Team notebook

November 14, 2019

Contents

1 1D-1D Optimization

```
//http://www.codah.club/tasks.php?show_task=5000000624
```

```cpp
//Batch Scheduling
#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

#define sf scanf
#define pf printf
#define pb push_back
#define mp make_pair
#define PI ( acos(-1.0) )
#define mod 1000000007
#define IN freopen("nocross.in","r",stdin)
#define OUT freopen("nocross.out","w",stdout)
#define FOR(i,a,b) for(i=a ; i<=b ; i++)
#define DBG pf("Hi\n")
#define INF 2000000000
#define i64 long long int
#define eps (1e-8)
#define xx first
#define yy second
#define LOG 19
#define off 2

using namespace __gnu_pbds;
using namespace std ;

typedef tree< pair<int,int>, null_type, less< pair<int,int> >,
    rb_tree_tag, tree_order_statistics_node_update> ordered_set;

#define maxn 1000005

int dp[maxn] ;
int t[maxn] , f[maxn] ;
int n , s ;

int w( int i, int x )
{
    return ( t[x]-t[i]+s )*( f[n]-f[i] ) ;
}

int main()
{
    scanf("%d %d",&n,&s) ;

    for(int i=1 ; i<=n ; i++)
    {
        scanf("%d %d",&t[i],&f[i]) ;
        t[i] += t[i-1] ;
        f[i] += f[i-1] ;
    }

    vector < pair<int,int> > vp ; // pos , best-k

    vp.pb( mp( 0 , 0 ) ) ;

    for(int x=1 ; x<=n ; x++)
    {
        int idx = upper_bound( vp.begin() , vp.end() , mp( x ,
            n+1 ) ) - vp.begin() ;
        idx-- ;

        dp[x] = dp[ vp[idx].yy ] + w( vp[idx].yy , x ) ;

        while( (int)vp.size() > 0 )
        {
            if( vp.back().xx > x && dp[x] + w( x , vp.back().xx )
                <= dp[ vp.back().yy ] + w( vp.back().yy ,
                vp.back().xx ) ) vp.pop_back();
            else break ;
        }
```

```
        if( vp.size() == 0 ) vp.push_back( mp( 0 , x ) ) ;
        else{

            int lo = max(vp.back().xx,x+1) , hi = n ;

            if( lo > hi || dp[ vp.back().yy ] + w( vp.back().yy ,
                hi ) <= dp[x] + w( x , hi ) ) continue ;

            while( lo < hi )
            {
                int mid = (lo+hi)/2 ;

                if( dp[ vp.back().yy ] + w( vp.back().yy , mid )
                    <= dp[x] + w( x , mid ) ) lo = mid + 1 ;
                else hi = mid ;
            }

            vp.pb( mp( lo , x ) ) ;
        }
    }

    printf("%d\n",dp[n]) ;

    return 0 ;
}
```

2 BlockCutTree

```
//LOJ-1308 Ant Network

#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

#define sf scanf
#define pf printf
#define pb push_back
#define mp make_pair
#define PI ( acos(-1.0) )
#define mod 1000000007LL
#define IN freopen("testing.txt","r",stdin)
#define OUT freopen("output.txt","w",stdout)
#define FOR(i,a,b) for(i=a ; i<=b ; i++)
#define DBG pf("Hi\n")
#define INF 1000000000000000000LL
#define i64 long long int
#define eps (1e-8)
#define xx first
#define yy second
#define ln 17
#define off 2

using namespace __gnu_pbds;
using namespace std ;

typedef pair<pair<i64, i64>,int> pi ;
typedef tree< pi, null_type, less<pi>, rb_tree_tag,
    tree_order_statistics_node_update> ordered_set;

#define maxn 100005

namespace BCT
{
    const int mx = 100005 ; // max( number of edge , number of
        node )
    bool isCutPoint[mx] ;
```

```
int low[mx] , pre[mx] , cnt2vcc , used[mx] , Timer = 0 ;
vector <int> biComp[mx] ;
int n , m ;
struct Edge{
    int v , id ;
};
vector <Edge> g[mx] ;
vector <int> bridges ; //for bridge
stack <int> stk ;

void init(int _n, int _m)
{
    n = _n ; m = _m ;
    for(int i=1 ; i<=max(n,m) ; i++) g[i].clear() ,
        biComp[i].clear() ;
    bridges.clear() ; // for bridge
}

void addEdge( int u, int v, int id )
{
    g[u].pb( {v,id} ) ; g[v].pb({u,id}) ;
}

void dfs(int u , int par )
{
//    printf("node-> %d par: %d edgeId: %d\n",u,par,edgeId) ;
    pre[u] = ++Timer ; low[u] = pre[u] ;

    int chCnt = 0 ;

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

        if( used[ edgeId ] ) continue ;
        used[ edgeId ] = true ;
        stk.push( edgeId ) ;

        int v = g[u][i].v ;

        if( pre[v]==-1 )
        {
            dfs( v , u ) ;

            low[u] = min( low[u] , low[v] ) ;
            if( low[ v ] == pre[ v ] ) bridges.pb(edgeId) ;
            if( low[v] >= pre[u] )
            {
                cnt2vcc++ ;
                while( stk.size() > 0 ) //making component
                {
                    biComp[cnt2vcc].pb( stk .top() ) ;
                    stk.pop() ;
                    if( biComp[cnt2vcc].back()==edgeId ) break
                        ;
                }

                if(par!=0) isCutPoint[u] = true ; //checking
                    if non-root
            }
            chCnt++ ;
        }
        else low[u] = min( low[u] , pre[v] ) ;
    }

    if(chCnt > 1 && par==0) isCutPoint[u] = true ; //checking
        for root

    return ;
```

```
    }

    int find2VCC()
    {
        int i , j ;
        Timer = 0 ;
        for(i=1 ; i<=m ; i++) used[i] = false ;
        for(i=1 ; i<=n ; i++)
        {
            isCutPoint[i] = false ;
            pre[i] = -1 ;
        }
        cnt2vcc = 0 ;
        for(i=1; i<=n ; i++)
        {
            if( pre[i]==-1 ) dfs(i,0) ;
        }
    }
}

struct Edge{
    int u , v , id ;
}edge[maxn];

int main()
{
    int i , j , k , l , m , n , t=1 , tc ;

    scanf("%d",&tc) ;

    while( t<=tc )
    {
        scanf("%d %d",&n,&m) ;

        BCT::init(n,m) ;

        for(i=1; i<=m ; i++)
        {
            int u , v ;
            scanf("%d %d",&u,&v) ;
            u++ ; v++ ;
            edge[i] = {u,v,i} ;
            BCT::addEdge(u,v,i) ;
        }

        BCT::find2VCC() ;

        int cntVcc = BCT::cnt2vcc ;

        int ans1 ;
        unsigned long long int ans2 ;

        if( cntVcc==1 )
        {
            ans1 = 2 ; ans2 = (n*(n-1))/2LL ;
        }
        else{
            ans1 = 0 , ans2=1LL ;
            for(i=1; i<=cntVcc ; i++)
            {
                set <int> nodes ;
                for(j=0 ; j<BCT::biComp[i].size() ; j++)
                {
                    int id= BCT::biComp[i][j] ;
                    nodes.insert( edge[id].u ) ; nodes.insert(
                        edge[id].v ) ;
                }
                set<int> :: iterator it = nodes.begin() ;
                int artCnt = 0 ;
                while( it!=nodes.end() )
```

```
            {
                if( BCT::isCutPoint[*it] ) artCnt++ ;
                it++ ;
            }
            if( artCnt==1 )
            {
                ans1++ ;
                ans2 *= (1LL*( (int)nodes.size() - artCnt )) ;
            }
        }

    }

    printf("Case %d: %d %llu\n",t++,ans1,ans2) ;

/*      for(i=1 ; i<=n ; i++)
    {
        printf("%d-->%d\n",i,BCT::isCutPoint[i]) ;
    }
    for(i=1 ; i<=cntVcc ; i++)
    {
        for(j=0 ; j<BCT::biComp[i].size() ; j++ ) printf("%d
            ",BCT::biComp[i][j]) ;
        printf("\n") ;
    }
*/
  }

  return 0 ;
}
/*
14 17
1 2
1 3
2 3
3 4
4 6
5 6
3 5
6 7
6 8
7 8

9 10
9 11
9 12
11 12
10 13
13 14
10 14
*/
```

3 Blossom(randomized)

```
#include <bits/stdc++.h>
using namespace std;
const int N=510;
vector<int>to[N];
int lnk[N],vis[N],tim=0;
inline void ae(int u,int v){
    to[u].push_back(v);
}
bool dfs(int x){
    if(x==0)return true;
    vis[x]=tim;
    vector<int>::iterator it=to[x].begin(),ti=to[x].end();
    random_shuffle(it,ti);
    for(int u,v;it!=ti;it++){
```

```
                u=*it,v=lnk[u];
                if(vis[v]<tim){
                        lnk[x]=u,lnk[u]=x,lnk[v]=0;
                        if(dfs(v))return true;
                        lnk[u]=v,lnk[v]=u,lnk[x]=0;
                }
        }
        return false;
}
int main(){
        int n,e,ans=0;
        scanf("%d%d",&n,&e);
        for(int u,v;e--;scanf("%d%d",&u,&v),ae(u,v),ae(v,u));
        srand(time(0));
        memset(lnk+1,0,n<<2);
        for(int tot=5;tot--;){
                for(int i=1;i<=n;i++){
                        if(!lnk[i]){
                                tim++,ans+=dfs(i);
                        }
                }
        }
        printf("%d\n",ans);
        for(int i=1;i<=n;i++){
                printf("%d ",lnk[i]);
        }
        putchar('\n');
        return 0;
}
```

4 Blossom

```
const int MAXN = 2020 + 1;
```

```
struct GM  // 1-based Vertex index
{
    int vis[MAXN], par[MAXN], orig[MAXN], match[MAXN],
        aux[MAXN], t, N;
    vector<int> conn[MAXN];
    queue<int> Q;
    void addEdge(int u, int v)
    {
        conn[u].push_back(v);
        conn[v].push_back(u);
    }
    void init(int n)
    {
        N = n;
        t = 0;
        for(int i=0; i<=n; ++i)
        {
            conn[i].clear();
            match[i] = aux[i] = par[i] = 0;
        }
    }

    void augment(int u, int v)
    {
        int pv = v, nv;
        do
        {
            pv = par[v];
            nv = match[pv];
            match[v] = pv;
            match[pv] = v;
            v = nv;
        }
        while(u != pv);
    }
```

```
int lca(int v, int w)
{
    ++t;
    while(true)
    {
        if(v)
        {
            if(aux[v] == t) return v;
            aux[v] = t;
            v = orig[par[match[v]]];
        }
        swap(v, w);
    }
}
void blossom(int v, int w, int a)
{
    while(orig[v] != a)
    {
        par[v] = w;
        w = match[v];
        if(vis[w] == 1) Q.push(w), vis[w] = 0;
        orig[v] = orig[w] = a;
        v = par[w];
    }
}
bool bfs(int u)
{
    fill(vis+1, vis+1+N, -1);
    iota(orig + 1, orig + N + 1, 1);
    Q = queue<int> ();
    Q.push(u);
    vis[u] = 0;
    while(!Q.empty())
    {
        int v = Q.front();
        Q.pop();
        for(int x: conn[v])
        {
            if(vis[x] == -1)
            {
                par[x] = v;
                vis[x] = 1;
                if(!match[x]) return augment(u, x), true;
                Q.push(match[x]);
                vis[match[x]] = 0;
            }
            else if(vis[x] == 0 && orig[v] != orig[x])
            {
                int a = lca(orig[v], orig[x]);
                blossom(x, v, a);
                blossom(v, x, a);
            }
        }
    }
    return false;
}
int Match()
{
    int ans = 0;
    //find random matching (not necessary, constant
        improvement)
    vector<int> V(N-1);
    iota(V.begin(), V.end(), 1);
    shuffle(V.begin(), V.end(), mt19937(0x94949));
    for(auto x: V) if(!match[x])
        {
            for(auto y: conn[x]) if(!match[y])
                {
                    match[x] = y, match[y] = x;
```

```
                    ++ans;
                    break;
                }
            }
        for(int i=1; i<=N; ++i) if(!match[i] && bfs(i)) ++ans;
        return ans;
    }
};
```

