PEPCODING QUESTION AND ANSWERS

1.leetcode 952 long pressed name(just check or simple while);dry run and pointer

class Solution {

public:

bool isLongPressedName(string name, string typed) {

int j=0;

for(int i=0;i<name.length();)

{ cout<<typed[j]<<" "<<name[i];

if(name[i]!=typed[j])return false;

while(name[i]==name[i+1]&&i+1<=name.size())

{

i++;

j++;

if(typed[j]!=name[i]) return false;

}

while(name[i]==typed[j])

{

j++;

// cout<<typed[i];

}

i++;

}

if(j==typed.size())return true;

return false;

}

};

2.range addition-leetcode 370:to learn that prefix sum and 0(n)

public class Solution {

/\*\*

\* @param length: the length of the array

\* @param updates: update operations

\* @return: the modified array after all k operations were executed

\*/

public int[] getModifiedArray(int length, int[][] updates) {

// Write your code here

int[]ans=new int[length];

// System.out.System.out.println(ans[0]);

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

{

ans[updates[i][0]]+=updates[i][2];

if(updates[i][1]+1<length)ans[updates[i][1]+1]-=updates[i][2];

}

for(int i=1;i<length;i++)

{

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

}

return ans;

}

}

3.cookoff max range queries:first find prefix sum array;then count array for i.e.prefix count of k;then find prefix count of k+1;then for each query find cost that is intial count of k-prefix count of k+prefix count of k+1

4.rotate array leetcode 189:reverse k,then reverse n-k;then reverse whole array

class Solution {

public:

void rotate(vector<int>& nums, int k) {

if(k<0)k=k+nums.size();

if(k%nums.size()==0)return;

k=k%nums.size();

int end=nums.size()-k;

for(int i=0;i<end/2;i++)

{

int tmp=nums[i];

nums[i]=nums[end-i-1];

nums[end-i-1]=tmp;

}

int st=end;

for(int i=st;i<st+((nums.size()-st)/2);i++)

{

int tmp=nums[i];

nums[i]=nums[nums.size()-i+st-1];

nums[nums.size()-i+st-1]=tmp;

}

for(int i=0;i<nums.size()/2;i++)

{

int tmp=nums[i];

nums[i]=nums[nums.size()-i-1];

nums[nums.size()-i-1]=tmp;

}

}

};

5.nextgreater leetcode 556

12.sieve erathonens:to find prime within range within range and complexity of loglogn

#include <iostream>

#include<vector>

#include<cmath>

using namespace std;

void ans(int n)

{

vector<bool> ar(n+1,false);

int s=sqrt(n);

for(int i=2;i<=s;i++)

{ if(ar[i]==false){

int j=2;

while(j\*i<=n)

{

ar[j\*i]=true;

j++;

}

}

}

for(int i=2;i<ar.size();i++)

{

if(ar[i]==false)cout<<i<<" ";

}

}

int main() {

//code

int tc;

cin>>tc;

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

{

int num;

cin>>num;

ans(num);

cout<<endl;

}

return 0;

}

13.all prime between two nos. but with space constraint,let n>m.So two array of size sqrt(n) and n-m

14.leetcode 977 squares of element in the array:0(n) by two pointer approach

class Solution {

public:

vector<int> sortedSquares(vector<int>& A) {

vector<int>ans(A.size(),0);

int i=0;int j=A.size()-1;int k=A.size()-1;

while(k>=0)

{

if(abs(A[i])>=abs(A[j]))

{

ans[k]=A[i]\*A[i];

i++;k--;

}

else if(abs(A[j])>abs(A[i]))

{

ans[k]=A[j]\*A[j];

j--;k--;

}

}

return ans;

}

};

15.fast exponential:logn

#include <iostream>

using namespace std;

int main() {

long exp,base;

cin>>base>>exp;

long t = 1L;

long N=1000000007L;

while(exp>0)

{

if (exp % 2 != 0)

t = (t \* base) % N;

base = (base \* base) % N;

exp /= 2;

}

cout<<t%N;

return 0;

}

16.fibonacci:in logn by making matrix of(fn+1,fn,fn,fn-1) leetcode 509

class Solution {

public:

vector<vector<int>> multipl(vector<vector<int>>a,vector<vector<int>>b)

{

vector<vector<int>>ans(2,vector<int>(2,0));

ans[0][0]=a[0][0]\*b[0][0]+a[0][1]\*b[1][0];

ans[0][1]=a[0][0]\*b[0][1]+a[0][1]\*b[1][1];

ans[1][0]=a[1][0]\*b[0][0]+a[1][1]\*b[1][0];

ans[1][1]=a[1][0]\*b[0][1]+a[1][1]\*b[1][1];

return ans;

}

vector<vector<int>>i={{1,1},{1,0}};

vector<vector<int>>fib(vector<vector<int>>nmat,int n)

{

if(n==1) return i;

vector<vector<int>>ans=multipl(fib(nmat,n/2),fib(nmat,n/2));

return n%2==0?ans:multipl(ans,i);

}

int fib(int N) {

if (N==0) return 0;

vector<vector<int>>ans=fib(i,N);

return ans[1][0];

}

};

17.segegrate 0’s and 1’s -gfg

#include <iostream>

#include<vector>

using namespace std;

int main() {

int tc;

cin>>tc;

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

{ int n;

cin>>n;

vector<int>arr(n,0);

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

{

cin>>arr[j];

}

int r=0,c=0;

while(c<n)

{ if(arr[c]==0)

{ swap(arr[r],arr[c]);

r++;

}

c++;

}

for(int g=0;g<n;g++)

{

cout<<arr[g]<<" ";

}

cout<<endl;

}

//code

return 0;

}

18.segregate 0,1,2:three pointer such that a is always infront of 0,b infront of 1,c at end or before start of 2

#include <iostream>

#include<vector>

using namespace std;

int main() {

int tc;

cin>>tc;

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

{ int n;

cin>>n;

vector<int>arr(n,0);

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

{

cin>>arr[j];

}

int r=0,c=0;int k=n-1;

while(c<=k)

{ if(arr[c]==0)

{ swap(arr[r],arr[c]);

r++;c++;

}

else if(arr[c]==2)

{

swap(arr[k],arr[c]);k--;

}

if(arr[c]==1){c++;}

}

for(int g=0;g<n;g++)

{

cout<<arr[g]<<" ";

}

cout<<endl;

}

//code

return 0;

}

19.leetcode 905 sort array by parity:two pointer approach

class Solution {

public:

vector<int> sortArrayByParity(vector<int>& A) {

int i=0,j=0;

while(j<A.size())

{

if(A[j]%2==0)

{

swap(A[j],A[i]);

i++;

}

j++;

}

return A;

}

};

20.optimal division leetcode 553:to remember parenthesis after i=1 index

class Solution {

public:

string optimalDivision(vector<int>& nums) {

if(nums.size()==0)return "";

if(nums.size()==1)return to\_string(nums[0]);

if(nums.size()==2)return to\_string(nums[0])+"/"+to\_string(nums[1]);

string s="";

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

{ if(i==1)s=s+"(";

if(i<=nums.size()-2)

s=s+to\_string(nums[i])+"/";

if(i==nums.size()-1)s=s+to\_string(nums[i])+")";

}

return s;

}

};

21.leetcode 296:best meeting points:median of points;

class Solution {

public:

/\*\*

\* @param grid: a 2D grid

\* @return: the minimize travel distance

\*/

int minTotalDistance(vector<vector<int>> &grid) {

// Write your code here

vector<int>row;

vector<int>col;

for(int i=0;i<grid.size();i++)

{

for(int j=0;j<grid[0].size();j++)

{

if(grid[i][j]==1)

{

row.push\_back(i);

}

}

}

for(int i=0;i<grid[0].size();i++)

{

for(int j=0;j<grid.size();j++)

{

if(grid[j][i]==1)

{

col.push\_back(i);

}

}

}

int midrow=row[(row.size()/2)];

int midcol=col[(col.size()/2)];

int sum=0;

for(int i=0;i<grid.size();i++)

{

for(int j=0;j<grid[0].size();j++)

{

if(grid[i][j]==1)

{

sum+=abs(midrow-i)+abs(midcol-j);

}

}

}

return sum;

}

};

22.leetcode 670 -maximum swap

23.gfg two sum:two pointer after sorting the array .nlogn

#include <iostream>

#include <algorithm>

#include<vector>

using namespace std;

bool hassum(vector<int>&arr,int sum)

{

int i=0,j=arr.size();

while(i<j)

{

if(arr[i]+arr[j]<sum)i++;

else if(arr[i]+arr[j]>sum)j--;

else if(arr[i]+arr[j]==sum)return true;

}return false;

}

int main() {

//code

int tc;

cin>>tc;

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

{

int n;

cin>>n;

int sum;

cin>>sum;

vector<int>arr(n,0);

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

{

cin>>arr[j];

}

sort(arr.begin(), arr.end());

if(hassum(arr,sum)==true)cout<<"Yes"<<endl;

else cout<<"No"<<endl;

}

return 0;

}

24.maximum diff of pair gfg:sort,two point i=0,j=0

#include <iostream>

#include <algorithm>

#include<vector>

using namespace std;

bool hasdiff(vector<int>& arr,int diff)

{

int i=0;int j=0;

while(j<arr.size()&&i<=j)

{

if(arr[j]-arr[i]==diff)return true;

else if(arr[j]-arr[i]>diff)i++;

if(arr[j]-arr[i]<diff)j++;

}

return false;

}

int main() {

//code

int tc;

cin>>tc;

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

{

int n;

cin>>n;

int sum;

cin>>sum;

vector<int>arr(n,0);

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

{

cin>>arr[j];

}

sort(arr.begin(),arr.end());

if(hasdiff(arr,sum))cout<<"1";

else cout<<"-1";

}

return 0;

}

25.leetcode 881:boats to save ppl:sort and then two pointer approach

class Solution {

public:

int numRescueBoats(vector<int>& people, int limit) {

sort(people.begin(),people.end());

int i=0;int j=people.size()-1;

int boat=0;

while(i<=j)

{

if(people[i]+people[j]<=limit){

i++;j--;boat++;

}

else {j--;boat++;}

}

return boat;

}

};

26.leetcode 169:majority element->keep val and count and change val on count==0

class Solution {

public:

int majorityElement(vector<int>& nums) {

int val=nums[0];int count=1;

for(int i=1;i<nums.size();i++)

{

if(nums[i]==val)count++;

else count--;

if(count==0){val=nums[i];count=1;}

}

return val;

}

};

27.majority element 11:leetcode 229:4 extra variable and update

class Solution {

public:

vector<int> majorityElement(vector<int>& nums) {

if(nums.size()==0||nums.size()==1)return nums;

int val1=nums[0],val2=nums[0],count1=1,count2=0;

for(int i=1;i<nums.size();i++)

{

if(val1==nums[i])count1++;

else if(val2==nums[i])count2++;

else if (count1==0)

{

val1=nums[i];count1=1;

}

else if (count2==0)

{

val2=nums[i];count2=1;

}

else

{

count1--;

count2--;

}

}

cout<<val1<<val2;

int s=0,c=0;

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

{

if(val1==nums[i])s++;

else if(val2==nums[i])c++;

}

vector<int>ans;

if(s>nums.size()/3)ans.push\_back(val1);

if(val1!=val2 &&c>nums.size()/3)ans.push\_back(val2);

return ans;

}

};

# 28. Given an array of size n and a number k, find all elements that appear more than n/k times:hashmap approach freq map types hashing

# 1.store in hashmap as in map[key]++

# 2.check if map[key]>=k by for( pair<int,int>s:map) if(s.second>=k) ans.add(s.first);

# 29.leetcode 769:maxchunks to make sorted array :if(temporary max=curpos)chunks++

# class Solution {

# public:

# int maxChunksToSorted(vector<int>& arr) {

# int chunks=0,i=0,max\_=-1;

# while(i<arr.size())

# {

# max\_=max(max\_,arr[i]);

# if(max\_==i)chunks++;

# i++;

# }

# return chunks;

# 

# }

# };

30.maxchunks leetcode 768->maximum from start and min from end if start<end chunk++

class Solution {

public:

int maxChunksToSorted(vector<int>& arr) {

vector<int>maxfromstart(arr.size(),arr[0]);

vector<int>minfromend(arr.size(),arr[arr.size()-1]);

int max\_=-1;

int chunks=0;

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

{

max\_=max(max\_,arr[i]);

maxfromstart[i]=max\_;

}

int min\_=(int )1e8;

for(int i=arr.size()-1;i>=0;i--)

{

min\_=min(min\_,arr[i]);

minfromend[i]=min\_;

}

for(int i=0;i<arr.size()-1;i++)

{

if(maxfromstart[i]<=minfromend[i+1])chunks++;

}

return chunks+1;

}

};

31.leetcode 795 number of subarray with bounded maximum:to define conditions for less than l greater than R and between them.when less between to first low ab tk add

class Solution {

public:

int numSubarrayBoundedMax(vector<int>& A, int L, int R) {

int smaller =0,j=0,ans=0;

for(int i=0;i<A.size();i++)

{

if(A[i]<=R && A[i]>=L)

{

smaller=i-j+1;

ans+=smaller;

}

else if(A[i]<L)

{

ans+=smaller;

}

else

{

smaller=0;

j=i+1;

}

}

return ans;

}

};

32.

33.leetcode 628 maximum product:o(nlogn)sort then either start or end product will that is last 3 or last1 and start 2

class Solution {

public:

int maximumProduct(vector<int>& nums) {

sort(nums.begin(),nums.end());

if(nums.size()<3)return -1;

int lastprod=nums[nums.size()-1]\*nums[nums.size()-2]\*nums[nums.size()-3];

int startprod=nums[nums.size()-1]\*nums[0]\*nums[1];

return lastprod>=startprod?lastprod:startprod;

}

};

34.leectcode 747 :calculate large and secondlarge

class Solution {

public:

int dominantIndex(vector<int>& nums) {

int large=-1;

int secondlarge=-1;int largeidx=-1;

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

{

if(large<nums[i])

{

large=nums[i];

largeidx=i;

}

}

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

{

if(large>nums[i]&& secondlarge<nums[i])

{

secondlarge=nums[i];

}

}

cout<<secondlarge<<large<<largeidx;

return 2\*secondlarge<=large?largeidx:-1;

}

};

35.wiggle sort leetcode 280:for odd and even either swap or no swap if eligible pos found

public class Solution {

/\*

\* @param nums: A list of integers

\* @return: nothing

\*/

public void wiggleSort(int[] nums) {

// write your code

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

{

if(i%2==0)

{

if(nums[i]>nums[i+1])

{

int tmp=nums[i];

nums[i]=nums[i+1];

nums[i+1]=tmp;

}

}

else

{

if(nums[i]<nums[i+1])

{

int tmp=nums[i];

nums[i]=nums[i+1];

nums[i+1]=tmp;

}

}

}

}

}

36.leetcode 487 maximum consecutiveones 2:store zero last occurrence and change length accordingly

class Solution {

public:

/\*\*

\* @param nums: a list of integer

\* @return: return a integer, denote the maximum number of consecutive 1s

\*/

int findMaxConsecutiveOnes(vector<int> &nums) {

// write your code here

int lastzero=-1;

int len=0,maxlen=-1;

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

{

if(nums[i]==1)len++;

if(nums[i]==0)

{

if(lastzero==-1)

{

lastzero=i;

len++;

}

else{

maxlen=max(maxlen,len);

len=i-lastzero;

lastzero=i;

}

}

maxlen=max(len,maxlen);

}

return maxlen;

}

};

37.leetcode 1004-maxconsecutive ones 3:so take a queue 3 conditions first when arr[i]=1,arr[i]=0 & k>=zerocount and k<zerocount

class Solution {

public:

int longestOnes(vector<int>& A, int K) {

queue<int>que;

int len=0,maxlen=0,countzero=0;

for(int i=0;i<A.size();i++)

{

if(A[i]==1)

{len++; maxlen=max(maxlen,len);}

if(A[i]==0)

{

que.push(i);

countzero++;

if(countzero<=K)

{

len++;

maxlen=max(maxlen,len);

}

else

{

int idx=que.front();

que.pop();

cout<<idx;

maxlen=max(len,maxlen);

len=i-idx;

countzero--;

}

}

}

return maxlen;

}

};

38.leetcode 763 partition labels:make hashmap or vector to store last occurance of char if char lastoccurance<map[s[i]] update lastoccurance else i==map[s[i]] then count idx=first-last+1

class Solution {

public:

vector<int> partitionLabels(string S) {

unordered\_map<char,int> map;

