Team Note of BFS_BROUGHT_ME_HERE

Seyun Roh, Geonhui Yoo, Min Shin

Contents

1 Graph

1.1 Hopcroft-Karp Bipartite Matching

Time Complexity: $\mathcal{O}(E\sqrt{V})$

```
#include <bits/stdc++.h>
using namespace std;
const int MX=1e6+5;
```

```
vector<int> c[MX];
int a[MX],b[MX],n,m,visit[MX],level[MX];

void bfs(){
    queue<int> Q;
    for(int i=1 ; i<=n ; i++){
        level[i]=0;
        if(!visit[i])Q.push(i);
    }
    while(!Q.empty()){
        int v=Q.front();    Q.pop();
        for(auto &u: c[v])
            if(b[u] && level[b[u]]==0){
                level[b[u]]=level[v]+1;
                Q.push(b[u]);
            }
    }
}

bool dfs(int v){
    for(auto &u: c[v]){
        if(b[u]==0 || level[b[u]]==level[v]+1 && dfs(b[u])){
            visit[v]=1;a[v]=u; b[u]=v;
            return 1;
        }
    }
    return 0;
}
int main(){
    cin>>n;
    int i;
    for(i=1 ; i<=n ; i++){
        int x,y;
        cin>>x>>y;
        c[i].push_back(x);
        c[i].push_back(y);
    }

    while(1){
        bfs();  int flow=0;
        for(i=1 ; i<=n ; i++) if(!visit[i] && dfs(i))flow++;
        if(flow==0)break;
        m+=flow;
    }
    if(m<n)cout<<-1;
    else for(i=1 ; i<=n ; i++)cout<<a[i]<<"\n";
    return 0;
}
```

1.2 2-SAT & SCC

```
#include <bits/stdc++.h>
using namespace std;
const int MX=1e4+5;
vector<int> a[2*MX],c[2*MX];
int b[MX*2],cnt,visit[MX*2],n,m;
stack<int> S;
int f(int x){
    if(x>n)return x-n;
    else return n+x;
}
void dfs2(int v){
    visit[v]=1;
    for(auto &u:c[v])if(!visit[u]) dfs2(u);
    b[v]=cnt;
}
void dfs(int v){
    visit[v]=1;
    for(auto &u:a[v])if(!visit[u])dfs(u);
    S.push(v);
}
int main(){
    scanf("%d %d",&n,&m);
    int i;
    for(i=1 ; i<=m ; i++){
        int x,y;
        scanf("%d %d",&x,&y);
        if(x<0)x=n-x;
        if(y<0)y=n-y;
        a[f(x)].push_back(y);
```

```
        a[f(y)].push_back(x);
        c[y].push_back(f(x));
        c[x].push_back(f(y));
    }
    for(i=1 ; i<=n*2 ; i++)if(visit[i]==0)dfs(i);
    for(i=1 ; i<=n*2 ; i++)visit[i]=0;
    while(!S.empty()){
        int v=S.top();
        S.pop();
        if(visit[v]==0){
            ++cnt;
            dfs2(v);
        }
    }
    for(i=1 ; i<=n ; i++){
        if(b[i]==b[n+i]){
            printf("0");
            return 0;
        }
    }
    printf("1\n");
    for(i=1 ; i<=n ; i++)printf("%d ",b[i]>b[n+i]);
    return 0;
}
```

1.3 Biconnected Component BCC

```
#include <bits/stdc++.h>
using namespace std;
const int MX=1e5+5;
int sz[MX], bccsz[MX],dth[MX], up[MX];
long long cnt[MX];
int w;
vector<int> a[MX];
int n,m;
long long f(int v,int p){
    long long ret=0;
    up[v]=dth[v]=dth[p]+1;
    bccsz[v]=1;
    long long k=1;
    for(auto u: a[v]){
        if(dth[u]){
            up[v]=min(dth[u],up[v]);
            continue;
        }
        ret+=f(u,v);
        up[v]=min(up[v],up[u]);
        if(up[u]<dth[v]){
            bccsz[v]+=bccsz[u];
            cnt[v]+=cnt[u];
        }
        else k+=sz[u];
    }
    cnt[v]+=k*(k-1);
    if(p && up[v]>=dth[p]) ret+=(long long)bccsz[v]*(cnt[v]+(long
    long)(w-sz[v])*(w-sz[v]-1));
    return ret;
}
void dfs(int v){
    sz[v]=1;
    for(auto u: a[v]){
        if(sz[u]==0){
            dfs(u);
            sz[v]+=sz[u];
        }
    }
}
int main(){
    cin>>n>>m;
    for(int i=0 ; i<m ; i++){
        int u,v;
        cin>>u>>v;
        a[u].push_back(v);
        a[v].push_back(u);
    }
    long long ans=0;
    for(int i=1 ; i<=n ; i++){
        if(!sz[i]){
            dfs(i);
            w=sz[i];
```

```
            ans+=(long long)w*(w-1)*(w-2)-f(i,0);
        }
    }
    cout<<ans;
    return 0;
}
```

1.4 Blossom Algorithm

```
const int MAXN = 2020 + 1;
// 1-based Vertex index
//shout out to DeobureoMinkyuParty
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;
}
```

1.5 Dinic's and L-R Flow

```
void add(int s,int e,int l,int r){
    a[s].push_back({e,r-l,a[e].size()});
    a[e].push_back({s,0,a[s].size()-1});
    d[s]+=l;
    d[e]-=l;
}

int dfs(int v,int f){
    if(v==sink)return f;
    for(int i=iter[v] ; i<a[v].size() ; iter[v]=++i){
        int u=a[v][i].e;
        int c=a[v][i].c;
        if(level[u]>level[v] && c>0){
            int ret=dfs(u,min(c,f));
            if(ret){
                a[v][i].c-=ret;
                a[u][a[v][i].inv].c+=ret;
                return ret;
            }
        }
    }
    return 0;
}

void bfs(){
    int i,j;
    for(i=0 ; i<=sink ; i++){
        level[i]=0;
        iter[i]=0;
    }
    queue<int> Q;
    Q.push(source);
    while(!Q.empty()){
        int v=Q.front();
        Q.pop();
        for(auto k: a[v]){
            if(k.c && level[k.e]==0 &&k.e!=source){
                level[k.e]=level[v]+1;
                Q.push(k.e);
            }
        }
    }
}
int main(){
    int i,j;
    cin>>n>>m;
    for(i=1 ; i<=n ; i++){
        for(j=1 ; j<=m ; j++)cin>>b[i][j];
        cin>>rs[i];
    }
    for(i=1 ; i<=m ; i++)cin>>cs[i];
    for(i=1 ; i<=n ; i++){
        add(0,i,(int)floor(rs[i]),(int)ceil(rs[i]));
        for(j=1 ; j<=m ; j++)

        add(i,200+200*(i-1)+j,(int)floor(b[i][j]),(int)ceil(b[i][j]));
    }

    for(i=1 ; i<=m ; i++){
        for(j=1 ; j<=n ; j++)add(200+200*(j-1)+i,200*201+i,0,2e9);
        add(200*201+i,source-1,(int)floor(cs[i]),(int)ceil(cs[i]));
    }

    int D=0;
    add(source-1,0,0,2e9);
    for(i=0 ; i<sink ; i++){
        if(d[i]<0)
            add(source,i,0,-d[i]);
        if(d[i]>0){
            add(i,sink,0,d[i]);
            D+=d[i];
        }
    }
```

```
    while(1){
        bfs();
        if(level[sink]==0)break;
        int flow;
        do{ flow=dfs(source,2e9); }while(flow);
    }
```

