

#include<bits/stdc++.h>

using namespace std;

int main()

{

    int a[500],i,j,n,b[500];

    cin>>n;

    for(i=0;i<n;i++)

    {

        cin>>b[i];

    }

    for(i=0;i<n;i++)

    {

        a[i]=i+1;

    }

    for(i=0;i<n;i++)

    {

        sort(a,a+n);

        if(i!=n-1)

        {

            cout<<a[b[i]]<<' ';

         a[b[i]]=n+1;

        }

        else

        {

            cout<<a[b[i]]<<endl;

        }

    }

    return 0;

}



#include<stdio.h>

#include<malloc.h>

typedef struct LinkedList\* List;

struct LinkedList

{

List right;

List left;

int n;

};

List create(int Num)

{

    List move=NULL;

    List leftList=NULL;

    List headList=NULL;

    for(int i=1;i<=Num;i++)

    {

            leftList=(List)malloc(sizeof(LinkedList));

            leftList->n=i;

            if(move==NULL)

            {

                move=headList=leftList;

            }

            else

            {

                move->left=leftList;

                leftList->right=move;

                move=leftList;

            }

    }

    move->left=headList;

    headList->right=move;

    return headList;

}

void solve(List move,int n,int m)

{

    while(n>2)

    {

        int temp=(m%n);

        while(temp--)

        {

            move=move->left;

        }

    List p,q;

    p=move->right;

    q=move->left;

    move->right=p->right;

    p->right->left=move;

    move->left=q->left;

    q->left->right=move;

    free(p);

    free(q);

    n=n-2;

}

    if(n==1)

    printf("%d\n",move->n);

    else

    printf("%d %d\n",(move->n)>(move->left->n)?(move->left->n):(move->n),

    (move->n)<(move->left->n)?(move->left->n):(move->n));

}

int main()

{

    int num,m,n;

    scanf("%d",&num);

    while(num--)

    {

        scanf("%d%d",&n,&m);

        List list;

        list=create(n);

        solve(list->right,n,m);

    }

return 0;

}



#include<stdio.h>

#include<stdlib.h>

#include<string.h>

char s[1000000];

int number[1000000]={0};

typedef int Position;

struct SNode {

int \*Data;

Position Top;

int MaxSize;

};

typedef struct SNode \*Stack;

Stack CreateStack( int MaxSize )

{

Stack S = (Stack)malloc(sizeof(struct SNode));

S->Data = (int \*)malloc(MaxSize \* sizeof(int));

S->Top = -1;

S->MaxSize = MaxSize;

return S;

}

bool IsFull( Stack S )

{

return (S->Top == S->MaxSize-1);

}

bool Push( Stack S, int X )

{

if ( IsFull(S) ) {

return false;

}

else {

S->Data[++(S->Top)] = X;

return true;

}

}

bool IsEmpty( Stack S )

{

return (S->Top == -1);

}

int Pop( Stack S )

{

if ( IsEmpty(S) ) {

return -1;

}

else

return ( S->Data[(S->Top)--] );

}

int main()

{

    Stack z=CreateStack(1000000);

    scanf("%s",s);

    int i,count=0;

    Push(z,-1);

    for(i=0;i<strlen(s);i++)

    {

        if(s[i]=='(')

        Push(z,i);

        else

        {

            Pop(z);

            if(IsEmpty(z))

            {

                Push(z,i);

            }

            else

            {

                number[count++]=i-(z->Data[z->Top]);

            }

        }

    }

        int max=0;

        for(i=0;i<count;i++)

        {

            if(number[i]>max)

            max=number[i];

        }

        int num=0;

        for(i=0;i<count;i++)

        {

            if(number[i]==max)

            num++;

        }

        if(max==0)

        num=1;

        printf("%d\n%d",max,num);

}



#include <cstdio>

#include <stack>

using namespace std;

stack <int> s;

int main()

{

int n,l,r=1000000;

int a[100005];

int post[100005];

scanf ("%d",&n);

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

{

scanf ("%d",&a[i]);

if (a[i]==n)

r=i;

if (i<=r)

s.push(a[i]);

            }

post[n]=a[n];

    for (int i=n-1;i>=1;i--)

    {

        if (a[i]>post[i+1])

            post[i]=a[i];

        else

            post[i]=post[i+1];

    }

printf ("%d",n);

    s.pop();

    r++;

while (r<=n)

{

while ((s.empty()) || ( r<=n && post[r] > s.top() ))

{

    s.push(a[r]);

    r++;

while ( !s.empty() && r<=n && s.top() > post[r] )

{

    printf (" %d",s.top());

    s.pop();

        }

}

while (!s.empty() )

    {

        printf (" %d",s.top());

