**Combinatorics:**

\*\*\*\*Combination 10^6.cpp\*\*\*

#include<bits/stdc++.h>

#define pb push\_back

#define ms(a,b) memset((a),(b),sizeof(a))

//#define clear(v,n)

for(\_typeof (n) i=0;i< (n) ; i++)  { v.clear(); }

#define ll long long

#define pii pair<ll,ll>

#define inf 100000000000

#define in(a) freopen(a,"r", stdin)

#define out(a) freopen(a,"w",stdout)

using namespace std;

vector<int>adj[1009];

ll dp[1009][1009], ara[1009], fact[1000009];

ll mod=1000000007;

ll pow1(int x, int n)

{

    if(n==0) return 1;

    if(n%2==0)

    {

        ll ret= pow1(x,n/2);

        return (ret\*ret)%mod;

    }

    return (x\* pow1(x,n-1))%mod;

}

ll nCr(int n, int r)

{

    if(r==0 || n==r) return 1;

    if(r==1) return n;

   ll num= fact[n];

   ll den=( fact[n-r] \* fact[r] )%mod;

   ll res= (num \* pow1(den, mod-2)   )%mod;

    return res;

}

ll f1(int n, int k)

{

    return nCr(n+k-1, k-1);  // or nCr(n+k-1,n);

}

int main()

{

    fact[0]=1;

    for(ll i=1;i<=1000009;i++)

    {

        fact[i]= (fact[i-1]\*i)%mod;

    }

  //  cout<<fact[50000]<<endl;

    ms(dp,-1);

    for(int i=1; i<=1006; i++)

    {

        for(int j=1; j<=i; j++)

            dp[i][j]= nCr(i,j);

    }

    return 0;

}

\*\*\*n th permutation of a string\*\*\*\*

// C++ program to print nth permutation with

// using next\_permute()

#include <bits/stdc++.h>

using namespace std;

// Function to print nth permutation

// using next\_permute()

void nPermute(string str, long int n)

{

    // Sort the string in lexicographically

    // ascending order

    sort(str.begin(), str.end());

    // Keep iterating until

    // we reach nth position

    long int i = 1;

    do {

        // check for nth iteration

        if (i == n)

            break;

        i++;

    } while (next\_permutation(str.begin(), str.end()));

    // print string after nth iteration

    cout << str;

}

// Driver code

int main()

{

    string str = "01234";

    long int n ;

    cin>>n;

    nPermute(str, n);

    return 0;

}

\*\*\*\*\*order of permutations and sum.cpp\*\*\*\*\*

///https://www.quora.com/How-do-you-do-addition-with-factorials

#include<bits/stdc++.h>

using namespace std;

int ara1[200009],ara2[200009], tree[4\*200009], sum[200009];

void build(int node, int beg, int endd)

{

    if(beg==endd)

    {

        tree[node]=1;

        return ;

    }

    int mid= (beg+endd)/2;

    int left= node\*2;

    int right= node\*2 +1;

    build(left, beg, mid);

    build(right, mid+1, endd);

    tree[node]= tree[left]+ tree[right];

}

int query1(int node, int beg, int endd, int x, int y)

{

    if(beg>y || endd<x) return 0;

    if(beg>=x && endd<=y)

        return tree[node];

    int mid= (beg+endd)/2;

    int left= node\*2;

    int right= node\*2 +1;

    int t1=query1(left, beg, mid, x,y);

    int t2=query1(right, mid+1, endd, x,y);

    return t1+t2;

}

int query2(int node, int beg, int endd, int x)

{

    if(beg==endd && x==tree[node])

    {

        tree[node]=0;

        return beg;

    }

    int mid= (beg+endd)/2;

    int left= node\*2;

    int right= node\*2 +1;

    int t;

    if(x>tree[left])

        t= query2(right, mid+1, endd, x-tree[left]);

    else t= query2(left, beg, mid, x);

    tree[node]=tree[left]+tree[right];

    return t;

}

void update(int node, int beg, int endd, int x)

{

    if(beg==endd && beg==x)

    {

        tree[node]=0;

        return;

    }

    int mid= (beg+endd)/2;

    int left= node\*2;

    int right= node\*2 +1;

    if(x<=mid)

        update(left, beg, mid, x);

    else update(right, mid+1, endd, x);

    tree[node]= tree[left]+tree[right];

}

int main()

{

    int n;

    scanf("%d",&n);

    for(int i=0; i<n; i++)

        scanf("%d",&ara1[i]);

    for(int i=0; i<n; i++)

        scanf("%d",&ara2[i]);

    build(1, 0, n-1);

    for(int i=0; i<n; i++)

    {

        int ret= query1(1, 0, n-1, 0, ara1[i])-1;

        update(1,0,n-1, ara1[i]);

        sum[i]+=ret;

    }

    build(1,0,n-1);

    for(int i=0; i<n; i++)

    {

        int ret= query1(1, 0, n-1, 0, ara2[i])-1;

        update(1,0,n-1, ara2[i]);

        sum[i]+=ret;

    }

    for(int i=n-1; i>0; i--)

    {

        int yy= n-i;

        int div= sum[i]/(yy);

        sum[i]%=yy;

        sum[i-1]+=div;

    }

    sum[0]= sum[0]%(n);

    build(1,0,n-1);

    vector<int>vec;

    for(int i=0; i<n; i++)

    {

        int ret= query2(1,0,n-1,sum[i]+1);

        vec.push\_back(ret);

    }

    for(int i=0;i<vec.size();i++)

        printf("%d ",vec[i]);

    puts("");

    return 0;

}

# Data Structure:

# \*\*\*\*2d segment tree.cpp\*\*\*\*\*

///This a implementation of 2d segment tree. I am trying to solve a problem ,so if i learn this by myself ,it may or may not

///help me in any contest :p  but implementation by myself is also like solving a problem.

#include<bits/stdc++.h>

#define pb push\_back

#define ms(a,b) memset((a),(b),sizeof(a))

#define i64 long long

#define pii pair<i64,i64>

#define INF 100000000000

#define in(a) freopen(a,"r", stdin)

#define out(a) freopen(a,"w",stdout)

#define rep(i,n) for(i64 i=0;i<n;i++)

using namespace std;

int T[2\*250009],ara[505][505];

struct point

{

    int r,c;

};

void build(int nd, point b, point e)

{

    if(b.r==e.r && b.c==e.c)

    {

        T[nd]=ara[b.r][b.c];

        return;

    }

    int ch1,ch2,ch3,ch4;

    ch1=nd\*4+1;

    ch2=nd\*4+2;

    ch3=nd\*4+3;

    ch4=nd\*4+4;

    point mid1b,mid1e,mid2b, mid2e,mid3b,mid3e, mid4b, mid4e;

    mid1b=b;

    mid1e= {(b.r+e.r)/2, (b.c+e.c)/2};

    mid2b= {b.r,  min((b.c+e.c)/2 +1,e.c) };

    mid2e= {(b.r+e.r)/2, e.c};

    mid3b= {min((b.r+e.r)/2 +1,e.r) , b.c};

    mid3e= {e.r,(b.c+e.c)/2 };

    mid4b= {min((b.r+e.r)/2 +1,e.r), min((b.c+e.c)/2+1,e.c)};

    mid4e=e;

    build(ch1, mid1b, mid1e);

    build(ch2, mid2b, mid2e);

    build(ch3, mid3b, mid3e);

    build(ch4, mid4b, mid4e);

    int r1=max(T[ch1],T[ch2]);

    int r2=max(T[ch3],T[ch4]);

    T[nd]=max(r1,r2);

}

bool check\_outside(point b, point e, point st, point en)

{

    if(e.r<st.r || b.r>en.r) return true;   /// up or down of my square

    if(e.c<st.c || b.c>en.c) return true;   /// left or right of my square

    return false;

}

bool check\_inside(point b, point e, point st, point en)

{

    if(b.r>=st.r && b.c>=st.c && e.r<=en.r && e.c<=en.c) return true;  ///Inside my square

    return false;

}

int query(int nd, point b, point e, point st, point en)

{

    if(check\_outside(b,e,st,en)==true) return 0;

    if(check\_inside(b,e,st,en)==true)

    {

         return T[nd];

    }

    int ch1,ch2,ch3,ch4;

    ch1=nd\*4+1;

    ch2=nd\*4+2;

    ch3=nd\*4+3;

    ch4=nd\*4+4;

    point mid1b,mid1e,mid2b, mid2e,mid3b,mid3e, mid4b, mid4e;

    mid1b=b;

    mid1e= {(b.r+e.r)/2, (b.c+e.c)/2};

    mid2b= {b.r,  min((b.c+e.c)/2 +1,e.c) };

    mid2e= {(b.r+e.r)/2, e.c};

    mid3b= {min((b.r+e.r)/2 +1,e.r) , b.c};

    mid3e= {e.r,(b.c+e.c)/2 };

    mid4b= {min((b.r+e.r)/2 +1,e.r), min((b.c+e.c)/2+1,e.c)};

    mid4e=e;

    int t1,t2,t3,t4;

    t1=t2=t3=t4=0;

    t1=query(ch1, mid1b, mid1e, st,en);

    t2=query(ch2, mid2b, mid2e, st,en);

    t3=query(ch3, mid3b, mid3e, st,en);

    t4=query(ch4, mid4b, mid4e, st,en);

    int r1=max(t1,t2);

    int r2=max(t3,t4);

    return max(r1,r2);

}

int main()

{

    int tt=0,test;

    scanf("%d",&test);

    while(tt<test)

    {

        printf("Case %d:\n",++tt);

        int n,q;

        scanf("%d %d",&n,&q);

        for(int i=0; i<n; i++)

        {

            for(int j=0; j<n; j++)

            {

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

            }

        }

        build(0, {0,0}, {n-1,n-1});

        while(q--)

        {

            int I,J,S;

            scanf("%d %d %d",&I,&J,&S);

            I--,J--;

            printf("%d\n",query(0, {0,0}, {n-1,n-1}, {I,J}, {I+S-1,J+S-1} ));

        }

        ms(T,0);

        ms(ara,0);

    }

    return 0;

}

**\*\*\*\*BITset\*\*\*\*\*\***

// C++ program to demonstrate various functionality of bitset

#include <bits/stdc++.h>

using namespace std;

#define M 32

int main()

{

    // default constructor initializes with all bits 0

    bitset<M> bset1;

    // bset2 is initialized with bits of 20

    bitset<M> bset2(20);

    // bset3 is initialized with bits of specified binary string

    bitset<M> bset3(string("1100"));

    // cout prints exact bits representation of bitset

    cout << bset1 << endl;  // 00000000000000000000000000000000

    cout << bset2 << endl;  // 00000000000000000000000000010100

    cout << bset3 << endl;  // 00000000000000000000000000001100

    cout << endl;

    // declaring set8 with capacity of 8 bits

    bitset<8> set8;    // 00000000

    // setting first bit (or 6th index)

    set8[1] = 1;    // 00000010

    set8[4] = set8[1];   //  00010010

    cout << set8 << endl;

    // count function returns number of set bits in bitset

    int numberof1 = set8.count();

    // size function returns total number of bits in bitset

    // so there difference will give us number of unset(0)

    // bits in bitset

    int numberof0 = set8.size() - numberof1;

    cout << set8 << " has " << numberof1 << " ones and "

         << numberof0 << " zeros\n";

    // test function return 1 if bit is set else returns 0

    cout << "bool representation of " << set8 << " : ";

    for (int i = 0; i < set8.size(); i++)

        cout << set8.test(i) << " ";

    cout << endl;

    // any function returns true, if atleast 1 bit

    // is set

    if (!set8.any())

        cout << "set8 has no bit set.\n";

    if (!bset1.any())

        cout << "bset1 has no bit set.\n";

    // none function returns true, if none of the bit

    // is set

    if (!bset1.none())

        cout << "bset1 has all bit set\n";

    // bset.set() sets all bits

    cout << set8.set() << endl;

    //  bset.set(pos, b) makes bset[pos] = b

    cout << set8.set(4, 0) << endl;

    // bset.set(pos) makes bset[pos] = 1  i.e. default

    // is 1

    cout << set8.set(4) << endl;

    // reset function makes all bits 0

    cout << set8.reset(2) << endl;

    cout << set8.reset() << endl;