1.6 Euler Path

```
#include <bits/stdc++.h>
using namespace std;
int a[1001][1001];
int b[1001];
int n;

void dfs(int v,int d){
    int cnt=0;
    if(v==d){
        printf("%d ",v);
        return;
    }
    for(int i=1 ; i<=n ; i++){
        while(a[v][i]){              ///v와 i 사이에 간선이 있을 시
            a[v][i]--;               ///간선 제거
            a[i][v]--;
            b[v]--;                  ///차수 감소
            b[i]--;
            if(cnt==0)dfs(i,d),cnt++;///더 이상 연결된 간선이 없으면
            가던 길 감
            else dfs(i,v);           ///남은 간선이 있으면 v로 돌아오는
            서킷 탐색
        }
    }
    printf("%d ",v);
}

int main(){
    int i,j;
    cin>>n;
    for(i=1 ; i<=n ; i++){
        for(j=1 ; j<=n ; j++){
            scanf("%d",&a[i][j]);
            b[i]+=a[i][j];
        }
    }

    for(i=1 ; i<=n ; i++){
        if(b[i]%2){            ///차수가 홀수일 경우
            printf("-1");
            return 0;
        }
    }

            ///1부터 시작
    for(i=1 ; i<=n ; i++){
        while(a[1][i]){          ///i와 연결되어있으면
            a[1][i]--;           ///간선 제거
            a[i][1]--;
            b[1]--;              ///차수 감소
            b[i]--;
            dfs(i,1);            ///i에서 1로 가는 경로 탐색
        }
    }
    printf("1 ");
    return 0;
}
```

1.7 Min Cost-Max Flow

```
int main(){
    cin>>n>>m;
    int i,j;
    for(i=1 ; i<=m ; i++)c[n+i][n+m+1]=1;
    for(i=1 ; i<=n ; i++){
        int s; cin>>s; [0][i]=1;
        for(j=0 ; j<s ; j++){
            int x,y;  cin>>x>>y; c[i][n+x]=1; w[i][n+x]=y;
            w[n+x][i]=-y;
```

```
        }
    }
    while(1){
        for(i=1 ; i<=n+m+1 ; i++)dist[i]=1e9;
        queue<int> Q;
        Q.push(0);
        inque[0]=1;
        while(!Q.empty()){
            int v=Q.front(); Q.pop();
            inque[v]=0;
            for(i=0 ; i<=n+m+1 ; i++){
                if( c[v][i] && dist[i]>dist[v]+w[v][i] ){
                    p[i]=v;
                    dist[i]=dist[v]+w[v][i];
                    if(!inque[i]){
                        Q.push(i);  inque[i]=1;
                    }
                }
            }
        }
        if(dist[n+m+1]==1e9)break;
        flow++;
        for(i=n+m+1 ; i ; i=j){
            j=p[i];] ans+=w[j][i];  c[j][i]--;  c[i][j]++;
        }
    }
    cout<<flow<<endl<<ans;
    return 0;
}
```

2 Data Structure

2.1 Binary Indexed Tree

```
int main(){
    scanf("%d %d %d",&n,&m,&k);
    int i,j;
    for(nn=1 ; nn<n ; nn*=2);
    for(i=1 ; i<=n ; i++){
        scanf("%lld",&a[i]);
        for(j=i ; j<nn ; j+=(j&-j))bit[j]+=a[i];
    }

    for(i=1 ; i<=m+k ; i++){
        int b,c,d;
        scanf("%d %d %d",&b,&c,&d);
        if(b==1){
            for(j=c ; j<nn ; j+=(j&-j))bit[j]+=d-a[c];
            a[c]=d;
        }
        else{
            long long ans=0;
            for(j=d ; j>=1 ; j-=(j&-j))ans+=bit[j];
            for(j=c-1 ; j>=1 ; j-=(j&-j))ans-=bit[j];
            printf("%lld\n",ans);
        }
    }
    return 0
}
```

2.2 Find k'th element in std::set

```
#include <ext/pb_ds/assoc_container.hpp> // Common file
#include <ext/pb_ds/tree_policy.hpp> // Including
tree_order_statistics_node_update
#include <ext/pb_ds/detail/standard_policies.hpp>
template<
    typename Key, // Key type
    typename Mapped, // Mapped-policy
    typename Cmp_Fn = std::less<Key>, // Key comparison functor
    typename Tag = rb_tree_tag, // Specifies which underlying data
    structure to use
    template<
    typename Const_Node_Iterator,
    typename Node_Iterator,
    typename Cmp_Fn_,
    typename Allocator_>
    class Node_Update = null_node_update, // A policy for updating
    node invariants
    typename Allocator = std::allocator<char> > // An allocator type
```

```
    class tree;
typedef tree<
int,
null_type,
less<int>,
rb_tree_tag,
tree_order_statistics_node_update>
ordered_set;
ordered_set X;
    X.insert(1);
    X.insert(2);
    X.insert(4);
    X.insert(8);
    X.insert(16);

    cout<<*X.find_by_order(1)<<endl; // 2
    cout<<*X.find_by_order(2)<<endl; // 4
    cout<<*X.find_by_order(4)<<endl; // 16
    cout<<(end(X)==X.find_by_order(6))<<endl; // true

    cout<<X.order_of_key(-5)<<endl;  // 0
    cout<<X.order_of_key(1)<<endl;   // 0
    cout<<X.order_of_key(3)<<endl;   // 2
    cout<<X.order_of_key(4)<<endl;   // 2
    cout<<X.order_of_key(400)<<endl; // 5
```