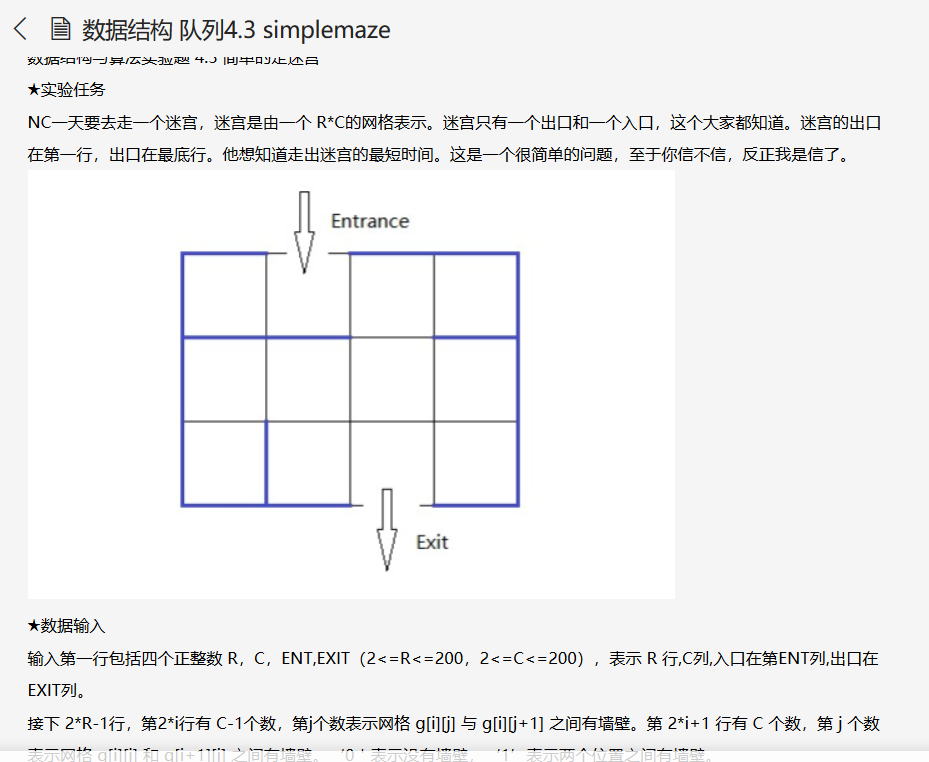
        s.pop();

    }

    return 0;

}

}



#include<iostream>

#include<queue>

using namespace std;

typedef pair<int, int> P;

int map[400][400];

int R, C;

int sx, sy; //起点的位置

int gx, gy; //终点的位置

int d[200][200];

void bfs()

{

    queue<P> que;

    for (int i = 0; i < R; i++)

        for (int j = 0; j < C; j++)

            d[i][j] = 666;  //初始化所有点的距离为666

    que.push(P(sx, sy));

    d[sx][sy] = 0;  //从起点出发将距离设为0，并放入队列首端

    while (que.size())

    {

        P p = que.front(); que.pop();

             //left

            if (0 <= p.first&&p.first<=R-1

             && 0 <= p.second&&p.second<=C-1

                &&map[2\*p.first][p.second-1]!=1

                )

            {

                int nx = p.first ;

                int ny = p.second -1;

                if (d[nx][ny] == 666)

                {

                    que.push(P(nx, ny));

                    d[nx][ny] = d[p.first][p.second] + 1;

                    //cout<<nx<<' '<<ny<<' '<<d[nx][ny]<<endl;

                    if(nx==gx && ny==gy) break;

                }

}

//right

         if (0 <=p.first&&p.first<=R-1

             && 0 <=p.second&&p.second<=C-1

                &&map[2\*p.first][p.second]!=1

                )

            {

                int nx = p.first ;

                int ny = p.second +1;

                if (d[nx][ny] == 666)

                {

                    que.push(P(nx, ny));

                    d[nx][ny] = d[p.first][p.second] + 1;

                    //cout<<nx<<' '<<ny<<' '<<d[nx][ny]<<endl;

                    if(nx==gx && ny==gy) break;

                }

}

            //up

            if (0 <=p.first&&p.first<=R-1

             && 0 <=p.second&&p.second<=C-1

                &&map[2\*p.first-1][p.second]!=1

                )

            {

                int nx = p.first-1;

                int ny = p.second ;

                if (d[nx][ny] == 666)

                {

                    que.push(P(nx, ny));

                    d[nx][ny] = d[p.first][p.second] + 1;

                    //cout<<nx<<' '<<ny<<' '<<d[nx][ny]<<endl;

                    if(nx==gx && ny==gy) break;

                }

}

            //down

            if (0 <= p.first&&p.first<=R-1

             && 0 <= p.second&&p.second<=C-1

                &&map[2\*p.first+1][p.second]!=1

                )

            {

                int nx = p.first +1 ;

                int ny = p.second ;

                if (d[nx][ny] == 666)

                {

                    que.push(P(nx, ny));

                    d[nx][ny] = d[p.first][p.second] + 1;

                    //cout<<nx<<' '<<ny<<' '<<d[nx][ny]<<endl;

                    if(nx==gx && ny==gy) break;

                }

}

    }

}

int main()

{

    cin>>R>>C;

    cin>>sy>>gy;

    sx=0;gy=gy-1;

    sy=sy-1;gx=R-1;

    for(int i=0;i<2\*R-1;i++)

    {

        if(i%2==0)

        {

            for(int j=0;j<C-1;j++)

            cin>>map[i][j];

        }

        else

        {

            for(int j=0;j<C;j++)

            cin>>map[i][j];

        }

    }

    bfs();

    if(d[gx][gy]>=666)

    cout<<"No Way";

    else

    cout<<d[gx][gy]+2;

    return 0;

}



#include<iostream>

#include<string>

#include<queue>

using namespace std;

queue<int> que[2001];

int main()

{

    int a,b;

    int x[2000]={0};

    string type;

    int n=0,i;

    while((cin>>type)&&type!="END")

    {

        cin>>a>>b;

        for(i=0;i<n;i++)

        {

            if(b==x[i])

             {

                que[b].push(a);

                break;

            }

        }

            if(i==n)

            {

                x[n]=b;

                n++;

                que[b].push(a);

            }

    }

    int temp;

    for(i=0;i<n;i++)

    {

        temp=x[i];

        while(!que[temp].empty()){

            cout<<que[temp].front()<<" ";

            que[temp].pop();