5 CHT Linear(example)

```
#include <bits/stdc++.h>

//#include <ext/pb_ds/assoc_container.hpp>
//#include <ext/pb_ds/tree_policy.hpp>

#define sf scanf
#define pf printf
#define pb push_back
#define mp make_pair
#define IN freopen("input.txt","r",stdin)
#define OUT freopen("output.txt","w",stdout)
#define FOR(i,a,b) for(i=a ; i<=b ; i++)
#define DBG pf("Hi\n")
#define INF 1000000000
#define i64 long long int
#define eps (1e-8)
#define xx first
#define yy second
#define ln 17
#define off 2

//using namespace __gnu_pbds;
using namespace std ;

typedef pair<int, int> pi ;
//typedef tree< pi, null_type, less<pi>, rb_tree_tag,
   tree_order_statistics_node_update> ordered_set;

const i64 mod = 1000000007LL ;

#define maxn 200005

/*
     *os.find_by_order(k) -> returns the k'th smallest element
        (indexing starts from 0)
      os.order_of_key(v) -> returns how many elements are
         strictly smaller than v
*/

struct Line{
    i64 m , c , id ;
    i64 f(i64 x){ return x*m + c ; }
    i64 iSect( Line other ) {
        // floored division
        i64 a = c - other.c , b = other.m - m ;
              return a / b - ((a ^ b) < 0 && a % b);
    }
    bool operator<(const Line& o)const{
       return mp(c,id) < mp(o.c,o.id) ;
    }
};

//for minimization
// slope increasing , query point increasing

struct chtLinear{
```

```
    deque <Line> dq ;

    void addLine( i64 m , i64 c , i64 id )
    {
        if( !dq.empty() && m==dq.front().m ) dq[0] = min( dq[0] ,
            {m,c,id} ) ;
        else dq.push_front( {m,c,id} ) ;
        Line l = dq[0] ; dq.pop_front() ;
        while( (int)dq.size() >= 2 && l.iSect(dq[0]) >
            dq[0].iSect(dq[1]) ) dq.pop_front() ;
        dq.push_front(l) ;
     }
    pair<i64,i64> f(i64 x)
    {
        while( (int)dq.size() > 1 && dq[0].f(x) >= dq[1].f(x) )
            dq.pop_front() ;
        return mp( dq[0].id , dq[0].f(x) ) ;
    }
}ch;

int main()
{
/*
    cout<<dq[0]<<" "<<dq[1]<<endl ;

    ch.addLine( 1 , -1 , 1 ) ;
    ch.addLine( -1, 1 , 2 ) ;

    pair<ll,ll> p1 = ch.query(0) , p2 = ch.query(1) , p3 =
        ch.query(2) ;

    cout<<p1.xx<<" "<<p2.xx<<" "<<p3.xx<<endl ;
    cout<<p1.yy<<" "<<p2.yy<<" "<<p3.yy<<endl ;

*/
```

```
    i64 n , m ;
    scanf("%lld %lld",&n,&m) ;

    ch.addLine(0,0,1) ;

    i64 S = 0 , B = 0, Sum = n ;

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

        scanf("%d",&q) ;

        if(q==1)
        {
            S = 0 , B = 0 ;
            ch.dq.clear() ;
            ch.addLine(0,0,1) ;
            i64 a ;
            scanf("%lld",&a) ;
            Sum += a ;
        }
        if(q==2)
        {
            i64 a ;
            scanf("%lld",&a) ;
            ch.addLine( (Sum) , ( - B - Sum*S) , Sum+1 ) ;
            Sum += a ;
        }
        if(q==3)
        {
            i64 b, s;
            scanf("%lld %lld",&b,&s) ;
            B += b ;
            S += s ;
```

```
        }
        pair<i64,i64> p = ch.f(S) ;
        printf("%lld %lld\n",p.xx, p.yy + B) ;
    }

    return 0 ;
}
/*
5 100
3 4 5
2 1
*/
```

6 ConncetedComponentDP

```
#include <bits/stdc++.h>

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

#define sf scanf
#define pf printf
#define pb push_back
#define mp make_pair
#define PI ( acos(-1.0) )
#define mod 1000000007LL
#define IN freopen("strings.in","r",stdin)
#define OUT freopen("strings.out","w",stdout)
#define FOR(i,a,b) for(i=a ; i<=b ; i++)
#define DBG pf("Hi\n")
#define INF 1000000000
#define i64 long long int
#define eps (1e-8)
#define xx first
#define yy second
#define ln 17
#define off 2

using namespace __gnu_pbds;
using namespace std ;

typedef pair<int, int> pi ;
typedef tree< pi, null_type, less<pi>, rb_tree_tag,
    tree_order_statistics_node_update> ordered_set;

i64 dp[102][32][12][12][2][2] , M ;
int a[102] ;

i64 f( int n ,int r , int k , int c , int st , int en )
{
    if( n==0 || c < 0 || st < 0 || en < 0 ) return 0 ;
    r = ( r + a[n]*( 2*c + st + en ) )%M ;
 //  printf("%d %d %d %d %d %d\n",n,r,k,c,st,en) ;
    if( dp[n][r][k][c][st][en] != -1 ) return
        dp[n][r][k][c][st][en] ;

//    int rem = r ;

    i64 ans = f(n-1,r,k,c,st,en) ; /* it is not used */

    if( k==1 )
    {
        if( c==0 && (st||en) && r==0 ) ans++ ; /* if this is the
            last element to take
        then is should either connect st and en , or be the first
            element or last */
    }
    else{
```

```
        if(st==0) ans += ( f( n-1, r , k-1 , c , 1 , en ) + f(
            n-1 , r , k-1 , c-1 , 1 , en )*c ) ; // this is
            starting element
        if(en==0) ans += ( f(n-1,r,k-1,c,st,1) + f(
            n-1,r,k-1,c-1,st,1)*c ) ; // this is ending element

        ans += f( n-1 , r , k-1 , c+1 , st , en ) ; // created &
            independent
        ans += f( n-1 , r , k-1, c , st , en)*2*c +
            f(n-1,r,k-1,c,st,en)*(st+en) ; /* created and
            connected with some other component
        possibly start or end component */
        ans += (f( n-1 , r , k-1 , c-1 , st , en)*c*(c-1) + f(
            n-1 , r , k-1, c-1 , st, en )*c*(st+en) ) ;/* created
        and connected between two component */

    }

    return dp[n][r][k][c][st][en] = ans%mod ;
}

int main()
{
    int n , k ;

    scanf("%d %d %d",&n,&M,&k) ;

    for(int i=1 ; i<n ; i++) scanf("%d",&a[i]) ;

    a[n] = 0 ;

    memset(dp,-1,sizeof(dp)) ;

    i64 ans = f( n , 0 , k , 0 , 0 , 0 ) ;

    ans = (ans%mod + mod)%mod ;

    printf("%lld\n",ans) ;

    return 0 ;
}
```

7 DominatorTree

```
#include <bits/stdc++.h>

#define mp make_pair
#define pb push_back
#define xx first
#define yy second
#define i64 long long int
#define MEM(a,x) memset(a,x,sizeof(x))
#define INF 1000000000
#define maxn 200005


using namespace std ;

namespace DominatorTree{
/*
Dominator Tree for General Graph ,Tr[u] stores all the
immediate children of node u (does not store the parent) in
the dominator tree. at first initialize with number of nodes.
then add edges(directed edges). call buildDominatorTree(r) ,
where r is the root. then just call dominator(u,v) to check
if v is u's dominator it returns false in case either u or v
is not connected to the root
```

```
*/
const int N = 202400;
vector <int> G[N] , pred[N] , dom[N] , Tr[N] ;
int old[N] , dfn[N] , up[N] , f[N] , semi[N] , g[N] , idom[N] ,
    cnt ;
int n , m ;
int Time , st[N] , en[N] ;

void init(int _n) {
    for (int i = 0 ; i < N ; i++) G[i].clear() , pred[i].clear()
        , dom[i].clear() , Tr[i].clear() ;
    memset (old , 0 , sizeof old) ;
    memset (dfn , 0 , sizeof dfn) ;
    memset (f , 0 , sizeof f) ;
    memset (up , 0 , sizeof up) ;
    memset (old , 0 , sizeof old) ;
    memset (g , 0 , sizeof g) ;
    memset (idom , 0 , sizeof idom) ;
    memset (st , -1 , sizeof st) ;
    memset (en , -1 , sizeof en) ;
    n = _n ;
    cnt = 0 ; Time = 0 ;
}

void addEdge(int u , int v){ return G[u].push_back(v) ; }

void dfs(int u){
        old[dfn[u]=++cnt] = u ;
        semi[cnt] = g[cnt] = f[cnt] = cnt;
        for(int v : G[u]){
                if(!dfn[v]){
                        dfs(v);
                        up[dfn[v]] = dfn[u];
                }
                pred[dfn[v]].push_back(dfn[u]);
        }
}

int ff(int x) {
        if(x == f[x]) return x;
        int y = ff(f[x]) ;
        if(semi[g[x]] > semi[g[f[x]]])
                g[x] = g[f[x]];
        return f[x] = y;
}

void dfs1(int u)
{
    Time++ ;
    st[u] = Time ;
    for(int i=0 ; i<Tr[u].size() ; i++)
    {
        dfs1( Tr[u][i] ) ; //par is not stored in Tr[u]
    }
    Time++ ;
    en[u] = Time ;
}

void buildDominatorTree(int r){
        dfs(r);
        for(int y = cnt ; y >= 2 ; y--){
                for(int z : pred[y]) {
                        ff(z);
                        semi[y]=min(semi[y],semi[g[z]]);
                }
                dom[semi[y]].push_back(y);
                int x=f[y]=up[y];
                for(int z:dom[x]){
                        ff(z);
                        idom[z]=semi[g[z]]<x? g[z]:x;
```

```
            }
            dom[x].clear();
        }
        for(int y = 2 ; y <= cnt ; ++y){
            if(idom[y]!=semi[y])
                idom[y]=idom[idom[y]];
            dom[idom[y]].push_back(y);
        }
        idom[r] = 0 ;
        for (int i = 1 ; i <= n ; i++) {
        for (int j = 0 ; j < dom[i].size() ; j++) {
            Tr[old[i]].push_back(old[dom[i][j]]) ;
        }
        }
        dfs1(r) ;
    }
    bool dominator( int u,int v )
    {
        //returns true if v is u's dominator
        if(st[u]==-1 || st[v]==-1) return false ;//if u or v is
            not connected to the root
        if( st[u] >= st[v] && st[u]<= en[v] ) return true ;
        return false ;
    }
}

vector <int> g[maxn], e[maxn] ;
int dis[maxn] ;
void dijkstra(int n)
{
    int i, j, k, l, m ;
    priority_queue < pair<int,int> > pq ;
    for(i=1 ; i<=n ; i++ ) dis[i] = INF ;
    dis[1] = 0 ;
    pq.push( mp(0,1) ) ;
    while( !pq.empty() )
    {
        pair<int,int> p = pq.top() ;
        pq.pop() ;
        if( dis[p.yy] != -p.xx ) continue ;
        for(i=0 ; i<g[p.yy].size() ; i++)
        {
            int city = g[ p.yy ][i] ;

            if( dis[city] > -p.xx + e[ p.yy][i] )
            {
                dis[city] = -p.xx + e[ p.yy][i] ;
                pq.push( mp( -dis[city], city ) ) ;
            }
        }
    }
    return ;
}

struct data
{
    int u, v, w ;
} edge[maxn];

int main()
{
    int i, j, k, l, m, n, t=1, tc ;

    scanf("%d",&tc);

    while(t<=tc)
    {
        scanf("%d %d",&n,&m) ;

        for(i=1 ; i<=n ; i++) g[i].clear(), e[i].clear() ;
```

```
    for(i=1 ; i<=m ; i++)
    {
        int u , v , w ;
        scanf("%d %d %d",&u,&v,&w) ;

        edge[i].u = u ; edge[i].v = v ; edge[i].w = w ;
        g[ u ].pb( v ) ;
        e[ u ].pb( w ) ;
    }

    dijkstra(n) ;

//      for(i=1 ; i<=n ; i++) printf("dis[%d]: %d\n",i,dis[i]) ;

    DominatorTree::init(n+m) ;
    for(i=1 ; i<=m ; i++)
    {
        int u = edge[i].u , v = edge[i].v , w=edge[i].w ;
        if( dis[u] + w == dis[ v ] )
        {
            DominatorTree::addEdge(u,n+i) ;
            DominatorTree::addEdge(n+i,v) ;
        }
    }
    DominatorTree::buildDominatorTree(1) ;

 /*     for(i=1; i<=n+m ; i++)
    {
        printf("%d:",i) ;
        for(j=0 ; j<dmt.Tr[i].size() ; j++) printf("
            %d",dmt.Tr[i][j]) ;
        printf("\n") ;
    }
    for(i=1 ; i<=n+m ; i++) printf("st[%d]: %d en[%d]:
        %d\n",i,dmt.st[i],i,dmt.en[i]) ; */

    long long int ans = 0LL ;

    for(i=1 ; i<=m ; i++)
    {
        int u = edge[i].u, v = edge[i].v, w = edge[i].w ;

        long long int a, b ;

        if( dis[u] == dis[v] )
        {
            if(w==0)
            {
                b = 0 ;

                if( DominatorTree::dominator(v,n+i) )
                {
                    a = 0 ;
                }
                else a = -1 ;
            }
            else
            {
                a = -1 ;
                b = w ;
            }
        }
        else
        {
            if( dis[u] + w == dis[v] )
            {
                b = 0 ;
                if( DominatorTree::dominator(v,n+i) ) a = 0 ;
```

```
                else a = -1 ;
            }
            else
            {
                a = -1 ;
                b = dis[u] + w - dis[v] ;
            }
        }

        ans += (i64)i*a + (i64)i*(i64)i*b ;

    //    printf("%d %lld %lld\n",i,a,b) ;
    }
    printf("Case %d: %lld\n",t++,ans) ;
}

    return 0 ;
}
```