2.3 Heavy-Light Decompositinon

```
#include <bits/stdc++.h>
using namespace std;
const int MX=1e5+5;
vector<int> a[MX];///인접리스트
int visit[MX],dth[MX],p[MX][20];
int s[MX];          ///서브트리크기
int ss[MX];       ///체인 i의 크기
int n,m,nn,idx; ///노드개수 쿼리개수 체인개수 세그트리인덱스
int chain[MX]; ///노드i의 체인번호
int head[MX]; /// 체인 i의 꼭대기
int rootid[MX]; /// 체인 i의 세그트리의 루트의 인덱스
struct Node{
    int l,r;
    long long c,lazy;
}seg[MX*3];

void dfs(int v){
    s[v]=1;
    visit[v]=1;
    for(auto u: a[v]){
        if(!visit[u]){
            dth[u]=dth[v]+1;
            p[u][0]=v;
            dfs(u);
            s[v]+=s[u];
        }
    }
    visit[v]=0;
}
inline int lca(int v,int u){
    int i,j;
    if(dth[v]>dth[u])swap(u,v);
    for(i=19 ; i>=0 ; i--)if(dth[u]-dth[v]>=(1<<i))u=p[u][i];
    if(u==v)return v;
    for(i=19 ; i>=0 ; i--){
        if(p[u][i]!=p[v][i]){
            u=p[u][i];
            v=p[v][i];
        }
    }
    return p[v][0];
}
void dfs2(int v,int k){
    chain[v]=k;
    ss[k]++;
    visit[v]=1;
    int t=0;
    for(auto u: a[v])if(!visit[u])if(s[u]>s[t])t=u;

    if(t==0)return;
    dfs2(t,k);
    for(auto u:a[v])if(!visit[u] && u!=t)head[++nn]=u,dfs2(u,nn);
```

```
}
void make_tree(int lef,int rig,int lev){
    if(lef==rig)return;
    int mid=(lef+rig)/2;
    seg[lev].l=++idx;
    seg[lev].r=++idx;
    make_tree(lef,mid,seg[lev].l);
    make_tree(mid+1,rig,seg[lev].r);
}
void update(int x,int y,int lef,int rig,int lev){
    seg[lev].c+=seg[lev].lazy*(rig-lef+1);
    seg[seg[lev].l].lazy+=seg[lev].lazy;
    seg[seg[lev].r].lazy+=seg[lev].lazy;
    seg[lev].lazy=0;
    if(x>rig || lef>y || x>y)return;
    if(x<=lef && rig<=y){
        seg[lev].lazy++;
        return;
    }
    int mid=(lef+rig)/2;
    update(x,y,lef,mid,seg[lev].l);
    update(x,y,mid+1,rig,seg[lev].r);
    seg[lev].c=seg[seg[lev].l].c+seg[seg[lev].r].c;
}
void upd(int u,int v){
    if(dth[u]<=dth[v] || u==0)return;
    int k=chain[u];
    while(k!=chain[v]){
        update(0,dth[u]-dth[head[k]],0,ss[k]-1,rootid[k]);
        u=p[head[k]][0];
        k=chain[u];
    }

    update(dth[v]-dth[head[k]]+1,dth[u]-dth[head[k]],0,ss[k]-1,rootid[k]);
}
long long g(int x,int y,int lef,int rig,int lev){
    seg[lev].c+=seg[lev].lazy*(rig-lef+1);
    seg[seg[lev].l].lazy+=seg[lev].lazy;
    seg[seg[lev].r].lazy+=seg[lev].lazy;
    seg[lev].lazy=0;
    if(x>rig || y<lef ||x>y)return 0;
    if(x<=lef && rig<=y)return seg[lev].c;
    int mid=(lef+rig)/2;
    return g(x,y,lef,mid,seg[lev].l)+g(x,y,mid+1,rig,seg[lev].r);
}
inline long long f(int u,int v){
    if(dth[u]<=dth[v] || u==0)return 0;
    long long ret=0;
    int k=chain[u];
    while(k!=chain[v]){
        ret+=g(0,dth[u]-dth[head[k]],0,ss[k]-1,rootid[k]);
        u=p[head[k]][0];
        k=chain[u];
    }
    return
    ret+g(dth[v]-dth[head[k]]+1,dth[u]-dth[head[k]],0,ss[k]-1,rootid[k]);
}
int main(){
cin>>n>>m;
    int i,j;
    for(i=1 ; i<n ; i++){
        int u,v;
        cin>>u>>v;
        a[u].push_back(v);
        a[v].push_back(u);
    }
    dfs(1);
    for(i=1 ; i<20 ; i++)for(j=1 ; j<=n ;
    j++)p[j][i]=p[p[j][i-1]][i-1];
    head[1]=1;
    dfs2(1,++nn);
    for(i=1 ; i<=nn ; i++){
        rootid[i]=++idx;
        make_tree(0,ss[i]-1,rootid[i]);
    }
    for(i=1 ; i<=m ; i++){
        char c;
        int u,v;
        cin>>c>>u>>v;
        int t=lca(u,v);
```

```
        if(c=='P'){
            upd(v,t);
            upd(u,t);
        }
        else  cout<<f(u,t)+f(v,t)<<"\n";
    }
    return 0;
}
```

2.4 Li Chao Tree

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const ll INF=2e18;
const ll MX=2e12;
int q;
struct Line{
    ll a,b;
    ll f(ll x){
        return a*x+b;
    }
};
struct Node{
    int lef,rig;
    Line l;
};
vector<Node> seg;

void update(int lev,ll xl,ll xr,Line m){
    Line n=seg[lev].l;
    if(m.f(xl)>n.f(xl))swap(m,n);
    ll xm=(xl+xr)/2;
    if(m.f(xr)<n.f(xr)){
        seg[lev].l=n;
        return;
    }
    if(m.f(xm)>=n.f(xm)){
        seg[lev].l=m;
        if(seg[lev].lef==-1){
            seg[lev].lef=seg.size();
            seg.push_back({-1,-1,{0,-INF}});
        }
        update(seg[lev].lef,xl,xm,n);
    }
    else{
        seg[lev].l=n;
        if(seg[lev].rig==-1){
            seg[lev].rig=seg.size();
            seg.push_back({-1,-1,{0,-INF}});
        }
        update(seg[lev].rig,xm,xr,m);
    }
}
ll g(int lev,ll xl,ll xr,ll x){
    if(lev==-1)return -INF;
    ll xm=(xl+xr)/2;
    if(x<=xm)return max(seg[lev].l.f(x),g(seg[lev].lef,xl,xm,x));
    return max(seg[lev].l.f(x),g(seg[lev].rig,xm,xr,x));
}
int main(){
seg.push_back({-1,-1,{0,-INF}});
    cin>>q;
    for(int i=1 ; i<=q ; i++){
        int k;
        cin>>k;
        if(k==1){ ll a,b;
            cin>>a>>b;
            update(0,-MX,MX,{a,b});
        }
        else{ ll x;
            cin>>x;
            cout<<g(0,-MX,MX,x)<<"\n";
        }
    }
return 0;
}
```

2.5 MO's Trick on Tree