        }

    }

    return 0;

}



#include<iostream>

#define debug(x) cout<<#x<<" = "<<x<<"\n"

using namespace std;

long long cnt;

int a[500005],temp[500005],n;

void Merge(int a[],int left,int right,int t[])

{

    int mid=(left+right)/2;

    int p1=left,p2=mid+1,p=left;

    for(int i=left;i<=right;i++) t[i]=a[i];

    while(p1<=mid&&p2<=right)

    {

        if(t[p1]<=t[p2]){

            a[p++]=t[p1++];

        }

        else{

            a[p++]=t[p2++];

            cnt+=(mid-p1+1);

        }

    }

    while(p1<=mid) a[p++]=t[p1++];

    while(p2<=right) a[p++]=t[p2++];

}

void Mergesort(int a[],int left,int right,int t[])

{

    if(left<right){

        int mid=(left+right)/2;

        Mergesort(a,left,mid,t);

        Mergesort(a,mid+1,right,t);

        Merge(a,left,right,t);

    }

}

int main()

{

    ios::sync\_with\_stdio(0);

    cin>>n;

    for(int i=0;i<n;i++) cin>>a[i];

    Mergesort(a,0,n-1,temp);

    cout<<cnt<<endl;

    return 0;

}



#include<bits/stdc++.h>

using namespace std;

int main()

{

    int x1,x2,y1,y2;

    cin>>x1>>y1>>x2>>y2;

    int n;

    cin>>n;

    int c[n][2];

    int i,j;

    for(i=0;i<n;i++)

    {

        cin>>c[i][0]>>c[i][1];

    }

    int a[n],b[n],d[n],e[n],m,x=0,y;

    for(i=0;i<n;i++)

    {

        a[i]=(x1-c[i][0])\*(x1-c[i][0])+(y1-c[i][1])\*(y1-c[i][1]);

        b[i]=(x2-c[i][0])\*(x2-c[i][0])+(y2-c[i][1])\*(y2-c[i][1]);

    }

    for(i=0;i<n;i++)

    {

        for(j=0;j<n;j++)

        {

            d[j]=b[j];

        }

        m=a[i];

        d[i]=-1;

        for(int k=0;k<n;k++)

        {

            if(a[i]>=a[k])

            {

                d[k]=-1;

            }

        }

        for(int k=0;k<n;k++)

        {

            if(d[k]==-1) x++;

        }

        if(x==n) e[i]=a[i];

        else

        {

            sort(d,d+n);

         e[i]=a[i]+d[n-1];

        }

        x=0;

    }

    if(x==1)

    {

        sort(b,b+n);

        cout<<b[n-1];

    }

    else if(x==n)

    {

        sort(a,a+n);

        cout<<a[n-1];

    }

    else

    {

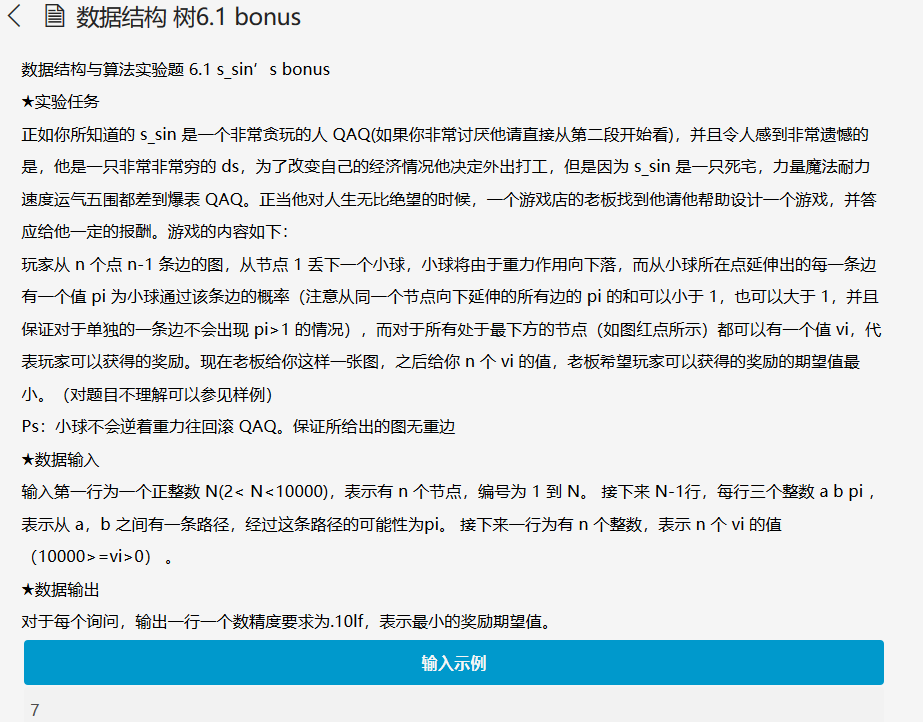
        sort(e,e+n);

    cout<<e[0];

    }

    return 0;

}



#include<stdio.h>

#include<iostream>

#include<algorithm>

#include<string.h>

using namespace std;

struct treenode{

    int num;

    double pi;

}p[10005];

int a[10005];

double b[10005];

int compare1(double a,double b)

{

    return a<b;

}

int compare2(int a,int b)

{

    return a<b;

}

int main(){

    int i = 0,j = 0;

    int n = 0;

    int Fat = 0,Son = 0;

    double pi = 0.0,sum = 0.0;

    cin>>n;

    for(i=0;i<n;i++){

        p[i].num = -1;