8 EulerPath

```
#include <bits/stdc++.h>

//#include <ext/pb_ds/assoc_container.hpp>
//#include <ext/pb_ds/tree_policy.hpp>

#define sf scanf
#define pf printf
#define pb push_back
#define mp make_pair
#define IN freopen("testing.txt","r",stdin)
#define OUT freopen("output.txt","w",stdout)
#define FOR(i,a,b) for(i=a ; i<=b ; i++)
#define DBG pf("Hi\n")
#define INF 1000000000
#define i64 long long int
#define eps (1e-8)
#define xx first
#define yy second
#define ln 17
#define off 2

//using namespace __gnu_pbds;
using namespace std ;

typedef pair<int, int> pi ;
//typedef tree< pi, null_type, less<pi>, rb_tree_tag,
   tree_order_statistics_node_update> ordered_set;

const i64 mod = 1000000007LL ;

#define maxn 500005

/*
     *os.find_by_order(k) -> returns the k'th smallest element
        (indexing starts from 0)
      os.order_of_key(v) -> returns how many elements are
         strictly smaller than v
*/

int c[maxn] , d[maxn] ;

map< int , multiset<int> > g ;
map <int,int> vis ;


void dfs1(int u)
{
```

```
    vis[u] = 1 ;
    for( auto v : g[u] ) if( vis.find(v) == vis.end() ) dfs1(v) ;
}

///--------Euler path printing-----------///
//just call dfs2 with the node you want to start your path
//at first you need to make sure, the graph is connected and
    euler path exist
vector <int> ans ;
void dfs2(int u)
{
    while( (int)g[u].size() !=0 )
    {
        int v = *g[u].begin() ;
        g[u].erase( g[u].find(v) ) ;
        g[v].erase( g[v].find(u) ) ;
        dfs2(v) ;
    }
    ans.pb(u) ;
}
///--------Euler path printing-----------///


int main()
{
    int n ;

    scanf("%d",&n) ;

    for(int i=1 ; i<n ; i++) scanf("%d",&c[i]) ;
    for(int i=1 ; i<n ; i++) scanf("%d",&d[i]) ;

    for(int i=1 ; i<n ; i++)
    {
        if( c[i] > d[i] )
        {
            printf("-1\n") ;
            return 0 ;
        }
        g[ c[i] ].insert( d[i] ) ;
        g[ d[i] ].insert( c[i] ) ;
    }

    int src = c[1] , cnt = 0 ;

    for( auto it : g )
    {
        if( (int)it.second.size() & 1 )
        {
            cnt++ ;
            src = it.first ;
        }
    }

    dfs1( src ) ;

    if( vis.size() != g.size() || ( cnt!=0 && cnt!=2 ) )
    {
        printf("-1\n") ;
        return 0 ;
    }

    //call for printing euler path
    dfs2(src) ;

    for(int i=0 ; i<ans.size() ; i++)
    {
        printf("%d",ans[i]) ;
        if( i == (int)ans.size() - 1 ) printf("\n") ;
        else printf(" ") ;
```

```
    }

    return 0 ;
}
```

9 Four Divisor Trick

```
/*
Let S(v,m) be the count of integers in the range 2..v that
    remain after sieving with all primes smaller
or equal than m. That is S(v,m) is the count of integers up to v
    that
are either prime or the product of primes larger than m.

S(v, p) is equal to S(v, p-1) if p is not prime or v is smaller
    than p2. Otherwise (p prime, p2<=v) S(v,p)
can be computed from S(v,p-1) by finding the count of integers
    that are removed while sieving with p.
An integer is removed in this step if it is the product of p
    with another integer that has no
divisor smaller than p. This can be expressed as

S(v,p)= S(v, p1 )    ( S(v/p,  p1 )    S( p1 , p1 ) ).

During computation of S(N,p) it is sufficient to compute S(v,p)
    for all positive integers v that
are representable as floor(N/k) for some integer k and all p
    v1/2.

NOTE: Pi(N) = S(N,N). When you call S(N,N) it will need to
    compute S(N/k,N/k) in its lower substate.
Hence you can have all required values of Pi(N/k).

In my code DSUM(N,P) do the job of calculating S(N,P).

I used two arrays H[] and L[] for storing the values of S(N/k,p)
    in H[k] for k<=N1/2 and
for k>=N^(1/2) I stored the values of S(N/k,p) in L[N/k].

For computation I started with p=2 and changed the values of all
    the state which is going to
be affected by this particular prime. Continue doing this for
    all prime till N1/2 and
at the end for k<=N1/2, H[k] will contain the values of Pi(N/k)
    and L[k] will contain the values of Pi(k).

PS: My complexity is O(N3/4).
*/
#include <bits/stdc++.h>

#define PI (acos(-1.0))
#define DBG printf("Hi\n")
#define loop(i,n) for(i =1 ; i<=n; i++)
#define mp make_pair
#define pb push_back
#define mod 998244353LL
#define INF 1000000000
#define xx first
#define yy second
#define sq(x) ((x)*(x))
#define eps 0.0000000001
#define i64 long long int
#define ui64 unsigned long long int

using namespace std ;

#define maxn 1000000
```

```
i64 Lo[maxn+5] , Hi[maxn+5] ;

void primeCount( i64 N )
{
    i64 i , j , k , l , m ;

    i64 s = sqrt(N+0.0) + 1 ;

    for(i=1 ; i<=s ; i++) Lo[i] = i-1 ;
    for(i=1 ; i<=s ; i++) Hi[i] = (N/i) - 1 ;

    for(i=2 ; i<=s ; i++)
    {
        if( Lo[i] == Lo[i-1] ) continue ;

        i64 isq = i*i , lim = N/isq ;

        // we need , ( N/j ) >= i*i
        //  => j <= ( N/(i*i) )

        for( j=1 ; j<=lim && j<=s ; j++ )
        {
            if( i*j > s ) Hi[j] = Hi[j] - ( Lo[N/(i*j)] - Lo[i-1]
                ) ;
            else Hi[j] = Hi[j] - ( Hi[i*j] - Lo[i-1] ) ;
        }


        // j >= i*i
        for( j=s ; j>=isq ; j-- )
        {
            Lo[j] = Lo[j] - ( Lo[j/i] - Lo[i-1] ) ;
        }
    }
    return ;
}


int main()
{
    i64 i , j , k , l , m , n ;

    scanf("%lld",&n) ;

    primeCount(n) ;

    i64 ans = 0LL ;

    for(i=2 ; (i*i*i)<=n ; i++)
    {
        if( Lo[i]!=Lo[i-1] ) ans++ ;
    }

    for(i=2 ; i*i <=n ; i++)
    {
        if( Lo[i]==Lo[i-1] ) continue ;
        ans += (Hi[i] - Lo[i] ) ;
    }

    printf("%lld\n",ans ) ;
    return 0 ;
}
```