```
int get_lca(int u,int v){
    int i;
    if(dth[v]<dth[u])swap(u,v);
    for(i=30 ; i>=0 ; i--)if(dth[v]-dth[u]>=(1<<i))v=p[v][i];
    if(u==v)return v;
    for(i=30 ; i>=0 ; i--)if(p[v][i]!=p[u][i])v=p[v][i],u=p[u][i];
    return p[v][0];
}
struct A{
    int u,v,l,r,lca,idx;
    void process(){
        if(in[u]>in[v])swap(u,v);
        lca=get_lca(u,v);
        if(lca==u){
            l=in[u];
            r=in[v];
        }
        else{
            l=out[u];
            r=in[v];
        }
    }
    bool operator<(const A&t){
        if(l/k==t.l/k)return r<t.r;
        return l/k<t.l/k;
    }
}q[MX];

void dfs(int v){
    in[v]=++nn;
    visit[v]=1;
    b[nn]=v;
    for(auto &u:a[v]){
        if(!visit[u]){
            p[u][0]=v;
            dth[u]=dth[v]+1;
            dfs(u);
        }
    }
    out[v]=++nn;
    b[nn]=v;
}
int main(){
    cin.tie(0);
    cout.tie(0);
    ios_base::sync_with_stdio(0);
    cin>>n;
    int i,j;
    for(i=1 ; i<=n ; i++)cin>>c[i];
    for(i=1 ; i<n ; i++){
        int u,v;
        cin>>u>>v;
        a[v].push_back(u);
        a[u].push_back(v);
    }
    dfs(1);
    k=sqrt(nn);
    for(i=1 ; i<30 ; i++)for(j=1 ; j<=n ;
    j++)p[j][i]=p[p[j][i-1]][i-1];
    cin>>m;
    for(i=1 ; i<=m ; i++){
        int u,v;
        cin>>u>>v;
        q[i]={u,v,0,0,0,i};
        q[i].process();
    }

    sort(q+1,q+m+1);
    int s=0;
    for(i=1 ; i<=m ; i++){
        for(j=q[i-1].r+1 ; j<=q[i].r ; j++){
            int t=b[j];
            check[t]++;
            if(check[t]==1){
                cnt[c[t]]++;
                if(cnt[c[t]]==1)s++;
            }
           if(check[t]==2 ){
```

```
            cnt[c[t]]--;
            if(cnt[c[t]]==0)s--;
        }
    }
    for(j=q[i].l ; j<q[i-1].l ; j++){
        int t=b[j];
        check[t]++;
        if(check[t]==1){
            cnt[c[t]]++;
            if(cnt[c[t]]==1)s++;
        }
        if(check[t]==2 ){
            cnt[c[t]]--;
            if(cnt[c[t]]==0)s--;
        }
    }
    for(j=q[i].r+1 ; j<=q[i-1].r ; j++){
        int t=b[j];
        check[t]--;
        if(check[t]==1){
            cnt[c[t]]++;
            if(cnt[c[t]]==1)s++;
        }
        if(check[t]==0){
            cnt[c[t]]--;
            if(cnt[c[t]]==0)s--;
        }
    }
    for(j=q[i-1].l ; j<q[i].l ; j++){
        int t=b[j];
        check[t]--;
        if(check[t]==1){
            cnt[c[t]]++;
            if(cnt[c[t]]==1)s++;
        }
        if( check[t]==0){
            cnt[c[t]]--;
            if(cnt[c[t]]==0)s--;
        }
    }
        ans[q[i].idx]=s;
       /// cout<<q[i].lca<<endl;
        if(q[i].lca!=q[i].u)if(cnt[c[q[i].lca]]==0)ans[q[i].idx]++;
    }

    for(i=1 ; i<=m ; i++)cout<<ans[i]<<"\n";
    return 0;
}
```

2.6 Offline Dynamic Connectivity

```
#include<bits/stdc++.h>
using namespace std;
const int MX=6e5+5;
vector<int> seg[MX*2];
int n,m;
int p[MX];
long long b[MX];
int sz[MX];
int c[MX];
int nn;
unordered_map<long long,int> mp;
stack<pair<int,int>> S;
struct A{
    int u,v,l,r;
};
vector<A> a;
int ans;

void f(int lef,int rig,int k,int lev){
    if(a[k].l>rig || a[k].r<lef)return;
    if(a[k].l<=lef && rig<=a[k].r){
        seg[lev].push_back(k);
        return;
    }
    if(lef==rig)return;
    int mid=(lef+rig)/2;
    f(lef,mid,k,lev*2);
    f(mid+1,rig,k,lev*2+1);
}
```

```
int F(int x){
    return x==p[x] ? x:F(p[x]);
}
void U(int x,int y,int k){
    int xx=F(x);
    int yy=F(y);
    if(xx==yy)return;
    if(sz[xx]>sz[yy])swap(xx,yy);
    sz[yy]+=sz[xx];
    p[xx]=yy;
    S.push({xx,k});
    ans--;
}
void g(int lef,int rig,int lev){
    for(auto k: seg[lev])U(a[k].u,a[k].v,lev);
    if(lef<rig){
        int mid=(lef+rig)/2;
        g(lef,mid,lev*2);
        g(mid+1,rig,lev*2+1);
    }
    if(lef==rig && c[lef])cout<<ans<<"\n";

    while(!S.empty() && S.top().second==lev){
        int x=S.top().first;
        S.pop();
        sz[p[x]]-=sz[x];
        p[x]=x;
        ans++;
    }

}
int main(){
    cin.tie(0);
    ios_base::sync_with_stdio(0);
    freopen("connect.in","r",stdin);
    freopen("connect.out","w",stdout);
    cin>>n>>m;
    ans=n;
    for(nn=1 ; nn<m ; nn*=2);
    int i;
    for(i=1 ; i<=m ; i++){
        char k;
        cin>>k;
        if(k=='?')c[i]=1;
        else{
            int u,v;
            cin>>u>>v;
            if(u>v)swap(u,v);
            long long w=(long long)n*u+v;
            if(k=='-'){
                a.push_back({u,v,mp[w],i});
                b[mp[w]]=0;
            }
            else{
                mp[w]=i;
                b[i]=w;
            }
        }
    }
    for(i=1 ; i<=m
    ;i++)if(b[i])a.push_back({(b[i]-1)/n,(b[i]-1)%n+1,i,m});

    for(i=0 ; i<a.size() ; i++)f(1,nn,i,1);


    for(i=1 ; i<=n ; i++)p[i]=i,sz[i]=1;
    g(1,nn,1);
    return 0;
}
```

3 Geometry

3.1 Closest Two Points

```
inline int dist(int x1,int x2,int y1,int y2){
    return (x1-x2)*(x1-x2)+(y1-y2)*(y1-y2);
}
int f(int l,int r){
    if(r==l+1)return dist(a[r].x,a[l].x,a[r].y,a[l].y);
    if(l>=r)return 2e9;
```

