1~n线性逆元

niyuan[1]=1;  
for(int i=2;i<=n;i++)  
 niyuan[i]=(p-(p/i))\*niyuan[p%i]%p;

ac自动机

#include<bits/stdc++.h>  
#define ll long long  
using namespace std;  
ll n;  
string s;  
const ll mxle=8e6+1;  
ll cnt,vis[mxle],ans,rev[mxle],rud[mxle];  
struct trie\_node{  
 ll son[27];  
 ll fail;  
 ll flag;  
 ll ans;  
 void init(){  
 memset(son,0,sizeof son);  
 fail=flag=0;  
 }  
}trie[mxle];  
  
void init(){  
 for(ll i=0;i<=cnt;i++)trie[i].init();  
 for(ll i=1;i<=n;i++)vis[i]=0;  
 cnt=1;  
 ans=0;  
}  
  
void insert(string &S,ll num){//trie的插入  
 ll u=1,len=S.size();  
 for(ll i=0;i<len;i++){//trie从1开始,1是空串  
 ll v=S[i]-'a';  
 if(!trie[u].son[v])trie[u].son[v]=++cnt;//cnt是trie节点数量  
 u=trie[u].son[v];  
 }  
 if(!trie[u].flag)trie[u].flag=num;//结束标记  
 rev[num]=trie[u].flag;//反向索引，第num个s的结束节点  
 return;  
}  
queue<ll> q;  
void getfail(){  
 for(ll i=0;i<26;i++)trie[0].son[i]=1;  
 q.emplace(1);  
 trie[1].fail=0;//1（空）的fail是0  
 while(!q.empty()){  
 ll u=q.front();  
 q.pop();  
 ll F=trie[u].fail;  
 for(ll i=0;i<26;i++){  
 ll v=trie[u].son[i];  
 if(!v){//u没有i的子节点  
 trie[u].son[i]=trie[F].son[i];//u的i子节点变成u失配的i  
 continue;  
 }//失配节点保证了返回到当前节点的最长后缀的相等前缀处  
 trie[v].fail=trie[F].son[i];//u的i子节点的失配变成u失配的i  
 rud[trie[F].son[i]]++;//F的i子节点入度在u有i子节点时增加  
 q.emplace(v);  
 }  
 }  
}  
  
void query(string &S){  
 ll u=1,len=S.size();  
 for(ll i=0;i<len;i++){  
 u=trie[u].son[S[i]-'a'];  
 trie[u].ans++;  
 }  
}  
  
void topu(){  
 for(ll i=1;i<=cnt;i++)  
 if(!rud[i])q.emplace(i);  
 while(!q.empty()){  
 ll f=q.front();  
 q.pop();  
 vis[trie[f].flag]=trie[f].ans;  
 ll u=trie[f].fail;  
 trie[u].ans+=trie[f].ans;  
 if(!(--rud[u]))q.emplace(u);  
 }  
}  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 cin>>n;  
 init();  
 for(ll i=1;i<=n;i++){//trie的构建  
 cin>>s;  
 insert(s,i);  
 }  
 getfail();  
 cin>>s;  
 query(s);  
 topu();//拓扑排序优化  
 for(ll i=1;i<=n;i++)  
 cout<<vis[rev[i]]<<'\n';  
 return 0;  
}