    // flip function flips all bits i.e.  1 <-> 0

    // and  0 <-> 1

    cout << set8.flip(2) << endl;

    cout << set8.flip() << endl;

    // Converting decimal number to binary by using bitset

    int num = 100;

    cout  << "\nDecimal number: " << num

         << "  Binary equivalent: " << bitset<8>(num);

    return 0;

}

\*\*\*\*LCA\*\*\*\*

#include<bits/stdc++.h>

#define pb push\_back

#define ms(a,b) memset((a),(b),sizeof(a))

//#define clear(v,n) for(\_typeof (n) i=0;i< (n) ; i++)  { v.clear(); }

#define i64 long long

#define pii pair<i64,i64>

#define inf 100000000000

#define in(a) freopen(a,"r", stdin)

#define out(a) freopen(a,"w",stdout)

#define rep(i,n) for(i64 i=0;i<n;i++)

using namespace std;

vector<int>adj[30009];

const int maxL=18;  ///log2(2\*1e5)

int Level[30009], par[30009][maxL+5];

void dfs(int src, int pr)

{

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

    {

        int node=adj[src][i];

        if(node!=pr)

        {

            Level[node]=Level[src]+1;

            par[node][0]=src;

            dfs(node,src);

        }

    }

}

void precomputeSparse(int N)

{

    for(int i=1; i<=maxL; i++)

    {

        for(int u=0; u<N; u++)

        {

            if(par[u][i-1]!=-1)

                {par[u][i]= par[ par[u][i-1] ][i-1];

               // printf("u=%d %dth= par=%d\n",u,i,par[u][i]);

                }

        }

    }

}

int getLca(int u, int v)

{

    if(Level[u]<Level[v])

        swap(u,v);

    int dif=Level[u]-Level[v];

    for(int i=maxL; i>=0; i--)

    {

        int d=1<<i;

        if(dif>=d)

        {

            dif-=d;

            u=par[u][i];

        }

    }

    if(u==v)

        return u;

    for(int i=maxL; i>=0; i--)

    {

      //  printf("i=%d %d->%d , %d->%d\n",i,u,par[u][i],v,par[v][i]);

        if(par[u][i]!=par[v][i])

            u=par[u][i], v=par[v][i];

    }

     return par[u][0];

}

int main()

{

    int n,u,v;

    scanf("%d",&n);

    for(int i=0; i<n-1; i++)

    {

        scanf("%d %d",&u,&v);

        adj[u].push\_back(v);

        adj[v].push\_back(u);

    }

    ms(par,-1);

    dfs(0,-1);   /// 1 no step

    precomputeSparse(n); /// 2 no step

    while(1){

    cin>>u>>v;

    cout<<getLca(u,v)<<endl;  /// 3 no step

    }

    return 0;

}

\*\*\*\*Mo’s Algorithm\*\*\*\*

#include<bits/stdc++.h>

#define pb push\_back

#define ms(a,b) memset((a),(b),sizeof(a))

#define loop(i,n) for(int i=0;i<n;i++)

#define ll long long

#define pii pair<int,int>

#define inf 100000000000

#define in(a) freopen(a,"r", stdin)

#define out(a) freopen(a,"w",stdout)

using namespace std;

int b\_size;

struct query

{

    int i,l,r;

} Q[200009];

ll answer[200009], sum=0;

int cntAra[1000009],  ara[200009];

bool cmp(query a, query b)

{

    if(a.l/b\_size != b.l/b\_size)

        return a.l/b\_size < b.l/b\_size;

    return a.r<b.r;

}

inline void Add(int pos)

{

    ll num= ara[pos];

    sum-= cntAra[num]\*(cntAra[num] \* num);

    cntAra[num]++;

    sum+= cntAra[num]\*(cntAra[num] \* num);

}

inline void Remove(int pos)

{

    ll num= ara[pos];

    sum-= cntAra[num]\*(cntAra[num] \* num);

    cntAra[num]--;

    sum+= cntAra[num]\*(cntAra[num] \* num);

}

int main()

{

    int  n,t;

    scanf("%d %d",&n,&t);

    for(int i=0; i<n; i++)

        scanf("%d",&ara[i]);

    b\_size= sqrt(n);

    for(int i=0; i<t; i++)

    {

        scanf("%d %d",&Q[i].l, &Q[i].r);

        Q[i].i=i;

    }

    sort(Q,Q+t,cmp);

    int curL=0, curR=-1;

    for(int i=0; i<t; i++)

    {

        int L= Q[i].l-1, R= Q[i].r-1;

        while(curR<R)

        {

            curR++;

            Add(curR);

        }

        while(curR>R)

        {

            Remove(curR);

            curR--;

        }

        while(curL<L)

        {

            Remove(curL);

            curL++;

        }

        while(curL>L)

        {

            curL--;

            Add(curL);

        }

        answer[Q[i].i]=sum;

    }

    for(int i=0; i<t; i++)

        printf("%lld\n",answer[i]);

    return 0;

}

\*\*\*\*nap sack with bitset\*\*\*\*

///https://agc020.contest.atcoder.jp/tasks/agc020\_c?lang=en

///http://petr-mitrichev.blogspot.com/2018/01/

#include<bits/stdc++.h>

#define i64 long long

#define inf 1000000000000000000

using namespace std;

int ara[2001];

bitset<2001\*2001>bset;//[2];

int main()

{

    int n;

    scanf("%d",&n);

    int sum=0,mx=0;

    for(int i=1; i<=n; i++)

        scanf("%d",&ara[i]);

  //  sort(ara+1,ara+n+1);

    for(int i=1; i<=n; i++)

    {

        sum+=ara[i];

        mx=max(mx,ara[i]);

    }

    int avg=(sum+2-1)/2, last=mx\*n;

    bset[0]=1;

    for(int i=1; i<=n; i++)

        bset= bset | (bset<<ara[i]);

    for(int i=avg; i<=last; i++)

    {

        if(bset[i]==1)

        {

            printf("%d\n",i);

            return 0;

        }

    }

    return 0;

}

\*\*\*\*PBDS\*\*\*\*

///using pbds  each operation in logn and this is using structure for any number of keys

#include<bits/stdc++.h>

///These are for pbds

#include<ext/pb\_ds/assoc\_container.hpp>

#include<ext/pb\_ds/tree\_policy.hpp>

#include<ext/pb\_ds/detail/standard\_policies.hpp>

#define pb push\_back

#define ms(a,b) memset((a),(b),sizeof(a))

//#define clear(v,n) for(\_typeof (n) i=0;i< (n) ; i++)  { v.clear(); }

#define ll long long

#define pii pair<int,int>

#define inf 100000000000

#define in(a) freopen(a,"r", stdin)

#define out(a) freopen(a,"w",stdout)

#define rep(i,n) for(int i=0;i<n;i++)

#define MP(x,y) make\_pair(x,y)

using namespace std;

using namespace \_\_gnu\_pbds;

struct info

{

    int solved,penalty,efficiency;

};

typedef pair<info,int>PI;

//

typedef tree<PI,null\_type, less<PI>, rb\_tree\_tag,

tree\_order\_statistics\_node\_update> set\_t;

//

const int mx=1e5+10;

int ps[mx],pp[mx],pe[mx];

//

inline bool operator<(const info& lhs, const info& rhs)

{

    if(lhs.solved==rhs.solved)

    {

        if(lhs.penalty==rhs.penalty)

            return lhs.efficiency>rhs.efficiency;

        return lhs.penalty<rhs.penalty;

    }

    return lhs.solved>rhs.solved;

}

int main()

{

    set\_t s;

    int n,m,t,p,e;

    cin>>n>>m;

    for(int i=0; i<n; i++)

    {

        info a;

        a= {0,0,0};

        s.insert(MP(a,i+1));

    }

    rep(i,m)

    {

        cin>>t>>p>>e;

        info a;

        a= {ps[t],pp[t], pe[t]};

        s.erase(MP(a,t));

        ps[t]++;

        pp[t]+=p;

        pe[t]+=e;

        a= {ps[t],pp[t], pe[t]};

        s.insert(MP(a,t));

        a= {ps[1],pp[1],pe[1]};

        cout<<s.order\_of\_key(MP(a,1))+1<<endl;

    }

//    set\_t ::iterator it;

//    for(it=s.begin(); it!=s.end();it++)

//    {

//        info a= it->first;

//        int per=it->second;

//        cout<<a.solved<<" "<<a.penalty<<" "<<a.efficiency<<" "<<per<<endl;

//    }

    return 0;

}

\*\*\*\*Segment Tree with new technique\*\*\*\*

#include<bits/stdc++.h>

///http://codeforces.com/problemset/problem/914/D

/// here instead of query and decide which child to go,, first go to that range then use the lc, rc val to decide

#define pb push\_back

#define ms(a,b) memset((a),(b),sizeof(a))

//#define clear(v,n) for(\_typeof (n) i=0;i< (n) ; i++)  { v.clear(); }

#define i64 long long

#define pii pair<i64,i64>

#define inf 100000000000

#define in(a) freopen(a,"r", stdin)

#define out(a) freopen(a,"w",stdout)

#define rep(i,n) for(i64 i=0;i<n;i++)

using namespace std;

const int MAXN=500009;

int n,foundIdx,foundVal;

int tree[MAXN\*4], ara[MAXN];

void init(int node, int beg, int endd)

{

    if(beg==endd)

    {

        tree[node]=ara[beg];

        return ;

    }

    int left=node\*2;

    int right=node\*2+1;

    int mid= (beg+endd)/2;

    init(left,beg,mid);

    init(right,mid+1,endd);

    tree[node]=\_\_gcd(tree[left], tree[right]);

}

void  update(int node, int beg, int endd, int x, int val)

{

    if(beg==x &&endd==x)

    {

        tree[node]=val;

        return;

    }

    int left=node\*2;

    int right=node\*2+1;

    int mid= (beg+endd)/2;

    if(x<=mid) update(left, beg, mid, x, val);

    else update(right, mid+1, endd, x, val);

    tree[node]=\_\_gcd(tree[left],tree[right]);

}

int query2(int node, int beg, int endd, int i, int j)

{

    if(beg>j || endd<i) return 0;

    if(beg>=i && endd<=j)

    {

        return tree[node];

    }

    int left=node\*2;

    int right=node\*2+1;

    int mid= (beg+endd)/2;

    int t1=query2(left, beg, mid, i,j);

    int t2=query2(right, mid+1,endd,i,j);

    return \_\_gcd(t1,t2);

}

void  query(int node, int beg, int endd, int i, int j, int flag, int x)

{

    if(beg>j || endd<i ) return ;

    if(beg>=i && endd<=j)

    {

        if(tree[node]%x==0)

            return ;

        if(foundIdx!=-1) return;

        flag=1;

    }

    if(beg==endd)

    {

        foundIdx=beg;

        foundVal=tree[node];

        return;

    }

    int left=node\*2;

    int right=node\*2+1;

    int mid= (beg+endd)/2;

    int t1,t2;

    if(flag==0)

    {

        query(left,beg,mid,i,j,flag,x);

        query(right,mid+1,endd,i,j,flag,x);

    }

    else

    {

        int lg=tree[left];

        int rg=tree[right];

        if(lg%x!=0)

            query(left,beg, mid, i,j,flag,x);

        else query(right,mid+1,endd,i,j,flag,x);

    }

}

int main()

{

    int l,r,x,q,cs,y;

    scanf("%d",&n);

    for(int i=0; i<n; i++)

        scanf("%d",&ara[i]);

    init(1,0,n-1);

    scanf("%d",&q);

    while(q--)

    {

        scanf("%d",&cs);

        if(cs==1)

        {

            scanf("%d %d %d",&l,&r,&x);

            foundIdx=-1;

            query(1,0,n-1,l-1,r-1,0,x);

            if(foundIdx==-1)

                printf("YES\n");

            else

            {

                update(1,0,n-1,foundIdx,x);

                int res=query2(1,0,n-1,l-1,r-1);

                if(res%x==0)

                    printf("YES\n");

                else

                    printf("NO\n");

                update(1,0,n-1,foundIdx,foundVal);

            }

        }

        else

        {

            scanf("%d %d",&x,&y);

            update(1,0,n-1,x-1,y);