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

{

map[S[i]]=i;

}

vector<int>ans;

int firstoccurance=0;

int lastoccurance=map[S[0]];

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

{

if(map[S[i]]>lastoccurance)

{

lastoccurance=map[S[i]];

}

if(lastoccurance==i)

{

ans.push\_back(lastoccurance+1-firstoccurance);

firstoccurance=i+1;

}

}

return ans;

}

};

39.leetcode 48:rotate matrix 90:so in rotation question we first take transpose then either row exchange orr col exchange

class Solution {

public:

void rotate(vector<vector<int>>& matrix) {

for(int i=0;i<matrix.size();i++)

{

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

{

int tmp=matrix[i][j];

matrix[i][j]=matrix[j][i];

matrix[j][i]=tmp;

}

}

for(int i=0;i<matrix.size();i++)

{ for(int j=0;j<matrix[0].size()/2;j++)

{

int tmp=matrix[i][j];

matrix[i][j]=matrix[i][matrix[0].size()-1-j];

matrix[i][matrix[0].size()-1-j]=tmp;

}

}

}

};

40.leetcode 829:consecutive sum of integer:brute force is two pointer with going till n/2

So basically if we prove let k be length then x+(x+1)….+x+k=n

After solving we get n-k(k-1)/2=kx

class Solution {

public:

int consecutiveNumbersSum(int n) {

int count=0,k=1;

while(k<sqrt(2\*n))

{

if(((n-((k\*k-k)/2))%k)==0)count++;

k++;

}

return count;

}

};

Brute force:

class Solution {

public:

int consecutiveNumbersSum(int N) {

if(N<=2)return 1;

int i=1,sum=1,j=1,count=0;

while(j<=N/2+1)

{

if(sum<N){j++;sum+=j;}

else if(sum>N){

sum=sum-i;

i++;

}

else if(sum==N)

{ count++;

sum=sum-i;

i++;

j++;

sum=sum+j;

}

}

return count+1;

}

};

41 leetcode 152 max prod subarray:if kadance approach then two variable needed to store maxneg and maxpos and if num<0 then maxpos=max(num,maxneg\*num) vice versa for maxpos and update grandprod

class Solution {

public:

int maxProduct(vector<int>& nums) {

//naive

int maxneg=nums[0];

int maxpos=nums[0],maxprod=nums[0];

for(int i=1;i<nums.size();i++)

{

if(nums[i]>0)

{ int a=maxneg;int b=maxpos;

maxneg=min(nums[i],a\*nums[i]);

maxpos=max(nums[i],b\*nums[i]);

maxprod=max(maxprod,maxpos);

}

else

{ int a=maxneg;int b=maxpos;

maxneg=min(nums[i],b\*nums[i]);

maxpos=max(nums[i],a\*nums[i]);

// cout<<maxpos<<" ";

maxprod=max(maxprod,maxpos);

}

}

return maxprod;

}

};

**OR:find prefix sum and prod and take take of zero case by updating at zero**

class Solution {

public:

int maxProduct(vector<int>& nums) {

int prefix=nums[0];

int prodans=nums[0];

int flag=0;

for(int i=1;i<nums.size();i++)

{ if(nums[i]==0)

{

prefix=0;

flag=1;

}

else

{ if(prefix==0)prefix=nums[i];

else prefix\*=nums[i];

}

prodans=max(prodans,prefix);

}

//prodans=max(prodans,prefix);

int suffix=nums[nums.size()-1];

prodans=max(prodans,suffix);

for(int i=nums.size()-2;i>=0;i--)

{ if(nums[i]==0)

{

suffix=0;

flag=1;

}

else

{ if(suffix==0)suffix=nums[i];

else suffix\*=nums[i];

}

prodans=max(prodans,suffix);

}

if(prodans<0 &&flag==1)return 0;

return prodans;

}

};

42.leetcode 1007 minimum dominoes rotation for equal row :to check all options as in two letter of start and output the minimum

class Solution {

public:

int minDominoRotations(vector<int>& A, vector<int>& B) {

int val1=A[0];

int val2=B[0];

int swapAval1=0;

for(int i=0;i<A.size();i++)

{

if(A[i]==val1);

else if(B[i]==val1)swapAval1++;

else {swapAval1=INT\_MAX;break;}

}

int swapAval2=0;

for(int i=0;i<A.size();i++)

{

if(A[i]==val2);

else if(B[i]==val2)swapAval2++;

else {swapAval2=INT\_MAX;break;}

}

int swapBval1=0;

for(int i=0;i<A.size();i++)

{

if(B[i]==val1);

else if(A[i]==val1)swapBval1++;

else {swapBval1=INT\_MAX;break;}

}

int swapBval2=0;

for(int i=0;i<A.size();i++)

{

if(B[i]==val2);

else if(A[i]==val2)swapBval2++;

else {swapBval2=INT\_MAX;break;}

}

return min(min(swapBval1,swapBval2),min(swapAval1,swapAval2))==INT\_MAX?-1:min(min(swapBval1,swapBval2),min(swapAval1,swapAval2));

}

};

43.leetcode 345:reverse vowel in a string:DO IT INPLACE just replace and dry run

class Solution {

public:

string reverseVowels(string s) {

string ans="";

int i=0;int j=s.length()-1;

while(i<j)

{

if(s[i]=='a'||s[i]=='e'||s[i]=='i'||s[i]=='o'||s[i]=='u'||s[i]=='A'||s[i]=='E'||s[i]=='I'||s[i]=='O'||s[i]=='U')

{

while(s[j]!='a'&&s[j]!='e'&&s[j]!='i'&&s[j]!='o'&&s[j]!='u'&&s[j]!='A'&&s[j]!='E'&&s[j]!='I'&&s[j]!='O'&&s[j]!='U')

{

j--;

}

char tmp=s[i];

s[i]=s[j];

s[j]=tmp;

cout<<s[i]<<s[j]<<" ";

j--;

}

i++;

}

return s;

}

};

44.leetcode 119:pascal triangle:do it inplace by starting from end and dry run

class Solution {

public:

vector<int> getRow(int rowIndex) {

vector<int>ans(rowIndex);

ans.push\_back(1);

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

{

int j=i-1;

ans[i]=1;

while(j>0)

{

ans[j]=ans[j]+ans[j-1];

j--;

}

}

return ans;

}

};

45.leetcode 915:partition array in disjoint interval:here we need to compare cmax and update with omax such that partition is created at indx

class Solution {

public:

int partitionDisjoint(vector<int>& A) {

int cmax=A[0],omax=A[0],indx=0;

for(int i=1;i<A.size();i++)

{

if(A[i]>=cmax)

{

if(omax<A[i])omax=A[i];

}

else

{

cmax=omax;

indx=i;

}

}

return indx+1;

}

};

46.leetcode 43 multipy strings:

47.leetcode 849:maximum distance from closest person:three cases and two main to find length update conditions when seats[i]==1

class Solution {

public:

int maxDistToClosest(vector<int>& seats) {

int idx=-1,maxlength=0,omax=0,oidx=-1;

for(int i=0;i<seats.size();i++)

{

if(seats[i]==0)

{

maxlength++;

if(i+1==seats.size())omax=max(omax,maxlength);

}

else if(seats[i]==1)

{

if(omax<maxlength)

{

if(idx==-1)omax=max(omax,maxlength);

else omax=max((maxlength+1)/2,omax);

}

maxlength=0;

idx=i;

}

}

return omax;

}

};

48.

49.valid palindrome leetcode 680:find the length until then it is valid and break and remove that start and end and check palindrome

class Solution {

public:

bool validPalindrome(string s) {

int st=-1,end=s.length();

for(int i=0;i<s.length()/2;i++)

{

if(s[i]!=s[s.length()-1-i])

{

st=i;

end=s.length()-1-i;

break;

}

}

if(st==-1 && end==s.length())return true;

bool res=true;

int st1=st,end1=end;

while(st<=end)

{

if(s[st]!=s[end-1])

{

res=false;

break;

}

st++;end--;

}

if (res==true) return true;

bool res2=true;

while(st1<=end1)

{

if(s[st1+1]!=s[end1])

{

res2=false;break;

}

st1++;end1--;

}

return res2;

}

};

50.leetcode 41 first missing positive:to understand no. is 1 to nums.size or num.size+1 so we can mark indexes

class Solution {

public:

int firstMissingPositive(vector<int>& nums) {

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

{

if(nums[i]<=0)nums[i]=nums.size()+1;

}

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

{

// if(abs(nums[i]) - 1 < nums.size() && nums[abs(nums[i])-1]>=0){

// nums[abs(nums[i])-1]\*= -1;

if(abs(nums[i])-1<nums.size()&&nums[abs(nums[i])-1]>=0) nums[abs(nums[i])-1]\*=-1;

//}

}

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

{

if(nums[i]>0)return i+1;

}

return nums.size()+1;

}

};

52.leetcode 775 :global and local inversions:to find global maxima by place value diff

class Solution {

public:

bool isIdealPermutation(vector<int>& A) {

for(int i=0;i<A.size();i++){

if(abs(A[i]-i)>=2)return false;

}

return true;

}

};

53.next greater element gfg:stack ngor stack will decreasing

61.valid parenthesis leetcode 20:check for closing bracket correct check stack empty and wrong bracket not in when stack is empty(

class Solution {

public:

bool isValid(string s) {

stack<int>st;

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

{

if(s[i]=='('||s[i]=='['||s[i]=='{')st.push(i);

else

{

if(st.empty())return false;

if(s[i]==')')

{

if(s[st.top()]=='(')st.pop();

else return false;

}

else if(s[i]==']')

{

if(s[st.top()]=='[')st.pop();

else return false;

}

if(s[i]=='}')

{

if(s[st.top()]=='{')st.pop();

else return false;

}

}

}

if(st.empty())return true;

return false;

}

};

62:longest valid substring leetcode 32:stack store invalid and find length(

class Solution {

public:

int longestValidParentheses(string s) {

stack<int>st;

st.push(-1);

int maxlen=0;

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

{

if(st.top()!=-1&& s[st.top()]=='('&&s[i]==')')

{

st.pop();

maxlen=max(maxlen,i-st.top());

}

else st.push(i);

}

return maxlen;

}

};

63.count duplicate parenthesis gfg:push in stack such that intil closing and check if count>0

#include <iostream>

#include<stack>

using namespace std;

int main() {

string s="((c+d)+(a+B))";

stack<int>st;

int count=0;

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

{

if(s[i]!=')')st.push(i);

else

{

while(s[st.top()]!='('){

if(s[st.top()=='+'])count++;

st.pop();

}

if(count==0) cout<<false;

st.pop();

}

count=0;

}

//cout<<true;

// cout<<"GfG!";

return 0;

}

64.min no. of bracket reversal to make parenthesis valid gfg:cases 4 such odd c and o open

#include <iostream>

#include<stack>

using namespace std;

int main() {

//code

int tc;

cin>>tc;

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

{

string s;

cin>>s;

stack<int>st;

st.push(-1);

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

{

if(s[i]=='}')

{

if(s[st.top()]=='{')st.pop();

else st.push(i);

}

else st.push(i);

}

int oc=0,cc=0;

while(st.top()!=-1)

{

if(s[st.top()]=='{')oc++;

else if(s[st.top()]=='}') cc++;

st.pop();

}

if(oc==0&&cc==0)cout<<0;

else if(((oc+cc)%2)==1)cout<<-1;

else

{

if(oc%2==1&&cc%2==1)cout<<(oc/2)+(cc/2)+2;

else cout<<(oc+cc)/2;

}

cout<<endl;

}

return 0;

}

65.leetcode 921 min add to make valid parenthesis:same invalid in stack and return size

class Solution {

public:

int minAddToMakeValid(string s) {

stack<int>st;

st.push(-1);

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

{

if(st.top()!=-1&& s[st.top()]=='('&&s[i]==')')

{

st.pop();

}

else st.push(i);

}

return st.size()-1;

}

};

66.leetcode 735 asteroid collision:4 cases with empty 5 cases pop until ele=<top and push if negative and push pos always

class Solution {

public:

vector<int> asteroidCollision(vector<int>& asteroids) {

stack<int>st;

for(int ele:asteroids)

{

if(ele>0)st.push(ele);

else

{

while(!st.empty()&&st.top()>0 &&st.top()<-ele)

{

st.pop();

}

if(!st.empty()&&st.top()>0 &&st.top()==-ele)

{

st.pop();

}

else if(st.empty()||st.top()<0)

{

st.push(ele);

}

}

}

vector<int>ans(st.size(),0);

for(int i=st.size()-1;i>=0;i--)

{

ans[i]=st.top();st.pop();

}

return ans;

}

};

67.leetcode 844,backspace string :simple string manipulation

class Solution {

public:

bool backspaceCompare(string s, string t) {

stack<int>st1;stack<int>st2;

st1.push(-1);st2.push(-1);

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

{

if(s[i]=='#'&&st1.top()!=-1)st1.pop();

else if(s[i]!='#')st1.push(i);

}

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

{

if(t[i]=='#'&&st2.top()!=-1)st2.pop();

else if(t[i]!='#')st2.push(i);

}

while(st1.top()!=-1&&st2.top()!=-1)

{

if(s[st1.top()]!=t[st2.top()])return false;

st1.pop();

st2.pop();

}

cout<<st1.top()<<st2.top();

return (st1.top()==-1&&st2.top()==-1)? true:false;

}

};

68.maximum subsequence of unbalanced valid parenthesis:so answer is either length or length-1;

69.

70.leetcode 134 gas station:to choose start and considering if while transversing start pehle nhi hai to agge bhi nhi hoga so store beforestart sum n aftr start sum

class Solution {

public:

int canCompleteCircuit(vector<int>& gas, vector<int>& cost) {

int epbeforestart=0;

int epafterstart=0;

int start=0;

int sum1=0,sum2=0;

for(int i=0;i<gas.size();i++)

{

sum1+=gas[i];

sum2+=cost[i];

}

if(sum1<sum2)return -1;

for(int i=0;i<gas.size();i++)

{

if(epafterstart+gas[i]-cost[i]>=0)

{

epafterstart=epafterstart+gas[i]-cost[i];

}

else

{

epbeforestart+=epafterstart+gas[i]-cost[i];

epafterstart=0;

start=i+1;

}

}

return start;

}

};

71.leetcode 853 car fleet:calculate time first sort wrt to position

class Solution {

public int carFleet(int target, int[] position, int[] speed) {

int N = position.length;

Car[] cars = new Car[N];

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

cars[i] = new Car(position[i], (double) (target - position[i]) / speed[i]);

Arrays.sort(cars, (a, b) -> Integer.compare(a.position, b.position));

int ans = 0, t = N;

while (--t > 0) {

if (cars[t].time < cars[t-1].time) ans++; //if cars[t] arrives sooner, it can't be caught

else cars[t-1] = cars[t]; //else, cars[t-1] arrives at same time as cars[t]

}

return ans + (t == 0 ? 1 : 0); //lone car is fleet (if it exists)

}

}

class Car {

int position;

double time;

Car(int p, double t) {

position = p;

time = t;

}

}

72.first negative in the window of k size gfg:start from last and save last negative

#include <iostream>

#include<vector>

using namespace std;

int main() {

int tc;

cin>>tc;

for(int t=0;t<tc;t++)

{

int n;

cin>>n;

vector<int>arr(n,0);

for(int i=0;i<n;i++){cin>>arr[i];}

int k;

cin>>k;

int lastneg=-1;

vector<int>ans(n,0);

for(int i=n-1;i>=0;i--)

{

if(arr[i]<0)

{lastneg=i;

ans[i]=arr[i];

}

else

{

if(lastneg!=-1)

{

if(i+k>lastneg)

{

ans[i]=arr[lastneg];

}

}

}

}

for(int i=0;i<n-k+1;i++)

{cout<<ans[i]<<" ";}

cout<<endl;

}

return 0;

}

73.a interesting method to generate binary no.s gfg:use queue as add last and remov e first

#include <iostream>

#include<queue>

using namespace std;

int main() {

//code

int tc;

cin>>tc;

for(int t=0;t<tc;t++)

{

int n;

cin>>n;

queue<string>que;

que.push("1");

while(n>0)

{

string s=que.front();que.pop();

que.push(s+"0");

que.push(s+"1");

n--;

cout<<s<<" ";

}

cout<<endl;

}

return 0;

}

74.max sum of smallest and second smallest subarray :intuition to see only adjacent sum