Dfs序欧拉序

#include<bits/stdc++.h>  
#define ll long long  
#define endl "\n"  
using namespace std;  
const ll N=3e5+5;  
vector<ll> g[N],tin(N),tout(N),id(N),par(N);  
ll T=0;  
void dfs(ll u,ll p){  
 id[u]=tin[u]=tout[u]=T++;  
 for(auto i:g[u]){  
 if(i!=p){  
 dfs(i,u);  
 par[i]=u;  
 tout[u]=tout[i];  
 }  
 }  
}  
vector<ll> p(N);  
ll ok(ll i){  
 if(p[i]==1){  
 if(i==1)  
 return 1;  
 return 0;  
 }  
 ll ant=p[i-1];  
 if(par[p[i]]==ant) return 1;//前一个是父节点  
 if(tin[ant]!=tout[ant])return 0;  
 ll pa=par[p[i]];  
 if(tin[ant]<tin[pa]||tin[ant]>tout[pa])return 0;  
 return 1;//是兄弟节点的叶子节点  
}  
void solve(){  
 ll n,q;  
 cin>>n>>q;  
 for(ll i=0;i<=n;i++)  
 g[i].clear();  
 T=0;  
 for(ll i=2;i<=n;i++){  
 ll fa;  
 cin>>fa;  
 g[fa].emplace\_back(i);  
 }  
 for(ll i=1;i<=n;i++){  
 cin>>p[i];  
 }  
 dfs(1,0);  
 ll cnt=0;  
 for(ll i=1;i<=n;i++){  
 cnt+=ok(i);  
 }  
 while(q--){  
 ll x,y;  
 cin>>x>>y;  
 set<ll> in;  
 in.insert(x),in.insert(y);//会对前后自己三个节点产生影响。  
 if(x-1>=1) in.insert(x-1);  
 if(x+1<=n) in.insert(x+1);  
 if(y-1>=1) in.insert(y-1);  
 if(y+1<=n) in.insert(y+1);  
 for(auto i:in)  
 cnt-=ok(i);  
 swap(p[x],p[y]);  
 for(auto i:in)  
 cnt+=ok(i);  
 cout<<(cnt==n?"YES":"NO")<<endl;  
 }  
}  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 ll t;  
 cin>>t;  
 while(t--){  
 solve();  
 }  
 return 0;  
}

Dijkstra

#include<bits/stdc++.h>//最短路加分层图  
#define ll long long  
#define maxn 2005  
using namespace std;  
struct cmp {  
public:  
 bool operator()(pair<ll,ll> a,pair<ll,ll> b){//f是dis s是节点  
 return a.first>b.first;  
 }  
};  
vector<pair<ll,ll>> e[maxn];//f是目的地，s是权重  
vector<ll> dis(maxn),vis(maxn);  
priority\_queue<pair<ll,ll>, vector<pair<ll,ll>>, cmp> q;  
void dijkstra(ll n, ll s) {//s出发，n目的  
 dis.assign(maxn,10000000000000005);  
 dis[s] = 0;  
 q.emplace(0, s);  
 while (!q.empty()) {  
 ll u = q.top().second;  
 q.pop();  
 if (vis[u]) continue;  
 vis[u] = 1;  
 for (auto ed : e[u]) {  
 ll v = ed.first, w = ed.second;  
 if (dis[v] > dis[u] + w) {  
 dis[v] = dis[u] + w;  
 q.emplace(dis[v], v);  
 }  
 }  
 }  
}  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 ll n,a,b,c;  
 cin>>n>>a>>b>>c;  
 for(ll i=1;i<=n;i++){  
 for(ll j=1;j<=n;j++){  
 ll d;  
 cin>>d;  
 e[i].emplace\_back(j,d);  
 }  
 }  
 for(ll i=1;i<=n;i++){  
 for(ll j=1;j<=n;j++){  
 e[i+n].emplace\_back(j+n,e[i][j-1].second\*b+c);  
 }  
 }  
 for(ll i=1;i<=n;i++){  
 for(ll j=0;j<n;j++)  
 e[i][j].second\*=a;  
 e[i].emplace\_back(i+n,0);  
 }  
 dijkstra(2\*n,1);  
 cout<<dis[2\*n];  
 return 0;  
}

Exgcd

ll gcd1;  
void exgcd(long long a,long long b,long long & x,long long & y){  
 if(!b){  
 gcd1=a;  
 x=1,y=0;  
 }  
 else{  
 exgcd(b,a%b,y,x);  
 y-=a/b\*x;  
 }  
}

exkmp

//s[i]开头的后缀和s的最长公共前缀。

#include<bits/stdc++.h>  
#define ll long long  
using namespace std;  
vector<ll> get\_z(string s) {  
 ll len=s.size();  
 vector<ll> z(len);  
 ll l=0,r=0;  
 for(ll i=1;i<len;i++) {  
 if(i<=r&&z[i-l]<r-i+1)  
 z[i]=z[i-l];  
 else {  
 z[i]=max((ll)0,r-i+1);  
 while(i+z[i]<len&&s[z[i]]==s[i+z[i]]) ++z[i];  
 }  
 if (i + z[i] - 1 > r) l = i, r = i + z[i] - 1;  
 }  
 return z;  
}  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 string s;  
 cin>>s;  
 vector<ll> z;  
 z=get\_z(s);  
 ll len=s.size();  
 ll ans=0,mx=0;  
 for(ll i=0;i<len;i++) {  
 if(z[i]<=mx&&z[i]+i==len&&ans<z[i])  
 ans=z[i];  
 mx=max(mx,z[i]);  
 }  
 string a=s.substr(0,ans);  
 cout<<a;  
 if(!ans)  
 cout<<"Just a legend";  
 return 0;  
}