        }

    }

    return 0;

}

\*\*\*\*sqrt decomposition\*\*\*\*

#include<bits/stdc++.h>

#define pb push\_back

#define ms(a,b) memset((a),(b),sizeof(a))

#define loop(i,n) for(int i=0;i<n;i++)

#define ll long long

#define pii pair<int,int>

#define inf 100000000000

#define in(a) freopen(a,"r", stdin)

#define out(a) freopen(a,"w",stdout)

using namespace std;

int ara[100002], store[320][10002], added[320];

bool lucky\_ara[10002];

int lucky[]= {4,7,44,47,74,77, 444,447,474,477,744,747,774,777,

              4444,4447,4474,4477,4744,4747,4774,4777,

              7444,7447,7474,7477,7744,7747,7774,7777

             };

int b\_size;

void update(int l, int r, int val)

{

    int cur\_buck;

    while(l<r and l%b\_size!=0 and l!=0)

    {

        cur\_buck= l/b\_size;

        store[cur\_buck][ara[l]]--;

        store[cur\_buck][ara[l]+val]++;

        ara[l]+=val;

        l++;

    }

    while(l+b\_size<=r)

    {

        cur\_buck= (l/b\_size);

        added[cur\_buck]+=val;

        l+= b\_size;

    }

    while(l<=r)

    {

        //printf("hello %d\n",l);

        cur\_buck= l/b\_size;

        store[cur\_buck][ara[l]]--;

        store[cur\_buck][ara[l]+val]++;

        ara[l]+=val;

        l++;

    }

}

int lucky\_in\_buck(int id, int sum)

{

    int tot=0;

    for(int i=0; lucky[i]; i++)

    {

        if(lucky[i]-sum>=0)

        {

            tot+=store[id][lucky[i]-sum];

        }

    }

    return tot;

}

int query(int l, int r)

{

    int cnt=0;

    while(l<r and l%b\_size!=0 && l!=0)

    {

        if(lucky\_ara[ara[l]+added[l/b\_size]]==true)

            cnt++;

        l++;

    }

    while(l+b\_size<=r)

    {

        cnt+=lucky\_in\_buck(l/b\_size, added[l/b\_size]);

        l+=b\_size;

    }

    while(l<=r)

    {

        if(lucky\_ara[ara[l]+added[l/b\_size]]==true)

            cnt++;

        l++;

    }

    return cnt;

}

void make\_bucket(int n)

{

    int b\_indx=-1;

    b\_size=sqrt(n);

    loop(i,n)

    {

        if(i%b\_size==0)

            b\_indx++;

        store[b\_indx][ara[i]]++;

    }

}

int main()

{

    for(int i=0; lucky[i]; i++) lucky\_ara[lucky[i]]=true;

    int n,m;

    scanf("%d %d",&n,&m);

    loop(i,n)

    {

        scanf("%d",&ara[i]);

    }

    make\_bucket(n);

    int l,r,val;

    char str[5];

    loop(i,m)

    {

        scanf("%s",&str);

        if(strlen(str)==5)

        {

            scanf("%d %d",&l,&r);

            printf("%d\n",query(l-1,r-1));

        }

        else

        {

            scanf("%d %d %d",&l,&r,&val);

            update(l-1,r-1,val);

        }

    }

    return 0;

}

\*\*\*\*Trie\*\*\*\*

struct node {

    bool endmark;

    node\* next[26 + 1];

    node()

    {

        endmark = false;

        for (int i = 0; i < 26; i++)

            next[i] = NULL;

    }

} \* root;

void insert(char\* str, int len)

{

    node\* curr = root;

    for (int i = 0; i < len; i++) {

        int id = str[i] - 'a';

        if (curr->next[id] == NULL)

            curr->next[id] = new node();

        curr = curr->next[id];

    }

    curr->endmark = true;

}

bool search(char\* str, int len)

{

    node\* curr = root;

    for (int i = 0; i < len; i++) {

        int id = str[i] - 'a';

        if (curr->next[id] == NULL)

            return false;

        curr = curr->next[id];

    }

    return curr->endmark;

}

void del(node\* cur)

{

    for (int i = 0; i < 26; i++)

        if (cur->next[i])

            del(cur->next[i]);

    delete (cur);

}

int main()

{

    puts("ENTER NUMBER OF WORDS");

    root = new node();

    int num\_word;

    cin >> num\_word;

    for (int i = 1; i <= num\_word; i++) {

        char str[50];

        scanf("%s", str);

        insert(str, strlen(str));

    }

    puts("ENTER NUMBER OF QUERY";);

    int query;

    cin >> query;

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

        char str[50];

        scanf("%s", str);

        if (search(str, strlen(str)))

            puts("FOUND");

        else

            puts("NOT FOUND");

    }

    del(root); //ট্রাইটা ধ্বংস করে দিলাম

    return 0;

}

\*\*\*\*Trie Implementation 2\*\*\*\*

#include<bits/stdc++.h>

#define pii pair<i64,i64>

#define i64 long long

using namespace std;

#define AS 2

struct node

{

    i64 endmarks, value;

    node\* next[AS+2];

    i64 cnt[AS+2];

    node()

    {

        endmarks=0;

        for(i64 i=0;i<=AS;i++){

            next[i]=NULL;

            cnt[i]=0;

        }

    }

};

node\* root;

i64 pri64(i64 id,i64 no)

{

    if(id==1)

    {

        printf("at no=%lld\n",no\*2);

        no=no\*2;

    }

    else

    {

        printf("at no=%lld\n",no\*2+1);

        no=no\*2+1;

    }

    return no;

}

void Insert(string str)

{

    node\* cur=root;

    i64 no=1, l= str.size();

    for(i64 i=0; i<l; i++)

    {

        i64 id=str[i]-'0';

        if(cur->next[id]==NULL)

            cur->next[id]= new node();

        cur->cnt[id]++;

        cur=cur->next[id];

    }

    cur->endmarks++;

}

void Remove(string str)

{

    node\* cur=root;

    i64 no=1, l= str.size();

    for(i64 i=0; i<l; i++)

    {

        i64 id=str[i]-'0';

        cur->cnt[id]--;

        cur=cur->next[id];

    }

    cur->endmarks--;

}

i64 Search(string str)

{

    node\* cur=root;

    i64 l=str.size();

    i64 res=0;

    for(i64 i=0;i<l;i++)

    {

        i64 id=str[i]-'0';

        i64 idr=id^1;

        if(cur->next[idr]!=NULL && cur->cnt[idr])

        {

            cur=cur->next[idr];

            res= (res<<1)+idr;

        }

        else if(cur->next[id]!=NULL) {cur=cur->next[id]; res=(res<<1) + id; }

        else return 0;

    }

    return res;

}

int main()

{

    root=new node();

    i64 q;

    cin>>q;

    while(q--)

    {

        char ch;

         i64 x;

        cin>>ch>>x;

        getchar();

        string tr=bitset<32>(x).to\_string();

        if(ch=='+')

            Insert(str);

        else if(ch=='-')

            Remove(str);

        else

        {

            node\* cur=root;

            i64 res=Search(str);

            printf("%lld\n",max(x,res^x));

        }    }}

**Divide and Conquer:**

**\*\*\*\*closest pair of points\*\*\*\***

#include<bits/stdc++.h>

using namespace std;

struct Point

{

    int x,y;

} ;

bool cmp1(Point a, Point b)

{

    return a.x<b.x;

}

bool cmp2(Point a, Point b)

{

    return a.y<b.y;

}

double dist(Point p1, Point p2)

{

    return sqrt( (p1.x-p2.x)\*(p1.x-p2.x) +

                 (p1.y-p2.y)\*(p1.y-p2.y)

               );

}

double bruteForce(Point P[], int n)

{

    double mini=FLT\_MAX;

    for(int i=0; i<n-1; i++)

        for(int j=i+1; j<n; j++)

            mini= min(mini, dist(P[i],P[j]));

    return mini;

}

double stripClosest(Point strip[], int n, double d)

{

    double mini=d;

    for(int i=0; i<n; i++)

    {

        for(int j=i+1; j<n && (strip[j].y-strip[i].y)<mini; j++)

        {

            if(dist(strip[j],strip[i])<mini)

                mini= dist(strip[j],strip[i]);

        }

    }

    return mini;

}

double closestUtil(Point Px[], Point Py[], int n)

{

    if(n<=3)

        return bruteForce(Px,n);

    int mid= n/2;

    Point midPoint= Px[mid];

    Point Pyl[mid+1];

    Point Pyr[n-mid+1];

    int li=0,ri=0;

    for(int i=0; i<n; i++)

    {

        if(Py[i].x<=midPoint.x)

            Pyl[li++]=Py[i];

        else Pyr[ri++]=Py[i];

    }

    double dl=closestUtil(Px,Pyl,mid);

    double dr=closestUtil(Px+mid,Pyr, n-mid);

    double d=min(dl,dr);

    Point strip[n];

    int j=0;

    for(int i=0; i<n; i++)

    {

        if(abs(Py[i].x-midPoint.x)<d)

            strip[j++]=Py[i];

    }

    double ret= min(d, stripClosest(strip,j,d));

   // cout<<ret<<" "<<d<<endl;

    return ret;

}

double closest(Point P[], int n)

{

    Point Px[n];

    Point Py[n];

    for(int i=0; i<n; i++)

    {

        Px[i]=P[i];

        Py[i]=P[i];

    }

    sort(Px, Px+n,cmp1);

    sort(Py, Py+n, cmp2);

    return closestUtil(Px,Py,n);

}

int main()

{

    Point P[]= {{2,3}, {12,30}, {40,50},{5, 1}, {12, 10}, {3, 4} };

    int n= sizeof(P)/ sizeof(P[0]);

    cout<<"The smallest distance :"<<closest(P,n)<<endl;

    return 0;

}

**\*\*\*\*Inversion Count\*\*\*\***

#include<bits/stdc++.h>

using namespace std;

int  \_mergeSort(int arr[], int temp[], int left, int right);

int mergex(int arr[], int temp[], int left, int mid, int right);

int mergeSort(int arr[], int array\_size)

{

    int \*temp=(int\*) malloc(sizeof(int)\*array\_size);

    return \_mergeSort(arr, temp, 0,array\_size-1);

}

int \_mergeSort(int arr[], int temp[], int left, int right)

{

    int mid, inv\_count=0;

    if(left>=right) return 0;

    mid= (left+right)/2;

    inv\_count+= \_mergeSort(arr,temp,left, mid);

    inv\_count+= \_mergeSort(arr,temp,mid+1, right);

    inv\_count+=mergex(arr,temp,left,mid+1, right);

    return inv\_count;

}

int mergex(int arr[], int temp[], int left, int mid, int right)

{

    int i,j,k;

    int inv\_count=0;

    i=left,j=mid,k=left;

    while((i<=mid-1) &&(j<=right))

    {

        if(arr[i]<=arr[j])

        {

            temp[k++]=arr[i++];

        }

        else

        {

            temp[k++]=arr[j++];

            inv\_count+=mid-i;

        }

    }

    while(i<=mid-1)

    {

        temp[k++]=arr[i++];

    }

    while(j<=right)

    {

        temp[k++]=arr[j++];

    }

    for(int i=left; i<=right;i++)

        arr[i]=temp[i];

    return inv\_count;

}

int main()

{

  int arr[] = {1, 20, 6, 4, 5};

  printf(" Number of inversions are %d \n", mergeSort(arr, 5));

  for(int i=0;i<5;i++)

    printf("%d ",arr[i]);

  getchar();

  return 0;

}

**DP and Backtrack**

\*\*\*\*Bitmask\*\*\*\*

int Set(int N,int pos){return N=N | (1<<pos);}

int reset(int N,int pos){return N= N & ~(1<<pos);}

bool check(int N,int pos){return (bool)(N & (1<<pos));}

\*\*\*\*Digit DP\*\*\*\*

const int NX = 70 ;

Long dp[2][2][NX][NX];

int vis[2][2][NX][NX];

int lim , tt ;

vector < int > inp ;

Long DP( int pos , int isSmall ,int isStart, int value)

{

    if( pos == lim ) return value ;