    }

    for(i=0;i<n-1;i++){

        cin>>Fat>>Son>>pi;

        p[Fat].num = 1;

        p[Son].num = 0;

        if(i==0){

            p[Fat].pi = 1.0;

        }

        p[Son].pi = pi\*p[Fat].pi;

    }

    for(i=0;i<=n;i++){

        if(p[i].num == 0){

            b[j++] = p[i].pi;

        }

    }

    sort(b,b+j,compare1);

    for(i=0;i<n;i++){

        cin>>a[i];

    }

    sort(a,a+n,compare2);

    i=0;

    j--;

    while(j>=0){

        sum += a[i]\*b[j];

        j--;

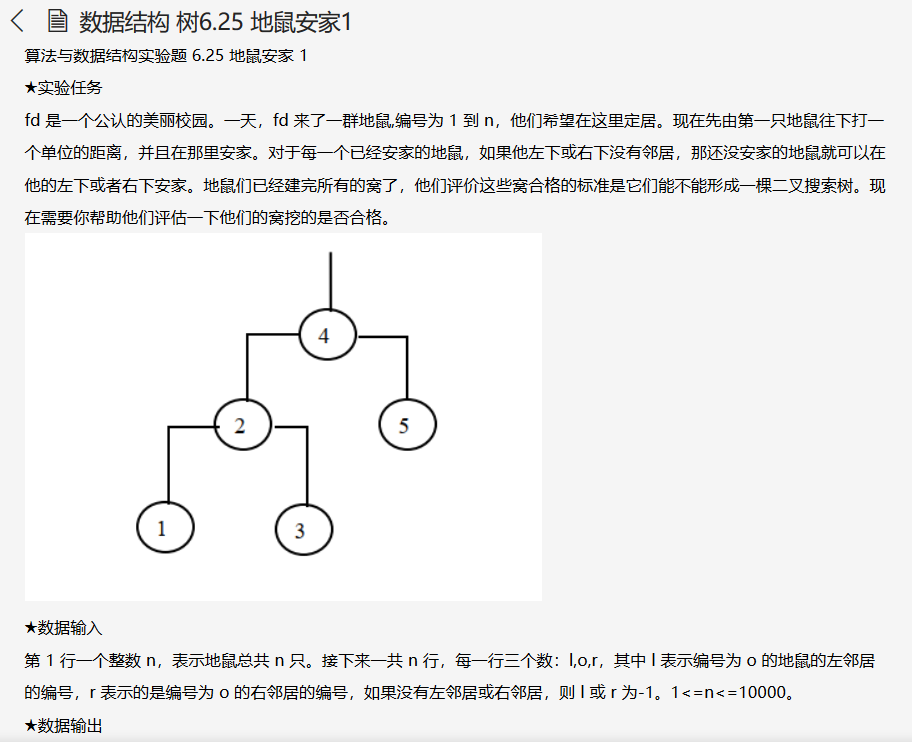
        i++;

    }

    printf("%.10lf",sum);

    return 0;

}



#include<stdio.h>

int p = 0;

int left[10005];

int right[10005];

int indeg[10005];

int a[10005];

void inorder(int x)

{

if (x != -1)

{

inorder(left[x]);

a[p++] = x;

inorder(right[x]);

}

}

int height(int x)

{

if (x == -1)

    {

        return -1;

    }

int lh = height(left[x]);

int rh = height(right[x]);

return lh>rh?++lh:++rh;

}

int main()

{

int i,n,l,r,fa;

scanf("%d",&n);

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

{

    scanf("%d%d%d",&l,&fa,&r);

left[fa] = l;

right[fa] = r;

indeg[l]++;

indeg[r]++;

}

int root;

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

{

if (!indeg[i])

{

root = i;

break;

}

}

inorder(root);

int flag = 0;

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

{

if (a[i] != a[i-1] + 1)

{

flag = 1;

break;

}

}

if (flag){

    printf("-1\n");

    }

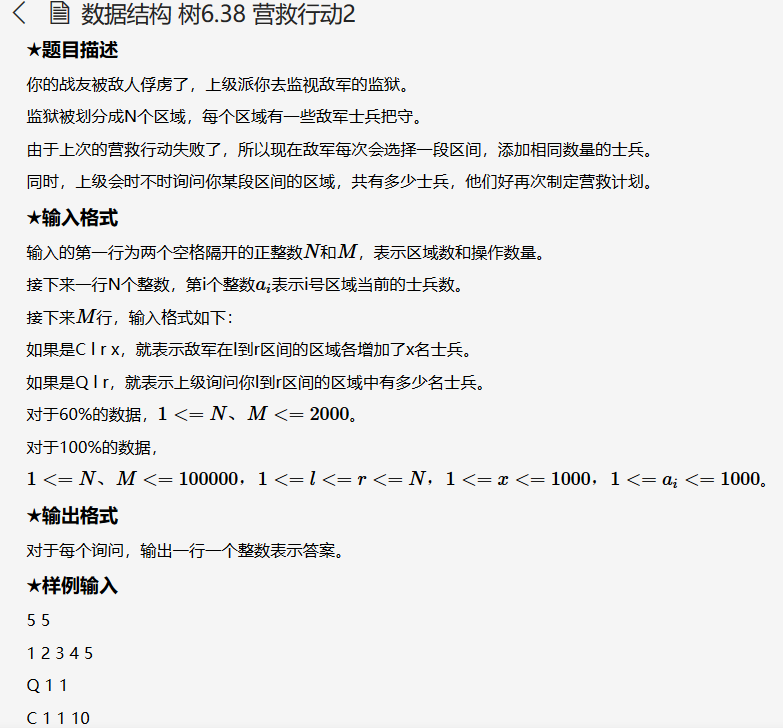
else{

    printf("%d",height(root)+1);