10 HLD

```
#include <bits/stdc++.h>

#include <ext/pb_ds/assoc_container.hpp>
```

```cpp
#include <ext/pb_ds/tree_policy.hpp>

#define sf scanf
#define pf printf
#define pb push_back
#define mp make_pair
#define IN freopen("transposition-115.txt","r",stdin)
#define OUT freopen("dichromatic.out","w",stdout)
#define FOR(i,a,b) for(i=a ; i<=b ; i++)
#define DBG pf("Hi\n")
#define INF 2000000000
#define i64 long long int
#define eps (1e-8)
#define xx first
#define yy second
#define sq(x) ((x)*(x))

using namespace __gnu_pbds;
using namespace std ;

#define maxn (1<<18)+5
#define mod 1000000007LL

typedef pair<i64,i64> pii ;
typedef long long int T ;

struct edge{
    int u , v , c , id ;
    bool operator<(const edge other)const{ return c < other.c ; }
};

namespace MST{
    int par[maxn] ;
    int findPar(int u)
    {
        if( par[u] != u ) return par[u] = findPar( par[u] ) ;
        return u ;
    }

    void findMST( int n , vector < edge > &e )
    {
        //after this function , e[0] to e[n-2] will contain the
            treeEdges
        //and other would be non-tree Edges

        sort( e.begin() , e.end() ) ;
        for(int i=1 ; i<=n ; i++) par[i] = i ;

        vector <edge> treeEdge , otherEdge ;

        for(int i=0 ; i< e.size() ; i++ )
        {
            int u = e[i].u , v = e[i].v ;
            int pu = findPar(u) , pv = findPar(v) ;
            if( pu==pv ) otherEdge.pb(e[i]) ;
            else{
                par[pu] = pv ;
                treeEdge.pb(e[i]) ;
            }
        }
        e.clear() ;
        for(int i=0 ; i<treeEdge.size() ; i++) e.pb( treeEdge[i]
            ) ;
        for(int i=0 ; i<otherEdge.size() ; i++) e.pb(
            otherEdge[i] ) ;
    }
}

int tr[2*maxn] , lazy[2*maxn] , ara[maxn] ;
```

```
void relax(int cn, int b, int e)
{
    tr[cn] = min( tr[cn] , lazy[cn] ) ;
    if( b!=e )
    {
        int lc = cn<<1 , rc = lc+1 , m = (b+e)>>1 ;
        lazy[lc] = min( lazy[cn] , lazy[lc] ) ;
        lazy[rc] = min( lazy[cn] , lazy[rc] ) ;
    }
    lazy[cn] = INF ;
}

void update(int cn, int b , int e, int l, int r,int val)
{
    if( lazy[cn] != INF ) relax(cn,b,e) ;
    if( e < l || b > r ) return ;
    if( l<=b && e<=r )
    {
        lazy[cn] = val ;
        relax(cn,b,e) ;
        return ;
    }
    int lc = cn<<1 , rc = lc+1 , m = (b+e)>>1 ;
    update(lc,b,m,l,r,val) ; update(rc,m+1,e,l,r,val) ;
    tr[cn] = max(tr[lc],tr[rc]) ;
}

int query(int cn, int b, int e, int l, int r)
{
    if(lazy[cn]!= INF ) relax(cn,b,e) ;
    if( e < l || b > r ) return -INF;
    if( l<=b && e<=r ) return tr[cn] ;
    int lc = cn<<1 , rc = lc+1 , m = (b+e)>>1 ;
    return max( query(lc,b,m,l,r) , query(rc,m+1,e,l,r) ) ;
}
```

```
void build(int cn, int b, int e)
{
    lazy[cn] = INF ;
    if( b==e )
    {
        tr[cn] = ara[b] ;
        return ;
    }
    int lc = cn<<1 , rc = lc+1 , m = (b+e)>>1 ;
    build(lc,b,m) ; build(rc,m+1,e) ;
    tr[cn] = max(tr[lc],tr[rc]) ;
}

namespace hld{
    int in[maxn] , out[maxn] , sub[maxn] , t = 1 , nxt[maxn] ,
        depth[maxn], par[maxn] , n ;
    vector <int> g[maxn] ;
    void init(int _n)
    {
        n = _n ;
        for(int i=0 ; i<=n ; i++) g[i].clear() ;
    }

    void addEdge(int u, int v)
    {
        g[u].pb(v) ;
        g[v].pb(u) ;
    }

    void dfsSZ(int u)
    {
        sub[u] = 1 ;

        for(int i=0 ; i<g[u].size() ; i++)
```

```
    {
        int v = g[u][i] ;
        for(int j=0 ; j<g[v].size() ; j++)
        {
            if( g[v][j] == u )
            {
                g[v].erase( g[v].begin() + j ) ;
                break ;
            }
        }
        dfsSZ(v) ;
        sub[u] += sub[v] ;
        if( sub[v] > sub[ g[u][0] ] ) swap( g[u][0] , g[u][i]
            ) ;
    }
}

void dfsHLD(int u)
{
    in[u] = ++t ;
    for(int i=0 ; i<g[u].size() ; i++)
    {
        int v = g[u][i] ;
        par[v] = u ;
        depth[v] = depth[u] + 1 ;
        if( i==0 ) nxt[v] = nxt[u] ;
        else nxt[v] = v ;

        dfsHLD(v) ;
    }
    out[u] = t ;
}

void preprocess(int root)
{
    dfsSZ(root) ;
    t = 0 ; nxt[root] = root ;
    depth[root] = 1 ;
    dfsHLD(root) ;
}

int hldQuery( int u , int v )
{
    int ans = -INF ;
    while( nxt[u] != nxt[v] )
    {
        if( depth[ nxt[u] ] < depth[ nxt[v] ] )
        {
            ans = max( ans , query(1,1,n, in[ nxt[v] ] , in[v]
                ) ) ;
            // do you thing here ( from in[v] to in[ nxt[v] ] )
            v = par[ nxt[v] ] ;
        }
        else{
            ans = max( ans , query(1,1,n, in[ nxt[u] ] , in[u]
                ) ) ;
            // do your thing here ( from in[u] to in[ nxt[u] ]
                )
            u = par[ nxt[u] ] ;
        }
    }
    int lc ;
    if( depth[u] > depth[v] ) swap(u,v) ;
    lc = u ;

    //here lc is the lca
    //if you are working on node , not on edge, then
        update/query upto u also
    //otherwise update/query from in[u]+1 to in[v]
```

```
    ans = max( ans , query(1,1,n,in[u]+1,in[v]) ) ;

    return ans ;
}

void hldUpdate( int u , int v , int val )
{
    while( nxt[u] != nxt[v] )
    {
        if( depth[ nxt[u] ] < depth[ nxt[v] ] )
        {
            update(1,1,n,in[ nxt[v] ] , in[v] , val ) ;
            // do you thing here ( from in[v] to in[ nxt[v] ] )
            v = par[ nxt[v] ] ;
        }
        else{
            update(1,1,n,in[ nxt[u] ] , in[u] , val ) ;
            // do your thing here ( from in[u] to in[ nxt[u] ]
                )
            u = par[ nxt[u] ] ;
        }
    }
    int lc ;
    if( depth[u] > depth[v] ) swap(u,v) ;
    lc = u ;

    //here lc is the lca
    //if you are working on node , not on edge, then
       update/query upto u also
    //otherwise update/query from in[u]+1 to in[v]

    update(1,1,n,in[u]+1,in[v],val) ;

    return ;
}
```

```
}

int ans[maxn] ;

int main()
{
    int n , m ;

    vector < edge > e ;

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

    for( int i=1 ; i<=m ; i++ )
    {
        int u ,v , c ;
        scanf("%d %d %d",&u,&v,&c) ;
        e.pb( {u,v,c,i} ) ;
    }

    MST::findMST( n , e ) ;

    hld::init(n) ;

    for(int i=0 ; i<n-1 ; i++)
    {
        hld::addEdge( e[i].u , e[i].v ) ;
    }

    hld::preprocess(1) ;

    for(int i=0 ; i<n-1 ; i++)
    {
        int u = e[i].u , v = e[i].v , c= e[i].c ;
```

```
    if( hld::depth[u] > hld::depth[v] ) swap(u,v) ;

    ara[ hld::in[v] ] = e[i].c ;
}

ara[ hld::in[1] ] = INF ;
build(1,1,n) ;

for(int i=n-1 ; i<m ; i++)
{
    int u = e[i].u , v = e[i].v , c= e[i].c ;

    ans[ e[i].id ] = hld::hldQuery(u,v) - 1 ;
}

for(int i=1 ; i<=n ; i++) ara[i] = INF ;
build(1,1,n) ;

for(int i=n-1 ; i<m ; i++)
{
    int u = e[i].u , v = e[i].v , c= e[i].c ;
    hld::hldUpdate(u,v,c) ;
}

for(int i=0 ; i<n-1 ; i++)
{
    int u = e[i].u , v = e[i].v , c= e[i].c ;
    int res = hld::hldQuery(u,v) ;
    if(res==INF) ans[ e[i].id ] = -1 ;
    else ans[ e[i].id ] = res-1 ;
}

for(int i=1 ; i<=m ; i++) printf("%d ",ans[i]) ;
printf("\n") ;
    return 0 ;
}
```

11 Matching(kuhn)+VertexCover

```
#include <bits/stdc++.h>
#define pf printf
#define sf scanf
#define INF 1000000000000000001LL
#define PI (acos(-1.0))
#define DBG printf("Hi\n")
#define loop(i,n) for(i =1 ; i<=n; i++)
#define mp make_pair
#define pb push_back
#define mod 1000000007
#define maxn 100005
#define ff first
#define ss second
#define sq(x) ((x)*(x))
#define eps 0.0000000001
#define i64 long long int
#define ui64 unsigned long long int

using namespace std ;

int pairs[2005] ;
bool vis[2005] ;
vector <int> g[2005] , g1[2005] ;

bool dfs1(int u)
{
    if(vis[u]) return false ;
    vis[u] = true ;
```

```
    for(int i = 0 ; i<g[u].size() ; i++)
    {
        int v = g[u][i] ;
        if( pairs[v]==-1 || dfs1(pairs[v]) ){
            pairs[v] = u ;
            pairs[u] = v ;
     //       printf("printing from dfs %d %d\n",u,v) ;
            return true ;
        }
    }
    return false ;
}

int kuhn(int r, int c)
{
    memset(pairs,-1,sizeof(pairs)) ;

    int ret = 0 ;

    for(int i=1 ; i<=r ; i++)
    {
        memset( vis , 0 , sizeof(vis) ) ;
        if(dfs1(i)) ret++ ;
    }
    return ret ;
}

void dfs2(int u)
{
    if(vis[u]) return ;
    vis[u] = 1 ;
    for(int i=0 ; i<g1[u].size() ; i++) dfs2(g1[u][i]) ;
    return ;
}
```

```
int main()
{
    int i , j , k , l , m , n , r , c , u , v , ans ;

    while( sf("%d %d %d",&r,&c,&n) )
    {
        if(r==0 &&c==0 && n==0) break ;

        for(i=1 ; i<=n ; i++)
        {
            sf("%d %d",&u,&v) ;
            g[u].pb(v+r) ;
            g[v+r].pb(u) ;
        }

        ans = kuhn(r,c) ;

    //    for(i=1 ; i<=r ; i++) pf("%d %d\n",i,pairs[i]-r) ;
     //   for(i=r+1 ; i<=r+c ; i++) pf("%d %d\n",i-r,pairs[i]) ;


        for(i=1 ; i<=r ; i++)
        {
            for(j=0 ; j<g[i].size() ; j++ )
            {
                if( pairs[i] == g[i][j] ) g1[ g[i][j] ].pb(i) ;
                else g1[i].pb( g[i][j] ) ;
            }
        }
        memset(vis,0,sizeof(vis)) ;

        for( i=1 ; i<=r ; i++ )
        {
```

```
            if( pairs[i]==-1 ) dfs2(i) ;
        }

        pf("%d",ans) ;

        for(i=1 ; i<=r+c ; i++)
        {
            if( pairs[i]!=-1 )
            {
                if( i<=r && !vis[i] ) pf(" r%d",i) ;
                else if( i>r && vis[i] ) pf(" c%d",i-r) ;
            }
        }
        pf("\n") ;
        for(i=1 ; i<=r+c ; i++){
            g[i].clear() ;
            g1[i].clear() ;
        }
    }

    return 0 ;
}
```

12 PointsInRectangle

```
/*
Add a point , and add a rectangle .
After each addition answer how many (rectangle,point) pair
    exists such that point is in rectangle.
*/

#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

#define sf scanf
#define pf printf
#define pb push_back
#define mp make_pair
#define PI ( acos(-1.0) )
#define mod 1000000007LL
#define IN freopen("C.in","r",stdin)
#define OUT freopen("output.txt","w",stdout)
#define FOR(i,a,b) for(i=a ; i<=b ; i++)
#define DBG pf("Hi\n")
#define INF 1000000000
#define i64 long long int
#define eps (1e-8)
#define xx first
#define yy second
#define ln 17
#define off 2

using namespace __gnu_pbds;
using namespace std ;

typedef pair<int, int> pii ;
typedef tree< pii, null_type, less<pii>, rb_tree_tag,
    tree_order_statistics_node_update> ordered_set;

#define maxn 300005

/*
     *os.find_by_order(k) -> returns the k'th smallest element
         (indexing starts from 0)
```

```
    os.order_of_key(v) -> returns how many elements are
       strictly smaller than v
*/
ordered_set os[maxn] ;
struct DataStructure
{
public:
    ordered_set os[maxn] ;
    int mxX , cur = 0 ;
    void init( int _mxX )
    {
        mxX = _mxX ;
        for(int i=1 ; i<=mxX ; i++) os[i].clear() ;
        cur = 0 ;
    }
    void add(int x ,int y)
    {
        cur++ ;
        pii element = mp( y, cur ) ;
        while( x<= mxX )
        {
            os[x].insert( element ) ;
            x += (x&(-x)) ;
        }
    }
    int query( int x ,int y )
    {
        int sum = 0 ;
        pii element = mp( y, cur+1 ) ;
        while( x>0 )
        {
            sum += os[x].order_of_key(element) ;
            x -= (x&(-x));
        }
        return sum ;
    }
};

struct Rectangle{
    int x1 , y1 , x2, y2 ;
}rect[maxn];
struct Point
{
    int x, y ;
}point[maxn];
set<int> xSet , ySet ;
map<int,int> xMap, yMap ;
int qType[maxn] ;

int main()
{
    int i , j , k , l , m , n , t ;

    scanf("%d",&n) ;

    int cntPoint = 0 , cntRect = 0 ;

    for(i=1 ; i<=n ; i++)
    {
        scanf("%d",&t) ;
        qType[i] = t ;
        if(t==1)
        {
            cntPoint++ ;
            scanf("%d %d",&point[cntPoint].x,&point[cntPoint].y) ;
            xSet.insert( point[cntPoint].x ) ;
            ySet.insert( point[cntPoint].y ) ;
        }
        else{
            cntRect++ ;
```

```
        scanf("%d %d %d %d",&rect[cntRect].x1,
            &rect[cntRect].y1, &rect[cntRect].x2,
            &rect[cntRect].y2) ;
        xSet.insert(rect[cntRect].x1 ) ;
            xSet.insert(rect[cntRect].x2 ) ;
        ySet.insert(rect[cntRect].y1 ) ;
            ySet.insert(rect[cntRect].y2 ) ;
    }
}

set<int> :: iterator it ;
it = xSet.begin();
int mxX = 0 ;
while( it!=xSet.end() )
{
    xMap[*it] = ++mxX ;
    it++ ;
}
it = ySet.begin();
int mxY = 0 ;
while( it!=ySet.end() )
{
    yMap[*it] = ++mxY ;
    it++ ;
}

for(i=1 ; i<=cntPoint ; i++)
{
    point[i].x = xMap[point[i].x] ; point[i].y =
        yMap[point[i].y] ;
}
for(i=1 ; i<=cntRect ; i++)
{
    rect[i].x1 = xMap[rect[i].x1] ; rect[i].x2 =
        xMap[rect[i].x2] ;
    rect[i].y1 = yMap[rect[i].y1] ; rect[i].y2 =
        yMap[rect[i].y2] ;
}

DataStructure pt , r1, r2 ;

pt.init(mxX+2) ; r1.init(mxX+2) ; r2.init(mxX+2) ;

cntRect = 0 ; cntPoint = 0 ;

i64 ans = 0LL ;

for(i=1 ; i<=n ; i++)
{
    t = qType[i] ;

    if( t==1 )
    {
        cntPoint++ ;
        int x = point[cntPoint].x , y = point[cntPoint].y ;
    //   printf("%d %d\n",x,y) ;
        ans += r1.query( x, y ) - r2.query( x, y ) ;
        pt.add( x, y ) ;
    }
    else{
        cntRect++ ;
        int x1 = rect[cntRect].x1 , x2=rect[cntRect].x2 ,
            y1=rect[cntRect].y1 , y2=rect[cntRect].y2 ;
  //      printf("%d %d %d %d\n",x1,y1,x2,y2) ;
        ans += pt.query(x2,y2)-pt.query(x1-1,y2) -
            pt.query(x2,y1-1)+pt.query(x1-1,y1-1) ;
        r1.add( x1,y1 ) ; r1.add( x2+1,y2+1 ) ;
        r2.add( x1,y2+1) ; r2.add(x2+1,y1) ;
    }
    printf("%lld\n",ans) ;
```

```
    }
    return 0 ;
}
/*
7
1 5 5
1 5 5
1 5 5
2 2 2 9 9
2 1 1 5 5
2 1 1 2 2
1 2 2
*/
```