#include <iostream>

using namespace std;

int main() {

int tc;

cin>>tc;

for(int t=0;t<tc;t++)

{

int n;

cin>>n;

int arr[n];

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

{

cin>>arr[i];

}

int max\_=(int)-1e8;;

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

{

max\_=max(arr[i]+arr[i+1],max\_);

}

cout<<max\_<<endl;

}

//code

return 0;

}

75.leetcode 946 validate stack:do stack operation only

class Solution {

public:

bool validateStackSequences(vector<int>& pushed, vector<int>& popped) {

stack<int>st;int j=0;

for(int i=0;i<pushed.size();i++)

{

while(!st.empty()&&st.top()==popped[j])

{

j++;

st.pop();

}

st.push(pushed[i]);

}

while(j<popped.size())

{

cout<<st.top()<<popped[j];

if(st.top()!=popped[j])return false;

j++;

st.pop();

}

return true;

}

};

77.leetcode 155 minstack:to find min use distance method

class MinStack {

stack<long int> s;

long int mine;

public:

/\*\* initialize your data structure here. \*/

MinStack() {

}

void push(int x) {

if(s.empty())

{

s.push(0);

mine=x;

}

else

{

s.push(x-mine);

if(x<mine)

{

mine=x;

}

}

}

void pop() {

long int m=s.top();

s.pop();

if(m<0)

{

mine=mine-m;

}

}

int top() {

if(s.top()>0)

{

return s.top()+mine;

}

return mine;

}

int getMin() {

return mine;

}

};

/\*\*

\* Your MinStack object will be instantiated and called as such:

\* MinStack\* obj = new MinStack();

\* obj->push(x);

\* obj->pop();

\* int param\_3 = obj->top();

\* int param\_4 = obj->getMin();

\*/

78.adapters:

Stacks by linkedlist:push()>add last0(1) remove last->0(n) addfirst0(1) remove first0(1)

Queue by ll:push>addfist(0[1])removelast0(n)

Stack using queue:push>addlast (0(1)) remove:take two queuesor 1 queue also

Queue using stack:push()addlast pop:two stacks

79.k stack

80.kqueue

81.infix evalution :leetcode calculator

class Solution {

public:

int calculate(string s) {

stack<long int> num;

stack<char> signs;

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

{

if(s[i]=='('||s[i]=='+'||s[i]=='-')signs.push(s[i]);

else if(s[i]==')')

{

if(signs.top()=='(')signs.pop();

else

{

stack<long int>small;

stack<char>ch;

long int pop1=num.top();num.pop();

long int pop2=num.top();num.pop();

small.push(pop1);

small.push(pop2);

ch.push(signs.top());

signs.pop();

while(signs.top()!='(')

{

ch.push(signs.top());

signs.pop();

small.push(num.top());

num.pop();

}

if(signs.top()=='(')

{signs.pop();cout<<"1";}

while(!ch.empty())

{ long int pop1=small.top();small.pop();

long int pop2=small.top();small.pop();

cout<<pop2<<ch.top()<<pop1<<endl;

if(ch.top()=='+')small.push(pop1+pop2);

if(ch.top()=='-')small.push(pop1-pop2);

ch.pop();}

num.push(small.top());

small.pop();

}

}

else

{

if(s[i]-'0'!=-16)

{ int n=s[i]-'0';

while(i+1<s.length()&&s[i+1]!=')'&&s[i+1]!='('&&s[i+1]!='+'&&s[i+1]!='-'&&s[i+1]-'0'!=-16){

int y=s[i+1]-'0';

cout<<y;

n=n\*10+y;i++;

// i++;

}

num.push(n);

}

}

}

if(!signs.empty())

{

stack<long int>small;

stack<char>ch;

long int pop1=num.top();num.pop();

long int pop2=num.top();num.pop();

small.push(pop1);

small.push(pop2);

ch.push(signs.top());

signs.pop();

while(!signs.empty())

{

ch.push(signs.top());

signs.pop();

small.push(num.top());

num.pop();

}

while(!ch.empty())

{ long int pop1=small.top();small.pop();

long int pop2=small.top();small.pop();

// cout<<pop2<<ch.top()<<pop1<<endl;

if(ch.top()=='+')small.push(pop1+pop2);

if(ch.top()=='-')small.push(pop1-pop2);

ch.pop();}

num.push(small.top());

small.pop();

}

//return num.size()==1?num.top():stoi(s);

return num.top();

}

};

Infix evaluation:two stacks

Infix to postfix:operator and operand and pop2 pop1 op

Infix to prefix: operator and operand and op pop1 pop2

Postfix evalution:if sign then solve else add

Postfix to infix:one stack if sign : (pop2 op pop1) add until end

Postfix to prefix:sign : (op pop1 pop2)

Prefix evaluation:start from last:if sign:pop1 op pop2

Prefix top infix: start from last:if sign: (pop1 op pop2)

Prefix to postfix:Start from last:if sign:pop1 pop2 op

82.leetcode 316:remove duplicate letters

83.reverse a linked list:use prev and curr break the bond of curr and curr.next(temp) &&curr.next point to prev and then updata curr and prev

/\*\*

\* Definition for singly-linked list.

\* public class ListNode {

\* int val;

\* ListNode next;

\* ListNode() {}

\* ListNode(int val) { this.val = val; }

\* ListNode(int val, ListNode next) { this.val = val; this.next = next; }

\* }

\*/

class Solution {

public ListNode reverseList(ListNode head) {

if(head==null) return null;

ListNode prev=null;

ListNode curr=head;

while(curr.next!=null)

{

ListNode temp=curr.next;

curr.next=prev;

prev=curr;

curr=temp;

}

curr.next=prev;

return curr;

}

}

84.leetcode 876 middle of linkedlist:slow and fast check foe end condition in while

/\*\*

\* Definition for singly-linked list.

\* public class ListNode {

\* int val;

\* ListNode next;

\* ListNode() {}

\* ListNode(int val) { this.val = val; }

\* ListNode(int val, ListNode next) { this.val = val; this.next = next; }

\* }

\*/

class Solution {

public ListNode middleNode(ListNode head) {

ListNode slow=head;

ListNode fast=head;

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

{

slow=slow.next;

fast=fast.next.next;

}

return fast.next==null?slow:slow.next;

}

}

85.floyd cycle leetcode 142:to find start of loop in linked list proof is 2(x+y)=x+(y+z)c this will imply x=z;

/\*\*

\* Definition for singly-linked list.

\* class ListNode {

\* int val;

\* ListNode next;

\* ListNode(int x) {

\* val = x;

\* next = null;

\* }

\* }

\*/

public class Solution {

public ListNode detectCycle(ListNode head) {

if(head==null||head.next==null)return null;

ListNode slow=head;

ListNode fast=head;

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

{

slow=slow.next;

fast=fast.next.next;

if(slow==fast)

{

slow=head;

while(slow!=fast)

{

slow=slow.next;

fast=fast.next;

}

return slow;

}

}

return null;

}

}

86. split a circular linked list into two halves gfg:same slow and fast pointers

void splitList(Node \*head, Node \*\*head1\_ref, Node \*\*head2\_ref)

{

if(head==NULL) return;

Node \* slow=head;

Node \* fast=head;

while(fast->next!=head&&fast->next->next!=head)

{

slow=slow->next;

fast=fast->next->next;

}

if(fast->next->next==head)

{

fast=fast->next;

}

\*head1\_ref=head;

if(head->next != head)

\*head2\_ref = slow->next;

/\* Make second half circular \*/

fast->next = slow->next;

/\* Make first half circular \*/

slow->next = head;

// your code goes here

}

87.clone a linkedlist with next random:3 steps:first create new list then copy random and separate list by dummy pointer

/\*

// Definition for a Node.

class Node {

int val;

Node next;

Node random;

public Node(int val) {

this.val = val;

this.next = null;

this.random = null;

}

}

\*/

class Solution {

public Node copyRandomList(Node head) {

Node copynode=head;

while(copynode!=null)

{

Node n=new Node(copynode.val);

Node forw=copynode.next;

copynode.next=n;

n.next=forw;

copynode=forw;

}

Node randomset=head;

while(randomset!=null)

{

if(randomset.random!=null)

{

randomset.next.random=randomset.random.next;

}

randomset= randomset.next.next;

}

Node curr=head;

Node newlist=new Node(-1);

Node newcurr=newlist;

Node cpynode=null;

while(curr!=null)

{

Node forw=curr.next.next;

cpynode=curr.next;

newcurr.next=cpynode;

curr.next=forw;

curr=forw;

newcurr=cpynode;

}

return newlist.next;

}

}

88.LRU cache

89.intersection of two linkedlist:headAbecomes tailb.next ,check for loop and find answer or size of both then same level find intersection

/\*\*

\* Definition for singly-linked list.

\* public class ListNode {

\* int val;

\* ListNode next;

\* ListNode(int x) {

\* val = x;

\* next = null;

\* }

\* }

\*/

public class Solution {

public ListNode getIntersectionNode(ListNode headA, ListNode headB) {

ListNode curr=headB;

if(headA==null ||headB==null)return null;

if(headA.val==headB.val)return headA;

int count1=0;

while(curr!=null)

{

curr=curr.next;

count1++;

}

int count2=0;

ListNode curA=headA;

while(curA!=null)

{

count2++;

curA=curA.next;

}

System.out.print(count1+" "+count2);

if(count1>count2)

{

int k=count1-count2;

while(k>0)

{

headB=headB.next;

k--;

}

}

else

{int k=count2-count1;

while(k>0)

{

headA=headA.next;

k--;

}

}

while(headA!=null &&headB!=null)

{

if(headA==headB)

{

return headA;

}

headA=headA.next;

headB=headB.next;

}

return null;

}

}

METHOD 2

/\*\*

\* Definition for singly-linked list.

\* public class ListNode {

\* int val;

\* ListNode next;

\* ListNode(int x) {

\* val = x;

\* next = null;

\* }

\* }

\*/

public class Solution {

public ListNode getIntersectionNode(ListNode headA, ListNode headB) {

ListNode curr=headB;

if(headA==null ||headB==null)return null;

if(headA.val==headB.val)return headA;

while(curr.next!=null)

{curr=curr.next;}

curr.next=headA;

ListNode slow=headB;

ListNode fast=headB;

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

{

slow=slow.next;

fast=fast.next.next;

if(slow==fast)

{

slow=headB;

while(slow!=fast)

{

slow=slow.next;

fast=fast.next;

}

curr.next=null;

return slow;

}

}

curr.next=null;

return null;

}

}

90. PREORDER ,POSTORDER,INORDER iteratively LEETCODE 94:

INORDER:go left and link all extreme right with curr and then while coming either print or disconnect ,print and point curr to right

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode(int x) { val = x; }

\* }

\*/

class Solution {

public List<Integer> inorderTraversal(TreeNode root) {

TreeNode curr=root;

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

while(curr!=null)

{

if(curr.left==null)

{

ll.add(curr.val);

curr=curr.right;

}

else

{

TreeNode currp1=curr.left;

while(currp1.right!=curr&&currp1.right!=null)

{

currp1=currp1.right;

}

if(currp1.right==null)

{

currp1.right=curr;

curr=curr.left;

}

else

{

currp1.right=null;

ll.add(curr.val);

curr=curr.right;

}

}

}

return ll;

}

}

Preoder:

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode() {}

\* TreeNode(int val) { this.val = val; }

\* TreeNode(int val, TreeNode left, TreeNode right) {

\* this.val = val;

\* this.left = left;

\* this.right = right;

\* }

\* }

\*/

class Solution {

public List<Integer> preorderTraversal(TreeNode root) {

List<Integer> ans=new ArrayList();

TreeNode curr=root;

while(curr!=null)

{

if(curr.left==null)

{

ans.add(curr.val);

curr=curr.right;

}

else

{

TreeNode currp1=curr.left;

while(currp1.right!=null &&currp1.right!=curr)

{

currp1=currp1.right;

}

if(currp1.right==null)

{

ans.add(curr.val);

currp1.right=curr;

curr=curr.left;

}

else

{

currp1.right=null;

curr=curr.right;

}

}

}

return ans;

}

}

91.leetcode 124 maxpathsum:remember return is max(node,node+left,node+right) while maxsum=(node,node+right,maxsum,node+left,node+right+left)

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode() {}

\* TreeNode(int val) { this.val = val; }

\* TreeNode(int val, TreeNode left, TreeNode right) {

\* this.val = val;

\* this.left = left;

\* this.right = right;

\* }

\* }

\*/

class Solution {

public int maxsum=(int)-1e8;

public int maxPathSum(TreeNode root) {

int y=maxPath(root);

return maxsum;

}

public int maxPath(TreeNode root)

{

if(root==null)return 0;

int left=maxPath(root.left);

int right=maxPath(root.right); maxsum=Math.max(Math.max(Math.max(left+root.val,Math.max(root.val+right,root.val)),root.val+left+right),maxsum);

return Math.max(left+root.val,Math.max(root.val+right,root.val));

}

}

92.levelorder leetcode 107:maintain a queue withpop and push

/\*\*

\* Definition for a binary tree node.

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode() : val(0), left(nullptr), right(nullptr) {}

\* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

\* TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

\* };

\*/

class Solution {

public:

vector<vector<int>> levelOrderBottom(TreeNode\* root) {

vector<vector<int>> ans;

if(root==NULL)return ans;

queue<TreeNode\*>que;

que.push(root);

while(!que.empty())

{

int size=que.size();

vector<int>a;

while(size>0)

{

size--;

TreeNode \* f=que.front();

que.pop();

a.push\_back(f->val);

if(f->left!=NULL)

{

que.push(f->left);

}

if(f->right!=NULL)

{

que.push(f->right);

}

}

ans.insert(ans.begin(),a);

}

return ans;

}

};

93.all nodes at k distance binary tree leetcode 863:

94.leetcode 1038:binary search tree to greater sum tree:reverse inorder and then value update

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode() {}

\* TreeNode(int val) { this.val = val; }

\* TreeNode(int val, TreeNode left, TreeNode right) {

\* this.val = val;

\* this.left = left;

\* this.right = right;

\* }

\* }

\*/

class Solution {

public int maxsum=0;

public TreeNode bstToGst(TreeNode root) {

if(root==null)return root;

TreeNode r=bstToGst(root.right);

maxsum+=root.val;

root.val=maxsum;

TreeNode l=bstToGst(root.left);

return root;

}

}

95.leetcode 199 binary tree right view: levelorder last value print

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode() {}

\* TreeNode(int val) { this.val = val; }

\* TreeNode(int val, TreeNode left, TreeNode right) {

\* this.val = val;

\* this.left = left;

\* this.right = right;

\* }

\* }

\*/

class Solution {

public List<Integer> rightSideView(TreeNode root) {

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

if(root==null)return ans;

Queue<TreeNode>que=new LinkedList();

que.add(root);

while(que.size()>0)

{

int s=que.size();

while(s>0)

{

s--;

TreeNode f=que.peek();

if(s==0)ans.add(f.val);

que.poll();

if(f.left!=null)

{

que.add(f.left);

}

if(f.right!=null)

{

que.add(f.right);

}

}

}

return ans;

}

}

96.left view of the tree gfg:leveloreder first element bfs

void leftView(Node root)

{

// Your code here

if(root==null)return;

Queue<Node>que=new LinkedList();

que.add(root);

while(que.size()>0)

{

int s=que.size();

int i=s;

while(s>0)

{

Node f=que.peek();

if(s==i)System.out.print(f.data+" ");

s--;

que.poll();

if(f.left!=null)

{

que.add(f.left);

}

if(f.right!=null)

{

que.add(f.right);

}

}

}

}

97.print nodes in topview :take hashmap to store first value of occurance and queue has pair <node,int> node and level;

#include<bits/stdc++.h>

using namespace std;

class Node {

    public:

        int data;

        Node \*left;

        Node \*right;

        Node(int d) {

            data = d;

            left = NULL;

            right = NULL;

        }

};

class Solution {

    public:

        Node\* insert(Node\* root, int data) {

            if(root == NULL) {

                return new Node(data);

            } else {

                Node\* cur;

                if(data <= root->data) {

                    cur = insert(root->left, data);

                    root->left = cur;

                } else {

                    cur = insert(root->right, data);

                    root->right = cur;

               }

               return root;

           }

        }

/\*

class Node {

    public:

        int data;

        Node \*left;

        Node \*right;

        Node(int d) {

            data = d;

            left = NULL;

            right = NULL;

        }

};

\*/

    void topView(Node \* root) {

    queue<pair<Node\*,int>>que;

    pair<Node\*,int> s(root,0);

    que.push(s);

    int max\_=INT\_MIN;

    int min\_=INT\_MAX;

    unordered\_map<int,int>map;

    while(!que.empty())

    {

        int s=que.size();

        while(s>0)

        {

            pair <Node\*,int>  p=que.front();

            que.pop();

            if(map.count(p.second)==0)

            {map[p.second]=p.first->data;

            }

            min\_=min(min\_,p.second);

            max\_=max(max\_,p.second);

            s--;

            if(p.first->left!=NULL)

            {

                pair <Node\*,int>  y(p.first->left,p.second-1);

                que.push(y);

            }

            if(p.first->right!=NULL)

            {

                pair <Node\*,int>  y(p.first->right,p.second+1);

                que.push(y);

            }

        }

    }

    while(min\_<=max\_)

    {

        cout<<map[min\_]<<" ";

        min\_++;

    }

    }

}; //End of Solution

int main() {

    Solution myTree;

    Node\* root = NULL;

    int t;

    int data;

    std::cin >> t;

    while(t-- > 0) {

        std::cin >> data;

        root = myTree.insert(root, data);

    }

    myTree.topView(root);

    return 0;

}

98.bottom view of tree gfg:hashmap and queue of pair such that queue of pair and hashmap last tak update

void bottomView(Node \*root)

{

if(root==NULL)return;

queue<pair<Node\*,int>> que;

pair<Node\*,int>p(root,0);

que.push(p);

int min\_=0;

int max\_=0;

unordered\_map<int,int>map;

while(que.size()>0)

{

int s=que.size();

while(s>0)

{

s--;

pair<Node\*,int> p=que.front();

map[p.second]=p.first->data;

que.pop();

min\_=min(min\_,p.second);

max\_=max(max\_,p.second);

if(p.first->left!=NULL)

{

pair<Node\*,int>y(p.first->left,p.second-1);

que.push(y);

}

if(p.first->right!=NULL)

{

pair<Node\*,int>y(p.first->right,p.second+1);

que.push(y);

}

}

}

while(min\_<=max\_){

cout<<map[min\_]<<" ";min\_++;}

// Your Code Here

}

99.vertical order leetcode 987: horizontal of vertical hould be sorted to create map of vectical hor and ver and que of hori and vert and node

/\*\*

\* Definition for a binary tree node.