Hash

using std::string;  
  
const int M = 1e9 + 7;  
const int B = 233;  
  
typedef long long ll;  
  
int get\_hash(const string& s) {  
 int res = 0;  
 for (int i = 0; i < s.size(); ++i) {  
 res = ((ll)res \* B + s[i]) % M;  
 }  
 return res;  
}  
  
bool cmp(const string& s, const string& t) {  
 return get\_hash(s) == get\_hash(t);  
}

kmp

vector<int> prefix\_function(string s) {  
 int n = (int)s.length();  
 vector<int> pi(n);  
 for (int i = 1; i < n; i++) {  
 int j = pi[i - 1];  
 while (j > 0 && s[i] != s[j]) j = pi[j - 1];  
 if (s[i] == s[j]) j++;  
 pi[i] = j;  
 }  
 return pi;  
}

Kruskal

const int MAXN = 5000 + 5;  
const int MAXM = 200000 + 5;  
const int INF = 0x3fffffff;  
struct edge {  
 int u;  
 int v;  
 int w;  
} e[MAXM];  
int f[MAXN],cnt,m,n,ans;  
bool cmp(edge x,edge y){  
 return x.w<y.w;  
}  
int find(int x){  
 if(f[x]==x) {  
 return x;  
 } else {  
 f[x]=find(f[x]);//路径压缩  
 return f[x];  
 }  
}  
  
void Kruskal()  
{  
 sort(e+1,e+m+1,cmp);  
 for(int i=1; i<=m; i++) {  
 int u = find(e[i].u);  
 int v = find(e[i].v);  
 if(u==v)continue;//判断两个点是否在同一颗树,同一棵树则成环跳过  
 ans+=e[i].w;//权重  
 f[v]=u;//v点的父亲为u，意思为(u,v)这条边加入  
 cnt++;  
 if(cnt==n-1)break;//所有的点构成构成一棵树  
 }  
}

lca倍增

const int MAXN = 5000 + 5;  
const int MAXM = 200000 + 5;  
const int INF = 0x3fffffff;  
struct edge {  
 int u;  
 int v;  
 int w;  
} e[MAXM];  
int f[MAXN],cnt,m,n,ans;  
bool cmp(edge x,edge y){  
 return x.w<y.w;  
}  
int find(int x){  
 if(f[x]==x) {  
 return x;  
 } else {  
 f[x]=find(f[x]);//路径压缩  
 return f[x];  
 }  
}  
  
void Kruskal()  
{  
 sort(e+1,e+m+1,cmp);  
 for(int i=1; i<=m; i++) {  
 int u = find(e[i].u);  
 int v = find(e[i].v);  
 if(u==v)continue;//判断两个点是否在同一颗树,同一棵树则成环跳过  
 ans+=e[i].w;//权重  
 f[v]=u;//v点的父亲为u，意思为(u,v)这条边加入  
 cnt++;  
 if(cnt==n-1)break;//所有的点构成构成一棵树  
 }  
}

manacher

const ll N=50005;  
ll n;  
vector<ll> d1(N),d2(N);  
void manacher(string s){  
 for (int i = 0, l = 0, r = -1; i < n; i++) {  
 int k = (i > r) ? 1 : min(d1[l + r - i], (ll)r - i + 1);  
 while (0 <= i - k && i + k < n && s[i - k] == s[i + k]) {  
 k++;  
 }  
 d1[i] = k--;  
 if (i + k > r) {  
 l = i - k;  
 r = i + k;  
 }  
 }  
 for (int i = 0, l = 0, r = -1; i < n; i++) {  
 int k = (i > r) ? 0 : min(d2[l + r - i + 1],(ll)r - i + 1);  
 while (0 <= i - k - 1 && i + k < n && s[i - k - 1] == s[i + k]) {  
 k++;  
 }  
 d2[i] = k--;  
 if (i + k > r) {  
 l = i - k - 1;  
 r = i + k;  
 }  
 }  
}

st表

#include<bits/stdc++.h>  
#define ll long long  
using namespace std;  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 ll n,m;  
 cin>>n>>m;  
 vector<ll> Log(n+1);  
 Log[1]=0;  
 for(ll i=2;i<=n;i++)  
 Log[i]=Log[i/2]+1;  
 vector<vector<ll>> f(n+1,vector<ll>(Log[n]+1));  
 for(ll i=1;i<=n;i++)  
 cin>>f[i][0];  
 for(ll i=1;i<=Log[n];i++){  
 for(ll j=1;j+(1<<i)-1<=n;j++){  
 f[j][i]=max(f[j][i-1],f[j+(1<<(i-1))][i-1]);  
 }  
 }  
 for(ll i=1;i<=m;i++){  
 ll l,r;  
 cin>>l>>r;  
 ll s=Log[r-l+1];  
 cout<<max(f[l][s],f[r-(1<<s)+1][s])<<'\n';  
 }  
 return 0;  
}