    }

return 0;

}



#include<iostream>

using namespace std;

int a[100005];

int main()

{

    char ch;

    int num,i,j;

    int n,m,l,r,q;

    cin>>n>>m;

    for(i=1;i<n+1;i++){

        cin>>a[i];

    }

    i=0;

    cin>>ch;

    while(i<m){

        num=0;

        if(ch=='Q'){

            cin>>l>>r;

            for(j=l;j<=r;j++){

                num+=a[j];

            }

            cout<<num<<endl;

        }

        else {

            cin>>l>>r>>q;

            for(j=l;j<=r;j++){

                a[j]+=q;

            }

        }

        if(i!=m-1)

         cin>>ch;

        i++;

    }

    return 0;

}



#include<stdio.h>

#include<iostream>

#include<string.h>

using namespace std;

int trie[1000000][26]={{0}};

int color[1000000]={0};

int k=1;

void insert(char \*s)

{

    int p=0;

    for(int i=0;i<strlen(s);i++)

    {

        int c=s[i]-'a';

        if(trie[p][c]==0)

        trie[p][c]=k++;

        p=trie[p][c];

        color[p]++;

    }

}

int match(char \*s)

{

    int p=0;

    for(int i=0;i<strlen(s);i++)

    {

        int c=s[i]-'a';

        if(trie[p][c]==0)

        return 0;

        p=trie[p][c];

    }

    return color[p];

}

int main()

{

    int n,m,i;

    char s[32];

    cin>>n;

    for(i=0;i<n;i++)

    {

        scanf("%s",s);

        insert(s);

    }

    cin>>m;

    for(i=0;i<m;i++)

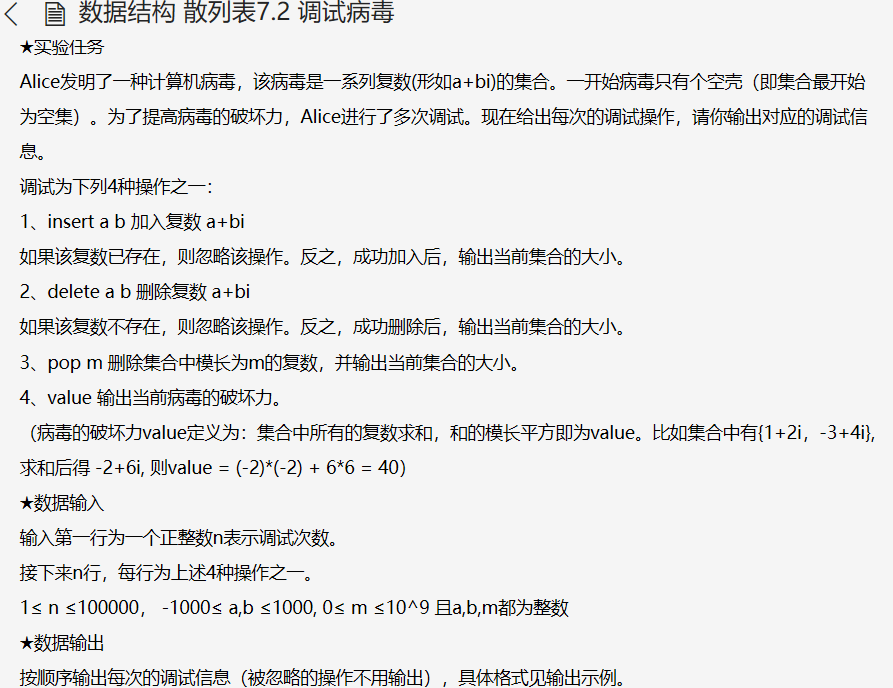
    {

        scanf("%s",s);

        cout<<match(s)<<endl;

    }

}



#include<stdio.h>

#include<stdlib.h>

#include<memory.h>

typedef struct dead death;

struct dead

{

short a,b;

death \*next;

};

death mod[2000000];

int main()

{

char sign[8];

short a,b;

int n,square;

death \*poi;

memset(mod,0,sizeof(death\*)\*2000000);

int afraid,bewail,total;

afraid=bewail=total=0;

scanf("%d",&n);

while(n--)

{

scanf("%s",sign);

if(sign[0]=='i')

{

scanf("%hd%hd",&a,&b);

poi=mod+a\*a+b\*b;

while(1)

{

if(poi->next)

{

poi=poi->next;

if(poi->a==a&&poi->b==b)break;

}

else

{

poi=poi->next=(death\*)malloc(sizeof(death));

poi->a=a;

poi->b=b;

poi->next=0;

afraid+=a;

bewail+=b;

printf("size = %d\n",++total);

break;

}

}

}

else if(sign[0]=='d')

{

scanf("%hd%hd",&a,&b);

poi=mod+a\*a+b\*b;

while(1)

{

if(poi->next)

{

if(poi->next->a==a&&poi->next->b==b)

{

death \*temp=poi->next;

poi->next=temp->next;

free(temp);

afraid-=a;

bewail-=b;

printf("size = %d\n",--total);

break;

}

poi=poi->next;

}

else break;

}

}

else if(sign[0]=='p')

{

scanf("%d",&square);

if(square<14142)

{

square\*=square;

poi=mod[square].next;

while(poi)