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode() : val(0), left(nullptr), right(nullptr) {}

\* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

\* TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

\* };

\*/

class Solution {

public:

vector<vector<int>> verticalTraversal(TreeNode\* root) {

queue<pair<TreeNode\*,pair<int,int>>>que;

map<int,map<int,vector<int>>>map;

if(root==NULL)return {{}};

pair<TreeNode\*,pair<int,int>> s(root,{0,0});

que.push(s);

int max\_=INT\_MIN;

int min\_=INT\_MAX;

while(!que.empty())

{

int s=que.size();

while(s>0)

{

s--;

pair<TreeNode\*,pair<int,int>> p=que.front();

que.pop();

map[p.second.first][p.second.second].push\_back(p.first->val);

int max\_=max(max\_,p.second.first);

int min\_=min(min\_,p.second.first);

if(p.first->left!=NULL)

{

pair<TreeNode\*,pair<int,int>> y(p.first->left,{p.second.first-1,p.second.second+1});

que.push(y);

}

if(p.first->right!=NULL)

{

pair<TreeNode\*,pair<int,int>> y(p.first->right,{p.second.first+1,p.second.second+1});

que.push(y);

}

}

}

vector<vector<int>>ans;

for(auto m:map)

{

vector<int>res1;

for(auto it:m.second)

{

sort(it.second.begin(),it.second.end());

for (auto a: it.second) {

res1.push\_back(a);

}

}

ans.push\_back(res1);

}

return ans;

}

};

100.diagonal transversal

101.boundary transversal gfg:left boundary that if root.left!=null call left else if(root.right!=null)call right and leaf in inorder and rightbourday reverse or call right and then left

void rightboundary(Node\* root)

{

if(root==NULL)return;

if(root->right)

{

rightboundary(root->right);

cout<<root->data<<" ";

}

else if(root->left)

{

rightboundary(root->left);

cout<<root->data<<" ";

}

}

void leftboundary(Node\* root)

{

if(root==NULL)return;

if(root->left)

{

cout<<root->data<<" ";

leftboundary(root->left);

}

else if(root->right)

{

cout<<root->data<<" ";

leftboundary(root->right);

}

}

void leaf(Node\* root)

{

if(root==NULL)return ;

leaf(root->left);

if(root->left==NULL&&root->right==NULL){cout<<root->data<<" ";}

leaf(root->right);

}

void printBoundary(Node \*root)

{

if(root==NULL)return;

//Your code here

cout<<root->data<<" ";

leftboundary(root->left);

leaf(root->left);

leaf(root->right);

rightboundary(root->right);

}

102.leetcode 968:binary camera:three cases 2 camera,1,covered,0-leaf or not covered

/\*\*

\* Definition for a binary tree node.

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode() : val(0), left(nullptr), right(nullptr) {}

\* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

\* TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

\* };

\*/

class Solution {

public:

int answer=0;

//1 covered,2=camera,0need camera

int camera(TreeNode\* root)

{

if(root==NULL)return 1;

int l=camera(root->left);

int r=camera(root->right);

if(l==0||r==0)

{

answer++;

return 2;

}

if(l==2||r==2)return 1;

return 0;

}

int minCameraCover(TreeNode\* root) {

if(root==NULL) return 0;

if(root->left==NULL &&root->right==NULL)return 1;

int a=camera(root);

return a==0?answer+1:answer;

}

};

103.lintcode 448 inorder succ in bst:3 option either in right or in parent to root path

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode(int x) { val = x; }

\* }

\*/

public class Solution {

/\*

\* @param root: The root of the BST.

\* @param p: You need find the successor node of p.

\* @return: Successor of p.

\*/

public TreeNode inorderSuccessor(TreeNode root, TreeNode p) {

// write your code here

if(root==null)return root;

//TreeNode s=p;

TreeNode ans=null;

if(p.right!=null)

{

p=p.right;

ans=p;

while(p.left!=null)

{

p=p.left;

ans=p;

}

return ans;

}

while(root!=null)

{

if(p.val<root.val)

{

ans=root;

root=root.left;

}

else if(p.val>root.val)

{

root=root.right;

}

else break;

}

return ans;

}

}

104.lca in bst leetcode 235:cases go right and left

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode(int x) { val = x; }

\* }

\*/

class Solution {

public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {

if(root==null)return root;

if(root.val>p.val&&root.val>q.val)

{

return lowestCommonAncestor(root.left,p,q);

}

else if(root.val<p.val&&root.val<q.val)

{

return lowestCommonAncestor(root.right,p,q);

}

return root;

}

}

105.binary tree lca leetcode 236:simple Boolean

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode(int x) { val = x; }

\* }

\*/

class Solution {

public TreeNode lcanode=null;

public TreeNode lowestCommonAncestor(TreeNode root, TreeNode p, TreeNode q) {

boolean res=lca(root,p,q);

return lcanode;

}

public boolean lca(TreeNode root,TreeNode p,TreeNode q)

{

if(root==null)return false;

boolean l=lca(root.left,p,q);

boolean r=lca(root.right,p,q);

boolean self=root==p||root==q;

if(self&&l||self&&r||l&&r)

{

lcanode=root;

return true;

}

return self||l||r;

}

}

106.leetcode 979 distribute coins:info and return value check

/\*\*

\* Definition for a binary tree node.

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode() : val(0), left(nullptr), right(nullptr) {}

\* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

\* TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

\* };

\*/

class Solution {

public:

int ans=0;

int s(TreeNode\* root)

{

if(root==NULL)return 0;

int l=s(root->left);

int r=s(root->right);

ans+=abs(l)+abs(r);

return l+r+root->val-1;

}

int distributeCoins(TreeNode\* root) {

int y=s(root);

return ans;

}

};

107.leetcode 450 deletion in bst:think about cases and think that tree is being constructed

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode() {}

\* TreeNode(int val) { this.val = val; }

\* TreeNode(int val, TreeNode left, TreeNode right) {

\* this.val = val;

\* this.left = left;

\* this.right = right;

\* }

\* }

\*/

class Solution {

public TreeNode deleteNode(TreeNode root, int key) {

if(root==null)return root;

if(key<root.val)

{

root.left=deleteNode(root.left,key);

}

else if(key>root.val)

{

root.right=deleteNode(root.right,key);

}

else

{

if(root.right!=null&&root.left==null)return root.right;

if(root.left!=null&&root.right==null)return root.left;

if(root.right==null&&root.left==null)return null;

if(root.right!=null&&root.left!=null)

{

TreeNode rootp1=root.right;

while(rootp1.left!=null)

{

rootp1=rootp1.left;

}

rootp1.left=root.left;

return root.right;

}

}

return root;

}

}

108.leetcode 105 construct tree from pre and inorder:creation of root is main to see

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode() {}

\* TreeNode(int val) { this.val = val; }

\* TreeNode(int val, TreeNode left, TreeNode right) {

\* this.val = val;

\* this.left = left;

\* this.right = right;

\* }

\* }

\*/

class Solution {

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

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

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

{

map.put(inorder[i],i);

}

return prepost(preorder,inorder,0,inorder.length-1,0,preorder.length-1);

}

public TreeNode prepost(int []pre,int[]in,int isi,int iei,int psi,int pei)

{

if(psi>pei||isi>iei)return null;

TreeNode n=new TreeNode(pre[psi]);

int idx=map.get(pre[psi]);

int count=idx-isi;

n.left=prepost(pre,in,isi,idx-1,psi+1,psi+count);

n.right=prepost(pre,in,idx+1,iei,psi+count+1,pei);

return n;

}

}

Leetcode 106:binary tree from post and in order

/\*\*

\* Definition for a binary tree node.

\* public class TreeNode {

\* int val;

\* TreeNode left;

\* TreeNode right;

\* TreeNode() {}

\* TreeNode(int val) { this.val = val; }

\* TreeNode(int val, TreeNode left, TreeNode right) {

\* this.val = val;

\* this.left = left;

\* this.right = right;

\* }

\* }

\*/

class Solution {

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

public TreeNode buildTree(int[] inorder, int[] postorder) {

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

map.put(inorder[i],i);

return inpost(inorder,postorder,0,postorder.length-1,0,inorder.length-1);

}

public TreeNode inpost(int[]in,int[]post,int psi,int pei,int isi,int iei)

{

if(isi>iei||psi>pei)return null;

TreeNode n=new TreeNode(post[pei]);

int idx=map.get(post[pei]);

int count=idx-isi;

n.left=inpost(in,post,psi,psi+count-1,isi,idx-1);

n.right=inpost(in,post,psi+count,pei-1,idx+1,iei);

return n;

}

}

109.

111.clone a binary tree:do all in post order, first create clone node to left ,then set random ,then 4 cases set myclone with r and l and node

void mapping(Node\* root){

if(!root){

return;

}

mapping(root -> left);

mapping(root -> right);

Node\* newroot = new Node(root -> data);

Node\* rootp1 = root -> left;

root -> left = newroot;

newroot -> left = rootp1;

}

void print(Node \* node)

{

if(node==NULL)return;

cout<<node->data;

print(node->left);

print(node->right);

}

void setrandom(Node\* node)

{

if(node==NULL)return;

setrandom(node->left->left);

setrandom(node->right);

if(node->random!=NULL)

{

//cout<<node->random->data<<" ";

node->left->random=node->random->left;

}

else

{

// cout<<node->random->data<<" ";

node->left->random=NULL;

}

}

Node \* extract(Node \*root)

{

if(root ==NULL)return NULL;

Node \* l=extract(root->left);

Node \* r=extract(root->right);

if(l==NULL && r==NULL)

{

Node\* myclone=root->left;

root->left=NULL;

return myclone;

}

else if(l==NULL && r!=NULL)

{

Node\* myclone=root->left;

root->left=NULL;

myclone->right=r;

return myclone;

}

else if(l!=NULL && r==NULL)

{

Node \* myclone=root->left;

root->left=myclone->left;

myclone->left=l;

return myclone;

}

else

{

Node \* myclone=root->left;

root->left=myclone->left;

myclone->left=l;

myclone->right=r;

return myclone;

}

}

Node\* cloneTree(Node\* node)

{

//Your code here

mapping(node);

setrandom(node);

//print(node);

return extract(node);

}

113.kthsmallest in bst:do morris travel with k counter

int KthSmallestElement(Node \*root, int k)

{

//add code here.

int ans=-1;

if(root==NULL)return -1;

while(root!=NULL)

{

if(root->left==NULL)

{

k--;

if(k==0)ans=root->data;

root=root->right;

}

else

{

Node\* rootp1=root->left;

while(rootp1->right!=NULL &&rootp1->right!=root)

{

rootp1=rootp1->right;

}

if(rootp1->right==NULL)

{

rootp1->right=root;

root=root->left;

}

else

{

k--;

if(k==0)ans=root->data;

rootp1->right=NULL;

root=root->right;

}

}

}

return ans;

}

114.leetcode 114 flatten :use stack for preorder and pop the element add to right and left should be null

/\*\*

\* Definition for a binary tree node.

\* struct TreeNode {

\* int val;

\* TreeNode \*left;

\* TreeNode \*right;

\* TreeNode() : val(0), left(nullptr), right(nullptr) {}

\* TreeNode(int x) : val(x), left(nullptr), right(nullptr) {}

\* TreeNode(int x, TreeNode \*left, TreeNode \*right) : val(x), left(left), right(right) {}

\* };

\*/

class Solution {

public:

void flatten(TreeNode\* root) {

if(root==NULL)return;

stack<TreeNode\*>st;

st.push(root);

while(!st.empty())

{

TreeNode\* node=st.top();st.pop();

if(node->right!=NULL)

{

st.push(node->right);

}

if(node->left!=NULL)

{

st.push(node->left);

}

if(!st.empty())

{

node->right=st.top();

}

node->left=NULL;

}

}

};

METHOD 2:POSTORDER and store left and right and then left==null and right should have left+right;

class Solution {

public:

void flatten(TreeNode\* root) {

if(root==NULL)return;

flatten(root->left);

flatten(root->right);

TreeNode\* l=root->left;

TreeNode\* r=root->right;

root->left=NULL;

root->right=l;

TreeNode\* temp=root;

while(temp->right!=NULL)

{

temp=temp->right;

}

temp->right=r;

}

118.bfs of graph:queue transverse and mark true in vis array

vector <int> bfs(vector<int> g[], int N) {

queue<int>que;

que.push(0);

vector<int>ans;

vector<bool>vis(N,false);

vis[0]=true;

while(!que.empty())

{

int size=que.size();

while(size>0)

{

size--;

int j=que.front();

que.pop();

ans.push\_back(j);

for(int i:g[j])

{

if(vis[i]!=true)

{

que.push(i);

vis[i]=true;

}

}

}

}

// Your code here

return ans;

}

119.leetcode 785 bipartite :to check odd length cycle queue and vis arr intialised with -1 and 10 and 12 one color

class Solution {

public:

bool bfsbi(vector<vector<int>>graph,vector<int>vis,int src)

{

queue<int>que;

que.push(src);

vis[src]=10;

while(!que.empty())

{

int size=que.size();

while(size>0)

{

size--;

int f=que.front();que.pop();

for(int i:graph[f])

{

if(vis[i]==-1)

{

que.push(i);

if(vis[f]==10)vis[i]=12;

else if (vis[f]==12)vis[i]=10;

}

else

{

if((vis[i]==10&& vis[f]==10)||(vis[i]==12&&vis[f]==12))

{ return false;}

}

}

}

}

return true;

}

bool isBipartite(vector<vector<int>>& graph) {

vector<int>vis(graph.size(),-1);

bool res=true;

for(int i=0;i<graph.size();i++)

{if(vis[i]==-1){

res=res&&bfsbi(graph,vis,i);}

}

return res;

}

};

120.leetcode 815 bus routes:here create hashmap of int,vector<int> busstop versus bus route and transverse such that bustop array buses array and graph hashmap

class Solution {

public:

int bfs(vector<vector<int>>routes,int src,int desti,unordered\_map<int,vector<int>>graph,int n)

{

vector<bool>buses(routes.size(),false);

vector<bool>busstop(n,false);

queue<int>que;

que.push(src);

int level=0;

while(!que.empty())

{

int size=que.size();

while(size>0)