    Long &ret = dp[isSmall][isStart][pos][value];

    int &v = vis[isSmall][isStart][pos][value];

    if( v == tt ) return ret ;

    v = tt ;

    int ses = isSmall ? 9 : inp[pos];

    int i ;

    ret = 0 ;

    if( !isStart )  // আগেই নাম্বার বসানো শুরু করে দিছি

    for ( i = 0 ; i <= ses ; i++ )

    {

        ret += DP( pos + 1 , isSmall | i < inp[pos] ,0, (i == 0) + value );

    }

    else

    {

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

    {

        ret += DP( pos + 1 , isSmall | i < inp[pos] ,0, (i == 0) + value );

    }

    ret += DP( pos + 1 , 1 ,1, 0 );

    }

    return ret ;

}

Long Cal( Long x )

{

    if( x < 0 ) return 0 ;

    if( x <= 9 ) return 1 ;

    inp.clear();

    while( x )

    {

        inp.pb(x%10);

        x/=10;

    }

    reverse(inp.begin(),inp.end()); // সুবিধার জন্য রিভারস করে নিচ্ছি , এইটা করতেই হবে

    lim = inp.size();

    tt++;

    return DP( 0 , 0 , 1 , 0 ) + 1; // শুধু ০ টা আলাদা এড করছি

}

int main()

{

   // I will always use scanf and printf

   // May be i won't be a good programmer but i will be a good human being

  // cout << fixed << setprecision(10) ;

     int cs , t = II ;

     for ( cs = 1 ; cs <= t ; cs++ )

     {

        Long n = LL , m = LL ;

        Long ans = Cal(m) - Cal(n-1);

        printf("Case %d: %lld\n",cs,ans);

     }

    return 0;

}

\*\*\*\*Minimum lines to connect all points in 2d\*\*\*

#include<bits/stdc++.h>

using namespace std;

int Set(int N, int pos)

{

    return N=N|(1<<pos);

}

int reset(int N, int pos)

{

    return N= N&~(1<<pos);

}

bool Check(int N, int pos)

{

    return (bool)(N&(1<<pos));

}

struct point

{

    int x, y;

} ara[17];

int Armask[17][17], dp[(1<<17)+5];

int n;

void clearr()

{

    memset(Armask,0,sizeof Armask);

}

double length(point a, point b)

{

    return sqrt( (a.x-b.x)\*(a.x-b.x)  + (a.y-b.y)\*(a.y-b.y) );

}

void make\_mask(int i, int j,point a, point b)

{

    int mask=0;

    double lenAB=length(a,b);

    //cout<<a.x<<","<<a.y<<" "<<b.x<<","<<b.y<<"="<<endl;

    for(int i=0; i<n; i++)

    {

        double l1= length(ara[i],a);

        double l2= length(ara[i],b);

        double a[3]= { lenAB, l1,l2 };

        sort(a,a+3);

        // printf("for %d th =%.2f %.2f = %.2f?\n",i, a[0],a[1],a[2]);

        if( fabs(a[0]+a[1] - a[2])<0.000000001 )

        {

            //printf("YES\n");

            mask=Set(mask,i);

        }

        //else printf("NO\n");

    }

    // printf("mask=%d\n",mask);

    Armask[i][j]=mask;

}

int f(int mask)

{

    if(mask==(1<<n)-1) return 0;

    int c=0;

    for(int i=0;i<n;i++)

        c+=!Check(mask,i);

    if(c<=2) return 1;

    if(dp[mask]!=-1) return dp[mask];

    int mn=100000000,ret=0;

    for(int i=0; i<n; i++)

    {

        if(Check(mask,i)) continue;

        for(int j=i+1; j<n; j++)

        {

            if( Check(mask,j)==0)

            {

                int temp= Armask[i][j];

                temp= temp|mask;

               // printf("%d and %d =mask->%d\n",i,j,temp);

                ret= 1+ f(temp);

                mn= min(ret,mn);

            }

        }

        break;

    }

    return dp[mask]=mn;

}

int main()

{

    int t=0,test;

    scanf("%d",&test);

    while(t<test)

    {

        //memset(Armask,-1,sizeof Armask);

        memset(dp,-1,sizeof dp);

        scanf("%d",&n);

        for(int i=0; i<n; i++)

        {

            scanf("%d %d",&ara[i].x,&ara[i].y);

        }

        for(int i=0; i<n; i++)

        {

            for(int j=i+1; j<n; j++)

            {

                //if(i==j) continue;

                make\_mask(i,j, {ara[i].x, ara[i].y }, {ara[j].x, ara[j].y});

                //printf("%d to %d = %d\n",i,j,Armask[i][j]);

            }

        }

        printf("Case %d: %d\n",++t,f(0));

        clearr();

    }

    return 0;

}

\*\*\*\*Two Recursions\*\*\*\*

///http://codeforces.com/problemset/problem/51/B

//html files, stack wise things

#include<bits/stdc++.h>

#define i64 long long

#define inf 1000000000000000000

using namespace std;

map<string,int>mp;

vector<int>tot;

void init()

{

    mp["<table>"]=3;

    mp["</table>"]=-3;

    mp["<tr>"]=2;

    mp["</tr>"]=-2;

    mp["<td>"]=1;

    mp["</td>"]=-1;

}

vector<int>vec,vec2;

void process(string str)

{

    string ret;

    for(int i=0; i<str.size(); i++)

    {

        if(str[i]=='<')

        {

            ret="";

            while(str[i]!='>'  && i<str.size())

                ret+=str[i], i++;

            ret+=str[i];

            if(mp[ret]!=2 && mp[ret]!=-2)

                vec.push\_back(mp[ret]);

        }

    }

//    for(int i=0; i<vec.size(); i++)

//        printf("%d ",vec[i]);

//

//    puts("");

}

int pos;

int Table1();

int Table2();

int Table2()

{

   // printf("in 2: %d -> %d\n",pos,vec[pos]);

    if(vec[pos+1]==-1)

    {

        pos++;

        //printf("returning from 2\n");

        return 1;

    }

    int res=0;

    while(vec[pos+1]==3)

    {

        pos++;

        res+=Table1();

    }

//    printf("2=>%d\n",res);

//    tot.push\_back(res);

    if(vec[pos+1]==-1)

    {

        pos++;

       // printf("returning from 2\n");

        return res;

    }

}

int Table1()

{

   // printf("in 1: %d -> %d\n",pos,vec[pos]);

    int res=0;

    while(vec[pos+1]==1)

    {

        pos++;

        res+=Table2();

    }

   // printf("1=>%d\n",res);

    tot.push\_back(res);

    if(vec[pos+1]==-3)

    {

       // printf("returning from 1\n");

        pos++;

        return 1;

    }

}

int main()

{

    init();

    //freopen("input.txt","r",stdin);

    string str;

    char ara[6009];

    while((scanf("%s",&ara))!=EOF)

        str+=ara;

    process(str);

    stack<int>stk;

    int res=Table1();

//    cout<<"--->"<<res<<endl;

//    tot.push\_back(res);

   // puts("");

    sort(tot.begin(),tot.end());

    for(int i=0; i<tot.size(); i++)

    {

        if(i) printf(" ");

        printf("%d",tot[i]);

    }

    puts("");

}

**Geometry**

\*\*\*\*Area of intersection circles\*\*\*\*

#include<bits/stdc++.h>

#define ll long long

#define pi acos(-1)

using namespace std;

struct circle

{

    double x,y,r;

};

double distance(int x1, int y1, int x2, int y2 )

{

    double d=(x1-x2)\*(x1-x2) +(y1-y2)\*(y1-y2);

    d= sqrt(d);

    return d;

}

double CosineRule(double b, double c, double a)

{

    return (b\*b + c\*c - a\*a)/ (2\*b\*c);

}

double section(double r, double theta)

{

    return r\*r\*0.5 \* (theta- sin(theta));

}

int main()

{

    int tt=0,test;

    cin>>test;

    while(tt<test)

    {

        circle c1,c2;

        cin>>c1.x>>c1.y>>c1.r>>c2.x>>c2.y>>c2.r;

        double area=0,area1=0,area2=0;

        double d=distance(c1.x,c1.y,c2.x,c2.y);

        if(c1.r+c2.r<=d)

        {

            area=0;

        }

        else if(d+ min(c1.r,c2.r) <= max(c1.r,c2.r))

        {

            area= min( pi\*c1.r\*c1.r, pi\*c2.r\*c2.r );

        }

        else

        {

            double theta= CosineRule(c1.r,d,c2.r);

            theta= acos(theta);

            theta\*=2;

            area1= section(c1.r,theta);

            theta= CosineRule(c2.r,d,c1.r);

            theta= acos(theta);

            theta\*=2;

            area2= section(c2.r, theta);

            area= area1+area2;

        }

       // cout<<area<<endl;

        printf("Case %d: %.10f\n",++tt,area);

    }

    return 0;

}

\*\*\*\*Convex Hull Graham Scan\*\*\*\*

#include<bits/stdc++.h>

using namespace std;

struct Point

{

    int x,y;

}p0;

Point nextToTop(stack<Point>&S)

{

    Point p=S.top();

    S.pop();

    Point res= S.top();

    S.push(p);

    return res;

}

int calc\_dist(Point p1, Point p2)

{

    return (p1.x - p2.x)\*(p1.x - p2.x) +

          (p1.y - p2.y)\*(p1.y - p2.y);

}

int orientation(Point p, Point q, Point r)

{

    int res= (q.y-p.y)\*(r.x-q.x) - (r.y-q.y)\*(q.x-p.x);

   if(res==0) return 0;

   return (res>0)? 1:2;  /// clock or counterclock wise

}

bool compare(Point a, Point b)

{

    int o=orientation(p0,a,b);

    if(o==0)

        return  calc\_dist(p0,a)< calc\_dist(p0,b);

    if(o==2)

        return true;  /// in ccw 2nd case so  ok no  swap needed

    else return false;  /// not ok swap is

}

void convexHull(Point points[], int n)

{

    int miny=1e9,mini=0;

    for(int i=0; i<n; i++)

    {

        int y=points[i].y;

        if((y<miny) || (y==miny && points[i].x<points[mini].x))

        {

            miny=y;

            mini=i;

        }

    }

    swap(points[0],points[mini]);

    p0=points[0];

    sort(points+1, points+n, compare);

    int m=1;

    for(int i=1;i<n;i++)

    {

       // printf("%d %d\n",points[i].x,points[i].y);

        while(i<n-1 && orientation(p0,points[i],points[i+1])==0)

            i++;

        points[m]=points[i];

        m++;

    }

    if(m<3) return;

    stack<Point>S;

    S.push(points[0]);

    S.push(points[1]);

    S.push(points[2]);

    for(int i=3;i<m;i++)

    {

        while(orientation(nextToTop(S),S.top(),points[i])!=2)

            S.pop();

        S.push(points[i]);

    }

    while(!S.empty())

    {

        Point p= S.top();

        printf(" (%d,%d)\n",p.x,p.y);

        S.pop();

    }

}

int main()

{

    Point points[] = {{0, 3}, {1, 1}, {2, 2}, {4, 4},

        {0, 0}, {1, 2}, {3, 1}, {3, 3}

    };

    int n = sizeof(points)/sizeof(points[0]);

    convexHull(points, n);

    return 0;

}

\*\*\*\*separating convex hull using straight line\*\*\*\*

///http://www.spoj.com/problems/DOORSPEN/en/

/// separate two convex hulls using one straight line

#include<bits/stdc++.h>

#define pb push\_back

#define loop(i,n) for(int i=0;i<n;i++)

using namespace std;

int d,p;

struct Points

{

    int x,y;

} p0;

vector<Points>pnts1,pnts2;

Points mp(int x,int y)

{

    Points ret;

    ret.x=x;

    ret.y=y;

    return ret;

}

int calc\_dist(Points a, Points b)

{

    return (a.x-b.x)\*(a.x-b.x)  + (b.x-b.y)\*(b.x-b.y);

}

int orientation(Points a, Points b, Points c)

{

    int res= (c.y-b.y)\*(b.x-a.x) - (b.y-a.y)\*(c.x-b.x);

    if(res==0)

        return 0;

    if(res>0) return -1;

    else return +1;

}

bool comaprePoints(Points a, Points b)