{

death \*temp=poi->next;

afraid-=poi->a;

bewail-=poi->b;

free(poi);

total--;

poi=temp;

}

mod[square].next=0;

printf("size = %d\n",total);

}

else printf("size = %d\n",total);

}

else if(sign[0]=='v')printf("value = %d\n",afraid\*afraid+bewail\*bewail);

}

return 0;

}



#include<iostream>

#include<algorithm>

#include<queue>

using namespace std;

struct quest

{

    long int start;

    long int time;

}a[200005];

priority\_queue<int, vector<int>, greater<int> > que;

int compare(quest x,quest y)

{

    return x.start<y.start;

}

long long int sum=0;

int main()

{

    int n,count=1;

    cin>>n;

    for(int i=0;i<n;i++)

    {

        cin>>a[i].start>>a[i].time;

    }

    sort(a,a+n,compare);

    long int now=a[0].start;

    que.push(a[0].time);

    while(count<n)

    {

        long int tmp=que.top();

        que.pop();

        if(now+tmp<=a[count].start)

        {

            now+=tmp;

            sum+=now;

        }

        else

        {

            tmp-=a[count].start-now;

            now=a[count].start;

            que.push(tmp);

        }

        while(a[count].start==now)

        {

            que.push(a[count].time);

            count++;

        }

        if(que.empty())

        {

            now=a[count].start;

            que.push(a[count].time);

            count++;

        }

    }

    while(!que.empty())

    {

        now+=que.top();

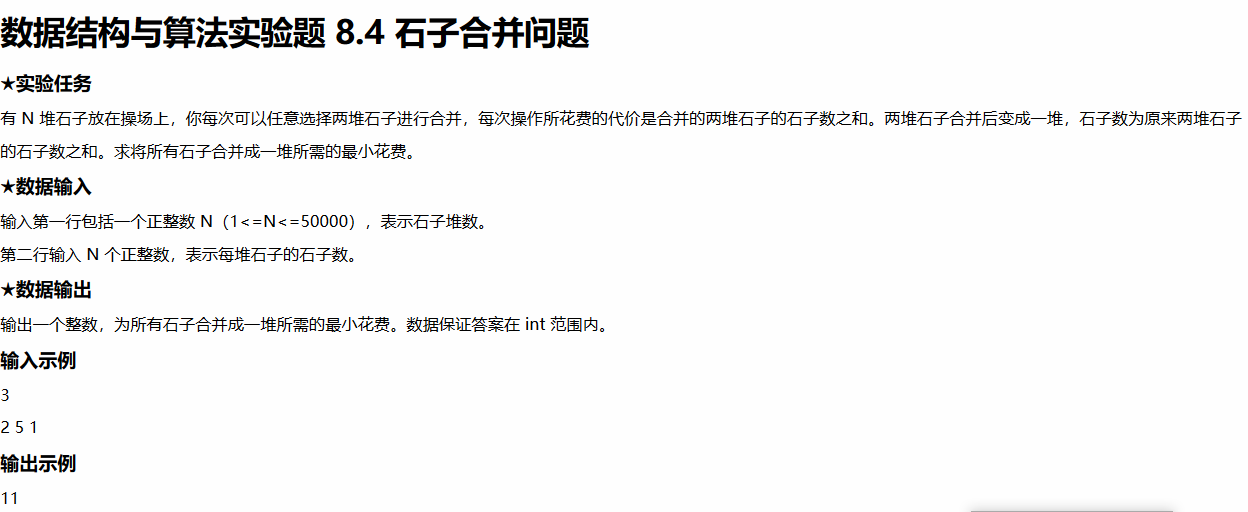
        que.pop();

        sum+=now;

    }

    cout<<sum;

}



#include<iostream>

#include<vector>

#include<algorithm>

using namespace std;

int compare(int x,int y)

{

    return x>y;

}

int main()

{

        int n;

        cin>>n;

        vector<int> v;

        while(n--){

            int x;

            cin>>x;

            v.push\_back(x);

        }

        long long sum=0;

        make\_heap(v.begin(),v.end(),compare);

        while(v.size()>1)

        {

            int min1=v.front();

            pop\_heap(v.begin(),v.end(),compare);

            v.pop\_back();

            int min2=v.front();

            pop\_heap(v.begin(),v.end(),compare);

            v.pop\_back();

            sum+=(min1+min2);

            v.push\_back(min1+min2);

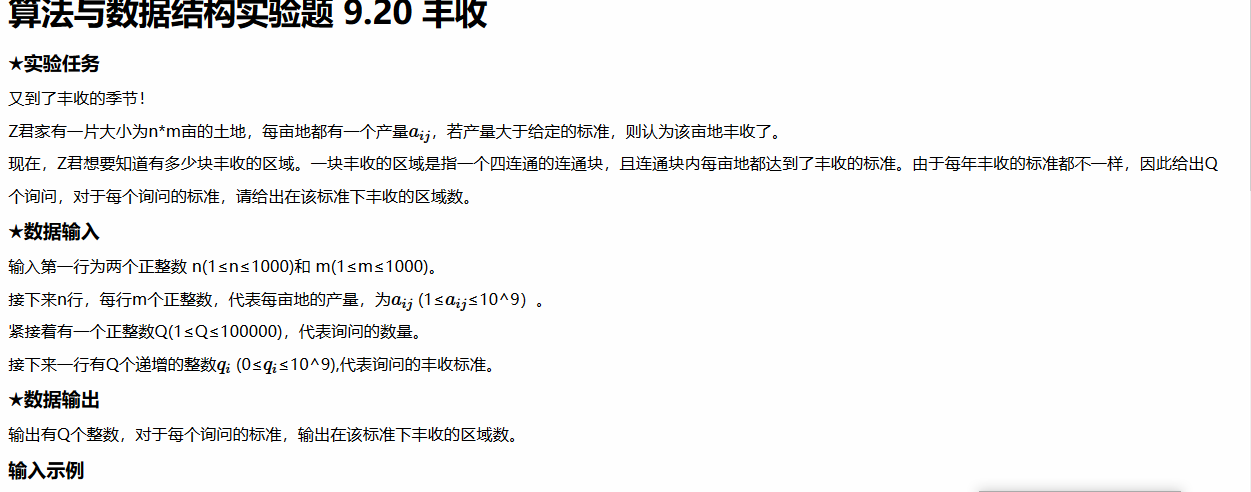
            push\_heap(v.begin(),v.end(),compare);