{

size--;

int f=que.front();

que.pop();

if(f==desti)return level;

for(int i:graph[f])

{

if(buses[i]==false)

{

buses[i]=true;

for(int j:routes[i])

{

if(busstop[j]==false)

{

que.push(j);

busstop[j]=true;

}

}

}

}

}

level++;

}

return -1;

}

int numBusesToDestination(vector<vector<int>>& routes, int S, int T) {

unordered\_map<int,vector<int>>graph;

int n=0;

for(int i=0;i<routes.size();i++)

{ n+=routes[i].size();

for(int j:routes[i])

{

graph[j].push\_back(i);

}

}

return bfs(routes,S,T,graph,n);

}

};

121.dfs

void dfs(int s, vector<int> g[], bool vis[])

{ cout<<s<<" ";

vis[s]=true;

for(int i:g[s])

{

if(!vis[i])

{

dfs(i,g,vis);

}

}

// Your code here

}

122.prims algo

123.dijikstra algo

124.chef and reversing:dijikstra pnly but modified by a linkedlist as time complexity and when 0 is added at front and 1 is at last and pop front

**#include <iostream>**

**#include<unordered\_map>**

**#include<utility>**

**#include<list>**

**#include<vector>**

**using namespace std;**

**int dijikstra(unordered\_map<int,vector<pair<int,int>>>&graph,int v)**

**{**

**//priority\_queue<pair<int,int>,vector<pair<int,int>>, greater<pair<int,int>> >pq;**

**list<pair<int,int>>ll;**

**pair<int,int>n(0,0);**

**ll.push\_back(n);**

**//int sum=0;**

**vector<int>vis(v,-1);**

**while(!ll.empty())**

**{**

**int size=ll.size();**

**while(size>0)**

**{**

**size--;**

**pair <int,int> f=ll.front();**

**ll.pop\_front();**

**if(vis[f.second]==-1)**

**{vis[f.second]=f.first;}**

**for(pair<int,int>s:graph[f.second])**

**{**

**if(vis[s.first]==-1)**

**{**

**pair<int,int>n(s.second+f.first,s.first);**

**// cout<<n.second<<"->"<<n.first<<";";**

**if(s.second==0)ll.push\_front(n);**

**else**

**{**

**ll.push\_back(n);**

**}**

**}**

**}**

**//cout<<endl;**

**}**

**}**

**// int a=pq.top().second;**

**return vis[v-1];**

**}**

**int main() {**

**int v,e;**

**cin>>v>>e;**

**unordered\_map<int,vector<pair<int,int>>>graph;**

**// your code goes here**

**for(int i=0;i<e;i++)**

**{**

**int a,b;**

**cin>>a>>b;**

**pair<int,int>p(b-1,0);**

**pair<int,int>r(a-1,1);**

**graph[b-1].push\_back(r);**

**graph[a-1].push\_back(p);**

**}**

**/\* for(int i=0;i<v;i++)**

**{**

**cout<<i<<"->";**

**{**

**for(pair<int,int>s:graph[i])**

**{**

**cout<<s.first<<"-"<<s.second<<";";**

**}**

**cout<<endl;**

**}**

**}\*/**

**cout<<dijikstra(graph,v);**

**//cout<<sum;**

**return 0;**

**}**

125.min cost to connect all cities:use prims for prims vis and pq queue min and the calculate ans

int Solution::solve(int A, vector<vector<int> > &B) {

unordered\_map<int,vector<pair<int,int>>>graph;

for(int i=0;i<B.size();i++)

{

pair<int,int>n(B[i][1],B[i][2]);

pair<int,int>r(B[i][0],B[i][2]);

graph[B[i][0]].push\_back(n);

graph[B[i][1]].push\_back(r);

}

priority\_queue<pair<int,int>,vector<pair<int,int>>,greater<pair<int,int>>>pq;

vector<bool>vis(A+1,false);

int ans=0;

pair<int,int>s(0,1);//wt,vtx

pq.push(s);

while(!pq.empty())

{

int size=pq.size();

while(size>0)

{

size--;

pair<int,int>s=pq.top();pq.pop();

if(vis[s.second]==false)

{

vis[s.second]=true;

ans+=s.first;

}

for(pair<int,int>i:graph[s.second])

{

if(vis[i.first]==false)

{

pair<int,int>l(i.second,i.first);

pq.push(l);

}

}

}

}

return ans;

}

126.optimize water distribution in a village

127.leetcode 399 evaluate division:trick is to make graph and then do bfs just like dijikstra

class Solution {

public:

double bfs(string src,string dest,unordered\_map<string,vector<pair<string,double>>>&graph,unordered\_map<string,int>&inlist)

{

queue<pair<string,double>>que;

pair<string,double>n(src,1.0);

que.push(n);

double ans=1;

unordered\_map<string,int>map;

while(!que.empty())

{

int size=que.size();

while(size>0)

{

size--;

pair<string,double>s=que.front();que.pop();

map[s.first]=1;

if(s.first==dest&&inlist[s.first]==1)

{

return s.second;

}

for(pair<string,double>f:graph[s.first])

{

if(map[f.first]!=1)

{pair<string,double>n(f.first,f.second\*s.second);

que.push(n);

}

}

}

}

cout<<"notfound";

return -1.0;

}

vector<double> calcEquation(vector<vector<string>>& equations, vector<double>& values, vector<vector<string>>& queries) {

vector<double>ans;

unordered\_map<string,int>inlist;

unordered\_map<string,vector<pair<string,double>>>graph;

for(int i=0;i<equations.size();i++)

{

cout<<equations[i][1]<<values[i];

pair<string,double>n;

n.first=equations[i][1];

n.second=values[i];

inlist[equations[i][0]]=1;

inlist[equations[i][1]]=1;

pair<string,double>r(equations[i][0],1/values[i]);

graph[equations[i][0]].push\_back(n);

graph[equations[i][1]].push\_back(r);

}

for(int i=0;i<queries.size();i++)

{

ans.push\_back(bfs(queries[i][0],queries[i][1],graph,inlist));

}

return ans;

}

};

128.leetcode 210 course schedule II:so basically here we have to think like graph form and relation such that parent comes first ,can check cycle also

class Solution {

public:

vector<int> findOrder(int n, vector<vector<int>>& prerequisites) {

unordered\_map<int,vector<int>>graph;

for(int i=0;i<prerequisites.size();i++)

{

graph[prerequisites[i][1]].push\_back(prerequisites[i][0]);

}

vector<int>ans;

vector<int>indegree(n,0);

for(int i=0;i<prerequisites.size();i++)

{

indegree[prerequisites[i][0]]++;

}

queue<int>que;

for(int i=0;i<indegree.size();i++)

{ if(indegree[i]==0)

que.push(i);

}

while(!que.empty())

{

int f=que.front();que.pop();

ans.push\_back(f);

for(int i:graph[f])

{

indegree[i]--;

if(indegree[i]==-1)return {};

if(indegree[i]==0)que.push(i);

}

}

if(ans.size()!=n)ans={};

return ans;

}

};

129.kosa raju:first find the topo in stack then reverse edge in new graph and pop and find ans ..

void dfs(stack<int>&ans,vector<bool>&vis,vector<int>adj[],int src)

{

vis[src]=true;

for(int i:adj[src])

{

if(vis[i]==false) dfs(ans,vis,adj,i);

}

// cout<<src;

ans.push(src);

}

void dfsnew(int src,vector<bool>&vis,vector<int>adj[])

{

vis[src]=true;

for(int i:adj[src])

{

if(vis[i]==false) dfsnew(i,vis,adj);

}

}

int kosaraju(int V, vector<int> adj[])

{

// Your code here

stack<int>ans;

vector<bool>vis(V,false);

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

{if(vis[i]==false)

dfs(ans,vis,adj,i);

}

vector<int>graph[V];

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

{

for(int j:adj[i])

{

graph[j].push\_back(i);

}

}

int finalans=0;

vector<bool>stackvis(V,false);

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

{

if(stackvis[ans.top()]==false)

{dfsnew(ans.top,stackvis,graph);

// ans.pop();

finalans++;

}

ans.pop();

}

return finalans;

}

130.mother vertex gfg:

void newdfs(int& ans,int src,vector<int>graph[],vector<bool>&vis)

{

vis[src]=true;

for(int i:graph[src])

{

if(vis[i]==false)newdfs(ans,i,graph,vis);

}

ans=src;

}

void dfs(vector<bool>&finvis,int src,vector<int>g[])

{

finvis[src]=true;

for(int i:g[src])

{

if(finvis[i]==false)

{

dfs(finvis,i,g);

}

}

//ans=src;

}

int findMother(int V, vector<int> g[])

{ vector<bool>vis(V,false);

int mother=-1;

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

{

if(vis[i]==false)newdfs(mother,i,g,vis);

// cout<<mother;

}

//cout<<mother;

vector<bool>finvis(V,false);

dfs(finvis,mother,g);

bool res=true;

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

{

res=res&&finvis[i];

}

// Your code here

return res==true?mother:-1;

}

131.leetcode 994 rotten oranges

class Solution {

public:

int bfs(queue<int>que,vector<vector<int>>& grid)

{

int level=-1;

vector<vector<int>>d={{1,0},{-1,0},{0,1},{0,-1}};

while(!que.empty())

{

int size=que.size();

while(size>0)

{

int pos=que.front();

que.pop();

size--;

for(int i=0;i<4;i++)

{

int x=pos/grid[0].size()+d[i][0];

int y=pos%grid[0].size()+d[i][1];

if(x>=0&&y>=0&&x<grid.size()&&y<grid[0].size()&&grid[x][y]==1)

{

grid[x][y]=0;

que.push(x\*grid[0].size()+y);

}

}

}

level++;

}

return level;

}

int orangesRotting(vector<vector<int>>& grid) {

int m=0;

queue<int>que;

for(int i=0;i<grid.size();i++)

{

for(int j=0;j<grid[0].size();j++)

{

if(grid[i][j]==2)

{

que.push(i\*grid[0].size()+j);

}

}

}

if(!que.empty()) m=bfs(que,grid);

for(int i=0;i<grid.size();i++)

{

for(int j=0;j<grid[0].size();j++)

{ if(grid[i][j]==1)return -1;

}}

return m;

}

};

132.bellmans ford: to learn that src vtx can have v-1 egdes max to any src to get right ans,first we store vertices with wt and intalise src=0 in vis vector and check where u+w<v and then updata;to check negative cycle we iterate once again and see whether any changes which is only possible if we have had negative cycle.negative cycle wt:no shortest path ,negative weight shortest path exists

#include <iostream>

#include<vector>

#include<bits/stdc++.h>

using namespace std;

int main() {

//code

int tc;

cin>>tc;

for(int t=0;t<tc;t++)

{

int v,e;

cin>>v>>e;

vector<vector<int>>uvw(e,vector<int>(3,0));

for(int i=0;i<uvw.size();i++)

{

cin>>uvw[i][0];

cin>>uvw[i][1];

cin>>uvw[i][2];

}

/\* for(int i=0;i<uvw.size();i++)

{

cout<<uvw[i][0];

cout<<uvw[i][1];

cout<<uvw[i][2]<<endl;

}\*/

vector<int>vtx(v,INT\_MAX);

vtx[0]=0;

for(int i=1;i<v;i++)

{

for(int j=0;j<uvw.size();j++)

{

if(vtx[uvw[j][0]]==INT\_MAX)continue;

else if(vtx[uvw[j][0]]+uvw[j][2]<vtx[uvw[j][1]])

{

vtx[uvw[j][1]]=vtx[uvw[j][0]]+uvw[j][2];

}

}

}

//for(int i=0;i<vtx.size();i++)cout<<vtx[i]<<" ";

int res=0;

for(int i=0;i<uvw.size();i++)

{

if(vtx[uvw[i][0]]+uvw[i][2]<vtx[uvw[i][1]])

{

res=1;

}

}

cout<<res<<endl;

}

return 0;

}

133.leetcode 200 numsinland:find 1 and call dfs to find no. of island

class Solution {

public:

void dfs(int r,int c,vector<vector<char>>&grid)

{

if(grid[r][c]=='0') return;

grid[r][c]='0';

if(r+1<grid.size())dfs(r+1,c,grid);

if(c+1<grid[0].size())dfs(r,c+1,grid);

if(r-1>=0)dfs(r-1,c,grid);

if(c-1>=0)dfs(r,c-1,grid);

}

int numIslands(vector<vector<char>>& grid) {

int count=0;

for(int i=0;i<grid.size();i++)

{

for(int j=0;j<grid[0].size();j++)

{

if(grid[i][j]=='1'){dfs(i,j,grid);

count++;}

}

}

return count;

}

};

134.leetcode 1020 no. of enclaves:go through edge and see for 1 and mark 0 and dfs

class Solution {

public:

void dfs(int r,int c,vector<vector<int>>& A)

{

if(A[r][c]==0)return ;

A[r][c]=0;

if(r+1<A.size())dfs(r+1,c,A);

if(r-1>=0)dfs(r-1,c,A);

if(c-1>=0)dfs(r,c-1,A);

if(c+1<A[0].size())dfs(r,c+1,A);

}

int numEnclaves(vector<vector<int>>& A) {

for(int i=0;i<A.size();i++)

{

if(A[i][0]==1)dfs(i,0,A);

if(A[i][A[0].size()-1]==1)dfs(i,A[0].size()-1,A);

}

for(int i=0;i<A[0].size();i++)

{

if(A[0][i]==1)dfs(0,i,A);

if(A[A.size()-1][i]==1)dfs(A.size()-1,i,A);

}

int count=0;

for(int i=0;i<A.size();i++)

{

for(int j=0;j<A[0].size();j++)

{

if(A[i][j]==1)count++;

}

}

return count;

}

};

135.leetcode 542 01 matrix:push all zeroes and ans add level

class Solution {

public:

void bfs(queue<int>que,vector<vector<int>>& grid,vector<vector<int>>&ans)

{

int level=0;

vector<vector<int>>d={{1,0},{-1,0},{0,1},{0,-1}};

while(!que.empty())

{

int size=que.size();

while(size>0)

{

int pos=que.front();

que.pop();

size--;

int r=pos/grid[0].size();

int c=pos%grid[0].size();

ans[r][c]=level;

for(int i=0;i<4;i++)

{

int x=pos/grid[0].size()+d[i][0];

int y=pos%grid[0].size()+d[i][1];

if(x>=0&&y>=0&&x<grid.size()&&y<grid[0].size()&&grid[x][y]==1)

{

grid[x][y]=0;

que.push(x\*grid[0].size()+y);

}

}

}

level++;

}

}

vector<vector<int>> updateMatrix(vector<vector<int>>& matrix) {

if(matrix.size()==0||matrix[0].size()==0)return matrix;

queue<int>que;

for(int i=0;i<matrix.size();i++)

{

for(int j=0;j<matrix[0].size();j++)

{

if(matrix[i][j]==0)

{

que.push(i\*matrix[0].size()+j);

}

}

}

vector<vector<int>>ans(matrix.size(),vector<int>(matrix[0].size(),0));

bfs(que,matrix,ans);

return ans;

}

};

136.lintcode 434 number of island 2:use dsu and assign parent and if same parent do nothing if different parent then ans— as they will combine

/\*\*

\* Definition for a point.