```
    int i,j;

    int mid=(l+r)/2;
    int ll=mid;
    int d=min(f(l,mid),f(mid+1,r));
    for(ll=mid ; ll>l &&
    (a[mid].x-a[ll-1].x)*(a[mid].x-a[ll-1].x)<d; ll--);
    int rr=mid;
    for(rr=mid ; rr<r && (a[mid].x-a[rr+1].x)*(a[mid].x-a[rr+1].x)<d
    ; rr++);
    for(i=ll ; i<=rr ; i++)b[i-ll].x=a[i].y,b[i-ll].y=a[i].x;
    sort(b,b+rr-ll+1);

    for(i=0 ; i<=rr-ll ; i++){
        for(j=i+1 ; j<=rr-ll && j<=i+6 ; j++){
            d=min(d,dist(b[i].x,b[j].x,b[i].y,b[j].y));
        }
    }
    return d;
}
int main(){
    scanf("%d",&n);
    int i;
    for(i=1 ; i<=n ; i++)scanf("%d %d",&a[i].x,&a[i].y);
    sort(a+1,a+n+1);

    printf("%d",f(1,n));
    return 0;
}
```

3.2 Rotating Calipers

```
#include <bits/stdc++.h>
using namespace std;
int n,t;
#define x first
#define y second
typedef long long ll;
typedef pair<ll,ll> pp;
pp a[300005];
pp v[300005];
pp s[300005];
pp b[300005];
ll ans=8e18+2e17;
int tt;
long double f(pp q,pp w,pp e){
    ll aa=q.y-w.y;
    ll bb=w.x-q.x;
    ll cc=q.x*w.y-q.y*w.x;
    return (long double)(aa*e.x+bb*e.y+cc)/sqrt((long
    double)(aa*aa+bb*bb));
}
ll ccw(pp q,pp w,pp e){
    return (w.x-q.x)*(e.y-w.y)-(w.y-q.y)*(e.x-w.x);
}
ll dist(pp q,pp w){
    return (w.x-q.x)*(w.x-q.x)+(w.y-q.y)*(w.y-q.y);
}
bool comp(pp q,pp w){
    ll k=ccw(b[1],q,w);
    return k ? k>0 : dist(b[1],q)<dist(b[1],w);
}
ll process(ll ti){
    if(ti<0 || ti>t)return 1e18;
    int i,j;
    ll ret=0;
    for(i=1 ; i<=n ; i++)b[i]={a[i].x+ti*v[i].x,a[i].y+ti*v[i].y};
    sort(b+1,b+n+1);
    sort(b+2,b+n+1,comp);
    ret=dist(b[1],b[n]);
    int to=2;
    s[1]=b[1];
    s[2]=b[2];
    b[n+1]=b[1];
    for(i=3 ; i<=n+1 ; i++){
        while(to>=2 && ccw(s[to-1],s[to],b[i])<=0)to--;
        s[++to]=b[i];
    }
    to--;
    if(to<3)return ret;
```

```
   /// cout<<to<<" "<<ti<<endl;
    j=2;
    for(i=1 ; i<=to ; i++){
        int ii=i%to+1;
        if(j>to)j=1;
        while(f(s[i],s[ii],s[j%to+1])>f(s[i],s[ii],s[j]) ){
            ret=max({ret,dist(s[i],s[j]),dist(s[ii],s[j])});
            j++;
            if(j>to)j=1;
        }
        ret=max({ret,dist(s[i],s[j]),dist(s[ii],s[j])});
    }
   /// cout<<ret<<endl;
    return ret;
}
int main(){
    cin.tie(0);
    cout.tie(0);


    cin>>n;
    int i;
    for(i=1 ; i<=n ; i++)cin>>a[i].x>>a[i].y;
    int lef=0;
    int rig=t;
    printf("%.8Lf",sqrt((long double)process(0)));
    return 0;
    while(rig>=lef){
        int mid=(lef+rig)/2;
        ll q=process(mid-1);
        ll w=process(mid);
        ll e=process(mid+1);
        if(q<ans){
            ans=q;
            tt=mid-1;
        }
        if(w<ans){
            ans=w;
            tt=mid;
        }
        if(e<ans){
            ans=e;
            tt=mid+1;
        }
        if(q>w && w>e)lef=mid+1;
        else rig=mid-1;
    }
    cout<<tt<<endl<<ans;

    return 0;
}
```

3.3 Segment Intersection

```
int ccw(pair<int, int> a, pair<int, int> b, pair<int, int> c) {
    int op = a.first*b.second + b.first*c.second + c.first*a.second;
    op -= (a.second*b.first + b.second*c.first + c.second*a.first);
    if (op > 0)return 1;
    else if (op == 0)return 0;
    else return -1;
}

///출처: https://jason9319.tistory.com/358 [ACM-ICPC 상 탈 사람]
int isIntersect(pair<pp, pp> x, pair<pp, pp> y) {
    pair<int, int> a = x.first;
    pair<int, int> b = x.second;
    pair<int, int> c = y.first;
    pair<int, int> d = y.second;
    int ab = ccw(a, b, c)*ccw(a, b, d);
    int cd = ccw(c, d, a)*ccw(c, d, b);
    if (ab == 0 && cd == 0) {
        if (a > b)swap(a, b);
        if (c > d)swap(c, d);
        return c <= b&&a <= d;
    }
    return ab <= 0 && cd <= 0;
}
```

3.4 Smallest Enclosing Sphere

```
#include <bits/stdc++.h>
using namespace std;
double x[1001];
double y[1001];
double z[1001];
int n;
double dist(double a,double b,double c){
    return a*a+b*b+c*c;
}
int main(){
    cin>>n;
    int i,j;
    double xx=0,yy=0,zz=0;
    for(i=1 ; i<=n;
    i++)cin>>x[i]>>y[i]>>z[i],xx+=x[i],yy+=y[i],zz+=z[i];
    xx/=n;
    yy/=n;
    zz/=n;
    double p=0.1;
    int k=0;
    double s=0;
    for(i=1 ; i<=10000 ; i++){
        k=s=0;
        for(j=1 ; j<=n ; j++){
            if(dist(x[j]-xx,y[j]-yy,z[j]-zz)>s){
                s=dist(x[j]-xx,y[j]-yy,z[j]-zz);
                k=j;
            }
        }
        xx+=(x[k]-xx)*p;
        yy+=(y[k]-yy)*p;
        zz+=(z[k]-zz)*p;
        p*=0.998;
    }
    printf("%.2f",sqrt(s));

    return 0;
}
```

4 DP

4.1 Convex Hull Trick

```
inline double f(pp x,pp y){return
(double)(y.first-x.first)/(x.second-y.second);}
int main(){
    cin>>n;
    int i,j;
    for(i=1 ; i<=n ; i++)cin>>a[i];
    for(i=1 ; i<=n  ;i++)cin>>b[i];
    int t=0;
    s[++t]={-b[n],b[1]};
    j=1;
    long long y=b[n];
    for(i=2 ; i<=n ; i++){
        while(j<t && f(s[j],s[j+1])<=(double)a[i])j++;
        if(j>t)j=t;
        y+=b[n];
        dp=s[j].second*a[i]+s[j].first+b[n]*(i-1);
        pp k={dp-y,b[i]};
        while(t>=2 && f(s[t],s[t-1])>=f(s[t-1],k))t--;
        if(t==1)if(s[1].first>=k.first)t--;
        s[++t]=k;
    }
    cout<<dp;
    return 0;
}
```

4.2 SOS DP

```
#include <bits/stdc++.h>
using namespace std;
#define f first
#define s second
pair<int,int> dp[(1<<21)];
int n;
int a[100005];
```