{

    int ret=orientation(p0,a,b);

    if(ret==0)

        return calc\_dist(p0,a)<calc\_dist(p0,b);

    return (ret==-1)? true: false;

}

Points nextToTop(stack<Points> &stk)

{

    Points temp=stk.top();

    stk.pop();

    Points ret= stk.top();

    stk.push(temp);

    return ret;

}

bool isSeparatingAxis(Points a, Points b,vector<Points>&ara1,vector<Points>&ara2)

{

    int sign=orientation(a,b,ara2[0]);

    loop(i,ara2.size())

    {

        if(orientation(a,b,ara2[i])!=sign)

            return false;

    }

    loop(i, ara1.size())

    {

        if(orientation(a,b,ara1[i])==sign)

            return false;

    }

    return true;

}

bool Convexhull(vector<Points>&ara1, vector<Points>&ara2, int n)

{

    int miny=1e8,mni=0;

    loop(i,n)

    {

        if(ara1[i].y<miny || (ara1[i].y==miny && ara1[i].x<ara1[mni].x))

        {

            miny=ara1[i].y;

            mni=i;

        }

    }

    swap(ara1[0], ara1[mni]);

    p0=ara1[0];

    sort(ara1.begin()+1,ara1.end(), comaprePoints);

    vector<Points>vec;

    vec.push\_back(p0);  ///remove co linear from p0

    for(int i=0; i<n; i++)

    {

        while(orientation(p0,ara1[i],ara1[i+1])==0  && i+2<n)

            i++;

        vec.push\_back(ara1[i]);

        // printf("\_\_%d %d\n",vec.back().x,vec.back().y);

    }

    stack<Points>stk;

    stk.push(vec[0]);

    stk.push(vec[1]);

    stk.push(vec[2]);

    for(int i=3; i<vec.size(); i++)

    {

        while(orientation(nextToTop(stk), stk.top(), vec[i])==+1)

            stk.pop();

        stk.push(vec[i]);

    }

    Points last=stk.top();

    while(stk.size()>1)

    {

        Points top1= stk.top(), top2=nextToTop(stk);

     //   printf("-> %d %d with %d %d\n",stk.top().x, stk.top().y, top2.x, top2.y);

        if(isSeparatingAxis(top1,top2,ara1,ara2)==true)

        {

            // printf("got= %d %d , %d %d\n",top1.x, top1.y, top2.x,top2.y);

            return true;

        }

        stk.pop();

    }

  //  printf("-> %d %d, %d %d\n",p0.x, p0.y, last.x, last.y);

    if(isSeparatingAxis(p0,last,ara1,ara2)==true)

        {

           // printf("->got  %d %d, %d %d\n",p0.x, p0.y, last.x, last.y);

            return true;

        }

    while(!stk.empty())

    stk.pop();

    vec.clear();

    return false;

}

int main()

{

    int test=0;

    while(scanf("%d %d",&d,&p)==2)

    {

        if(d==0 && p==0) return 0;

        if(test)

            puts("");

        int x1,y1,x2,y2;

        loop(i,d)

        {

            scanf("%d %d %d %d",&x1, &y1, &x2,&y2);

            pnts1.pb(mp(x1,y1)),pnts1.pb(mp(x2,y2)), pnts1.pb(mp(x1,y2)), pnts1.pb(mp(x2,y1));

        }

//

        loop(i,p)

        {

            scanf("%d %d %d %d",&x1,  &y1, &x2,&y2);

            pnts2.pb(mp(x1,y1)),pnts2.pb(mp(x2,y2)), pnts2.pb(mp(x1,y2)), pnts2.pb(mp(x2,y1));

        }

        int res1=0, res2=0;

        res1=Convexhull(pnts1, pnts2,pnts1.size());

        res2=Convexhull(pnts2,pnts1, pnts2.size());

        //printf("res1= %d  res2=%d\n",res1,res2);

        if(res1==1 || res2==1)

            printf("Case %d: It is possible to separate the two groups of vendors.\n",++test);

        else printf("Case %d: It is not possible to separate the two groups of vendors.\n",++test);

        pnts1.clear(),pnts2.clear();

    }

    return 0;

}

\*\*\*\*std complex and easy geometry\*\*\*\*

#include<iostream>

#include<complex>

#include<bits/stdc++.h>

using namespace std;

/// define x, y as real(), imag()

typedef complex<double> point ;

#define x real()

#define y imag()

int main()

{

    //double num=20;

    point a(5,3);

    point b(6,2);

    point c(1,1);

    cout<< a<<" "<<b<<endl;

    cout<< (conj(b-c)\*(a-c)).y<<endl;

    cout<< (conj(a-c)\*(b-c)).y<<endl;

///// vector addition and subtraction

//    printf("Addition , subtraction ,Multiplication \n");

//    cout<<a+b<<endl;

//    cout<<a-b<<endl;

//    cout<<a\*b<<endl;

//

///// scalar multiplication

//    printf("Scalar multiplication:\n");

//    cout<<3.0\*a<<endl;

//    cout<< a/5.0<<endl;

//

/////dot product

//    printf("Dot product:\n");

//    cout<<  (conj(a)\*b).x <<endl;

//    cout<<  (conj(b)\*a).x <<endl;

//

//    cout<< (conj(a)\*b).y<<endl;

//    cout<< cross(a,b)<<endl;

//

//    cout<< norm(a-b)<<endl;

//    cout<< abs(a-b)<<endl;

//

//

//     cout<<arg(b-a)<<endl;

//     cout<<tan(arg(b-a))<<endl;

//

//     cout<<polar(1,90)<<endl;

//     cout<< point(abs(b-a), arg(b-a))<<endl;

    return 0;

}

\*\*\*\*vector geometry\*\*\*\*

#include<bits/stdc++.h>

using namespace std;

#define pi      acos(-1.00)

#define eps     1e-9

#define D(x)    cout << #x " = " << (x) << endl

const int inf = numeric\_limits<int>::max();

bool eq(double a, double b) { return fabs( a - b ) < eps; } //two numbers are equal

struct point{

    double x, y;

    point(){}

    point(double xx, double yy) {x = xx, y = yy;} // NEVER USE xx = 0 or yy = 0 HERE

} origin = point(0, 0);

point operator+(const point &u, const point &v) {return point(u.x + v.x, u.y + v.y);} //OK

point operator-(const point &u, const point &v) {return point(u.x - v.x, u.y - v.y);} //OK

point operator\*(const point &u, double v) {return point(u.x\*v, u.y\*v);} //OK

point operator\*(double v, const point &u) {return point(u.x\*v, u.y\*v);} //OK

point operator\*(const point &u, const point &v) {return point(u.x \* v.x - u.y \* v.y, u.x \* v.y + v.x \* u.y);} // multiplying two complex numbers

point operator/(const point &u, double v) {assert(abs(v) > eps); return point(u.x/v, u.y/v);} //OK

bool operator != (const point &u, const point &v) {return !(eq(u.x, v.x) && eq(u.y, v.y));} //OK

ostream &operator <<(ostream &os, const point &p) {

  os << "(" << p.x << "," << p.y << ")";

} //OK

bool operator <(const point &u, const point &v){

    if(fabs(u.x - v.x ) < eps) return u.y + eps < v.y;

    return u.x + eps < v.x;

}

double norm(point u){return sqrt(u.x \* u.x + u.y \* u.y);} //OK

double arg(point u){ assert(u != origin); return atan2(u.y, u.x);} //OK

point polar(double r, double theta) {return point(r \* cos(theta), r \* sin(theta));} //OK

double dotp(point u, point v) {return u.x \* v.x + u.y \* v.y;} //OK

double crsp(point u, point v) {return u.x \* v.y - u.y \* v.x;} //OK

point unit\_vector(point u) { return u / norm(u); } //OK

point rtt(point piv, point u, double theta) {return (u - piv) \* polar(1.00, theta) + piv;} //OK

point projection(point p, point st, point ed) { return dotp(ed - st, p - st) / norm(ed - st) \* unit\_vector(ed - st) + st;} //OK

point extend(point st, point ed, double len) { return ed + unit\_vector(ed-st) \* len;} //OK

point segmentProjection(point p, point st, point ed)

{

    double d = dotp(p - st, ed - st) / norm(ed - st);

    if(d < 0) return st;

    if(d > norm(ed - st) + eps) return ed;

    return st + unit\_vector(ed - st) \* d;

} //OK

double distancePointSegment(point p, point st, point ed) {return norm(p - segmentProjection(p, st, ed)); } //OK

double distancePointLine( point P, point st, point ed) { return norm( projection(P, st, ed) - P ); } //OK

point reflection(point p, point st, point ed){

    point proj = projection(p, st, ed);

    if(p != proj) return extend(p, proj, norm(p - proj));

    return proj;

} //OK

int main()

{

    return 0;

}

**GRAPH**

\*\*\*\*  
Articulation bridge\*\*\*\*

#include<bits/stdc++.h>

#define pb push\_back

#define ms(a,b) memset((a),(b),sizeof(a))

//#define clear(v,n) for(\_typeof (n) i=0;i< (n) ; i++)  { v.clear(); }

#define ll long long

#define pii pair<ll,ll>

#define inf 100000000000

#define in(a) freopen(a,"r", stdin)

#define out(a) freopen(a,"w",stdout)

using namespace std;

set<pii>bridge;

vector<int>adj[100009];

int low[100009],disc[100009],par[100009],vis[100009], times;

void dfs1(int src)

{

    disc[src]=low[src]=++times;

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

    {

        int node= adj[src][i];

        if(vis[node]==0 && node!= par[src])

        {

            par[node]=src;

            vis[node]=1;

            dfs1(node);

            low[src]= min(low[src], low[node]);

            if(low[node]> disc[src])

            {

                bridge.insert({ min(src,node), max(src,node) });

                // prllf("%d -> %d\n",src, node);

            }

        }

        else if(node!= par[src])

        {

            low[src]= min(low[src], disc[node]);

        }

    }

}

int main()

{

}

\*\*\*\*\*Articulation point\*\*\*\*\*

#include<bits/stdc++.h>

#define pb push\_back

#define ms(a,b) memset((a),(b),sizeof(a))

//#define clear(v,n) for(\_typeof (n) i=0;i< (n) ; i++)  { v.clear(); }

#define ll long long

#define pii pair<int,int>

#define inf 100000000000

#define in(a) freopen(a,"r", stdin)

#define out(a) freopen(a,"w",stdout)

using namespace std;

vector<int>adj[10009];

int discTime=0, low[10009], disc[10009];

bool isAp[10009];

void dfs\_findAp(int src, int parent)

{

    low[src]= disc[src]= ++ discTime;

    int child=0;

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

    {

        int node= adj[src][i];

        if(!disc[node])

        {

            child++;

            dfs\_findAp(node, src);

            low[src]= min(low[src], low[node]);

            if(parent==-1 && child>1)    isAp[src]=true;

            if(parent!=-1 && low[node]>=disc[src])  isAp[src]=true;

        }

        else if( node!=parent)

        {

            low[src]= min(low[src], disc[node]);

        }

    }

}

int main()

{

    int n,m,u,v;

    scanf("%d %d",&n,&m);

    for(int i=0;i<m;i++)

    {

        scanf("%d %d",&u,&v);

        adj[u].push\_back(v);

        adj[v].push\_back(u);

    }

    //dfs\_findAp(0,-1);

    for(int i=0;i<n;i++)

    {

        if(disc[i]==0)

        {

            dfs\_findAp(i,-1);

        }

    }

    for(int i=0;i<n;i++)

    {

       if(isAp[i]==true)

         printf("%d\n",i);

    }

}

\*\*\*\*BellMan Fornd\*\*\*\*

#include<bits/stdc++.h>

#define pb push\_back

#define ms(a,b) memset((a),(b),sizeof(a))

//#define clear(v,n) for(\_typeof (n) i=0;i< (n) ; i++)  { v.clear(); }

#define ll long long

#define pii pair<ll,ll>

#define inf 100000000

#define in(a) freopen(a,"r", stdin)

#define out(a) freopen(a,"w",stdout)

using namespace std;

vector<pii>vec;

int n;

int dis[209],busy[209],inCycle[209];

int calc(int a, int b)

{

    int c= b-a;

    return c\*c\*c;

}

void Bellman\_ford(int src)