13 SOS(on the fly)

```
public class TestProctoring {
    public double expectedTime(int[] p, int[] q) {
        int n = p.length;
        double[] prob = new double[n];
        for (int i = 0; i < n; i++) {
            prob[i] = p[i] * 1.0 / q[i];
        }
        double[][] t = new double[n+1][1<<n];
        double[] dp = new double[1<<n];

       /* t[i][mask] is sum of all submask of mask where
       difference of mask and submask is before i'th bit( 0 based
          ) , that means
       difference can be in 0 to i-1 th bit
       t[0][mask] contains nothing other than just value of this
          mask
       t[n][mask] contains result of all submask of this mask
       */

       for (int mask = 1; mask < 1 << n; mask++) {
           double fail = 1;
           double mult = 1;
           double am = 1;
           for (int j = 0; j < n; j++) { t[j+1][mask] =
               t[j][mask]; if (((mask>>j)&1) == 1) {
                   t[j+1][mask] += t[j][mask^(1<<j)];
                   fail *= (1 - prob[j]);
                   mult *= prob[j];
                   am *= (1 - prob[j]) / prob[j];
               }
           }
           dp[mask] = (1 + mult * t[n][mask]) / (1 - fail);
           for (int j = 0; j <= n; j++) {
               t[j][mask] += dp[mask] * am;
           }
       }
       return dp[(1<<n)-1];
    }
}
```

14 SmallToLarge(nlogn)

```
#include <bits/stdc++.h>
// #include <ext/pb_ds/assoc_container.hpp>/
// #include <ext/pb_ds/tree_policy.hpp>

#define pb push_back
#define mp make_pair

#define mod 998244353LL
```

```
#define IN freopen("input.txt","r",stdin)
#define OUT freopen("output.txt","w",stdout)
#define FOR(i,a,b) for(i=a ; i<=b ; i++)
#define DBG printf("Hi\n")
#define INF 1000000000
#define i64 long long int
#define eps (1e-8)
#define xx first
#define yy second
#define ln 17
#define off 2
#define SZ(z) ((int)z.size())
#define sq(x) ((x)*(x))

#define FastIO ios_base::sync_with_stdio(false); cin.tie(NULL)

#define EPS 1e-7

// using namespace __gnu_pbds;
using namespace std ;

// typedef tree< i64, null_type, less<i64>, rb_tree_tag,
   tree_order_statistics_node_update> ordered_set;

typedef pair<i64, i64> pii;

#define maxn 500005
#define alpha 22

int ans[maxn] ;
int lenOff[maxn] , mask[maxn] ;
int len[(1<<alpha)+2] ;
int sub[maxn] ;
vector <int> g[maxn] , e[maxn] ;
```

```
void Add(int u , int root , int curMask, int l)
{
    len[ (curMask^mask[root]) ] = max( len[ (curMask^mask[root])
       ] , l-lenOff[root] ) ;

    for(int i=0 ; i<g[u].size() ; i++)
    {
        int v = g[u][i] ;
        Add( v , root , (curMask^(1<<e[u][i])) , l+1 ) ;
    }
}

void Remove(int u , int root , int curMask, int l)
{
    len[ (curMask^mask[root]) ] = -INF ;

    for(int i=0 ; i<g[u].size() ; i++)
    {
        int v = g[u][i] ;
        Remove( v , root , (curMask^(1<<e[u][i])) , l+1 ) ;
    }
}

void addAnswer(int u , int root , int curMask, int l)
{
    ans[root] = max( ans[root] , len[ (curMask^mask[root]) ] +
       lenOff[root] + l ) ;

    for(int i=0 ; i<alpha ; i++)
    {
        int m = (curMask^(1<<i)) ;
        ans[root] = max( ans[root] , len[ m^mask[root] ] +
           lenOff[root] + l ) ;
    }
```

```
    for(int i=0 ; i<g[u].size() ; i++)
    {
        int v = g[u][i] ;
        addAnswer( v , root , (curMask^(1<<e[u][i])) , l+1 ) ;
    }
}

void dfs(int u, bool keep)
{
    bool hasChild = ( (int)g[u].size() > 0 ) ;

    for(int i=1 ; i<g[u].size() ; i++)
    {
        dfs(g[u][i],0) ;
    }
    if( hasChild )
    {
        int bigChild = g[u][0] ;
        dfs( bigChild , 1) ;
        mask[u] = (mask[bigChild]^( 1<<e[u][0] )) ;
        lenOff[u] = lenOff[ bigChild ] + 1 ;
    }

    len[ mask[u] ] = max( len[ mask[u] ] , -lenOff[u] ) ;

    for(int i=1 ; i<g[u].size() ; i++)
    {
        addAnswer( g[u][i] , u , (1<<e[u][i]) , 1 ) ;
        Add( g[u][i] , u , (1<<e[u][i]) , 1 ) ;
    }

    ans[u] = max( ans[u] , 0 ) ;

    ans[u] = max( ans[u] , len[ mask[u] ] + lenOff[u] ) ;

    for(int i=0 ; i<alpha ; i++)
    {
        ans[u] = max( ans[u] , len[ (mask[u]^(1<<i)) ] +
            lenOff[u] ) ;
    }

    for(auto v:g[u]) ans[u] = max( ans[u] , ans[v] ) ;

    if( keep == 0 ) Remove(u,u,0,0) ;
}

void dfsSize(int u )
{
    sub[u] = 1 ;
    for( int i=0 ; i<g[u].size() ; i++ )
    {
        int v = g[u][i] ;
        dfsSize(v) ;
        sub[u] += sub[v] ;
        if( sub[v] > sub[ g[u][0] ] )
        {
            swap( g[u][0] , g[u][i] ) ;
            swap( e[u][0] , e[u][i] ) ;
        }
    }
}

int main()
{
    int n ;
    scanf("%d",&n) ;

    for(int i=2 ; i<=n ; i++)
    {
        int p ;
```

```
        char s[4] ;
        scanf("%d %s",&p,s) ;
        g[p].pb(i) ; e[p].pb( s[0]-'a' ) ;
    }

    for(int i=0 ; i<(1<<alpha) ; i++) len[i] = -INF ;

    dfsSize(1) ;
    dfs(1,1) ;

    for(int i=1 ; i<=n ; i++)
    {
        printf("%d ",ans[i]) ;
    }
    printf("\n") ;

    return 0 ;
}
```

15 SuffixAutomata(short)

```
/*
https://www.spoj.com/problems/STRSOCU/
given two strings s,t and an integer k, how many distinct
    substrings of s occurs exactly k times in t ?
O(n) solution
*/
#include <bits/stdc++.h>

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

#define sf scanf
#define pf printf
#define pb push_back
#define mp make_pair
#define PI ( acos(-1.0) )
#define mod 1000000007LL
#define IN freopen("C.in","r",stdin)
#define OUT freopen("output.txt","w",stdout)
#define FOR(i,a,b) for(i=a ; i<=b ; i++)
#define DBG pf("Hi\n")
#define INF 1000000000
#define i64 long long int
#define eps (1e-8)
#define xx first
#define yy second
#define ln 17
#define off 2

using namespace __gnu_pbds;
using namespace std ;

typedef pair<int, int> pi ;
typedef tree< pi, null_type, less<pi>, rb_tree_tag,
    tree_order_statistics_node_update> ordered_set;

#define maxn 100005

/*
     *os.find_by_order(k) -> returns the k'th smallest element
        (indexing starts from 0)
      os.order_of_key(v) -> returns how many elements are
         strictly smaller than v
*/
class SuffixAutomaton{
public:
    struct state{
```

```
    int edge[27] , len , link , cnt ;
};

state *st ;
int sz , last , alpha = 26 ;

SuffixAutomaton( string &s )
{
    int l = s.length() ;
    int i , j ;
    st = new state[l*2] ;

    st[0].link = -1 ; st[0].len = 0 ; st[0].cnt = 0 ; sz = 1
        ; last = 0 ;
    for(i=0 ; i<alpha ; i++) st[0].edge[i] = -1 ;

    for(i=0 ; i<l ; i++)
    {
        int cur = sz++ ;

        for(j=0 ; j<alpha ; j++) st[cur].edge[j] = -1 ;
        st[cur].len = st[last].len+1 ; st[cur].cnt = 1 ;

        int p = last , c = s[i]-'a' ;

        while( p!=-1 && st[p].edge[c] == -1 )
        {
            st[p].edge[c] = cur ;
            p = st[p].link ;
        }

        if( p == -1 )
        {
            st[cur].link = 0 ;
        }
        else
        {
            int q = st[p].edge[c] ;
            if( st[p].len+1 == st[ q ].len ) st[cur].link = q ;
            else{
                int clone = sz++ ;
                for(j=0 ; j<alpha ; j++) st[clone].edge[j] =
                    st[q].edge[j] ;
                st[clone].len = st[p].len+1 ;
                st[clone].link = st[q].link ;
                st[clone].cnt = 0 ;

                while( p!=-1 && st[p].edge[c] == q )
                {
                    st[p].edge[c] = clone ;
                    p = st[p].link ;
                }
                st[q].link = st[cur].link = clone ;
            }
        }
        last = cur ;
    }
    vector <pi> vp ;
    for(i=0 ; i<sz ; i++) vp.pb( mp( st[i].len , i ) ) ;
    sort(vp.begin(),vp.end()) ;

    for( i=sz-1 ; i>0 ; i-- )
    {
        int state = vp[i].yy ;
        st[st[ state].link].cnt += st[ state].cnt ;
//        printf("%d %d\n",st[state].cnt[0],st[state].cnt[1])
    ;
    }
}
```

```
    int f( string t )
    {
        int len = (int)t.size() ;
        t = t+t ;
//        t.pop_back() ;

        int cur = 0 , myLen = 0 ;

        vector <int> vis ;

        for(int i=0 ; i<t.size() ; i++)
        {
            int ch = t[i] - 'a' ;
            while( st[ cur ].edge[ch] == -1 && st[cur].link!=-1 )
            {
                cur = st[cur].link ;
                myLen = st[cur].len ;
            }
            if( st[cur].edge[ch]!=-1 ) cur = st[cur].edge[ch] ,
                myLen++ ;

            while( cur!=0 && st[ st[cur].link ].len >= len )
            {
                cur = st[cur].link ;
                myLen = st[cur].len ;
            }
            if( cur !=0 && myLen >= len ) vis.pb( cur ) ;

//            if( myLen >= len && st[cur].len >= len ) cout<<i<<"
    "<<st[cur].cnt<<endl ;

        }
        sort( vis.begin() , vis.end() ) ;

        int ans = 0 ;

        for( int i=0 ; i<vis.size() ; i++ )
        {
            int cur = vis[i] ;
            if( i > 0 && vis[i]==vis[i-1] ) continue ;
            if( st[cur].len >= len && st[ st[cur].link ].len <
                len ) ans += st[cur].cnt ;
        }
        return ans ;
    }

    ~SuffixAutomaton()
    {
        delete []st ;
    }
};

int main()
{
    ios_base::sync_with_stdio(false);
    cin.tie(NULL);

    string s , t ;

    cin>>s ;

    int m ;
    cin>>m ;

    SuffixAutomaton sa(s) ;

    for(int i=1 ; i<=m ; i++)
    {
        cin>>t ;
        cout<<sa.f(t)<<"\n" ;
```

```
    }

    return 0 ;
}
```