```
int ans;
int main(){
    cin>>n;
    int i,j;
    for(i=1 ; i<=n ; i++){
        cin>>a[i];
        dp[a[i]].s=dp[a[i]].f;
        dp[a[i]].f=i;
    }
    for(j=0 ; j<21 ; j++){
        for(i=0 ; i<(1<<21) ; i++){
            if(i&(1<<j))continue;
            int ii=i+(1<<j);
            if(dp[ii].s>dp[i].f){
                dp[i].f=dp[ii].f;
                dp[i].s=dp[ii].s;
            }
            else if(dp[ii].f>dp[i].f){
                dp[i].s=dp[i].f;
                dp[i].f=dp[ii].f;
            }
            else if(dp[ii].f>dp[i].s){
                dp[i].s=dp[ii].f;
            }
        }
    }
    for(i=1 ; i<=n-2 ; i++){
        int w=0;
        for(j=20 ; j>=0 ; j--){
            if(a[i]&(1<<j))continue;
            if(dp[w|(1<<j)].s>i)w|=(1<<j);
        }
        ans=max(ans,w|a[i]);
    }
    cout<<ans;
    return 0;
}
```

5 Math

5.1 Extended Euclidian Algorithm

```
int ee(int a,int b,int &s,int &t) {
    if (!b){
        s = 1, t = 0;
        return a;
    }
    int q = a/b, r = a%b, sp, tp;
    int g = ee(b,r,sp,tp);
    s = tp; t = sp-tp*q;
    return g;
}
```

5.2 Fast Fourier Transform

```
#define _USE_MATH_DEFINES
#include <bits/stdc++.h>
#include <math.h>
#include <complex>
#include <vector>
#include <algorithm>
using namespace std;
#define sz(v) ((int)(v).size())
#define all(v) (v).begin(),(v).end()
typedef complex<double> base;
void fft(vector <base> &a, bool invert)
{
    int n = sz(a);
    for (int i=1,j=0;i<n;i++){
        int bit = n >> 1;
        for (;j>=bit;bit>>=1) j -= bit;
        j += bit;
        if (i < j) swap(a[i],a[j]);
    }
    for (int len=2;len<=n;len<<=1){
        double ang = 2*M_PI/len*(invert?-1:1);
        base wlen(cos(ang),sin(ang));
        for (int i=0;i<n;i+=len){
            base w(1);
```

```
            for (int j=0;j<len/2;j++){
                base u = a[i+j], v = a[i+j+len/2]*w;
                a[i+j] = u+v;
                a[i+j+len/2] = u-v;
                w *= wlen;
            }
        }
    }
    if (invert){
        for (int i=0;i<n;i++) a[i] /= n;
    }
}

void multiply(const vector<int> &a,const vector<int> &b,vector<int>
&res)
{
    vector <base> fa(all(a)), fb(all(b));
    int n = 1;
    while (n < max(sz(a),sz(b))) n <<= 1;
    fa.resize(n); fb.resize(n);
    fft(fa,false); fft(fb,false);
    for (int i=0;i<n;i++) fa[i] *= fb[i];
    fft(fa,true);
    res.resize(n);
    for (int i=0;i<n;i++) res[i] =
    int(fa[i].real()+(fa[i].real()>0?0.5:-0.5));
}
```

5.3 Number Theoretic Transform

```
#include <cstdio>

const int A = 7, B = 26, P = A << B | 1, R = 3;
const int SZ = 20, N = 1 << SZ;

int Pow(int x, int y) {
    int r = 1;
    while (y) {
        if (y & 1) r = (long long)r * x % P;
        x = (long long)x * x % P;
        y >>= 1;
    }
    return r;
}

void FFT(int *a, bool f) {
    int i, j, k, x, y, z;
    j = 0;
    for (i = 1; i < N; i++) {
        for (k = N >> 1; j >= k; k >>= 1) j -= k;
        j += k;
        if (i < j) {
            k = a[i];
            a[i] = a[j];
            a[j] = k;
        }
    }
    for (i = 1; i < N; i <<= 1) {
        x = Pow(f ? Pow(R, P - 2) : R, P / i >> 1);
        for (j = 0; j < N; j += i << 1) {
            y = 1;
            for (k = 0; k < i; k++) {
                z = (long long)a[i | j | k] * y % P;
                a[i | j | k] = a[j | k] - z;
                if (a[i | j | k] < 0) a[i | j | k] += P;
                a[j | k] += z;
                if (a[j | k] >= P) a[j | k] -= P;
                y = (long long)y * x % P;
            }
        }
    }
    if (f) {
        j = Pow(N, P - 2);
        for (i = 0; i < N; i++) a[i] = (long long)a[i] * j % P;
    }
}

int X[N];

int main() {
```

```
    int i, n;
    scanf("%d", &n);
    for (i = 0; i <= n; i++) scanf("%d", &X[i]);
    FFT(X, false);
    for (i = 0; i < N; i++) X[i] = (long long)X[i] * X[i] % P;
    FFT(X, true);
    for (i = 0; i <= n + n; i++) printf("%d ", X[i]);
}
```

5.4 Kitamasa Method

```
void Kitamasa(long long m,int n) {
    int i,j;
    if(m==1){
        d[1]=1;
        return;
    }
    Kitamasa(m>>1,n);
    for(i=1 ; i<=(n<<1) ; i++)b[i]=0;
    for(i=1 ; i<=n ; i++)for(j=1 ; j<=n ;
    j++)b[i+j]=(b[i+j]+d[i]*d[j])%D;
    for(i=n*2 ; i>=n+1 ; i--)for(j=1 ; j<=n ;
    j++)b[i-j]=(c[j]*b[i]+b[i-j])%D;
    for(i=1 ; i<=n ; i++)d[i]=b[i];

    if(m&1){
        for(i=1 ; i<=n ; i++)b[i]=c[n-i+1]*d[n]%D;
        for(i=2 ; i<=n ; i++)b[i]=(d[i-1]+b[i])%D;
        for(i=1 ; i<=n ; i++)d[i]=b[i];
    }
}
int main(){
    int i,j;   long long n,m;
    cin>>n>>m;
    long long ans=0;
    dp[0]=1;
    for(i=1 ; i<=n ; i++)for(j=0 ; j<i ; j++)dp[i]=(dp[j]+dp[i])%D;

    c[n]=dp[n];
    for(i=1 ; i<n ; i++)c[i]=1;
    Kitamasa(m,n);
    a[0]=1;
    for(i=1 ; i<=n ; i++)for(j=1 ; j<=i ;
    j++)a[i]=(a[i]+a[i-j]*c[j])%D;
    for(i=1 ; i<=n ; i++)ans=(a[i]*d[i]+ans)%D;

    cout<<ans;
    return 0;
}
```

5.5 Miller-Rabin Test

```
vector<ull> alist = {2, 7, 61};
//vector<ull> alist = {2, 325, 9375, 28178, 450775, 9780504,
1795265022};
// calculate x^y % m
ull powmod(ull x, ull y, ull m) {
    x %= m;
    ull r = 1ULL;
    while (y > 0) {
        if (y % 2 == 1)
            r = (r * x) % m;
        x = (x * x) % m;
        y /= 2;
    }
    return r;
}