{

    for(int i=1;i<=n;i++)

        dis[i]=inf;

    dis[src]=0;

    for(int i=1;i<=n-1;i++)

    {

        for(int j=0;j<vec.size();j++)

        {

            pii tp= vec[j];

            int u=tp.first;

            int v=tp.second;

            int w= calc(busy[u],busy[v]);

            if(dis[u]!=inf && dis[u]+ w <dis[v])

            {

               // printf("u=%d v=%d w=%d\n",u,v,w);

                dis[v]= dis[u]+w;

            }

        }

    }

    bool flag=false;

    for(int i=0;i<vec.size();i++)

    {

            pii tp= vec[i];

            int u=tp.first;

            int v=tp.second;

            int w= calc(busy[u],busy[v]);

            if( dis[u]!=inf &&  dis[u]+ w <dis[v])

            {

               // printf("Fuck\n");

               inCycle[v]=1;

               inCycle[u]=1;

                break;

            }

    }

    return ;

}

int main()

{

    int tt=0,test;

    cin>>test;

    while(tt<test){

    printf("Case %d:\n",++tt);

    int m,u,v;

    scanf("%d",&n);

    for(int i=1;i<=n;i++)

        scanf("%d",&busy[i]);

    scanf("%d",&m);

    for(int i=0;i<m;i++)

    {

        scanf("%d %d",&u,&v);

        vec.push\_back({u,v});

    }

    Bellman\_ford(1);

    int q;

    scanf("%d",&q);

    for(int i=0;i<q;i++)

    {

        scanf("%d",&v);

       // printf(" v=%d dis[v]=%d\n",v,dis[v]);

        if(inCycle[v]==1 || dis[v]<3 || dis[v]==inf)

            printf("?\n");

        else

            printf("%d\n",dis[v]);

    }

    vec.clear();

    ms(inCycle,0);

    ms(dis,0);

    }

    return 0;

}

\*\*\*\*Dijkstra Using priority queue\*\*\*\*

struct node{

    int city,dist;

    bool operator < (const node &n) const{

        if(dist==n.dist)

            return city>n.city;

        return dist > n.dist;

    }

};

void dijkstra(int source){

    for(int i = 1; i <= n; i++) d[i] = inf;

    d[source] = 0;

    node u;

    u.city = source;

    u.dist = 0;

    priority\_queue <node> pq;

    pq.push(u);

    while(!pq.empty()){

        node u= pq.top();

        pq.pop();

    //if(vis[u.city]==1) continue;

        for(int i = 0; i < G[u.city].size(); i++){

            node v;

            v.city = G[u.city][i].first;

            v.dist = u.dist + G[u.city][i].second;

            if(d[v.city] > v.dist){

                d[v.city] = v.dist;

                pq.push(v);

            }

        }

    }

}

\*\*\*\*  
Distance Matrix to Tree\*\*\*\*

#include<bits/stdc++.h>

#define pb push\_back

#define ms(a,b) memset((a),(b),sizeof(a))

//#define clear(v,n) for(\_typeof (n) i=0;i< (n) ; i++)  { v.clear(); }

#define ll long long

#define pii pair<int,int>

#define inf 100000000000

#define in(a) freopen(a,"r", stdin)

#define out(a) freopen(a,"w",stdout)

using namespace std;

struct node

{

    int u,v,w;

    bool operator < (const node &p) const

    {

        return w<p.w;

    }

};

vector<node>vec;

vector<int>adj[2009],edge[2009];

int ara[2009][2009],dis[2009][2009];

int par[2009], vis[2009];

int n;

bool check()

{

    for(int i=1; i<=n; i++)

    {

        if(ara[i][i]!=0) return false;

        for(int j=i+1; j<=n; j++)

        {

            if(ara[i][j]!= ara[j][i])

                return false;

            if(ara[i][j]==0)

                return false;

        }

    }

    return true;

}

int findd(int r)

{

    return par[r]= (par[r]==r)? r:findd(par[r]);

}

void mst()

{

    sort(vec.begin(),vec.end());

    for(int i=1; i<=n; i++)

        par[i]=i;

    for(int i=0; i<vec.size(); i++)

    {

        int u= findd(vec[i].u);

        int v= findd(vec[i].v);

        if(u!=v)

        {

            par[u]=v;

            int u1= vec[i].u;

            int v1=vec[i].v;

            int w1=vec[i].w;

            adj[u1].push\_back(v1);

            edge[u1].push\_back(w1);

            adj[v1].push\_back(u1);

            edge[v1].push\_back(w1);

        }

    }

}

void dfs(int src, int ith)

{

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

    {

        int node= adj[src][i];

        int ege= edge[src][i];

        if(vis[node]==0)

        {

            vis[node]=1;

            dis[ith][node]=dis[ith][src]+ege;

            dfs(node,ith);

        }

    }

}

int main()

{

    scanf("%d",&n);

    for(int i=1; i<=n; i++)

    {

        for(int j=1; j<=n; j++)

            scanf("%d",&ara[i][j]);

    }

    if(check()==false) {printf("NO\n"); return 0; }

    for(int i=1; i<=n; i++)

    {

        for(int j=i+1; j<=n; j++)

        {

            vec.push\_back({ min(i,j), max(i,j), ara[i][j]});

        }

    }

    mst();

    for(int i=1; i<=n; i++)   /// Here is the main tree.

    {

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

        {

            printf("%d %d %d\n",i,adj[i][j], edge[i][j]);

        }

    }

    puts("-------");

    for(int i=1; i<=n; i++)     /// N=2009 so , n^2 loop is for determining all pair shortest paths since its a tree

    {

        ms(vis,0);

        vis[i]=1;

        dfs(i,i);

    }

    for(int i=1; i<=n; i++)     /// checking if the given matrix is correct

    {

        for(int j=1; j<=n; j++)

        {

            //printf("%d ",dis[i][j]);

            if(ara[i][j]!=dis[i][j])

            {

                printf("NO\n");

                return 0;

            }

        }

        //puts("");

    }

    printf("YES\n");

    return 0;

}

\*\*\*\*Floyd Warshall\*\*\*\*

#include<bits/stdc++.h>

#define pb push\_back

#define ms(a,b) memset((a),(b),sizeof(a))

//#define clear(v,n) for(\_typeof (n) i=0;i< (n) ; i++)  { v.clear(); }

#define ll long long

#define inf 1000099

using namespace std;

int ara[25][25];

vector<string>name;

int main()

{

    int t=0;

    int n,m,u,v,w   ;

    while(1){

    scanf("%d %d",&n,&m);

    if(m==0 && n==0 ) break;

    string str;

    for(int i=0;i<n;i++)

    {

        cin>>str;

        name.push\_back(str);

    }

    for(int i=1;i<=n;i++)

        for(int j=1;j<=n;j++)

            ara[i][j]= (i==j)? 0:100009;

    for(int i=0;i<m;i++)

    {

        scanf("%d %d %d",&u,&v,&w);

        ara[u][v]=w;

        ara[v][u]=w;

    }

    for(int k=1;k<=n;k++)

        for(int i=1;i<=n;i++)

            for(int j=1;j<=n;j++)

                if(ara[i][k]+ara[k][j]<ara[i][j])

                    ara[i][j]=ara[i][k]+ara[k][j];

    int mn=100000,mni=0;

    for(int i=1;i<=n;i++){

        int sum=0;

        for(int j=1;j<=n;j++)

            {

                sum+=ara[i][j];

                //printf("%d ",ara[i][j]);

            }

        if(sum<mn) { mn=sum; mni=i; }

       //    puts("");

    }

    printf("Case #%d : ",++t);

    cout<<name[mni-1]<<endl;

    ms(ara,0);

    name.clear();

    }

    return 0;

}

\*\*\*\*grid bfs\*\*\*\*

#include<bits/stdc++.h>

#define pb push\_back

#define ms(a,b) memset((a),(b),sizeof(a))

//#define clear(v,n) for(\_typeof (n) i=0;i< (n) ; i++)  { v.clear(); }

#define ll long long

#define pii pair<int,int>

#define inf 100000000000

using namespace std;

int R,C;

struct par

{

    int r,c;

};

int dirr[4]= { 0, 0, 1, -1 };

int dirc[4]= { 1, -1,0,  0 };

int dis[21][21], vis[21][21];

char ara[21][21];

vector<par>vec;

void clearr()

{

    vec.clear();

    ms(dis,0);

    ms(vis,0);

}

void bfs(par src)

{

    queue<par>q;

    q.push(src);

    dis[src.r][src.c]=0;

    vis[src.r][src.c]=1;

    while(!q.empty())

    {

        par t= q.front();

        q.pop();

        int r=t.r;

        int c=t.c;

        for(int i=0; i<4; i++)

        {

            int  r1=r+dirr[i];

            int  c1=c+dirc[i];

            if(r1>=1 && r1<=R && c1>=1 && c1<=C && ara[r1][c1] !='#' &&  ara[r1][c1]!='m' && vis[r1][c1]==0) // && visf[r1][c1][num]==0

            {

                vis[r1][c1]=1;

                dis[r1][c1]=dis[r][c]+1;

                q.push({r1,c1});

            }

        }

    }

}

int main()

{

    int tt=0,test;

    cin>>test;

    getchar();

    while(tt<test)

    {

        scanf("%d %d",&R,&C);

        getchar();

        int cnt=0;

        par t;

        for(int i=1; i<=R; i++)

        {

            for(int j=1; j<=C; j++)

            {

                scanf("%c",&ara[i][j]);

                if(ara[i][j]=='a' || ara[i][j]=='b'  || ara[i][j]=='c')

                {

                    vec.push\_back({i,j});

                }

                if(ara[i][j]=='h')

                    t= {i,j};

            }

            getchar();

        }

        int mx=0;

        for(int i=0; i<3; i++)

        {

            ms(vis,0);

            bfs( vec[i] );

            mx= max(mx, dis[ t.r ][ t.c ] );

        }

        printf("Case %d: %d\n",++tt, mx);

        clearr();

    }

    return 0;

}

\*\*\*\*MST Kruskal\*\*\*\*

struct edge {

int u, v, w;

bool operator<(const edge& p) const

{

return w < p.w;

}

};

int pr[MAXN];

vector<edge> e;

int find(int r)

{

return (pr[r] == r) ? r : find(pr[r]);

}

int mst(int n)

{

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

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

pr[i] = i;

int count = 0, s = 0;

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

int u = find(e[i].u);

int v = find(e[i].v);

if (u != v) {

pr[u] = v;

count++;

s += e[i].w;

if (count == n - 1)

break;

}

}

return s;

}

int main()

{

// READ("in");

int n, m;

cin >> n >> m;

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

int u, v, w;

cin >> u >> v >> w;

edge get;

get.u = u;

get.v = v;

get.w = w;

e.push\_back(get);

}

cout << mst(n) << endl;

return 0;

}

\*\*\*\*   
Number of Nodes in a DAG\*\*\*\*

///https://www.hackerrank.com/contests/accel-hack/challenges/acyclic-graph

#include<bits/stdc++.h>

#define i64 long long

#define inf 1000000000000000000

using namespace std;

const int MAXN=5\*10009;

vector<int>adj[MAXN];

int keeps[MAXN],vis[MAXN];

bitset<5\*10009>bset[50009];

void dfs(int src)

{

    vis[src]=1;

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

    {

        int nd=adj[src][i];

        if(vis[nd]==0)

        {

            dfs(nd);

            bset[src]|=bset[nd];

        }

        else bset[src]|=bset[nd];

    }

    bset[src][src]=1;

}

int main()

{

    int n,m,res=0,u,v;

    scanf("%d %d",&n,&m);

    for(int i=0;i<m;i++)

    {

        scanf("%d %d",&u,&v);

        adj[u].push\_back(v);

    }

    for(int i=1;i<=n;i++)

    {

        dfs(i);

        int ret=bset[i].count();

        if(ret\*2>=n) res++;

    }

   printf("%d\n",res);

    return 0;

}

**Linear Algebra And Math**

\*\*\*\* matrix expo\*\*\*\*

#include <iostream>

#include <cassert>

using namespace std;

struct matrix {

  int v[5][5];

  int row, col; // number of row and column

};

int mod = 10000;