16 Weighted Blossoms

```
//from facelessman&vfleaking
#include<bits/stdc++.h>
#define cin kin
#define DIST(e) (lab[e.u]+lab[e.v]-g[e.u][e.v].w*2)
using namespace std;
typedef long long ll;
const int N=1023,INF=1e9;
struct Edge
{
	int u,v,w;
} g[N][N];
int
    n,m,n_x,lab[N],match[N],slack[N],st[N],pa[N],flower_from[N][N],S[N],vis[N];
vector<int> flower[N];
deque<int> q;
void update_slack(int u,int x)
{
	if(!slack[x]||DIST(g[u][x])<DIST(g[slack[x]][x]))slack[x]=u;
}
void set_slack(int x)
{
	slack[x]=0;
	for(int u=1; u<=n; ++u)
		if(g[u][x].w>0&&st[u]!=x&&S[st[u]]==0)update_slack(u,x);
}
void q_push(int x)
{
	if(x<=n)return q.push_back(x);
	for(int i=0; i<flower[x].size(); i++)q_push(flower[x][i]);
}
void set_st(int x,int b)
{
	st[x]=b;
	if(x<=n)return;
	for(int i=0; i<flower[x].size();
	    ++i)set_st(flower[x][i],b);
}
int get_pr(int b,int xr)
{
	int
	    pr=find(flower[b].begin(),flower[b].end(),xr)-flower[b].begi
	if(pr%2==1) //
	{
		reverse(flower[b].begin()+1,flower[b].end());
		return (int)flower[b].size()-pr;
	}
	else return pr;
}
void set_match(int u,int v)
{
	match[u]=g[u][v].v;
	if(u<=n)return;
	Edge e=g[u][v];
	int xr=flower_from[u][e.u],pr=get_pr(u,xr);
	for(int i=0; i<pr;
	    ++i)set_match(flower[u][i],flower[u][i^1]);
	set_match(xr,v);
	rotate(flower[u].begin(),flower[u].begin()+pr,flower[u].end());
}
void augment(int u,int v)
{
```

```
	int xnv=st[match[u]];
	set_match(u,v);
	if(!xnv)return;
	set_match(xnv,st[pa[xnv]]);
	augment(st[pa[xnv]],xnv);
}
int get_lca(int u,int v)
{
	static int t=0;
	for(++t; u||v; swap(u,v))
	{
		if(u==0)continue;
		if(vis[u]==t)return u;
		vis[u]=t;//		v
		u=st[match[u]];
		if(u)u=st[pa[u]];
	}
	return 0;
}
void add_blossom(int u,int lca,int v)
{
	int b=n+1;
	while(b<=n_x&&st[b])++b;
	if(b>n_x)++n_x;
	lab[b]=0,S[b]=0;
	match[b]=match[lca];
	flower[b].clear();
	flower[b].push_back(lca);
	for(int x=u,y; x!=lca; x=st[pa[y]])
		flower[b].push_back(x),flower[b].push_back(y=st[match[x]]),q_push(y);
	reverse(flower[b].begin()+1,flower[b].end());
	for(int x=v,y; x!=lca; x=st[pa[y]])
		flower[b].push_back(x),flower[b].push_back(y=st[match[x]]),q_push(y);
	set_st(b,b);
	for(int x=1; x<=n_x; ++x)g[b][x].w=g[x][b].w=0;
	for(int x=1; x<=n; ++x)flower_from[b][x]=0;
	for(int i=0; i<flower[b].size(); ++i)
	{
		int xs=flower[b][i];
		for(int x=1; x<=n_x; ++x)
			if(g[b][x].w==0||DIST(g[xs][x])<DIST(g[b][x]))
				g[b][x]=g[xs][x],g[x][b]=g[x][xs];
		for(int x=1; x<=n; ++x)
			if(flower_from[xs][x])flower_from[b][x]=xs;
	}
	set_slack(b);
}
void expand_blossom(int b) // S[b] == 1
{
	for(int i=0; i<flower[b].size(); ++i)
		set_st(flower[b][i],flower[b][i]);
	int xr=flower_from[b][g[b][pa[b]].u],pr=get_pr(b,xr);
	for(int i=0; i<pr; i+=2)
	{
		int xs=flower[b][i],xns=flower[b][i+1];
		pa[xs]=g[xns][xs].u;
		S[xs]=1,S[xns]=0;
		slack[xs]=0,set_slack(xns);
		q_push(xns);
	}
	S[xr]=1,pa[xr]=pa[b];
	for(int i=pr+1; i<flower[b].size(); ++i)
	{
		int xs=flower[b][i];
		S[xs]=-1,set_slack(xs);
	}
	st[b]=0;
}
bool on_found_Edge(const Edge &e)
{
```

```
        int u=st[e.u],v=st[e.v];
        if(S[v]==-1)
        {
                pa[v]=e.u,S[v]=1;
                int nu=st[match[v]];
                slack[v]=slack[nu]=0;
                S[nu]=0,q_push(nu);
        }
        else if(S[v]==0)
        {
                int lca=get_lca(u,v);
                if(!lca)return augment(u,v),augment(v,u),1;
                else add_blossom(u,lca,v);
        }
        return 0;
}
bool matching()
{
        fill(S,S+n_x+1,-1),fill(slack,slack+n_x+1,0);
        q.clear();
        for(int x=1; x<=n_x; ++x)
                if(st[x]==x&&!match[x])pa[x]=0,S[x]=0,q_push(x);
        if(q.empty())return 0;
        for(;;)
        {
                while(q.size())
                {
                        int u=q.front();
                        q.pop_front();
                        if(S[st[u]]==1)continue;
                        for(int v=1; v<=n; ++v)
                                if(g[u][v].w>0&&st[u]!=st[v])
                                {
                                        if(DIST(g[u][v])==0)
                                        {
                                                if(on_found_Edge(g[u][v]))r
                                                        1;
                                        }
                                        else update_slack(u,st[v]);
                                }
                }
        int d=INF;
        for(int b=n+1; b<=n_x; ++b)
                if(st[b]==b&&S[b]==1)d=min(d,lab[b]/2);
        for(int x=1; x<=n_x; ++x)
                if(st[x]==x&&slack[x])
                {
                        if(S[x]==-1)d=min(d,DIST(g[slack[x]][x]));
                        else
                                if(S[x]==0)d=min(d,DIST(g[slack[x]][x]
                }
        for(int u=1; u<=n; ++u)
        {
                if(S[st[u]]==0)
                {
                        if(lab[u]<=d)return 0;
                        lab[u]-=d;
                }
                else if(S[st[u]]==1)lab[u]+=d;
        }
        for(int b=n+1; b<=n_x; ++b)
                if(st[b]==b)
                {
                        if(S[st[b]]==0)lab[b]+=d*2;
                        else if(S[st[b]]==1)lab[b]-=d*2;
                }
        q.clear();
        for(int x=1; x<=n_x; ++x)
                if(st[x]==x&&slack[x]&&st[slack[x]]!=x&&DIST(g[sl
```

```
                        if(on_found_Edge(g[slack[x]][x]))return
                            1;
                for(int b=n+1; b<=n_x; ++b)
                        if(st[b]==b&&S[b]==1&&lab[b]==0)expand_blossom(b);
        }
        return 0;
}
pair<ll,int> weight_blossom()
{
        fill(match,match+n+1,0);
        n_x=n;
        int n_matches=0;
        ll tot_weight=0;
        for(int u=0; u<=n; ++u)st[u]=u,flower[u].clear();
        int w_max=0;
        for(int u=1; u<=n; ++u)
                for(int v=1; v<=n; ++v)
                {
                        flower_from[u][v]=(u==v?u:0);
                        w_max=max(w_max,g[u][v].w);
                }
        for(int u=1; u<=n; ++u)lab[u]=w_max;
        while(matching())++n_matches;
        for(int u=1; u<=n; ++u)
                if(match[u]&&match[u]<u)
                        tot_weight+=g[u][match[u]].w;
        return make_pair(tot_weight,n_matches);
}
struct Istream
{
        char b[20<<20],*i,*e;
        Istream(FILE*
            in):i(b),e(b+fread(b,sizeof(*b),sizeof(b)-1,in)) {}
        Istream& operator>>(int &val)
        {
                while(*i<'0')++i;
                for(val=0; *i>='0';
                    ++i)val=(val<<3)+(val<<1)+*i-'0';
                return *this;
        }
} kin(stdin);
int main()
{
        cin>>n>>m;
        for(int u=1; u<=n; ++u)
                for(int v=1; v<=n; ++v)
                        g[u][v]=Edge {u,v,0};
        for(int i=0,u,v,w; i<m; ++i)
        {
                cin>>u>>v>>w;
                g[u][v].w=g[v][u].w=w;
        }
        cout<<weight_blossom().first<<'\n';
        for(int u=1; u<=n; ++u)cout<<match[u]<<' ';
}
```