// true for probable prime, false for composite
inline bool miller_rabin(ull n, ull a) {
    ull d = n - 1;
    while (d % 2 == 0) {
        if (powmod(a, d, n) == n-1)
            return true;
        d /= 2;
    }
    ull tmp = powmod(a, d, n);
    return tmp == n-1 || tmp == 1;
}
```

```
bool is_prime(ull n) {
    if (n <= 1)
        return false;
    if (n <= 10000ULL) {
        for (ull i = 2; i*i <= n; i++)
            if (n % i == 0)
                return false;
        return true;
    }
    for (ull a : alist)
        if (!miller_rabin(n, a))
            return false;
    return true;
}
```

5.6 Miller-Rabin & Pollard-Rho

```
namespace miller_rabin{
    lint mul(lint x, lint y, lint mod){ return (__int128) x * y %
    mod; }
  lint ipow(lint x, lint y, lint p){
    lint ret = 1, piv = x % p;
    while(y){
      if(y&1) ret = mul(ret, piv, p);
      piv = mul(piv, piv, p);
      y >>= 1;
    }
    return ret;
  }
  bool miller_rabin(lint x, lint a){
    if(x % a == 0) return 0;
    lint d = x - 1;
    while(1){
      lint tmp = ipow(a, d, x);
      if(d&1) return (tmp != 1 && tmp != x-1);
      else if(tmp == x-1) return 0;
      d >>= 1;
    }
  }
  bool isprime(lint x){
    for(auto &i : {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37}){
      if(x == i) return 1;
      if(x > 40 && miller_rabin(x, i)) return 0;
    }
    if(x <= 40) return 0;
    return 1;
  }
}

namespace pollard_rho{
  lint f(lint x, lint n, lint c){
    return (c + miller_rabin::mul(x, x, n)) % n;
  }
  void rec(lint n, vector<lint> &v){
    if(n == 1) return;
    if(n % 2 == 0){
      v.push_back(2);
      rec(n/2, v);
      return;
    }
    if(miller_rabin::isprime(n)){
      v.push_back(n);
      return;
    }
    lint a, b, c;
    while(1){
      a = rand() % (n-2) + 2;
      b = a;
      c = rand() % 20 + 1;
      do{
        a = f(a, n, c);
        b = f(f(b, n, c), n, c);
      }while(gcd(abs(a-b), n) == 1);
      if(a != b) break;
    }
    lint x = gcd(abs(a-b), n);
    rec(x, v);
    rec(n/x, v);
  }
  vector<lint> factorize(lint n){
```

```
    vector<lint> ret;
    rec(n, ret);
    sort(ret.begin(), ret.end());
    return ret;
  }
};
```

5.7 XOR Maximization with Gaussian Elimination

```
    cin>>n;
    for(int i=1 ; i<=n ; i++){
        long long x;
        cin>>x;
        for(auto &j:b)x=min(x,x^j);
        if(x)b.insert(x);
    }

    for(auto &i:b)ans=max(ans,ans^i);
    cout<<ans;
```

6 String

6.1 Aho-Corasick

```
void make_trie(int lev,int k){
    if(a[k]==0){
        c[lev]=1;
        return;
    }
    if(b[lev][a[k]-'a']==0)b[lev][a[k]-'a']=++nn;
    int y=b[lev][a[k]-'a'];
    make_trie(y,k+1);
}
int dfs(int lev,int k){
    if(c[lev])return 1;
    if(a[k]==0)return 0;
    if(b[lev][a[k]-'a'])return dfs(b[lev][a[k]-'a'],k+1);
    if(lev==1)return dfs(1,k+1);
    return dfs(f[lev],k);
}
int main(){
    cin>>n;
    int i;
    f[1]=1;
    for(i=0 ; i<n ; i++){
        cin>>a;
        make_trie(1,0);
    }
    queue<int> Q;
    Q.push(1);

    while(!Q.empty()){
        int v=Q.front();
        Q.pop();
        for(i=0 ; i<26 ; i++){
            if(!b[v][i])continue; Q.push(b[v][i]);
            int x=f[v],y=b[v][i];
            while(1){
                if(b[x][i] && b[x][i]!=y){
                    f[y]=b[x][i];
                    break;
                }
                if(x==1)f[y]=1; break;
                x=f[x];
            }
            if(c[f[y]])c[y]=1;
        }
    }
    cin>>m;
    for(i=0 ; i<m ; i++){
        cin>>a;
        if(dfs(1,0))cout<<"YES\n";
        else cout<<"NO\n";
    }
    return 0;
}
```

6.2 KMP

```
for(i=1 ; i<m ; i++){
    while(j>0 && b[i]!=b[j]) j=f[j-1];
    if(b[i]==b[j])f[i]=++j;
}
j=0;
for(i=0 ; i<n ; i++){
    while(j>0 && a[i]!=b[j])j=f[j-1];
    if(a[i]==b[j]){
        j++;
        if(j==m){
            j=f[m-1];
            ans.push_back(i+2-m);
        }
    }
}
```

6.3 Manacher's Algorithm

```
for(i=1 ; i<=n ; i++)c[i*2]=a[i]; /// a is given string
n=2*n+1;
for(i=1 ; i<=n ; i++)if(i%2)c[i]='.';
for(i=1 ; i<=n ; i++){
    if(k+b[k]>=i)b[i]=min(b[2*k-i],k+b[k]-i);
    while(i-b[i]-1>=1 && i+b[i]+1<=n
    &&c[i-b[i]-1]==c[i+b[i]+1])b[i]++;
    ans=max(ans,b[i]);
    if(i+b[i]>k+b[k])k=i;
}
```

6.4 Suffix-Array and LCP