// multiplies two matrices and returns the result

matrix multiply(matrix a, matrix b) {

  assert(a.col == b.row);

  matrix r;

  r.row = a.row;

  r.col = b.col;

  for (int i = 0; i < r.row; i++) {

    for (int j = 0; j < r.col; j++) {

      int sum = 0;

      for (int k = 0; k < a.col;  k++) {

        sum += a.v[i][k] \* b.v[k][j];

        sum %= mod;

      }

      r.v[i][j] = sum;

    }

  }

  return r;

}

// returns mat^p

matrix power(matrix mat, int p) {

  assert(p >= 1);

  if(p==0) return indentity mat;

  //if (p == 1) return mat; this one gives wa

  if (p % 2 == 1)

    return multiply(mat, power(mat, p - 1));

  matrix ret = power(mat, p / 2);

  ret = multiply(ret, ret);

  return ret;

}

int main() {

  int tcase;

  int a, b, n, m;

  cin >> tcase;

  while (tcase--) {

    // input routine

    cin >> a >> b >> n >> m;

    // preparing the matrix

    matrix mat;

    mat.row = mat.col = 2;

    memset(mat.v,0,sizeof mat.v);

    mat.v[0][0] = mat.v[0][1] = mat.v[1][0] = 1;

    mat.v[1][1] = 0;

    // preparing mod value

    mod = 1;

    for (int i = 0; i < m; i++) mod \*= 10;

    a %= mod, b %= mod;

    if (n < 3) {

      if (n == 0) cout << a << endl;

      if (n == 1) cout << b << endl;

      if (n == 2) cout << (a+b) % mod << endl;

    } else {

      mat = power(mat, n - 1);

      int ans = b \* mat.v[0][0] + a \* mat.v[0][1];   /// here multiply the whole row with the whole column of the M matrix which is A^(n-2)

      ans %= mod;

      cout << ans << endl;

    }

  }

  return 0;

}

\*\*\*\* iterative BiMod\*\*\*\*

ll expo(ll base, ll exponent, ll mod) {

    ll ans = 1;

    while(exponent !=0 ) {

        if((exponent&1) == 1) {

            ans = ans\*base ;

            ans = ans%mod;

        }

        base = base\*base;

        base %= mod;

        exponent>>= 1;

    }

    return ans%mod;

}

\*\*\*\*\* differentiational equation\*\*\*\*\*

/\*\*http://codeforces.com/contest/932/problem/E

see its tutorial

differentiate this equation and multiply by x:

x^b \* (1+x)^c

= b\*x^b \* (1+x)^c  + c\*x^(b+1)  \* (1+x)^(c-1)

same as diff,  nCr x^r;  = nCr\* r\*x^r;

this can be written as the following dp function

\*/

#include<bits/stdc++.h>

#define i64 long long

#define pii pair<i64,i64>

#define mod 1000000007

using namespace std;

i64 pow1(int x, int n)

{

    if(n==0)

        return 1;

    if(n%2==0)

    {

        i64 res= pow1(x,n/2);

        return (res\*res)%mod;

    }

    return (x\*pow1(x,n-1))%mod;

}

int dp[5001][5001];

int f(int k, int a, int n)

{

    if(k==0)

    {

        i64 res= pow1(2,n-a);

        return (int)res;

    }

    if(dp[k][a]!=-1)

        return dp[k][a];

    int rem=n-a;

    int res= ( (a? 1LL\*a\* f(k-1,a,n) :0LL)  +  (rem? 1LL\*rem\*f(k-1,a+1,n):0LL) )%mod;

    return dp[k][a]= res;

}

int main()

{

    memset(dp,-1,sizeof dp);

    int n,k;

    cin>>n>>k;

    cout<<f(k,0,n)<<endl;

}

\*\*\*\*\*Big Integer Jan vai\*\*\*\*

/\*

    Author       :    Jan

    Problem Name :    Big int for contest

    Algorithm    :

    Complexity   :

\*/

#include <cstdio>

#include <string>

#include <algorithm>

using namespace std;

struct Bigint {

    string a;

    int sign;

    Bigint() {}

    Bigint( string b ) { (\*this) = b; }

    int size() { return a.size(); }

    Bigint inverseSign() { sign \*= -1; return (\*this); }

    Bigint normalize( int newSign ) {

        sign = newSign;

        for( int i = a.size() - 1; i > 0 && a[i] == '0'; i-- ) a.erase(a.begin() + i);

        if( a.size() == 1 && a[0] == '0' ) sign = 1;

        return (\*this);

    }

    void operator = ( string b ) {

        a = b[0] == '-' ? b.substr(1) : b;

        reverse( a.begin(), a.end() );

        this->normalize( b[0] == '-' ? -1 : 1 );

    }

    bool operator < ( const Bigint &b ) const {

        if( a.size() != b.a.size() ) return a.size() < b.a.size();

        for( int i = a.size() - 1; i >= 0; i-- ) if( a[i] != b.a[i] ) return a[i] < b.a[i];

        return false;

    }

    Bigint operator + ( Bigint b ) {

        if( sign != b.sign ) return (\*this) - b.inverseSign();

        Bigint c;

        for( int i = 0, carry = 0; i < (int)a.size() || i < (int)b.size() || carry; i++ ) {

            carry += (i < (int)a.size() ? a[i] - 48 : 0) + (i < (int)b.a.size() ? b.a[i] - 48 : 0);

            c.a += (carry % 10 + 48);

            carry /= 10;

        }

        return c.normalize(sign);

    }

    Bigint operator - ( Bigint b ) {

        if( sign != b.sign ) return (\*this) + b.inverseSign();

        if( (\*this) < b ) return (b - (\*this)).inverseSign();

        Bigint c;

        for( int i = 0, borrow = 0; i < (int)a.size(); i++ ) {

            borrow = a[i] - borrow - (i < b.size() ? b.a[i] : 48);

            c.a += borrow >= 0 ? borrow + 48 : borrow + 58;

            borrow = borrow >= 0 ? 0 : 1;

        }

        return c.normalize(sign);

    }

    Bigint operator \* ( Bigint b ) {

        Bigint c("0");

        for( int i = 0, k = a[i]; i < (int)a.size(); i++, k = a[i] ) {

            while(k-- - 48) c = c + b;

            b.a.insert(b.a.begin(), '0');

        }

        return c.normalize(sign \* b.sign);

    }

    Bigint operator / ( Bigint b ) {

        if( b.size() == 1 && b.a[0] == '0' ) b.a[0] /= ( b.a[0] - 48 ) ;

        Bigint c("0"), d;

        for( int j = 0; j < (int)a.size(); j++ ) d.a += "0";

        int dSign = sign \* b.sign; b.sign = 1;

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

            c.a.insert( c.a.begin(), '0');

            c = c + a.substr( i, 1 );

            while( !( c < b ) ) c = c - b, d.a[i]++;

        }

        return d.normalize(dSign);

    }

    Bigint operator % ( Bigint b ) {

        if( b.size() == 1 && b.a[0] == '0' ) b.a[0] /= ( b.a[0] - 48 ) ;

        Bigint c("0");

        int cSign = sign \* b.sign; b.sign = 1;

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

            c.a.insert( c.a.begin(), '0');

            c = c + a.substr( i, 1 );

            while( !( c < b ) ) c = c - b;

        }

        return c.normalize(cSign);

    }

    void print() {

        if( sign == -1 ) putchar('-');

        for( int i = a.size() - 1; i >= 0; i-- ) putchar(a[i]);

    }

};

int main() {

    Bigint a, b, c;

    a = "511";

    b = "10";

    c = a + b;

    c.print();

    putchar('\n');

    c = a - b;

    c.print();

    putchar('\n');

    c = a \* b;

    c.print();

    putchar('\n');

    c = a / b;

    c.print();

    putchar('\n');

    c = a % b;

    c.print();

    putchar('\n');

    return 0;

}

**String**

\*\*\*\*Hashing\*\*\*\*

///http://codeforces.com/problemset/problem/51/B

//html files, stack wise things

#include<bits/stdc++.h>

#define i64 long long

#define inf 1000000000000000000

using namespace std;

map<string,int>mp;

vector<int>tot;

void init()

{

    mp["<table>"]=3;

    mp["</table>"]=-3;

    mp["<tr>"]=2;

    mp["</tr>"]=-2;

    mp["<td>"]=1;

    mp["</td>"]=-1;

}

vector<int>vec,vec2;

void process(string str)

{

    string ret;

    for(int i=0; i<str.size(); i++)

    {

        if(str[i]=='<')

        {

            ret="";

            while(str[i]!='>'  && i<str.size())

                ret+=str[i], i++;

            ret+=str[i];

            if(mp[ret]!=2 && mp[ret]!=-2)

                vec.push\_back(mp[ret]);

        }

    }

//    for(int i=0; i<vec.size(); i++)

//        printf("%d ",vec[i]);

//

//    puts("");

}

int pos;

int Table1();

int Table2();

int Table2()

{

   // printf("in 2: %d -> %d\n",pos,vec[pos]);

    if(vec[pos+1]==-1)

    {

        pos++;

        //printf("returning from 2\n");

        return 1;

    }

    int res=0;

    while(vec[pos+1]==3)

    {

        pos++;

        res+=Table1();

    }

//    printf("2=>%d\n",res);

//    tot.push\_back(res);

    if(vec[pos+1]==-1)

    {

        pos++;

       // printf("returning from 2\n");

        return res;

    }

}

int Table1()

{

   // printf("in 1: %d -> %d\n",pos,vec[pos]);

    int res=0;

    while(vec[pos+1]==1)

    {

        pos++;

        res+=Table2();

    }

   // printf("1=>%d\n",res);

    tot.push\_back(res);

    if(vec[pos+1]==-3)

    {

       // printf("returning from 1\n");

        pos++;

        return 1;

    }

}

int main()

{

    init();

    //freopen("input.txt","r",stdin);

    string str;

    char ara[6009];

    while((scanf("%s",&ara))!=EOF)

        str+=ara;

    process(str);

    stack<int>stk;

    int res=Table1();

//    cout<<"--->"<<res<<endl;

//    tot.push\_back(res);

   // puts("");

    sort(tot.begin(),tot.end());

    for(int i=0; i<tot.size(); i++)

    {

        if(i) printf(" ");

        printf("%d",tot[i]);

    }

    puts("");

}

\*\*\*\*KMP\*\*\*\*

#include<bits/stdc++.h>

void computeLPSArray(char \*pat, int M, int \*lps);

// Prints occurrences of txt[] in pat[]

void KMPSearch(char \*pat, char \*txt)

{

    int M = strlen(pat);

    int N = strlen(txt);

    // create lps[] that will hold the longest prefix suffix

    // values for pattern

    int lps[M];

    // Preprocess the pattern (calculate lps[] array)

    computeLPSArray(pat, M, lps);

    int i = 0;  // index for txt[]

    int j  = 0;  // index for pat[]

    while (i < N)

    {

        if (pat[j] == txt[i])

        {

            j++;

            i++;

        }

        if (j == M)

        {

            printf("Found pattern at index %d n", i-j);

            j = lps[j-1];

        }

        // mismatch after j matches

        else if (i < N && pat[j] != txt[i])

        {

            // Do not match lps[0..lps[j-1]] characters,

            // they will match anyway

            if (j != 0)

                j = lps[j-1];

            else

                i = i+1;

        }

    }

}

// Fills lps[] for given patttern pat[0..M-1]

void computeLPSArray(char \*pat, int M, int \*lps)

{

    // length of the previous longest prefix suffix

    int len = 0;

    lps[0] = 0; // lps[0] is always 0

    // the loop calculates lps[i] for i = 1 to M-1

    int i = 1;

    while (i < M)

    {

        if (pat[i] == pat[len])

        {

            len++;

            lps[i] = len;

            i++;

        }

        else // (pat[i] != pat[len])

        {

            // This is tricky. Consider the example.