\* struct Point {

\* int x;

\* int y;

\* Point() : x(0), y(0) {}

\* Point(int a, int b) : x(a), y(b) {}

\* };

\*/

class Solution {

public:

/\*\*

\* @param n: An integer

\* @param m: An integer

\* @param operators: an array of point

\* @return: an integer array

\*/

int find(vector<int>&par,int x)

{

if(x==par[x])return x;

int px=find(par,par[x]);

par[x]=px;

return px;

}

void unin(vector<int>&rank,vector<int>&par,int x,int y,int &ans)

{

int px=find(par,x);

int py=find(par,y);

if(px!=py)

{

if(rank[px]==rank[py])

{

par[px]=py;

rank[py]++;

}

else if(rank[px]>rank[py])

{

par[py]=px;

}

else

{

par[px]=py;

}

ans--;

}

}

vector<int> numIslands2(int n, int m, vector<Point> &operators) {

// write your code here

vector<int>par(n\*m,0);

vector<vector<int>>dir={{0,1},{1,0},{-1,0},{0,-1}};

vector<vector<int>>grid(n,vector<int>(m,0));

for(int i=0;i<n\*m;i++)par[i]=i;

vector<int>rank(n\*m,1);

vector<int>a;

int ans=0;

// cout<<operators[1].x;

for(int i=0;i<operators.size();i++)

{

int x=operators[i].x;

int y=operators[i].y;

if(grid[x][y]!=1)ans++;

grid[x][y]=1;

for(int j=0;j<dir.size();j++)

{

int r=x+dir[j][0];

int c=y+dir[j][1];

if(r<n &&c<m&&r>=0&&c>=0&&grid[r][c]==1)

{

unin(rank,par,r\*m+c,x\*m+y,ans);

// cout<<r<<c<<" "<<ans<<":";

}

}

a.push\_back(ans);

}

return a;

}

};

139.leetcode 947 :most stones removed:store par and rank in hashmap.ans++ if not in hashmap before as in set creation and ans— if diff parent;

Return stones-parent size

class Solution {

public:

int ans = 0;

unordered\_map<int,int> parent;

unordered\_map<int,int> rank;

int find(int x){

if(parent[x] == x ){

return x;

}

int px=find(parent[x]);

parent[x]=px;

return px;

}

void merge(int x, int y){

int lx = find(x);

int ly = find(y);

if(lx!=ly){

if(rank[lx]>rank[ly]){

parent[ly]=lx;

} else if(rank[lx]<rank[ly]){

parent[lx]=ly;

} else {

parent[lx]=ly;

rank[ly]++;

}

ans--;

}

}

int removeStones(vector<vector<int>>& stones) {

for(int i=0;i<stones.size();i++){

int row = stones[i][0];

int col = ((-1)\*stones[i][1]) -1; //to distinguish row 0 and col 0

if(parent.find(row)==parent.end()) {

parent[row]=row;

rank[row]=1;

ans++;

}

if(parent.find(col)==parent.end()){

parent[col]=col;

rank[col]=1;

ans++;

}

merge(row,col);

}

//ans is group leader

return stones.size() - ans;

}

};

140.leetcode 990-satisfiablity of equation:see equations are connect and we have to find same group or parent so use dsu at equality and check parent for inequality

class Solution {

public:

int find(int x,vector<int>&par)

{

if(par[x]==x)return x;

int px=find(par[x],par);

par[x]=px;

return px;

}

void unin(int x,int y,vector<int>&par,vector<int>&rank)

{

int px=find(x,par);

int py=find(y,par);

if(px!=py)

{

if(rank[px]<rank[py])

par[py]=px;

else if(rank[px]>rank[py])par[px]=py;

else

{ par[px]=py;

rank[px]++;

}

}

}

bool equationsPossible(vector<string>& equations) {

vector<int>par(26,0);

for(int i=0;i<par.size();i++)par[i]=i;

vector<int>rank(26,1);

for(int i=0;i<equations.size();i++)

{

if(equations[i][1]=='=')

{

unin(equations[i][0]-'a',equations[i][3]-'a',par,rank);

}

}

for(int i=0;i<equations.size();i++)

{

if(equations[i][1]=='!')

{

if(find(equations[i][0]-'a',par)==find(equations[i][3]-'a',par))return false;

}

}

return true;

}

};

141.kruskal mst:for dynamic graph mostly:sort the graph and merge points if diff par add in ans else don’t add

143.leetcode 127 word ladder:you have to bfs only but see the collection of make word.length\*26 words and check in list then output popping time check

class Solution {

public:

int ladderLength(string beginWord, string endWord, vector<string>& wordList) {

unordered\_map<string,int>map;

for(int i=0;i<wordList.size();i++)

map[wordList[i]]=1;

queue<string>que;

int level=0;

que.push(beginWord);

while(!que.empty())

{

int size=que.size();

level++;

while(size>0)

{

string f=que.front();

if(f==endWord)return level;

que.pop();

size--;

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

{

for(int j=0;j<26;j++)

{

string newwrd=f;

char ch=j+'a';

string x=string(1,ch);

newwrd[i]=ch;

if(map[newwrd]==1)

{

cout<<newwrd<<" ";

map[newwrd]=0;

que.push(newwrd);

}

}

}

}

}

return 0;

}

};

144.number of island 1:dfs and save string and use backtrack

class Solution {

public:

/\*\*

\* @param grid: a list of lists of integers

\* @return: return an integer, denote the number of distinct islands

\*/

void mark\_dfs(int i,int j,vector<vector<int>>& grid,int r,int c,string& path){

grid[i][j]=0;

if(i-1>=0 && grid[i-1][j]==1){

path=path+"U";

mark\_dfs(i-1,j,grid,r,c,path);

}

if(j-1>=0 && grid[i][j-1]==1){

path=path+"L";

mark\_dfs(i,j-1,grid,r,c,path);

}

if(i+1<r && grid[i+1][j]==1){

path=path+"D";

mark\_dfs(i+1,j,grid,r,c,path);

}

if(j+1<c && grid[i][j+1]==1){

path=path+"R";

mark\_dfs(i,j+1,grid,r,c,path);

}

path=path+"X";

}

int numberofDistinctIslands(vector<vector<int>> &grid) {

// write your code here

unordered\_map<string,int> mp;

int cnt=0;

int m=grid.size();

int n=grid[0].size();

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

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

if(grid[i][j]==1){

string path="S";

mark\_dfs(i,j,grid,m,n,path);

if(mp.find(path)==mp.end()){

mp[path]=1;

cnt++;

}

}

}

}

return cnt;

}

};

145.eulerian path:undirected count indegree it should be even for path 0 (Cycle)or 2(path) odd indegree and for directed indegree ==outdegree (cycle)and two place indegree+1=outdegree (path)

\* The function returns one of the following values

   0 --> If grpah is not Eulerian

   1 --> If graph has an Euler path (Semi-Eulerian)

   2 --> If graph has an Euler Circuit (Eulerian)  \*/

int Graph::isEulerian()

{

    // Check if all non-zero degree vertices are connected

    if (isConnected() == false)

        return 0;

    // Count vertices with odd degree

    int odd = 0;

    for (int i = 0; i < V; i++)

        if (adj[i].size() & 1)

            odd++;

    // If count is more than 2, then graph is not Eulerian

    if (odd > 2)

        return 0;

    // If odd count is 2, then semi-eulerian.

    // If odd count is 0, then eulerian

    // Note that odd count can never be 1 for undirected graph

    return (odd)? 1 : 2;

}

146.leetcode 684:redundant connection :simple dsu and see when par are same

class Solution {

public:

int find(int x,vector<int>&par)

{

if(x==par[x])return x;

//int px=find(par[x],par);

//par[x]=px;

return find(par[x],par);

//return px;

}

bool unin(int x,int y,vector<int>&par,vector<int>&rank)

{

int px=find(x,par);

int py=find(y,par);

if(px!=py)

{

if(rank[px]>rank[py])par[py]=px;

else if(rank[px]<rank[py])par[px]=py;

else

{

par[px]=py;

rank[px]++;

}

par[px]=py;

return false;

}

return true;

}

vector<int> findRedundantConnection(vector<vector<int>>& edges) {

vector<int>par(edges.size()+1,0);

for(int i=0;i<=edges.size();i++)par[i]=i;

vector<int>rank(edges.size()+1,1);

for(int i=0;i<edges.size();i++)

{

bool res=unin(edges[i][0],edges[i][1],par,rank);

if(res==true)return edges[i];

}

return {};

}

};

147.leetcode 685 redundant connection 2:directed graph so basically three cases 2 par ,cycle or both..check for 2 par by indegree then cycle if cycle change then that is the ans else this is the answer

149.leetcode 1203 sort items by groups respectively:so make a graph such that end of groups and then do topological sort for groups for last and then for first

class Solution {

public:

bool ispossible=true;

void dfs( vector<vector<int>> &adj,int src,stack<int> &st,bool \*visited,bool\*recstack,int &n,int &m){

visited[src]=true;

recstack[src]=true;

for(int i=0;i<adj[src].size();i++)

{

if(visited[adj[src][i]]==false)

{

dfs(adj,adj[src][i],st,visited,recstack,n,m);

}

else{

if(recstack[adj[src][i]]==true) ispossible=false;

}

}

recstack[src]=false;

if(src<n){

st.push(src);

}

}

vector<int> sortItems(int n, int m, vector<int>& group, vector<vector<int>>& beforeItems) {

vector<vector<int>> adj(n+2\*m);

for(int i=0;i<group.size();i++)

{

int x=group[i];

if(x==-1) continue;

int startg=n+(2\*x);

int endg=n+(2\*x)+1;

adj[startg].push\_back(i);

adj[i].push\_back(endg);

}

//stage 1 complete with building graph via group numbers

for(int i=0;i<beforeItems.size();i++)

{

for(int j=0;j<beforeItems[i].size();j++)

{

int before=beforeItems[i][j];

int groupB=group[before];

int groupI=group[i];

//now 4 cases arise

if(groupB==-1&&groupI==-1){

adj[before].push\_back(i);

}

else if(groupB!=-1&&groupI==-1){

int end=n+(2\*groupB)+1;

adj[end].push\_back(i);

}

else if(groupB==-1&&groupI!=-1)

{

int start=n+(2\*groupI);

adj[before].push\_back(start);

}

else if(groupB!=-1&&groupI!=-1)

{

if(groupB==groupI)

{

adj[before].push\_back(i);

}

else if(groupB!=groupI)

{

int start=n+(2\*groupI);

int end=n+(2\*groupB)+1;

adj[end].push\_back(start);

}

}

}

}

/\*for(int i=0;i<n+(2\*m);i++)

{

cout<<i<<" ";

for(int j=0;j<adj[i].size();j++)

{

cout<<adj[i][j]<<" ";

}

cout<<endl;

}

\*/

//complete graph. now initialize array for visited

stack<int> st;

vector<int> ans;

bool \*visited=new bool[n+(2\*m)];

bool \*recstack=new bool[n+(2\*m)];

for(int i=0;i<(n+(2\*m));i++)

{

visited[i]=false;

recstack[i]=false;

}

for(int i=n;i<n+(2\*m);i++)

{

if(visited[i]==false)

{

dfs(adj,i,st,visited,recstack,n,m);

}

}

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

{

if(visited[i]==false)

{

dfs(adj,i,st,visited,recstack,n,m);

}

}

if(ispossible==false) return ans;

while(!st.empty())

{

ans.push\_back(st.top());

st.pop();

}

return ans;

}

};

150.leetcode 1162 as far as possible:simple bfs just put 1 in que and manhattan

class Solution {

public:

int bfs(queue<pair<int,int>>&que,vector<vector<int>>&grid)

{

int max\_=-1;

vector<vector<int>>dir={{1,0},{0,1},{-1,0},{0,-1}};

while(!que.empty())

{

pair<int,int>f=que.front();que.pop();

// cout<<f.second;

max\_=max(max\_,f.second);

for(int i=0;i<dir.size();i++)

{

int x=f.first/grid[0].size()+dir[i][0];

int y=f.first%grid[0].size()+dir[i][1];

if(x>=0&&x<grid.size()&&y>=0&&y<grid[0].size()&&grid[x][y]==0)

{

grid[x][y]=1;

pair<int,int>p(x\*grid[0].size()+y,f.second+1);

que.push(p);

}

}

}

return max\_;

}

int maxDistance(vector<vector<int>>& grid) {

queue<pair<int,int>>que;//rclevel

for(int i=0;i<grid.size();i++)

{

for(int j=0;j<grid[0].size();j++)

{

if(grid[i][j]==1){

pair<int,int>n(i\*grid[0].size()+j,0);

que.push(n);

}

}

}

if(que.size()==grid.size()\*grid[0].size())return -1;

return bfs(que,grid);

}

};

151.leetcode 886 possible bipartion :look likes directed but first create a graph such that next level in different group and then is bipartite

class Solution {

public:

bool bfsbi(unordered\_map<int,vector<int>>graph,vector<int>&vis,int src)

{

queue<int>que;

que.push(src);

vis[src]=10;

while(!que.empty())

{

int f=que.front();que.pop();

for(int i:graph[f])

{

if(vis[i]==-1)

{

que.push(i);

if(vis[f]==10)vis[i]=12;

else if (vis[f]==12)vis[i]=10;

}

else

{

if((vis[i]==10&& vis[f]==10)||(vis[i]==12&&vis[f]==12))

{ return false;}

}

}

}

return true;

}

bool possibleBipartition(int N, vector<vector<int>>& dislikes) {

unordered\_map<int,vector<int>>graph;

for(int i=0;i<dislikes.size();i++)

{

graph[dislikes[i][0]].push\_back(dislikes[i][1]);

graph[dislikes[i][1]].push\_back(dislikes[i][0]);

}

vector<int>vis(N+1,-1);

bool res=true;

for(int i=0;i<graph.size();i++)

{if(vis[i]==-1){

res=res&&bfsbi(graph,vis,i);}

}

return res;

}

};

152.leetcode 934 shortest bridge :mark using bfs 1 island then put in que and bfs to find next 1

class Solution {

public:

vector<vector<int>>dir={{1,0},{0,1},{-1,0},{0,-1}};

void dfs(vector < vector < int >> & A, int r, int c) {

A[r][c]=2;

for(int i=0;i<dir.size();i++)

{

int x=r+dir[i][0];

int y=c+dir[i][1];

if(x<A.size()&&y<A[0].size()&&x>=0&&y>=0&&A[x][y]==1)

{

dfs(A,x,y);

}

}

/\*if(A[r][c]==0)return;

A[r][c]=2;

if (r + 1 < A.size()&&A[r][c]==1) dfs(A, r + 1, c);

if (r - 1 >= 0&&A[r][c]==1) dfs(A, r - 1, c);

if (c + 1 < A[0].size()&&A[r][c]==1) dfs(A, r, c + 1);

if (c - 1 >= 0&&A[r][c]==1) dfs(A, r, c - 1);\*/

}

int bfs(queue < int > que, vector < vector < int >> & A) {

int level = -1;

while (!que.empty()) {

int size = que.size();

level++;

while (size > 0) {

int rc = que.front();

que.pop();

size--;

int x = rc / A[0].size();

int y = rc % A[0].size();

for (int i = 0; i < dir.size(); i++) {

int r = x + dir[i][0];

int c = y + dir[i][1];

if (r >= 0 && c >= 0 && r < A.size() && c < A.size() && A[r][c] == 1)return level;

if (r >= 0 && c >= 0 && r < A.size() && c < A.size() && A[r][c] == 0) {

A[r][c] = 2;

que.push(r \* A[0].size() + c);

}

}

}

}

return 0;

}

int shortestBridge(vector<vector<int>>& A) {

int posi=-1,posj=-1;

int flag=0;

for(int i=0;i<A.size();i++)

{

for(int j=0;j<A[0].size();j++)

{

if(A[i][j]==1&&flag==0)

{

posi=i,posj=j;

flag=1;

}

}

}

// A[posi][posj]=2;

dfs(A,posi,posj);

queue<int>que;

for(int i=0;i<A.size();i++)

{

for(int j=0;j<A[0].size();j++)

{

cout<<A[i][j];

if(A[i][j]==2)

que.push(i\*A.size()+j);

}

cout<<endl;

}

int l=bfs(que,A);

return l;

}

};

159.leetcode 773 sliding puzzle

class Solution {

public:

int slidingPuzzle(vector<vector<int>>& board) {

vector<vector<int>>dir={{1,3},{0,2,4},{1,5},{0,4},{1,3,5},{2,4}};

string s="";

for(int i=0;i<2;i++)

{

for(int j=0;j<3;j++)

{

s=s+to\_string(board[i][j]);

}

}

unordered\_map<string,int>map;

string ans="123450";

queue<string>que;

que.push(s);

int level=-1;

while(!que.empty())

{

int size=que.size();

level++;

while(size>0)

{

size--;

string f=que.front();que.pop();

int pos=-1;

if(f==ans)return level;

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

{

if(f[i]=='0')

{

pos=i;

break;

}

}

for(int d:dir[pos])

{

string n=f;

swap(n[pos],n[d]);

if(map[n]==1)continue;

else

{ map[n]=1;

que.push(n);

}

}

}

}

return -1;

}

};

160.minimum no of swaps to make array sorted gfg

vector<bool>vis(N,false);

int move=0;

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

{

if(vis[i]==true)continue;

else

{

int cycle=0;

int j=vect[i].second;

while(vis[j]==false)

{

vis[j]=true;

j=vect[j].second;

cycle++;

}

move+=cycle-1;

}

}

167.leetcode 781 rabbits in forrest:first make freq map and then tranverse map count groups and and groupsize=key+1 and gc=ciel(map[key]/gs]

class Solution {

public:

int numRabbits(vector<int>& answers) {

unordered\_map<int,int>map;

for(int i=0;i<answers.size();i++)