```
int main(){
    int i,j,k;
    cin>>n>>a+1;
    int m=max(n,26);
    for(i=1 ; i<=n ; i++)ord[i]=a[i]-'a'+1,cnt[ord[i]]++;
    for(i=1 ; i<=m ; i++)cnt[i]+=cnt[i-1];
    for(i=1 ; i<=n ; i++)sa[cnt[ord[i]]--]=i;
    for(k=1 ; k<n ; k*=2){
        for(i=0 ; i<=m ; i++)cnt[i]=0;
        for(i=1 ; i<=n ; i++)cnt[ord[i+k]]++;
        for(i=1 ; i<=m ; i++)cnt[i]+=cnt[i-1];
        for(i=1 ; i<=n ; i++)b[cnt[ord[i+k]]--]=i;
        for(i=0 ; i<=m ; i++)cnt[i]=0;
        for(i=1 ; i<=n ; i++)cnt[ord[i]]++;
        for(i=1 ; i<=m ; i++)cnt[i]+=cnt[i-1];
        for(i=n ; i>=1 ; i--)sa[cnt[ord[b[i]]]--]=b[i];
        b[sa[1]]=j=1;
        for(i=2 ; i<=n ; i++)
            if(ord[sa[i]]!=ord[sa[i-1]] ||
            ord[sa[i]+k]!=ord[sa[i-1]+k])b[sa[i]]=++j;
            else b[sa[i]]=b[sa[i-1]];
        for(i=1 ; i<=n  ;i++)ord[i]=b[i];
    }
    k=0;
    for (i=1;i<=n;i++) r[sa[i]] = i;
    for (i=1;i<=n;lcp[r[i++]]=k)
    for (k?k--:0,j=sa[r[i]-1];a[i+k]==a[j+k];k++);
    int ans=0;
    for(i=2 ; i<=n ; i++)ans=max(ans,lcp[i]);
    cout<<ans;
    return 0;
}
```

7 Formula

7.1 세 점을 지나는 평면

$$\begin{vmatrix} y_2-y_1 & z_2-z_1 \\ y_3-y_1 & z_3-z_1 \end{vmatrix}(x-x_1) - \begin{vmatrix} x_2-x_1 & z_2-z_1 \\ x_3-x_1 & z_3-z_1 \end{vmatrix}(y-y_1) + \begin{vmatrix} x_2-x_1 & y_2-y_1 \\ x_3-x_1 & y_3-y_1 \end{vmatrix}(z-z_1) = 0$$

7.2 카탈란 수

$$C_0 = 1 \qquad C_n = \sum_{i=0}^{n-1} C_i C_{n-1-i} \qquad \text{for } n \geq 1$$
$$C_n = \frac{1}{n+1}\binom{2n}{n} = \frac{2n!}{n!(n+1)!}$$

7.3 외심

$$x = -\frac{({x_2}^2 - {x_1}^2 + {y_2}^2 - {y_1}^2)(y_3 - y_2) - ({x_2}^2 - {x_3}^2 + {y_2}^2 - {y_3}^2)(y_1 - y_2)}{2(x_1 - x_2)(y_3 - y_2) - 2(x_3 - x_2)(y_1 - y_2)}$$
$$y = -\frac{({y_2}^2 - {y_1}^2 + {x_2}^2 - {x_1}^2)(x_3 - x_2) - ({y_2}^2 - {y_3}^2 + {x_2}^2 - {x_3}^2)(x_1 - x_2)}{2(x_3 - x_2)(y_1 - y_2) - 2(x_1 - x_2)(y_3 - y_2)}$$

7.4 Knuth's and D&C Optimizaiton

7.4.1 Knuth's

조건 1. DP 점화식 꼴

$$D[i][j] = min_{i<k<j}(D[i][k] + D[k][j]) + C[i][j]$$

조건 2. Quadrangle Inequalty (사각부등식)

$$C[a][c] + C[b][d] \leq C[a][d] + C[b][c], a \leq b \leq c \leq d$$

조건 3. Monotonicity (단조성)

$$C[b][c] \leq C[a][d], a \leq b \leq c \leq d$$

조건 2와 조건 3을 만족하면 $A[i][j] = D[i][j]$가 최소가 되기 위한 가장 작은 k라고 했을 때 아래 식을 만족한다.

$$A[i][j-1] \leq A[i][j] \leq A[i+1][j]$$

7.4.2 Divide&Conquer

조건 1. DP 점화식 꼴

$$D[t][i] = min_{k<i}(D[t-1][k] + C[k][i])$$

조건 2. $A[t][i]$는 $D[t][i]$를 만족시키는 최소 k라 할 때 아래 부등식을 만족

$$A[t][i] \leq A[t][i+1]$$

사각부등식을 만족하면 조건 2도 만족한다.

7.5 Grundy Number

어떤 상황 S에 대한 Grundy Number를 구하는 방법은 아래와 같다. 상황 S에서 다음 상황들 중 하나를 S′라고 하자.

$$G(S) = min(v) \qquad (v \notin G(S'))$$

즉, 상황 S의 Grundy Number G(S)는 상황 S의 다음 상황들 S′의 Grundy Number G(S′)들 중 존재하지 않는 가장 작은 수다.

각 상황들을 Grundy Number로 표현했을 때, 필승법의 여부는 Nim Game과 마찬가지로 구하면 된다. Grundy Number로 표현된 각 상황들을 XOR했을 때 결과가 0이면 무조건 지는 것이고, 0이 아니면 무조건 이길 수 있다.

7.6 중국인의 나머지 정리

7.6.1 정리

서로소인 k개의 자연수 n_1,n_2,...,n_k와 임의의 정수 a_1,a_2,...,a_k가 있을 때, 임의의 $i(1 \leq i \leq k)$에 대해 이 방정식이 성립하는 해 $x = a$가 항상 존재하며, $n_1, n_2...n_k$ 모듈로 안에서 유일하다. 즉, 이 방정식의 해는 $x \equiv a(mod\ n_1 n_2...n_k)$로 표현 가능하다.

7.6.2 증명과 계산 방법

$N = n_1 n_2...n_k$ 라고 하자. N/n_i는 n_i와 서로소이기 때문에 $r_i n_i + s_i(N/n_i) = 1$인 정수 r_i, s_i가 항상 존재한다. (확장 유클리드) 여기에서, $e_i = s_i(N/n_i)$라고 하면 아래와 같은 등식이 성립한다.

$$e_i \equiv 1(mod\ n_i)$$
$$e_i \equiv 0(mod\ n_j)$$

여기에서 $a = \sum_{i=1}^{k} a_i e_i$라고 하면, 임의의 i에 대해 $a \equiv a_i (mod\ n_i)$가 성립한다. 즉, a는 우리가 구하고자 하는 해 중 하나이다.

7.7 Lucas' Theorem

음이 아닌 정수 n,k,소수 p에 대해서 다음과 같이 n과 k를 p진법 전개식으로 나타냈을 때

$$n = n_m p^m + n_{m-1} p^{m-1} + ...n_1 p + n_0$$
$$k = k_m p^m + k_{m-1} p^{m-1} + ...k_1 p + k_0$$

다음 합동식이 성립한다

$$\binom{n}{k} = \prod_{j=0}^{k} \binom{n_j}{k_j} (mod\ p)$$

7.8 Hall's Theorem

어떤 bipartite graph $G = (L \cup R, E)$가 주어졌다고 하자.
어떤 부분집합 $S \subseteq L$에 대해서, S에 인접한 정점들의 집합을 $N(S) \subseteq R$라고 할 때, L의 모든 정점이 참여하는 mathcing이 존재하는 필요충분조건은 모든 L의 부분집합 S가 $|S| \leq |N(S)|$를 만족하는 것이다.

7.9 Primes for NTT

$p = 3221225473,$	$a = 3,$	$b = 30,$	$primitive = 5$
$p = 2281701377,$	$a = 17,$	$b = 27,$	$primitive = 3$
$p = 469762049,$	$a = 7,$	$b = 26,$	$primitive = 3$
$p = 2013265921,$	$a = 15,$	$b = 27,$	$primitive = 31$
$p = 998244353,$	$a = 119,$	$b = 23,$	$primitive = 3$