            // AAACAAAA and i = 7. The idea is similar

            // to search step.

            if (len != 0)

            {

                len = lps[len-1];

                // Also, note that we do not increment

                // i here

            }

            else // if (len == 0)

            {

                lps[i] = 0;

                i++;

            }

        }

    }

}

// Driver program to test above function

int main()

{

    char \*txt = "ABABDABACDABABCABAB";

    char \*pat = "ABABCABAB";

    KMPSearch(pat, txt);

    return 0;

}

**MAXFLOW**

\*\*\*\*Bipartite matching\*\*\*\*

int matchR[55],Graph[55][55];

bool vis[55];

struct person

{

    int h,a,d;

} mp[55], fp[55];

bool bpm(int u, int m)  // for each node,  match with m elements in 2nd set

{

    for(int v=1;v<=m;v++)

    {

        if(Graph[u][v]==1 && vis[v]==false)

        {

            vis[v]=true;

            if(matchR[v]<0 || bpm(matchR[v],m))

            {

                matchR[v]=u;

                return true;

            }

        }

    }

    return false;

}

int maxBPM(int n, int m)  // n= number in 1st set,  m is # in another set

{

    memset(matchR, -1, sizeof matchR);

    int result=0;

    for(int u=1;u<=n;u++)

    {

        memset(vis,0,sizeof vis);

        if(bpm(u,m))

            result++;

    }

    return result;

}

\*\*\*\*\*Dinic Implementation 1\*\*\*\*

const int N = 3003;

typedef int T;

struct Edge

{

    int u, v;

    T cap, flow;

    Edge(int u, int v, T c, T f):u(u), v(v), cap(c), flow(f) {}

};

struct Dinic

{

    int n, m, s, t;

    const T oo = 1e9;

    vector<Edge> edge;

    vector<int> G[N];

    bool vis[N];

    int d[N];

    int cur[N];

    void init(int n)

    {

        this->n=n;

        for(int i=0; i<=n; i++)

            G[i].clear();

        edge.clear();

    }

    void addEdge(int u, int v, int cap)

    {

        edge.push\_back(Edge(u, v, cap, 0));

        edge.push\_back(Edge(v, u, cap, 0));

        m=edge.size();

        G[u].push\_back(m-2);

        G[v].push\_back(m-1);

    }

    bool bfs()

    {

        memset(vis, 0, sizeof vis);

        queue<int> q;

        q.push(s);

        d[s]=0;

        vis[s]=1;

        while(!q.empty())

        {

            int x=q.front();

            q.pop();

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

            {

                Edge& e=edge[G[x][i]];

                if(!vis[e.v] && e.cap>e.flow)

                {

                    vis[e.v]=true;

                    d[e.v]=d[x]+1;

                    q.push(e.v);

                }

            }

        }

        return vis[t];

    }

    T dfs(int x, T a)

    {

        if(x==t || a==0)return a;

        T flow=0, f;

        for(int& i=cur[x]; i<G[x].size(); i++)

        {

            Edge& e=edge[G[x][i]];

            if(d[x]+1==d[e.v] && (f=dfs(e.v, min(a, e.cap-e.flow)))>0)

            {

                e.flow+=f;

                edge[G[x][i]^1].flow-=f;

                flow+=f;

                a-=f;

                if(a==0)break;

            }

        }

        return flow;

    }

    T dinitz(int s, int t)

    {

        this->s=s;

        this->t=t;

        int flow=0;

        while(bfs())

        {

            memset(cur, 0, sizeof cur);

            flow+=dfs(s, oo);

        }

        return flow;

    }

} MaxF;

int main() {

      int n;

      int cs = 0;

      while(scanf("%d",&n) && n ) {

            Diii(u,v,m);

            MaxF.init(n);

            forn(i,m) {

                  Diii(a,b,c);

                  MaxF.addEdge(a,b,c);

            }

            printf("Network %d\nThe bandwidth is %lld.\n\n",++cs, MaxF.dinitz(u,v));

      }

      return 0;

}

\*\*\*\* Dinic implementation2\*\*\*\*

const int maxnodes = 5000;

int nodes = maxnodes, src, dest;

int dist[maxnodes], q[maxnodes], work[maxnodes];

struct Edge {

  int to, rev;

  int f, cap;

};

vector<Edge> g[maxnodes];

// Adds bidirectional edge

void addEdge(int s, int t, int cap){

  Edge a = {t, g[t].size(), 0, cap};

  Edge b = {s, g[s].size(), 0, cap};

  g[s].push\_back(a);

  g[t].push\_back(b);

}

bool dinic\_bfs() {

  fill(dist, dist + nodes, -1);

  dist[src] = 0;

  int qt = 0;

  q[qt++] = src;

  for (int qh = 0; qh < qt; qh++) {

    int u = q[qh];

    for (int j = 0; j < (int) g[u].size(); j++) {

      Edge &e = g[u][j];

      int v = e.to;

      if (dist[v] < 0 && e.f < e.cap) {

        dist[v] = dist[u] + 1;

        q[qt++] = v;

      }

    }

  }

  return dist[dest] >= 0;

}

int dinic\_dfs(int u, int f) {

  if (u == dest)

    return f;

  for (int &i = work[u]; i < (int) g[u].size(); i++) {

    Edge &e = g[u][i];

    if (e.cap <= e.f) continue;

    int v = e.to;

    if (dist[v] == dist[u] + 1) {

      int df = dinic\_dfs(v, min(f, e.cap - e.f));

      if (df > 0) {

        e.f += df;

        g[v][e.rev].f -= df;

        return df;

      }

    }

  }

  return 0;

}

int maxFlow(int \_src, int \_dest) {

  src = \_src;

  dest = \_dest;

  int result = 0;

  while (dinic\_bfs()) {

    fill(work, work + nodes, 0);

    while (int delta = dinic\_dfs(src, INT\_MAX))

      result += delta;

  }

  return result;

}

int main() {

    int n = 3;

    nodes = n;

    int capacity[][3] = { { 0, 3, 2 }, { 0, 0, 2 }, { 0, 0, 0 } };

    for (int i = 0; i < n; i++)

        for (int j = 0; j < n; j++)

            if (capacity[i][j] != 0)

                addEdge(i, j, capacity[i][j]);

    cout << (4 == maxFlow(0, 2)) << endl;

}

\*\*\*\*Dinic Zobayer VAi\*\*\*\*

#include<bits/stdc++.h>

#define pb push\_back

#define ms(a,b) memset((a),(b),sizeof(a))

#define i64 long long

#define pii pair<int,int>

#define INF 1000000009

#define in(a) freopen(a,"r", stdin)

#define out(a) freopen(a,"w",stdout)

using namespace std;

int src,snk, nNode,nEdge;

const int MAXN =100009;

char input[105][105];

bool isOccup[MAXN];

int Q[MAXN], fin [MAXN], pro[MAXN], dist[MAXN];

int flow[MAXN], cap[MAXN], nextt[MAXN], to[MAXN];

inline void init(int \_src, int \_snk, int \_n)

{

ms(Q,0);

ms(pro,0);

ms(dist,0);

ms(flow,0);

ms(cap,0);

ms(nextt,0);

ms(to,0);

ms(isOccup,0);

ms(input,0);

src= \_src, snk=\_snk, nNode= \_n, nEdge=0;

ms(fin,-1);

}

inline void addEdge(int u, int v, int \_cap)

{

if(u!=0 && isOccup[v]==true) return;

to[nEdge]=v, cap[nEdge]=\_cap, flow[nEdge]=0;

nextt[nEdge]=fin[u], fin[u]=nEdge++;

to[nEdge]=u, cap[nEdge]=0, flow[nEdge]=0;

nextt[nEdge]=fin[v], fin[v]=nEdge++;

}

bool bfs()

{

int st, en, i,u, v;

ms(dist,-1);

dist[src]=st=en=0;

Q[en++]=src;

while(st<en)

{

u=Q[st++];

for(i=fin[u]; i>=0; i=nextt[i])

{

v=to[i];

// printf("%d %d i=%d cap= %d flo=%d\n",u,v,i,flow[i],cap[i]);

if(flow[i]<cap[i] && dist[v]==-1)

{

dist[v]=dist[u]+1;

Q[en++]=v;

}

}

}

return (dist[snk]!=-1);

}

int dfs(int u, int f1)

{

if(u==snk ) return f1;

for(int &e=pro[u], v, df ; e>=0 ; e=nextt[e])

{

v=to[e];

if(flow[e]< cap[e] && dist[v]==dist[u]+1)

{

// printf("%d to %d?\n",u,v);

df= dfs(v, min(cap[e]-flow[e], f1));

if(df>0)

{

flow[e]+=df;

flow[e^1]-=df;

//cap[e]-=df;

return df;

}

}

}

return 0;

}

int dinitz()

{

int ret=0;

int df;

while(bfs()){

for(int i=0; i<= nNode; i++) pro[i]=fin[i];

int cnt=0;

while(true){

df=dfs(src, INF);

if(df) ret+= (int)df;

else break;

cnt+=df;

}

}

return ret;

}

int main()

{

int tt,test=0;

scanf("%d",&test);

while(tt<test)

{

int n,m;

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

getchar();

int n2=n\*m;

int src= 0, sink= m\*n+n2+1, pep=0;

init(src,sink,2\*n2+1 );

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

{

scanf("%s",&input[i]);

getchar();

}

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

{

for(int j=1; j<=m; j++)

{

char ch;

ch= input[i][j-1];

if(ch=='\*')

{

isOccup[(i-1)\*m+j]=true;

addEdge(src, (i-1)\*m+j, 1);

pep++;

}

}

}

for(int i=0; i<n; i++) /// n row, m col

{

for(int j=1; j<=m; j++)

{

addEdge(i\*m+j, i\*m+j+n2, 1);

if(j>1)

{

addEdge(i\*m+j+n2, i\*m+j-1, 1);

}

if(j<m)

{

addEdge(i\*m+j+n2, i\*m+j+1, 1);

}

if(i>0) ///changed here

{

addEdge(i\*m+j+n2, (i-1)\*m+j, 1);

}

if(i<n-1) /// changed here

{

addEdge(i\*m+j+n2, (i+1)\*m+j, 1);

}

if(i==0 || j==1 || i==n-1 || j==m)

{

addEdge(i\*m+j+n2, sink, 1);

}

}

}

int tot=dinitz();

// printf("tot= %d pep=%d\n",tot,pep);

printf("Case %d: ",++tt);

if(tot==pep) printf("yes\n");

else printf("no\n");

}

return 0;

}

\*\*\*\*FordFulkerSon\*\*\*\*

#include<bits/stdc++.h>

#define pii pair<int,int>

using namespace std;

int Graph[104][104], rGraph[104][104],parent[105];

int n;

bool bfs(int s, int t)

{

    bool vis[104];

    memset(vis,0,sizeof vis);

    queue<int>q;

    q.push(s);

    vis[s]=true;

    parent[s]=-1;

    while(!q.empty())

    {

        int u=q.front();

        q.pop();

        for(int v=1; v<=n; v++)

        {

            if(vis[v]==false  && rGraph[u][v]>0)

            {

                // printf("%d -> %d\n",u,v);

                q.push(v);

                parent[v]=u;

                vis[v]=true;

            }

        }

    }

    return (vis[t] == true);

}

int fordFulkerson(int s, int t)

{

    int u,v;

    for(int i=1; i<=n; i++)

        for(int j=1; j<=n; j++)

            rGraph[i][j]=Graph[i][j];

    int max\_flow=0;

    while(bfs(s,t))

    {

        int path\_flow=INT\_MAX;

        for(v=t ; v!=s ; v=parent[v])

        {

            u= parent[v];

            path\_flow= min(path\_flow,rGraph[u][v]);

        }

        for(v=t; v!=s ; v=parent[v])

        {

            u=parent[v];

            rGraph[u][v]-= path\_flow;

            rGraph[v][u]+= path\_flow;

        }

        // cout<<path\_flow<<endl;

        max\_flow+=path\_flow;

    }

    return max\_flow;

}

int main()

{

    int tt=0,test;

    cin>>test;

    while(tt<test)

    {

        int s,t,c,u,v,w;

        scanf("%d",&n);

        scanf("%d %d %d",&s,&t,&c);

        for(int i=0; i<c; i++)

        {

            scanf("%d %d %d",&u,&v,&w);

            Graph[u][v]+=w;

            Graph[v][u]+=w;

        }

        printf("Case %d: %d\n",++tt,fordFulkerson(s,t));

        memset(Graph,0,sizeof Graph);

        memset(rGraph,0,sizeof rGraph);

        memset(parent,0,sizeof parent);

    }

    return 0;

}