分块

#include<bits/stdc++.h>  
#define ll long long  
using namespace std;  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 ll n,m;  
 cin>>n>>m;  
 vector<ll> Log(n+1);  
 Log[1]=0;  
 for(ll i=2;i<=n;i++)  
 Log[i]=Log[i/2]+1;  
 vector<vector<ll>> f(n+1,vector<ll>(Log[n]+1));  
 for(ll i=1;i<=n;i++)  
 cin>>f[i][0];  
 for(ll i=1;i<=Log[n];i++){  
 for(ll j=1;j+(1<<i)-1<=n;j++){  
 f[j][i]=max(f[j][i-1],f[j+(1<<(i-1))][i-1]);  
 }  
 }  
 for(ll i=1;i<=m;i++){  
 ll l,r;  
 cin>>l>>r;  
 ll s=Log[r-l+1];  
 cout<<max(f[l][s],f[r-(1<<s)+1][s])<<'\n';  
 }  
 return 0;  
}

哈夫曼树

#include<bits/stdc++.h>  
#define ll long long  
#define endl "\n"  
using namespace std;  
struct node{  
 ll w,h;  
 node(){w=0,h=0;}  
 node(ll w,ll h):w(w),h(h){}  
 bool operator <(const node &a)const{return a.w==w?h>a.h:w>a.w;}  
};  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 ll n,k;  
 cin>>n>>k;  
 vector<ll> a(n+1);  
 priority\_queue<node> q;  
 for(ll i=1;i<=n;i++){  
 cin>>a[i];  
 q.emplace(node(a[i],1));  
 }  
 while((q.size()-1)%(k-1)!=0)q.emplace(node(0,1));  
 ll ans=0;  
 while (q.size()>=k){  
 ll h=-1,w=0;  
 for(ll i=1;i<=k;i++){  
 node t=q.top();  
 q.pop();  
 h=max(h,t.h);  
 w+=t.w;  
 }  
 ans+=w;  
 q.emplace(node(w,h+1));  
 }  
 cout<<ans<<endl;  
 cout<<q.top().h-1<<endl;  
 return 0;  
}

快读、int128

#include<bits/stdc++.h>  
#define ll \_\_int128  
using namespace std;  
inline ll read() {  
 int x = 0, w = 1;  
 char ch = 0;  
 while (ch < '0' || ch > '9') { // ch 不是数字时  
 if (ch == '-') w = -1; // 判断是否为负  
 ch = getchar(); // 继续读入  
 }  
 while (ch >= '0' && ch <= '9') { // ch 是数字时  
 x = x \* 10 + (ch - '0'); // 将新读入的数字「加」在 x 的后面  
 ch = getchar(); // 继续读入  
 }  
 return x \* w; // 数字 \* 正负号 = 实际数值  
}  
void write(ll x) {  
 static ll sta[35];  
 int top = 0;  
 do {  
 sta[top++] = x % 10, x /= 10;  
 } while (x);  
 while (top) putchar(sta[--top] + 48); // 48 是 '0'  
}

拓扑排序

#include<bits/stdc++.h>  
#define ll long long  
using namespace std;  
int n;  
vector<int> G[105];//图  
int in[105]; // 存储每个结点的入度  
bool toposort() {  
 vector<int> L;  
 queue<int> S;  
 for (int i = 1; i <= n; i++)  
 if (in[i] == 0) S.push(i);  
 while (!S.empty()) {//bfs  
 int u = S.front();  
 S.pop();  
 L.push\_back(u);  
 for (auto v : G[u]) {  
 if (--in[v] == 0) {  
 S.push(v);  
 }  
 }  
 }  
 if (L.size() == n) {  
 for (auto i : L) cout << i << ' ';  
 return true;  
 } else {  
 return false;//有环  
 }  
}  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 cin>>n;  
 for(ll i=1;i<=n;i++){  
 while(true){  
 ll x;  
 cin>>x;  
 if(x==0)  
 break;  
 G[i].emplace\_back(x);  
 in[x]++;  
 }  
 }  
 toposort();  
 return 0;  
}