        }

        cout<<sum;

    return 0;

}



#include<bits/stdc++.h>

using namespace std;

const int maxn = 1005;

const int maxq = 1e5+1;

int q[maxq];

int pa[maxn\*maxn];

int Find(int x){

    return x==pa[x]?x:pa[x]=Find(pa[x]);

}

int h[maxn][maxn];

struct Node

{

int x;

int y;

int val;

}nd[maxn\*maxn];

bool operator < (const Node &x,const Node &y){

    return x.val < y.val;

}

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

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

int main()

{

int m,n;

    scanf("%d%d",&m,&n);

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

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

scanf("%d",h[i]+j);

int t=i\*n+j;

nd[t].x=i;

            nd[t].y=j;

nd[t].val=h[i][j];

}

}

sort(nd,nd+m\*n);

memset(pa,-1,sizeof(int)\*(m\*n));

int Ts;

    scanf("%d",&Ts);

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

scanf("%d",q+i);

}

int k=m\*n-1;

int ans=0;

for(int i=Ts-1;i>=0;i--){

if(q[i]<nd[k].val){

while(k>=0&&q[i]<nd[k].val){

int id=nd[k].x\*n+nd[k].y;

if(!~pa[id]){

    ans++;

    pa[id]=id;

                }

for(int d=0;d<4;d++){

int nx=nd[k].x+dx[d];

                    int ny=nd[k].y+dy[d];

if(nx>=0&&nx<m&&ny>=0&&ny<n&&h[nx][ny]>q[i]){

int nid=nx\*n+ny;

if(~pa[nid]){

int a=Find(nid);

                            int b=Find(id);

if(a!=b){

pa[a]=b;

ans--;

}

}

}

}

k--;

}

if(k<0){

for(;i>=0;i--){

q[i]=ans;

}

break;

}

}

q[i]=ans;

}

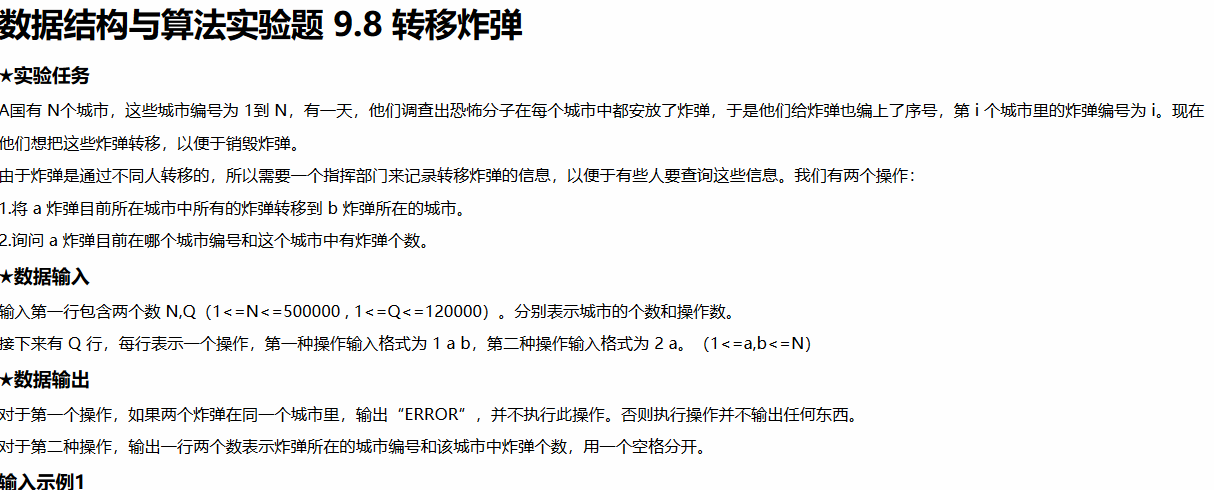
for(int i=0;i<Ts;i++){

    printf("%d\n",q[i]);

    }

return 0;

}



#include<iostream>

using namespace std;

int boom[500005];

int city[500005];

int find(int x)

{

if(boom[x] == x)

return x;

else

return find(boom[x]);

}

int sufind(int x) {

if (x != boom[x])

boom[x] = sufind(boom[x]);

return boom[x];

}

bool same(int x,int y)

{

return sufind(x) == sufind(y);

}

int main()

{

int N=0,Q=0;

cin>>N>>Q;

int i = 0;

int j = 0;

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

boom[i] = i;

city[i] = 1;

}

for(i=0;i<Q;i++){

int flag = 0;

cin>>flag;

if(flag==1){

int a=0,b=0;

cin>>a>>b;

int acity = sufind(a);

int bcity = sufind(b);

if(same(a,b)){

cout<<"ERROR"<<endl;

