

            ans.add(root.data);

        }

        flaa(root.left,level+1,ans);

        flaa(root.right,level+1,ans);

    }

Level Order Traversal

class Solution {

    public List<List<Integer>> levelOrder(TreeNode root) {

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

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

        if(root==null){

            return finalAns;

        }

        q.add(root);

        while(!q.isEmpty()){

            int levels = q.size();

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

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

                TreeNode s=q.peek();

                q.remove();

                if(s.left!=null){

                    q.add(s.left);

                }

                if(s.right!=null){

                    q.add(s.right);

                }

                subLevels.add(s.val);

            }

            finalAns.add(subLevels);

        }

        return finalAns;

    }

}

Lowest Ancestor

Node findLCA(Node node, int n1, int n2)

    {

        if (node == null ||node.data == n1 || node.data == n2)

            return node;

        Node left\_lca = findLCA(node.left, n1, n2);

        Node right\_lca = findLCA(node.right, n1, n2);

        if (left\_lca != null && right\_lca != null)

            return node;

        if(left\_lca != null){

            return left\_lca;

        }

            return right\_lca;

    }

MaxDepth/Height

public static int height(Node root)

    {

        if(root==null){

            return 0;

        }

        int lh=height(root.left);

        int rh=height(root.right);

        return 1+Math.max(lh,rh);

    }

Min Distnce b/w two Nodes

static int distance(Node root,int n1,int n2){

        Node lca=findlca(root, n1,n2);

        int d1=dist(lca, n1, 0);

        int d2=dist(lca, n2,0);

        return d1+d2;

    }

    static int dist(Node root,int k,int distan){

        if(root==null){

            return -1;

        }

        if(root.data==k){

            return distan;

        }

        int left=dist(root.left, k, distan+1);

        if(left!=-1){

            return left;

        }

        return dist(root.right, k, distan+1);

    }

Inorder and preorder array to Binary tree

    public TreeNode buildTree(int[] preorder, int[] inorder)

    {

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

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

            map.put(inorder[i],i);

        }

         TreeNode root= build(preorder,0,preorder.length-1, inorder,0,inorder.length-1,map);

         return root;

    }

    public TreeNode build(int[] preorder, int prestart, int preend,int[] inorder,int instart,int inend ,HashMap<Integer,Integer> map){

        if(prestart>preend || instart>inend){

            return null;

        }

        TreeNode root= new TreeNode(preorder[prestart]);

        int index=map.get(root.val);

        int area=index-instart;

        root.left=build(preorder, prestart+1, prestart+area, inorder, instart, index-1, map);

        root.right=build(preorder, prestart+area+1, preend, inorder, index+1, inend, map);

        return root;

    }

Reverse Traversal

 Stack<Node> S = new Stack();

        Queue<Node> Q = new LinkedList();

        Q.add(root);

        while (!Q.isEmpty())

        {

            root = Q.peek();

            Q.remove();

            S.push(root);

            if (root.right != null)

                Q.add(root.right);

            if (root.left != null)

                Q.add(root.left);

        }

        while (!S.empty())

        {

            int l = S.peek().data;

            System.out.print(l + " ");

            S.pop();

        }

Right View

static void flaa(Node root, int level,ArrayList ans){

        if(root==null){

            return;

        }

        if(ans.size()==level){

            ans.add(root.data);

        }

        flaa(root.right,level+1,ans);

        flaa(root.left,level+1,ans);

    }

Tree From String

static Node treeFromString(String s)

{

    if(s==null ||s.length()==0){

        return null;

    }

    return str2(s);

}

static int start=0;

static Node str2(String s){

    if(start>=s.length()){

        return null;

    }

    boolean neg=false;

    if(s.charAt(start)=='-'){

        neg=true;

        start++;

    }

    int num=0;

    while(start<s.length() && Character.isDigit(s.charAt(start))){

        int digit=Character.getNumericValue(s.charAt(start));

        num=num\*10+digit;

        start++;

    }

    System.out.println(num);

    if(neg==true){

        num=-num;

        neg=false;

    }

    Node root= newNode(num);

    if(start>=s.length()){

        return root;

    }

    if(start<s.length() && s.charAt(start)=='('){

        start++;

        root.left=str2(s);

    }

    if(start<s.length() && s.charAt(start)==')'){

        start++;

        return root;

    }

    if(start<s.length() && s.charAt(start)=='('){

        start++;

        root.right=str2(s);

    }

    if(start<s.length() && s.charAt(start)==')'){

        start++;

        return root;

    }

    return root;

}

Sum tree their child

int toSumTree(Node node)

    {

        if(node==null){

            return 0;

        }

        int old\_val= node.data;

        node.data=toSumTree(node.left)+toSumTree(node.right);

        return old\_val+node.data;

    }

Top View

private void TopView(Node root)

    {

        class QueueObj {

            Node node;

            int hd;

            QueueObj(Node node, int hd)

            {

                this.node = node;

                this.hd = hd;

            }

        }

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

        Map<Integer, Node> map= new TreeMap<Integer, Node>();

        if (root == null) {

            return;

        }else {

            q.add(new QueueObj(root, 0));

        }

        System.out.println(

            "The top view of the tree is : ");

        while (!q.isEmpty()) {

            QueueObj tmp = q.poll();

            int hd=tmp.hd;

            Node h=tmp.node;

            if (!map.containsKey(hd)) {

                map.put(hd,h);

            }

            if (h.left != null) {

                q.add(new QueueObj(h.left,hd - 1));

            }

            if (h.right != null) {

                q.add(new QueueObj(h.right,hd + 1));

            }

        }

        for (Map.Entry<Integer, Node> entry :

            map.entrySet()) {

            System.out.print(entry.getValue().key + " ");

        }

    }

Zigzag Traversal

class Solution {

    public List<List<Integer>> zigzagLevelOrder(TreeNode root) {

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

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

        boolean f=true;

        if(root==null){

            return finalAns;

        }

        q.add(root);

        while(!q.isEmpty()){

            int levels = q.size();

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

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

                TreeNode node=q.peek();

                q.poll();

                if(f){

                    l.add(node.val);

                }else{

                    l.add(0,node.val);

                }

                if(node.left!=null){

                    q.add(node.left);

                }

                if(node.right!=null){

                    q.add(node.right);

                }

            }

            f=!f;

            finalAns.add(l);

    }

    return finalAns;

}}

Queue using stack

class MyQueue {

public Stack<Integer> s1 = new Stack<>();

public Stack<Integer> s2 = new Stack<>();

public MyQueue() {

}

public void push(int x) {

while(!s1.isEmpty()){

s2.push(s1.pop());

}

s2.push(x);

while(!s2.isEmpty()){

s1.push(s2.pop());

}

}

public int pop() {

if(!s1.isEmpty()){

return s1.pop();

}

return -1;

}

public int peek() {

return s1.peek();

}

public boolean empty() {

return s1.isEmpty();

}

}

Stack Using queue

class MyStack {

Queue<Integer> q1=new LinkedList<Integer>();

Queue<Integer> q2=new LinkedList<Integer>();

public MyStack() {

}

public void push(int x) {

while(!q1.isEmpty()){

q2.add(q1.peek());

q1.remove();

}

q1.add(x);

while(!q2.isEmpty()){

q1.add(q2.peek());

q2.remove();

}

}

public int pop() {

int x=q1.peek();

q1.remove();

return x;

}

public int top() {

return q1.peek();

}

public boolean empty() {

return q1.isEmpty();

}

}