17 basisFinding

```
#include <bits/stdc++.h>

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

#define sf scanf
#define pf printf
#define pb push_back
#define mp make_pair
#define PI ( acos(-1.0) )
```

```
#define IN freopen("dichromatic.in","r",stdin)
#define OUT freopen("dichromatic.out","w",stdout)
#define FOR(i,a,b) for(i=a ; i<=b ; i++)
#define DBG pf("Hi\n")
#define INF 1000000000
#define i64 long long int
#define eps (1e-8)
#define xx first
#define yy second
#define sq(x) ((x)*(x))

using namespace __gnu_pbds;
using namespace std ;

#define maxn 200005

int bit[(1<<18)+5] ;

vector <int> Generate( int x )
{
    vector <int> vec ;
    vector <int> res ;
    vec.pb(0) ;
    vec.pb(1) ;

    for(int i=1 ; i<x ; i++)
    {
        int j = (1<<i) - 1 ;
        for( ; j>=0 ; j--)
        {
            vec.pb( (vec[j]|(1<<i)) ) ;
        }
    }
    for( int i=1 ; i<(1<<x) ; i++ )
    {
        res.pb( bit[ vec[i]^vec[i-1] ] ) ;
    }
    return res ;
}

int a[maxn] ;

/*
given some numbers in range [0 to 2^x), this function will
    return you with a set of basis
vectors from this numbers
*/
vector <int> findBasis( vector <int> b , int x )
{
    vector <int> idx , num ;
    for(int i=0 ; i<b.size() ; i++) idx.pb(i) , num.pb(b[i]) ;

    for(int i=x-1 , j=0 ; i>=0 ; i--,j++)
    {
        for(int k= j+1 ; k<idx.size() ; k++)
        {
            if( b[k] > b[j] )
            {
                swap(b[k],b[j]) ;
                swap(idx[k],idx[j]) ;
            }
        }
        for(int k=j+1 ; k<idx.size() ; k++)
        {
            if( (b[k]^b[j]) < b[k] ) b[k] ^= b[j] ;
        }

    }
    int i = (int)idx.size() - 1 ;
    while( i > 0 && b[i] == 0 )
```

```
    {
        b.pop_back() ;
        idx.pop_back() ;
        i-- ;
    }
    vector <int> basis ;
    for(int i=0 ; i<idx.size() ; i++) basis.pb( num[idx[i]] ) ;
    return basis ;
}

//

const int ln = 18 ;

int main()
{
    for(int i=0 ; i<=ln ; i++) bit[(1<<i)] = i ;
    int n ;
    scanf("%d",&n) ;

    for(int i=0; i<n ; i++ )
    {
        scanf("%d",&a[i]) ;
    }

    vector <int> ans ;
    ans.pb(0) ;

    for(int i=ln ; i>=1 ; i-- )
    {
        vector <int> b ;
        for(int j=0 ; j<n ; j++) if( a[j] < (1<<i) ) b.pb(a[j]) ;
        vector <int> basis = findBasis(b,i) ;
        if( (int)basis.size() == i )
        {
            vector <int> res = Generate(i) ;
            for(int j=0 ; j<res.size() ; j++) ans.pb(
                ans.back()^basis[ res[j] ] ) ;
            break ;
        }
    }

    printf("%d\n",bit[(int)ans.size()]) ;
    for(int i=0 ; i<ans.size() ; i++)
    {
        printf("%d ",ans[i]) ;
    }

    return 0 ;
}
```

18 fft(anymod)

```
//fft with any mod
//be careful in choosing MAXN , it should be double of next
   power of 2 of your needed n
//example, here n is 131072 = 2^17 , so MAXN is 2^19 ( double of
   2^18 )
// if n was 10, then MAXN = 32 suffices


#include <bits/stdc++.h>

#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

#define sf scanf
#define pf printf
```

```
#define pb push_back
#define mp make_pair
#define PI ( acos(-1.0) )
#define IN freopen("dichromatic.in","r",stdin)
#define OUT freopen("dichromatic.out","w",stdout)
#define FOR(i,a,b) for(i=a ; i<=b ; i++)
#define DBG pf("Hi\n")
#define INF 1000000000
#define i64 long long int
#define eps (1e-8)
#define xx first
#define yy second
#define ln 17
#define off 2
#define sq(x) ((x)*(x))

using namespace __gnu_pbds;
using namespace std ;

typedef tree< i64, null_type, less<i64>, rb_tree_tag,
   tree_order_statistics_node_update> ordered_set;
typedef pair<i64, i64> pii;

#define MAX 131073
#define MAXN 524288
#define MOD 258280327
#define clr(ar) memset(ar, 0, sizeof(ar))
#define read() freopen("lol.txt", "r", stdin)
#define dbg(x) cout << #x << " = " << x << endl
#define ran(a, b) ((((rand() << 15) ^ rand()) % ((b) - (a) + 1))
   + (a))

using namespace std;

namespace fft{
int len, last = -1, step = 0, rev[MAXN];

    struct complx{
    double real, img;

    inline complx(){
        real = img = 0.0;
    }

    inline complx conjugate(){
        return complx(real, -img);
    }

    inline complx(double x){
        real = x, img = 0.0;
    }

    inline complx(double x, double y){
        real = x, img = y;
    }

    inline complx operator + (complx other){
        return complx(real + other.real, img + other.img);
    }

    inline complx operator - (complx other){
        return complx(real - other.real, img - other.img);
    }

    inline complx operator * (complx other){
        return complx((real * other.real) - (img *
            other.img), (real * other.img) + (img *
            other.real));
    }
} u[MAXN], v[MAXN], f[MAXN], g[MAXN], dp[MAXN];
```

```
void build(int& a, int *A, int& b, int* B){
    while (a > 1 && A[a - 1] == 0) a--;
    while (b > 1 && B[b - 1] == 0) b--;

        len = 1 << (32 - __builtin_clz(a + b) -
            (__builtin_popcount(a + b) == 1));
    for (int i = a; i < len; i++) A[i] = 0;
    for (int i = b; i < len; i++) B[i] = 0;

            if (!step++){
        dp[1] = complx(1);
        for (int i = 1; (1 << i) < MAXN; i++){
            double theta = (2.0 * acos(0.0)) / (1 << i);
            complx mul = complx(cos(theta), sin(theta));

            int lim = 1 << i;
            for (int j = lim >> 1; j < lim; j++){
                dp[2 * j] = dp[j];
                dp[2 * j + 1] = dp[j] * mul;
            }
        }
    }

    if (last != len){
        last = len;
        int bit = (32 - __builtin_clz(len) -
            (__builtin_popcount(len) == 1));
        for (int i = 0; i < len; i++) rev[i] = (rev[i >> 1]
            >> 1) + ((i & 1) << (bit - 1));
    }
    }

    void transform_unrolled(complx *in, complx *out, complx*
        ar){
        for (int i = 0; i < len; i++) out[i] = in[rev[i]];
        for (int k = 1; k < len; k <<= 1){
        for (int i = 0; i < len; i += (k << 1)){
            complx z, *a = out + i, *b = out + i + k, *c = ar
                + k;
            if (k == 1){
                z = (*b) * (*c);
                *b = *a - z, *a = *a + z;
            }

            for (int j = 0; j < k && k > 1; j += 2, a++, b++,
                c++){
                z = (*b) * (*c);
                *b = *a - z, *a = *a + z;
                a++, b++, c++;
                z = (*b) * (*c);
                *b = *a - z, *a = *a + z;
            }
        }
        }
}

bool equals(int a, int* A, int b, int* B){
if (a != b) return false;
for (a = 0; a < b && A[a] == B[a]; a++){}
return (a == b);
}

int mod_multiply(int a, int* A, int b, int* B, int mod){
    build(a, A, b, B);
    int flag = equals(a, A, b, B);
    for (int i = 0; i < len; i++) A[i] %= mod, B[i] %=
        mod;
    for (int i = 0; i < len; i++) u[i] = complx(A[i] &
        32767, A[i] >> 15);
```

```
        for (int i = 0; i < len; i++) v[i] = complx(B[i] &
            32767, B[i] >> 15);

            transform_unrolled(u, f, dp);
            for (int i = 0; i < len; i++) g[i] = f[i];
            if (!flag) transform_unrolled(v, g, dp);

            for (int i = 0; i < len; i++){
        int j = (len - 1) & (len - i);
        complx c1 = f[j].conjugate(), c2 = g[j].conjugate();

        complx a1 = (f[i] + c1) * complx(0.5, 0);
                  complx a2 = (f[i] - c1) * complx(0, -0.5);
                  complx b1 = (g[i] + c2) * complx(0.5 /
                      len, 0);
                  complx b2 = (g[i] - c2) * complx(0, -0.5 /
                      len);
                  v[j] = a1 * b2 + a2 * b1;
                  u[j] = a1 * b1 + a2 * b2 * complx(0, 1);
            }
            transform_unrolled(u, f, dp);
            transform_unrolled(v, g, dp);

    long long x, y, z;
            for (int i = 0; i < len; i++){
        x = f[i].real + 0.5, y = g[i].real + 0.5, z =
            f[i].img + 0.5;
        A[i] = (x + ((y % mod) << 15) + ((z % mod) << 30)) %
            mod;
            }
            return a + b - 1;
    }
}

int black[17][MAXN] , red[17][MAXN] , ans[MAXN] ;

int dp[MAXN] ;

int main()
{
    IN ;
    OUT ;
    black[0][0] = 1 ; red[0][0] = 1 ; red[0][1] = 1 ;
    black[1][1] = 1 ; black[1][2] = 2 ; black[1][3] = 1 ;
    red[1][3] = 1 ; red[1][4] = 4 ; red[1][5] = 6 ; red[1][6] =
        4 ; red[1][7] = 1 ;

    for(int i=2; i<=16 ; i++)
    {
     //   printf("%d\n",i) ;
        for(int j=0 ; j<MAX ; j++) dp[j] =
            (red[i-1][j]+black[i-1][j])%MOD ;
        fft::mod_multiply(MAX,dp,MAX,dp,MOD) ;
        for(int j=1 ; j<MAX ; j++) black[i][j] = dp[j-1] ;
        for(int j=0 ; j<MAX ; j++) dp[j] = black[i][j] ;
        fft::mod_multiply(MAX,dp,MAX,dp,MOD) ;
        for(int j=0 ; j<MAX ; j++) red[i][j] = dp[j-1] ;
 //       for(int j=0 ; j<10 ; j++) printf("%d: %d
    %d\n",j,black[i][j],red[i][j]) ;
    }

    int t , h ;
    scanf("%d %d",&t,&h) ;

    for(int i=0 ; i<MAX ; i++)
    {
        for(int j=0 ; j<=h ; j++) ans[i] = (ans[i] +
            black[j][i]+red[j][i])%MOD ;
        ans[i] = (ans[i]%MOD + MOD)%MOD ;
    }
```

```
    for(int i=0 ; i<t ; i++)
    {
        int n ;
        scanf("%d",&n) ;
        printf("%d\n",ans[n] ) ;
    }

    return 0 ;
}
```

19 lca

```
#include <bits/stdc++.h>

#define i64 long long int

using namespace std ;

//starts O(1) lca
//preprocessing nlogn

#define MAX 100010
#define LOG 18
namespace LCA{
    i64 sum[MAX] ;
    int st[MAX] , en[MAX] , lg[MAX] , par[MAX] , a[MAX] ,
        id[MAX] , dp[LOG][MAX] ;
    vector <int> weight[MAX] , g[MAX] ;
    int n , r , Time , cur ;

    void init(int nodes, int root){
        n = nodes, r = root, lg[0] = lg[1] = 0;
        for (int i = 2; i <= n; i++) lg[i] = lg[i >> 1] + 1;
        for (int i = 0; i <= n; i++) g[i].clear(),
            weight[i].clear();
    }

    void addEdge(int u, int v, int w){
        g[u].push_back(v), weight[u].push_back(w);
        g[v].push_back(u), weight[v].push_back(w);
    }

    int lca(int u, int v)
    {
        if( en[u] > en[v] )swap(u,v) ;
        if( st[v] <= st[u] && en[u] <= en[v] ) return v ;
        int l = lg[id[v] - id[u] + 1] ;
        int p1 = id[u] , p2 = id[v] - (1<<l) + 1 ;
        if( sum[ dp[l][p1] ] < sum[ dp[l][p2] ] ) return par[
            dp[l][p1] ] ;
        else return par[ dp[l][p2] ] ;
    }

    i64 dis( int u ,int v )
    {
        int l = lca(u,v) ;
        return (sum[u] + sum[v] - ( sum[l]<<1LL )) ;
    }

    void dfs(int u, int p , i64 curSum){
        st[u] = ++Time ; par[u] = p ; sum[u] = curSum ;
        for(int i=0 ; i<g[u].size() ; i++)
        {
            if( g[u][i]==p ) continue ;
            dfs( g[u][i] ,u,curSum+weight[u][i]) ;
        }
        en[u] = ++Time ;
        a[++cur] = u ;
```

```
        id[u] = cur ;
    }

    void build(){
        cur = Time = 0 ;
        dfs( r , r , 0 );
        for(int i=1 ; i<=n ; i++) dp[0][i] = a[i] ;
        for(int l=0 ; l<LOG-1 ; l++)
        {
            for(int i=1 ; i<=n ; i++)
            {
                dp[l+1][i] = dp[l][i] ;
                if( (1<<l)+i <= n && sum[dp[l][i+(1<<l)]] <
                    sum[dp[l][i]]) dp[l+1][i] = dp[l][ i+(1<<l)] ;
            }
        }
    }
}
int main() { return 0 ; }
```