{

map[answers[i]]++;

}

int ans=0;

for(pair<int,int>p:map)

{

int groupsize=p.first+1;

if(p.first>=p.second)ans+=groupsize;

else

{

double g=double(p.second)/double(groupsize);

cout<<g;

int groupcount=ceil(g);

ans+=(groupcount)\*groupsize;

}

}

return ans;

}

};

168.max consecutive one and zeroes in array:just update cmax with every zero and omax=max(omax,cmax)

// Returns count of maximum consecutive 1's

// in binary array arr[0..n-1]

int getMaxLength(bool arr[], int n)

{

    int count = 0; //intitialize count

    int result = 0; //initialize max

    for (int i = 0; i < n; i++)

    {

        // Reset count when 0 is found

        if (arr[i] == 0)

            count = 0;

        // If 1 is found, increment count

        // and update result if count becomes

        // more.

        else

        {

            count++;//increase count

            result = max(result, count);

        }

    }

    return result;

}

169.leetcode 560 with subarray sum k:in this make prefix sum array and put in hashmap after check whether in hashmap or not

class Solution {

public:

int subarraySum(vector<int>& nums, int k) {

unordered\_map<int,int>map;

int sum=0;

map[0]=1;

vector<int>prefix(nums.size(),0);

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

{

sum+=nums[i];

prefix[i]=sum;

}

int ans=0;

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

{

if(map[prefix[i]-k]!=0)

{ans+=map[prefix[i]-k];}

map[prefix[i]]++;

}

return ans;

}

};

170. count all subarray with sum divisible by k gfg:so first prefix sum array and then rem array and if negative add k and before adding to array check if that rem exist of not

#include <iostream>

#include<unordered\_map>

#include<vector>

using namespace std;

int main() {

int tc;

cin>>tc;

for(int t=0;t<tc;t++)

{

int n;

int k;

cin>>n>>k;

int arr[n];

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

{

cin>>arr[i];

}

int rem[n];

int sum=0;

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

{

sum+=arr[i];

rem[i]=sum%k>=0?sum%k:(sum%k)+k;

// cout<<rem[i];

}

unordered\_map<int,int>map;

map[0]=1;

int ans=0;

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

{

if(map[rem[i]]!=0)

{

ans+=map[rem[i]];

}

map[rem[i]]++;

}

cout<<ans<<endl;

}

//code

return 0;

}

171.k closest to origin leetcode 973: make priority of maximum of size size k and add distance and cordibates and later pop

class Solution {

public:

vector<vector<int>> kClosest(vector<vector<int>>& points, int k) {

if(points.size()==0||k==0)return {};

if(points.size()==1)return points;

priority\_queue<vector<int>,vector<vector<int>>>pq;

vector<int>p={points[0][0]\*points[0][0]+points[0][1]\*points[0][1],points[0][0],points[0][1]};

pq.push(p);

for(int i=1;i<points.size();i++)

{

vector<int>n=pq.top();

vector<int>f={points[i][0]\*points[i][0]+points[i][1]\*points[i][1],points[i][0],points[i][1]};

if(pq.size()<k)

{

pq.push(f);

}

else if(pq.size()==k)

{

if(n[0]>f[0]){pq.pop();pq.push(f);}

}

}

vector<vector<int>>ans;

while(!pq.empty())

{

vector<int>s=pq.top();pq.pop();

vector<int>l={

s[1],s[2]

};

ans.push\_back(l);

}

//cout<<pq.top()[0];

return ans;

}

};

172.no of subarray with equal no.s of 0,s and 1’s gfg:first keep two array count of zero and 1,then do hashing if same pattern exist add++and ans+=map[pc0-pc1] map[pc0-pc1]++

#include <iostream>

#include<vector>

#include<unordered\_map>

using namespace std;

int main() {

int tc;

cin>>tc;

for(int t=0;t<tc;t++)

{

int n;

cin>>n;

int arr[n];

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

{

cin>>arr[i];

}

int pc0[n];

int pc1[n];

pc0[0]=arr[0]==0?1:0;

pc1[0]=arr[0]==1?1:0;

//cout<<pc0[0];

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

{

if(arr[i]==0)

{

pc0[i]=1+pc0[i-1];

pc1[i]=pc1[i-1];

}

else

{

pc1[i]=1+pc1[i-1];

pc0[i]=pc0[i-1];

}

//cout<<pc0[i];

}

// cout<<" ";

unordered\_map<int,int>map;

map[0]=1;

int ans=0;

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

{

if(map[pc0[i]-pc1[i]]!=0)ans+=map[pc0[i]-pc1[i]];

map[pc0[i]-pc1[i]]++;

}

cout<<ans<<endl;

}

//code

return 0;

}

173.count substring with equal no of 0 1 2 gfg:keep 3 array and pc0 pc1 and pc2 and do hashing of pc0-pc1\*pc0-pc2

#include <iostream>

#include<unordered\_map>

#include<vector>

using namespace std;

int main() {

//code

int tc;

cin>>tc;

for(int t=0;t<tc;t++)

{

string n;

cin>>n;

int s=n.length();

vector<int>pc0(s,0);

vector<int>pc1(s,0);

vector<int>pc2(s,0);

pc0[0]=n[0]=='0'?1:0;

pc1[0]=n[0]=='1'?1:0;

pc2[0]=n[0]=='2'?1:0;

for(int i=1;i<s;i++)

{

if(n[i]=='0')

{

pc0[i]=1+pc0[i-1];

pc1[i]=pc1[i-1];

pc2[i]=pc2[i-1];

}

else if(n[i]=='1')

{

pc1[i]=1+pc1[i-1];

pc0[i]=pc0[i-1];

pc2[i]=pc2[i-1];

}

else if(n[i]=='2')

{

pc0[i]=pc0[i-1];

pc1[i]=pc1[i-1];

pc2[i]=1+pc2[i-1];

}

}

// for(int i=0;i<s;i++)cout<<pc2[i];

unordered\_map<string,int>map;

int ans=0;

map["0\*0"]=1;

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

{

int a=pc0[i]-pc1[i];

int b=pc0[i]-pc2[i];

string c=to\_string(a)+'\*'+to\_string(b);

if(map[c]!=0)

{

ans+=map[c];

}

map[c]++;

// cout<<c<<map[c];

}

cout<<ans<<endl;

}

return 0;

}

174.leetcode 871 min no of refueling shop:keep pq and check whether you reach the coordinate and if not then pq.pop and cant bool then return -1 else add and check whether and then push fuel at that stop

class Solution {

public:

int minRefuelStops(int target, int startfuel, vector<vector<int>>& stations) {

priority\_queue<int,vector<int>>pq;

int ans=0;

for(int i=0;i<stations.size();i++)

{

if(stations[i][0]<=startfuel)

{

pq.push(stations[i][1]);

cout<<stations[i][0];

}

else

{

while(startfuel<stations[i][0]&&pq.size()>0)

{startfuel+=pq.top();

pq.pop();

ans++;

}

if(startfuel<stations[i][0]&&pq.size()==0)return -1;

pq.push(stations[i][1]);

}

}

//cout<<" "<<startfuel<<" "<<ans;

while(startfuel<target&&pq.size()>0)

{

startfuel+=pq.top();pq.pop();

// cout<<";"<<startfuel;

ans++;

}

if(startfuel<target)return -1;

return ans;

}

};

175.check whether ap can be formed or not gfg:find smallest ,second small ,diff and check if no. exist in hashmap or not if not then size of hashmap and return ans;

176.x in kind of deck of cards leetcode 914: basically u have to form group maximum possible of size so,find freq map and then gcd

class Solution {

public:

int gcd(int a,int b)

{

if(b==0)return a;

return gcd(b,a%b);

}

bool hasGroupsSizeX(vector<int>& deck) {

unordered\_map<int,int>map;

for(int i=0;i<deck.size();i++)

{

map[deck[i]]++;

}

vector<int>a;

for(pair<int,int>p:map)

{

a.push\_back(p.second);

}

int ans=a[0];

for(int i=1;i<a.size();i++)

{

ans=gcd(ans,a[i]);

}

return ans>=2?true:false;

}

};

177leetcode 954:use ordered map and check whether double existsand frequency matching also.

class Solution {

public:

bool canReorderDoubled(vector<int>& A) {

map<int, int> m;

for(int x: A) m[abs(x)]++;

for(auto & p: m) { // start with smallest number

int x = p.first;

if(m[x]!=0) {

if(m[2\*x] < m[x]) return false;

m[2\*x] -= m[x];

}

}

return true;

}

};

178.morning assembly gfg: here find sequence of consective no by checking before and then len -lcs

#include <iostream>

#include<unordered\_map>

using namespace std;

int main() {

int tc;

cin>>tc;

for(int t=0;t<tc;t++)

{

int n;

cin>>n;

int arr[n];

int max\_=-1;

for(int i=0;i<n;i++)cin>>arr[i];

unordered\_map<int,int>map;

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

{

if(map[arr[i]-1]!=0)

{

map[arr[i]]=map[arr[i]-1]+1;

}

else

{ map[arr[i]]=1;}

max\_=max(max\_,map[arr[i]]);

}

cout<<n-max\_<<endl;

}

//code

return 0;

}

179.leetcode 128 max consecutive seq:put all element in map,then transvere array check next and before longest consecutive seq and mark 0 in map and len=max(len,next-before+1)

class Solution {

public:

int longestConsecutive(vector<int>& nums) {

if(nums.size()==0)return 0;

unordered\_map<int,int>map;

for(int ele:nums)

{

map[ele]=1;

}

int len=1;

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

{

if(map[nums[i]]==0)continue;

map[nums[i]]=0;

int prevele=nums[i]-1;

int nextele=nums[i]+1;

while(map[prevele]==1){map[prevele]=0;prevele--;}

while(map[nextele]==1){map[nextele]=0;nextele++;}

len=max(len,nextele-prevele-1);

}

return len;

}

};

180.leetcode 554 brick wall:keep a hashmap in which u store wall edge and calculate max and return wall no-max

class Solution {

public:

int leastBricks(vector<vector<int>>& walls) {

unordered\_map<int,int>map;

int max\_=0;

for(int i=0;i<walls.size();i++)

{

int sum=0;

for(int j=0;j<walls[i].size()-1;j++)

{

sum+=walls[i][j];

map[sum]++;

}

}

for(pair<int,int>p:map)

{

cout<<p.first<<"->"<<p.second<<":";

max\_=max(max\_,p.second);

}

return walls.size()-max\_;

}

};

181.leetcode 205 isomeric string:keep hashset and hashmap and check if curr char in map not then add and check t[I ]not in map if false return false and else unique mapping and in vis

class Solution {

public:

bool isIsomorphic(string s, string t) {

unordered\_map<char,int>vis;

unordered\_map<char,char>map;

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

{

if(!map.count(s[i]))

{

if(vis[t[i]]==0)

{

vis[t[i]]=1;

map[s[i]]=t[i];

}

else

{

return false;

}

}

else

{

if(vis[t[i]]==0||map[s[i]]!=t[i])return false;

}

}

return true;

}

};

182.

183.

184. leetcode 463 perimeter of island:transver if u find 1 ans+4 and if any dir 1 again ans—

class Solution {

public:

int islandPerimeter(vector<vector<int>>& grid) {

vector<vector<int>>dir={{1,0},{0,1},{-1,0},{0,-1}};

int ans=0;

for(int i=0;i<grid.size();i++)

{

for(int j=0;j<grid[i].size();j++)

{

if(grid[i][j]==1)

{

ans+=4;

for(int k=0;k<4;k++)

{

int x=i+dir[k][0];

int y=j+dir[k][1];

if(x>=0&&y>=0&&x<grid.size()&&y<grid[0].size()&&grid[x][y]==1)

{

ans--;

}

}

}

}

}

return ans;

}

};

185.leetcode 895 max freq stack:keep two map 1 no. freq and freq and stack n maxfreq while pushing add to both and update max freq and while popping pop fron stack of maxfreq if empty maxfreq—and freq[no]—

class FreqStack {

public:

unordered\_map<int,stack<int>>fs;

unordered\_map<int,int>freq;

int maxfreq=0;

FreqStack() {

}

void push(int x) {

freq[x]++;

maxfreq=max(freq[x],maxfreq);

fs[freq[x]].push(x);

}

int pop() {

int ans=fs[maxfreq].top();

fs[maxfreq].pop();

freq[ans]--;

if(fs[maxfreq].empty())maxfreq--;

return ans;

}

};

/\*\*

\* Your FreqStack object will be instantiated and called as such:

\* FreqStack\* obj = new FreqStack();

\* obj->push(x);

\* int param\_2 = obj->pop();

\*/

186.length of the subarray with contigious element gfg:here keep max and min max-min=i-j two loops(O(n2))

int findLength(int arr[], int n)

{

    int max\_len = 1;  // Initialize result

    for (int i=0; i<n-1; i++)

    {

        // Initialize min and max for all subarrays starting with i

        int mn = arr[i], mx = arr[i];

        // Consider all subarrays starting with i and ending with j

        for (int j=i+1; j<n; j++)

        {

            // Update min and max in this subarray if needed

            mn = min(mn, arr[j]);

            mx = max(mx, arr[j]);

            // If current subarray has all contiguous elements

            if ((mx - mn) == j-i)

                max\_len = max(max\_len, mx-mn+1);

        }

    }

    return max\_len;  // Return result

}

187.length of the largest subarray with contigious elements repeating:do same as above only keep a hashset and check if element occurred in set before if yes then simply break or i++

int findLength(int arr[], int n)

{

    int max\_len = 1; // Inialize result

    // One by one fix the starting points

    for (int i=0; i<n-1; i++)

    {

        // Create an empty hash set and

        // add i'th element to it.

        set<int> myset;

        myset.insert(arr[i]);

        // Initialize max and min in

        // current subarray

        int mn = arr[i], mx = arr[i];

        // One by one fix ending points

        for (int j=i+1; j<n; j++)

        {

            // If current element is already

            // in hash set, then this subarray

            // cannot contain contiguous elements

            if (myset.find(arr[j]) != myset.end())

                break;

            // Else add current element to hash

            // set and update min, max if required.

            myset.insert(arr[j]);

            mn = min(mn, arr[j]);

            mx = max(mx, arr[j]);

            // We have already checked for

            // duplicates, now check for other

            // property and update max\_len

            // if needed

            if (mx - mn == j - i)

                max\_len = max(max\_len, mx - mn + 1);

        }

    }

    return max\_len; // Return result

}

188.sliding window max leetcode 239:divide the array in k groups and calculate maxr and maxl in the interval such that for i%k==0 or start has maxl same ansi+1%k==0 is end same maxr and calculate for start it is right max else rightmax or leftmax[i+k-1]

class Solution {

public:

vector<int> maxSlidingWindow(vector<int>& nums, int k) {

vector<int>leftmax(nums.size(),0);

vector<int>rightmax(nums.size(),0);

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

{

if(i%k==0)

{

leftmax[i]=nums[i];

}

else

{

if(i>0)leftmax[i]=max(leftmax[i-1],nums[i]);

}

}

for(int i=nums.size()-1;i>=0;i--)

{

if((i+1)%k==0)

{

rightmax[i]=nums[i];

}

else

{

if(i+1!=nums.size())rightmax[i]=max(rightmax[i+1],nums[i]);

else

{

rightmax[i]=nums[i];

}

}

}

vector<int>ans(nums.size()-k+1,0);

for(int i=0;i<=nums.size()-k;i++)

{

if(i%k==0)

{

ans[i]=rightmax[i];

}

else

{

ans[i]=max(rightmax[i],leftmax[i+k-1]);

}

}

for(int i=0;i<ans.size();i++)cout<<ans[i];

return ans;

}

};

189.leetcode 42 trapping rainwater:first method to calculate ngol and ngor and find min of two and subtract with height that is water return water else keep two variavle leftmax and rightmax now if left array is less than right then move left++ check if leftmax could be upadated or calulate water by leftmax- arr[i] i++ do same for right side until i==j

class Solution {

public:

int trap(vector<int>& height) {

if(height.size()==0)return {};

int rightmax=height[height.size()-1],leftmax=height[0];

int ans=0,i=0,j=height.size()-1;

while(i<j)

{

if(height[i]<=height[j])

{

if(leftmax<=height[i])

{

leftmax=height[i];

}

else

{

ans+=leftmax-height[i];

}

i++;

}

else

{

if(rightmax<=height[j])rightmax=height[j];

else

{

ans+=rightmax-height[j];

}

j--;

}

}

return ans;

}

};

190.