}

else{

for(j=1;j<=N;j++){

if(boom[j]==acity){

boom[j]=bcity;

}

}

city[bcity] += city[acity];

city[acity] = 0;

}

}

else{

int a=0;

cin>>a;

int b=0;

b=sufind(a);

cout<<b<<" "<<city[b]<<endl;

}

}

return 0;

}



#include<iostream>

#include<string>

using namespace std;

#define MAXVEX 1003

#define INFINITY 65535

int jteam[1000];

int map[1000][1000];

int aaa[500000];

void toposort(int N,int M){

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

for(i=1;i<=N;i++){

for(j=1;j<=N;j++){

if(!jteam[j]){

jteam[j]--;

cout<<j;

if(i!=N)

cout<<" ";

else

cout<<endl;

for(k=1;k<=N;k++){

if(map[j][k])

jteam[k]--;

}

break;

}

}

}

}

int main()

{

int N=0,M=0;

cin>>N>>M;

int x=0,y=0;

int i=0,j=0;

for(i=0;i<1000;i++){

for(j=0;j<1000;j++){

map[i][j] = 0;

}

}

for(i=0;i<1000;i++){

jteam[i]=0;

}

for(i=1;i<=M;i++){

cin>>x>>y;

if(!map[x][y]){

map[x][y]=1;

jteam[y]++;

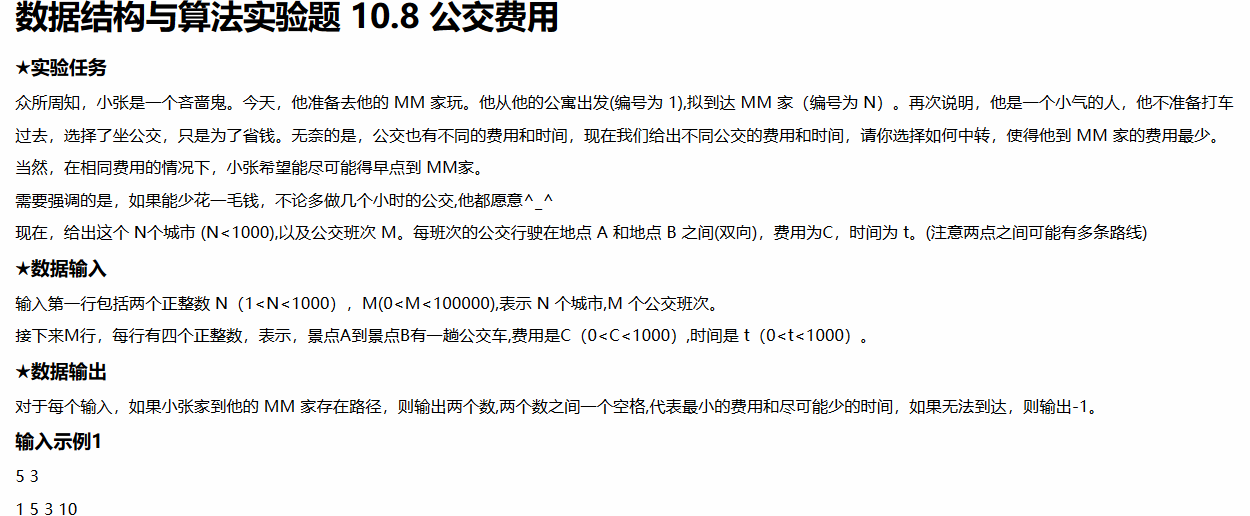
}

}

toposort(N,M);

return 0;

}



#include<stdio.h>

#include<string.h>

#include<math.h>

#include<iostream>

#include<string>

#include<queue>

#include<stack>

#include<map>

#include<set>

#include<algorithm>

#include<unordered\_map>

#include<unordered\_set>

#define PI acos(-1)

#define pb push\_back

#define INF 0x3f3f3f3f

#define dd(x) cout<<#x<<" = "<<x<<","

#define de(x) cout<<#x<<" = "<<x<<'\n'

using namespace std;

typedef long long ll;

typedef pair<int,int> pii;

const int N=1e3+5;

const int M=1e5+5;

struct edge{

int to,c,t,next;

};

edge e[2\*M];

int head[N],dis[N],cost[N],vis[N],cnt;

void add(int from,int to,int c,int t){

cnt++;

e[cnt].to=to;

e[cnt].c=c;

e[cnt].t=t;

e[cnt].next=head[from];

head[from]=cnt;

}

int bbb[1000000];

int main()

{

int n,m;

scanf("%d%d",&n,&m);

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

int from,to,c,t;

scanf("%d%d%d%d",&from,&to,&c,&t);

add(from,to,c,t);

add(to,from,c,t);

}

memset(dis,0x3f,sizeof(dis));

memset(cost,0x3f,sizeof(cost));

dis[1]=cost[1]=0;

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

int mn=INF,pos=0;

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

if(cost[i]<mn&&!vis[i]){

mn=cost[i];

pos=i;

}

}

vis[pos]=1;

for(int i=head[pos];i>0;i=e[i].next){

if(cost[pos]+e[i].c<cost[e[i].to]){

cost[e[i].to]=cost[pos]+e[i].c;

dis[e[i].to]=dis[pos]+e[i].t;

}

else if(cost[pos]+e[i].c==cost[e[i].to]){

dis[e[i].to]=min(dis[e[i].to],dis[pos]+e[i].t);

}

}

}

if(cost[n]!=INF&&dis[n]!=INF)

printf("%d %d",cost[n],dis[n]);

else

printf("-1");

}