20 moSet

```
/*
CF-375D
A undirected tree is given, each node having a colour , we'll
    have some queries of form (v,k).
We've to answer how many color occurs at least k'times in the
    subtree rooted at v.
*/

#include <bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>

#define sf scanf
#define pf printf
#define pb push_back
#define mp make_pair
#define PI ( acos(-1.0) )
#define mod 1000000007LL
#define IN freopen("nocross.in","r",stdin)
#define OUT freopen("nocross.out","w",stdout)
#define FOR(i,a,b) for(i=a ; i<=b ; i++)
#define DBG pf("Hi\n")
#define INF 1000000001
#define i64 long long int
#define eps (1e-8)
#define xx first
#define yy second
#define ln 17
#define off 2
#define sq(x) ((x)*(x))

using namespace __gnu_pbds;
using namespace std ;

typedef pair< i64,i64 > pii ;
typedef tree< i64, null_type, less<i64>, rb_tree_tag,
    tree_order_statistics_node_update> ordered_set;

/* Special functions:

        find_by_order(k) --> returns iterator to the kth largest
            element counting from 0
        order_of_key(val) --> returns the number of items in a
            set that are strictly smaller than our item
*/
```

```
#define maxn 100005

class moSet{

    /*
    intution
    say, now I have my cnt array like this cnt[] = { 0 , 4 , 1 ,
       2 , 3 , 4, 0 , 2 }
    and each time cnt changes by only one, so if a element
       occurred 2 times before, now it's
    occurrence can be either 1 or 3. And each time we want to
       know, how many element occurred
    at least k times. Then we can keep our cnt[] array in
       another form by sorting the occurrence.
    occ[] = { 0,0,1,2,2,3,4,4 }. So when an increase an elements
       occurrence , say previously an element
    occurred 2 times, now it is occurring 3 times , then there
       will be only a slight change in occ array,
    one 2 will be just turned into 3.

     Description of the data structure
     this is a data-structure mainly for keeping track of
        occurrences
     lets say currently we have n=6 and our sortedArray[] =
        {0,0,1,2,2,3,4,4}
     if we increase(2) , then sortedArray[] = { 0,0,1,2,3,3,4,4
        } , so one 2 will be turned into 3
     if we decrease(3) now ,then sortedArray[] = {
        0,0,1,2,2,3,4,4 } , so one 3 will be turned into 2
     along with this we've start , and en array and exist array,
        start[1] = 2 , en[2] = 4 , exist[1] = true,
     exist[5] = false, also remember here we can only handle
        elements from 0 to n (inclusive)
    */
public:
    bool *exist ;
    int *sortedArray, *Start , *End , n ;

    moSet(int _n)
    {
        n = _n ;
        exist = new bool[n+2] ;
        sortedArray = new int[n+2] ;
        Start = new int[n+2] ;
        End = new int[n+2] ;

        for(int i=0 ; i<=n ; i++) exist[i] = false ;
    }
    moSet( int _n , int initVal )
    {
        n = _n ;
        exist = new bool[n+2] ;
        sortedArray = new int[n+2] ;
        Start = new int[n+2] ;
        End = new int[n+2] ;

        for(int i=0 ; i<=n ; i++) exist[i] = false ;
        exist[initVal] = true ;
        for(int i=0 ; i<=n ; i++) sortedArray[i] = initVal ;
        Start[initVal] = 0 ; End[initVal] = n ;
    }

    void increase(int v)
    {
        if(!exist[v] || v<0 || v>=n ) return ; // 0 to n-1
        sortedArray[ End[v] ] = v+1 ;
        if( !exist[v+1] )
        {
            exist[v+1] = true ;
            Start[ v+1 ] = End[v+1] = End[v] ;
```

```
        }
        else Start[v+1]-- ;
        End[v]-- ;
        if(Start[v]>End[v]) exist[v] = false ;
        return ;
    }
    void decrease(int v)
    {
        if( !exist[v] || v<=0 || v>n ) return ; // 1 to n
        sortedArray[ Start[v] ] = v-1 ;
        if( !exist[v-1] )
        {
            exist[v-1] = true ;
            Start[v-1] = End[v-1] = Start[v] ;
        }
        else End[v-1]++ ;
        Start[v]++ ;
        if(Start[v]>End[v]) exist[v] = false ;
        return ;
    }

    int howMany(int x)
    {
        if( sortedArray[n] <x ) return 0 ;
        int lo = 0 , hi = n ;
        while(lo<hi)
        {
            int mid = (lo+hi)/2;
            if( sortedArray[mid]>=x ) hi = mid ;
            else lo = mid+1 ;
        }
        return n-lo+1 ;
    }

};
```

```
int posInArray[maxn] , endPos[maxn] , Time , s ;
int c[maxn] , col[maxn] , cnt[maxn] , ans[maxn] ;
vector <int> g[maxn] ;

void dfs(int u , int par)
{
    posInArray[u] = ++Time ;
    col[Time] = c[u] ;
    for(int i=0 ; i<g[u].size() ; i++ )
    {
        int v = g[u][i] ;
        if(v==par) continue ;
        dfs(v,u) ;
    }
    endPos[u] = Time ;
}

struct Query{
    int l ,r ,id , k ;
}query[maxn];

bool moComp( Query q1 , Query q2 )
{
    if( q1.l/s == q2.l/s ) return q1.r<q2.r ;
    return q1.l<q2.l ;
}

int main()
{
    int i , j , k , m , n , l , r ;

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

```
    moSet mySet(n,0) ;
/*
    while(1)
    {
        scanf("%d %d",&k,&l) ;
        if( k==1 ) mySet.increase(l) ;
        else mySet.decrease(l) ;
        for(i=0 ; i<=10 ; i++) printf("%d ",mySet.sortedArray[i])
            ;
        printf("\n") ;
    }
*/
    for(i=1 ; i<=n ; i++) scanf("%d",&c[i]) ;

    for(i=1 ; i<n ; i++)
    {
        int u , v ;
        scanf("%d %d",&u,&v) ;
        g[u].pb(v) ; g[v].pb(u) ;
    }
    Time = 0 ;
    dfs(1,-1) ;

    for(i=1 ; i<=m ; i++)
    {
        int v ;
        scanf("%d %d",&v,&k) ;
        query[i].l = posInArray[v] ; query[i].r = endPos[v] ;
            query[i].k = k ;
        query[i].id = i ;
    }

    s = sqrt(n+0.0) + 1 ;

    sort(query+1, query+m+1,moComp) ;
```

```
    memset(cnt,0,sizeof(cnt)) ;

    l = 1 ; r = 0 ;

    for(i=1 ; i<=m ; i++)
    {
      // printf("%d %d")
        while(r<query[i].r)
        {
            r++ ;
            j = r ;
            mySet.increase( cnt[ col[j] ] ) ;
            cnt[ col[j] ]++ ;
        }
        while(l>query[i].l)
        {
            l-- ;
            j = l ;
            mySet.increase( cnt[ col[j] ] ) ;
            cnt[ col[j] ]++ ;
        }
        while(r>query[i].r)
        {
            j = r ;
            mySet.decrease(cnt[col[j]]) ;
            cnt[ col[j] ]-- ;
            r-- ;
        }
        while(l<query[i].l)
        {
            j = l ;
            mySet.decrease(cnt[col[j]]) ;
            cnt[ col[j] ]-- ;
            l++ ;
```

```
        }
        ans[ query[i].id ] = mySet.howMany(query[i].k) ;
    /*    printf("%d\n",query[i].id) ;
        for(j=0 ; j<=n ; j++) printf("%d ",mySet.sortedArray[j]) ;
        printf("\n") ; */
    }

    for(i=1 ; i<=m ; i++) printf("%d\n", ans[i] ) ;

    return 0 ;
}
/*
10 10
82 48 59 48 32 83 34 46 47 79
2 1
3 1
4 3
5 4
6 1
7 2
8 3
9 2
10 2
1 2
1 1
1 3
1 5
2 1
2 2
3 1
3 3
6 1
9 2
*/
```

21 non negative soln extended euclid

```
#include <bits/stdc++.h>
// #include <ext/pb_ds/assoc_container.hpp>/
// #include <ext/pb_ds/tree_policy.hpp>

#define sf scanf
#define pf printf
#define pb push_back
#define mp make_pair
#define PI ( acos(-1.0) )
#define mod 1000000007
#define IN freopen("nocross.in","r",stdin)
#define OUT freopen("nocross.out","w",stdout)
#define FOR(i,a,b) for(i=a ; i<=b ; i++)
#define DBG pf("Hi\n")
#define INF 2000000000
#define i64 long long int
#define eps (1e-8)
#define xx first
#define yy second
#define ln 17
#define off 2
#define SZ(z) ((int)z.size())

// using namespace __gnu_pbds;
using namespace std ;

// typedef tree< i64, null_type, less<i64>, rb_tree_tag,
    tree_order_statistics_node_update> ordered_set;

#define maxn 50005

typedef pair<i64, i64> pii;
```

```
// f(a,b,c) returns how many non_negative (x,y) are there such
    that a*x+b*y = c
//tested for (a,b,c) > 0
// for negative (a,b) equation changes

pii extendedEuclid(i64 a, i64 b) { // returns x, y | ax + by =
    gcd(a,b)

    if(b == 0) return pii( a >= 0 ? 1 : -1 , 0LL);
    else {
        pii d = extendedEuclid(b, a % b);
        return pii(d.yy, d.xx - d.yy * (a / b));
    }

}
i64 gcd(i64 a, i64 b)
{
    if(b==0) return a ;
    return gcd(b,a%b) ;
}
i64 Floor(i64 a, i64 b)
{
    if(b<0) b *= (-1) , a *= (-1) ;
    i64 c = a/b ;
    if( a<0 && b*c!=a ) c-- ;
    return c ;
}
i64 Ceil(i64 a, i64 b)
{
    if(b<0) b *= (-1) , a *= (-1) ;
    i64 c = a/b ;
    if( a>0 && b*c!=a ) c++ ;
    return c ;
}
i64 f( i64 a, i64 b,i64 c )
{
    if( a==0 && b==0 ) return (c==0) ;
    if(a==0) return (c%b == 0) ;
    if(b==0) return (c%a == 0) ;

    i64 g = gcd(a,b) ;
    if(c%g!=0) return 0 ;

    c /= g ;
    a /= g ; b/= g ;

    pii soln = extendedEuclid(a,b) ;

    soln.xx*= c ; soln.yy *= c ;

    i64 lo = Ceil(-soln.xx,b) , hi = Floor(soln.yy,a) ;

    return max( 0LL , hi-lo+1 ) ;
}

int main()
{
    i64 i , j , k , l , m , n ;

    i64 t =1 , tc ;

    i64 a , b , c , p ;

    scanf("%lld",&tc) ;

    while(t<=tc)
    {
        scanf("%lld %lld %lld %lld",&a,&b,&c,&p) ;

        printf("Case %lld: ",t++) ;
```

```
    i64 g = gcd( a , gcd(b,c) ) ;

    if(p%g!=0)
    {
        printf("0\n") ;
        continue ;
    }

    a /= g ; b /= g ; c/=g ; p/=g ;

    i64 ans = 0LL ;

    while( p>=0 )
    {
        ans += f(a,b,p) ;
        p -= c ;
    }

    printf("%lld\n",ans) ;
}

return 0 ;
}
```

22 sum of (p*i+r) q

```
/*
i64 res = 0 ;
for(int i=0 ; i<n ; i++) res += ( p*i+r )/q ;
return res ;
*/
//this function does the above thing in logarithmic time
i64 findSum(i64 n, i64 p, i64 r, i64 q) {
  if (p == 0) {
    return (r / q) * n;
  }
  if (p >= q || r >= q) {
    return ((p / q) * (n - 1) + 2 * (r / q)) * n / 2 +
        findSum(n, p % q, r % q, q);
  }
  return findSum((p * n + r) / q, q, (p * n + r) % q, p);
}
```