191.

192.leetcode 378 kth smallest in sorted matrix:add first col in matrix and remove samllest and add next col and row do until k

class Solution {

public:

int kthSmallest(vector<vector<int>>& matrix, int k) {

priority\_queue<vector<int>,vector<vector<int>>,greater<vector<int>>>pq;

for(int i=0;i<matrix[0].size();i++)

{

vector<int>p={matrix[i][0],i,0};

pq.push(p);

}

int ans=0;

while(k>0)

{

k--;

vector<int>s=pq.top();

pq.pop();

if(s[2]+1<matrix[0].size())

{

vector<int>n={matrix[s[1]][s[2]+1],s[1],s[2]+1};

pq.push(n);

}

ans=s[0];

}

return ans;

}

};

193.kth smallest fraction leetcode 786:right comaparator and then add first all i=0 to nums,size()-2 and j=nums.size()-1 fraction and then pop and add next to it;or binary search in whin hi=0 and lo=1 and double everything and add all element with val above it if range k geater then reduce range else increasr or cout<<ans;keep count =nums.size()-j;j=1 intially for loop if j<arr arr[j]>mid\*arr[i] then j++ temp—and count+=temp

class Solution {

public:

vector<int> kthSmallestPrimeFraction(vector<int>& A, int K) {

double lo=0,hi=1;

int n=0,d=1;

while(lo<=hi)

{

double mid=(hi+lo)/2;

int j=1;

int temp=A.size()-j;

int count=0;

for(int i=0;i<A.size()-1;i++)

{

while(j<A.size() && A[i]>(double)A[j]\*mid )

{

j++;

temp--;

}

count+=temp;

if( j<A.size()&&n\*A[j]<d\*A[i])

{

n=A[i];

d=A[j];

}

}

if(count<K)

{

lo=mid;

n=0;

d=1;

}

else if(count>K)

{

hi=mid;

n=0;

d=1;

}

else if(count==K)

{

return {n,d};

}

}

return {};

}

};

194. leetcode 319 bulb switcher:intuition is odd factor no. will be on so u will have to find total square below n

class Solution {

public:

int bulbSwitch(int n) {

int ans=0;

for(int i=1;i\*i<=n;i++)

{

ans++;

}

return ans;

}

};

195.count pair array who sum us divible by k gfg:calculate rem and negative rem +k =newrem add in hashmap and check if k-rem exists or not if exists then add in ans and map[rem]++

int divisibleSumPairs(int n, int k, vector<int> ar) {

unordered\_map<int,int>map;

int ans=0;

for(int i=0;i<n;i++)

{

    int rem=ar[i]%k;

    if(rem<0)rem+=k;

    if(rem==0)

    {

        ans+=map[rem];

    }

    else if(map[k-rem]!=0)ans+=map[k-rem];

    map[rem]++;

}

return ans;

}

196.

197.pair of non coinciding points:here x1=x2 or y1=y2 three maps if x1=x2 add ans map[x1] map[x1]++ if y1==y2 ans+=map[y] map[y1]++ and if map[x1\*y1] ans-=map[x1\*y1] and map[x1\*y1]++

#include <iostream>

#include<vector>

#include<unordered\_map>

using namespace std;

int main() {

int tc;

cin>>tc;

for(int t=0;t<tc;t++)

{

int n;

cin>>n;

vector<vector<int>>points(n,vector<int>(2,0));

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

{cin>>points[i][0];

cin>>points[i][1];

}

unordered\_map<int,int>row;

unordered\_map<int,int>col;

unordered\_map<string,int>rc;

int ans=0;

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

{

if(row[points[i][0]]!=0)

{

ans+=row[points[i][0]];

}

row[points[i][0]]++;

if(col[points[i][1]]!=0)

{

ans+=col[points[i][1]];

}

col[points[i][1]]++;

string s=to\_string(points[i][0])+"\*"+to\_string(points[i][1]);

//cout<<s;

if(rc[s]!=0)

{

ans=ans-2\*rc[s];

}

rc[s]++;

}

cout<<ans<<endl;

}

//code

return 0;

}

198.smallest no whose digits are multiple to given no.n gfg:take the no while n>0 for (9 to 1) if n%i==0 n/I and i++ and make ans and return reversed ans

#include <iostream>

using namespace std;

int main() {

int tc;

cin>>tc;

for(int t=0;t<tc;t++)

{

long int n;

cin>>n;

if(n==1)cout<<"1";

else{long int ans=0;

for(int i=9;i>=2;i--)

{

if(n%i==0)

{

ans=ans\*10+i;

n=n/i;

i++;

}

}

if(n!=1)cout<<"-1";

else

{

long int revans=0;

while(ans>0)

{

revans=revans\*10+ans%10;

ans=ans/10;

}

cout<<revans;

}

}

cout<<endl;

}

//code

return 0;

}

199.check if freq of all characters can become sane by removal of 1 char

200.a simple fraction gfg:firstly calulate ans if rem store in map with pos equal to 0 while rem store in map if rem there from before break and stor pos now break string and add brackets

#include <iostream>

#include<unordered\_map>

using namespace std;

int main() {

//code

int tc;

cin>>tc;

for(int t=0;t<tc;t++)

{

int a,b;

cin>>a>>b;

string ab=to\_string(a/b);

unordered\_map<int,int>map;

if(a%b>0)

{

int pos=0,rem=0;

string ans;

rem=a%b;

while(map.count(rem)==0&&rem!=0)

{

map[rem]=pos;

ans+=to\_string(10\*rem/b);

rem=(10\*rem)%b;

//cout<<rem;

pos++;

}

if(map.count(rem)!=0)

{

int si=map[rem];

// cout<<ans;

ab=ab+"."+ans.substr(0,si)+"("+ans.substr(si,pos)+")";

}

else if(rem==0)

{

ab+="."+ans;

}

}

cout<<ab<<endl;

}

return 0;

}

201

202.check if the anagram of string is palindrome or not gfg:In this even char except 1 or none

#include <iostream>

#include<vector>

using namespace std;

int main() {

//code

int tc;

cin>>tc;

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

{

string s;

cin>>s;

vector<int>v(26,0);

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

{

v[s[j]-'a']++;

}

int flag=0;

for(int j=0;j<26;j++)

{

if(v[j]%2==1&&flag==0)

{

flag=1;

}

else if(v[j]%2==1&&flag==1)

{

flag=2;

break;

}

}

if(flag==2)cout<<"No";

else cout<<"Yes";

cout<<endl;

}

return 0;

}

203.group anagrams leetcode 49:make 26 size vector and then do hashing with \* after each char or prime factor multiplication hashing

class Solution {

public:

vector<vector<string>> groupAnagrams(vector<string>& strs) {

/\* vector<int>ch={2 ,3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,73 , 79, 83, 89, 97, 101};

unordered\_map<long int,vector<string>>map;\*/

unordered\_map<string,vector<string>>map;

for(int i=0;i<strs.size();i++)

{

//long int num=1;

vector<int>v(26,0);

for(char c:strs[i])

{

v[c-'a']++;

}

string num;

for(int j=0;j<26;j++)

{

num+=to\_string(v[j])+"\_";

}

map[num].push\_back(strs[i]);

}

vector<vector<string>>ans;

for(pair<string,vector<string>>p:map)

{

vector<string>a;

for(string s:p.second)

{

a.push\_back(s);

}

ans.push\_back(a);

}

return ans;

}

};

204

205.smallest subarray with all occurences of most freq element :keep a hashmap storing freq ,si,ei

206.check if two strings are k anagrams:transverse on 1 string and strore freq map and transverse on if freq>letter match++ if total-match<=kreturn true else false

class Solution {

public:

vector<vector<string>> groupAnagrams(vector<string>& strs) {

/\* vector<int>ch={2 ,3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71,73 , 79, 83, 89, 97, 101};

unordered\_map<long int,vector<string>>map;\*/

unordered\_map<string,vector<string>>map;

for(int i=0;i<strs.size();i++)

{

//long int num=1;

vector<int>v(26,0);

for(char c:strs[i])

{

v[c-'a']++;

}

string num;

for(int j=0;j<26;j++)

{

num+=to\_string(v[j])+"\_";

}

map[num].push\_back(strs[i]);

}

vector<vector<string>>ans;

for(pair<string,vector<string>>p:map)

{

vector<string>a;

for(string s:p.second)

{

a.push\_back(s);

}

ans.push\_back(a);

}

return ans;

}

};

207.length of the largest substring without repeating character gfg:store thr last location and update length as per last length

#include <iostream>

#include<vector>

using namespace std;

int substringlength(string str)

{

int n = str.size();

int res = 0; // result

// last index of all characters is initialized

// as -1

vector<int> lastIndex(256, -1);

// Initialize start of current window

int i = 0;

// Move end of current window

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

// Find the last index of str[j]

// Update i (starting index of current window)

// as maximum of current value of i and last

// index plus 1

i = max(i, lastIndex[str[j]] + 1);

// Update result if we get a larger window

res = max(res, j - i + 1);

//

// Update last index of j.

lastIndex[str[j]] = j;

}

return res;

}

int main() {

//code

int tc;

cin>>tc;

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

{

string s;

cin>>s;

cout<<substringlength(s)<<endl;

}

return 0;

}

208.leetcode 380 insert and delete get random: keep hashmap and vector.in hashmap store element with array pos and vect.push\_back ele if element before return false in pop if not ele return false else find idx of ele swap with last remove from map update map and get random insert

vect[rand()%vector.size]

class RandomizedSet {

unordered\_map<int, int> map;

vector<int> v;

public:

/\*\* Initialize your data structure here. \*/

RandomizedSet() {

}

/\*\* Inserts a value to the set. Returns true if the set did not already contain the specified element. \*/

bool insert(int val) {

if(map.count(val)!=0)return false;

map[val] = v.size();

v.push\_back(val);

return true;

}

/\*\* Removes a value from the set. Returns true if the set contained the specified element. \*/

bool remove(int val) {

if(map.count(val)==0)return false;

int temp=v.back();

v.back()=v[map[val]];

v[map[val]]=temp;

v.pop\_back();

map[temp]=map[val];

map.erase(val);

return true;

}

/\*\* Get a random element from the set. \*/

int getRandom() {

return v[rand() % v.size()];

}

};

209.leetcode 381 insert delete getrandom:make unorderedmap of unorderedmap and while pushing same as above just push pos while deleting unordermap pos delete their and swap and delete array and update array

class RandomizedCollection {

public:

/\*\* Initialize your data structure here. \*/

unordered\_map<int,unordered\_set<int>>map;

vector<int>v;

RandomizedCollection() {

}

/\*\* Inserts a value to the collection. Returns true if the collection did not already contain the specified element. \*/

bool insert(int val) {

if(map.count(val)==0)

{ map[val].insert(v.size());

v.push\_back(val);

return true;}

else

{map[val].insert(v.size());

v.push\_back(val);

return false;

}

}

/\*\* Removes a value from the collection. Returns true if the collection contained the specified element. \*/

bool remove(int val) {

if(map.count(val)==0)return false;

int ridx=\*map[val].begin();

map[val].erase(map[val].begin());

int temp=v[ridx];

v[ridx]=v[v.size()-1];

v[v.size()-1]=temp;

map[v[ridx]].erase(v.size()-1);

map[v[ridx]].insert(ridx);

v.pop\_back();

return true;

}

/\*\* Get a random element from the collection. \*/

int getRandom() {

return v[rand()%v.size()];

}

};

/\*\*

\* Your RandomizedCollection object will be instantiated and called as such:

\* RandomizedCollection\* obj = new RandomizedCollection();

\* bool param\_1 = obj->insert(val);

\* bool param\_2 = obj->remove(val);

\* int param\_3 = obj->getRandom();

\*/

210.build heap from array:for each par calculate child swap with largest child and call heapify such that with largest idx

// C++ program for building Heap from Array

#include <iostream>

using namespace std;

// To heapify a subtree rooted with node i which is

// an index in arr[]. N is size of heap

void heapify(int arr[], int n, int i)

{

int largest = i; // Initialize largest as root

int l = 2 \* i + 1; // left = 2\*i + 1

int r = 2 \* i + 2; // right = 2\*i + 2

// If left child is larger than root

if (l < n && arr[l] > arr[largest])

largest = l;

// If right child is larger than largest so far

if (r < n && arr[r] > arr[largest])

largest = r;

// If largest is not root

if (largest != i) {

swap(arr[i], arr[largest]);

// Recursively heapify the affected sub-tree

heapify(arr, n, largest);

}

}

// Function to build a Max-Heap from the given array

void buildHeap(int arr[], int n)

{

// Index of last non-leaf node

int startIdx = (n / 2) - 1;

// Perform reverse level order traversal

// from last non-leaf node and heapify

// each node

for (int i = startIdx; i >= 0; i--) {

heapify(arr, n, i);

}

}

// A utility function to print the array

// representation of Heap

void printHeap(int arr[], int n)

{

cout << "Array representation of Heap is:\n";

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

cout << arr[i] << " ";

cout << "\n";

}

211.heapsort:heap above in postorder

public class HeapSort

{

    public void sort(int arr[])

    {

        int n = arr.length;

        // Build heap (rearrange array)

        for (int i = n / 2 - 1; i >= 0; i--)

            heapify(arr, n, i);

        // One by one extract an element from heap

        for (int i=n-1; i>0; i--)

        {

            // Move current root to end

            int temp = arr[0];

            arr[0] = arr[i];

            arr[i] = temp;

            // call max heapify on the reduced heap

            heapify(arr, i, 0);

        }

    }

    // To heapify a subtree rooted with node i which is

    // an index in arr[]. n is size of heap

    void heapify(int arr[], int n, int i)

    {

        int largest = i; // Initialize largest as root

        int l = 2\*i + 1; // left = 2\*i + 1

        int r = 2\*i + 2; // right = 2\*i + 2

        // If left child is larger than root

        if (l < n && arr[l] > arr[largest])

            largest = l;

        // If right child is larger than largest so far

        if (r < n && arr[r] > arr[largest])

            largest = r;

        // If largest is not root

        if (largest != i)

        {

            int swap = arr[i];

            arr[i] = arr[largest];

            arr[largest] = swap;

            // Recursively heapify the affected sub-tree

            heapify(arr, n, largest);

        }

    }

211.binary search:logn for searching sorted array:comparsion be like 1\*logn whereas for more than that 2\*logn calculation

int binarySearch(int arr[], int l, int r, int x)

    {

        if (r >= l) {

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

            // If the element is present at the

            // middle itself

            if (arr[mid] == x)

                return mid;

            // If element is smaller than mid, then

            // it can only be present in left subarray

            if (arr[mid] > x)

                return binarySearch(arr, l, mid - 1, x);

            // Else the element can only be present

            // in right subarray

            return binarySearch(arr, mid + 1, r, x);

        }

        // We reach here when element is not present

        // in array

        return -1;

    }

212.median of a sorted array leetcode 4: calculate gl , mid1 using size keep first array of big

223.leetcode 70 climbings stairs:thinking count+=f(n-1)+f(n-2) if n<=1 return n;just like Fibonacci

class Solution {

public int climbStairs(int n) {

int a=0,b=1,sum=0;

if(n<=1)return n;

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

{

sum=a+b;

a=b;

b=sum;

}

return sum;

}

}

224.leetcode 45 jump game II:using dp is as like keep if last=0 jump and then jump<=arr[i] find min jumps ;

int Solution::jump(vector<int> &A) {

vector<int>dp(A.size(),INT\_MAX);

for(int i=A.size()-1;i>=0;i--)

{

if(i==A.size()-1){dp[i]=0;continue;}

for(int j=1;j<=A[i];j++)

{

if(i+j<A.size()&&dp[i+j]!=INT\_MAX)

{

dp[i]=min(dp[i],dp[i+j]+1);

}

}

}

return dp[0]==INT\_MAX?-1:dp[0];

}

Or using greedy keep three variables cmax cend st,ansj=1 start a loop for st<cend for(int i=st to cend) update cmax=max(camx,arr[i]+i) out of for loop and ansj++ cend=cmax and st=cend+1;

class Solution {

public:

int jump(vector<int>& nums) {

if(nums.size()==0||nums.size()==1)return 0;

int cmax=0,ansj=1,cend=0;

cend=nums[0];

int st=1;

while(cend<nums.size()-1)

{

for(int i=st;i<=cend;i++)

{

cmax=max(cmax,nums[i]+i);

}

ansj++;

st=cend+1;

cend=cmax;

}

return ansj++;

}

};