Prim

#include<bits/stdc++.h>  
#define ll long long  
using namespace std;  
const ll N = 5050, M = 2e5 + 10;//节点数和边数  
  
struct E {//边的信息  
 ll v, w, x;//v节点w权重x下一条边  
} e[M \* 2];//存边  
  
ll n, m, h[N], cnte;//cnte第几条边，h节点的边的头指针  
  
void adde(ll u, ll v, ll w) { e[++cnte] = E{v, w, h[u]}, h[u] = cnte; }//向图中添加边，更新头指针  
//S存节点和距离  
  
class cmp{  
public:  
 bool operator()(pair<ll,ll> a,pair<ll,ll> b){  
 return a.second>b.second;  
 }  
};//小顶堆  
  
priority\_queue<pair<ll,ll>,vector<pair<ll,ll>>,cmp> q;//1是点，2是dis  
ll dis[N];//存从源点到节点i的最短距离，初始化源点0,其余无穷大  
bool vis[N];//标记顶点i是否被访问，被加入设为true  
  
ll res = 0, cnt = 0;//res总权重，cnt顶点数  
  
void Prim() {  
 memset(dis, 0x3f, sizeof(dis));  
 dis[1] = 0;  
 q.emplace(1,0);  
 while (!q.empty()) {  
 if (cnt >= n) break;  
 ll u = q.top().first, d = q.top().second;  
 q.pop();  
 if (vis[u]) continue;  
 vis[u] = true;  
 ++cnt;  
 res += d;  
 for (ll i = h[u]; i; i = e[i].x) {  
 ll v = e[i].v, w = e[i].w;  
 if (w < dis[v]) {  
 dis[v] = w, q.emplace(v, w);  
 }  
 }  
 }  
}  
  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 cin>>n>>m;  
 for(ll i=1,u,v,w;i<=m;i++){  
 cin>>u>>v>>w;  
 adde(u,v,w);  
 adde(v,u,w);  
 }  
 Prim();  
 if(cnt==n)  
 cout<<res;  
 else  
 cout<<"orz";  
 return 0;  
}

树堆（弱平衡树）

#include<bits/stdc++.h>  
#define ll long long  
using namespace std;  
const ll N = 5050, M = 2e5 + 10;//节点数和边数  
  
struct E {//边的信息  
 ll v, w, x;//v节点w权重x下一条边  
} e[M \* 2];//存边  
  
ll n, m, h[N], cnte;//cnte第几条边，h节点的边的头指针  
  
void adde(ll u, ll v, ll w) { e[++cnte] = E{v, w, h[u]}, h[u] = cnte; }//向图中添加边，更新头指针  
//S存节点和距离  
  
class cmp{  
public:  
 bool operator()(pair<ll,ll> a,pair<ll,ll> b){  
 return a.second>b.second;  
 }  
};//小顶堆  
  
priority\_queue<pair<ll,ll>,vector<pair<ll,ll>>,cmp> q;//1是点，2是dis  
ll dis[N];//存从源点到节点i的最短距离，初始化源点0,其余无穷大  
bool vis[N];//标记顶点i是否被访问，被加入设为true  
  
ll res = 0, cnt = 0;//res总权重，cnt顶点数  
  
void Prim() {  
 memset(dis, 0x3f, sizeof(dis));  
 dis[1] = 0;  
 q.emplace(1,0);  
 while (!q.empty()) {  
 if (cnt >= n) break;  
 ll u = q.top().first, d = q.top().second;  
 q.pop();  
 if (vis[u]) continue;  
 vis[u] = true;  
 ++cnt;  
 res += d;  
 for (ll i = h[u]; i; i = e[i].x) {  
 ll v = e[i].v, w = e[i].w;  
 if (w < dis[v]) {  
 dis[v] = w, q.emplace(v, w);  
 }  
 }  
 }  
}  
  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 cin>>n>>m;  
 for(ll i=1,u,v,w;i<=m;i++){  
 cin>>u>>v>>w;  
 adde(u,v,w);  
 adde(v,u,w);  
 }  
 Prim();  
 if(cnt==n)  
 cout<<res;  
 else  
 cout<<"orz";  
 return 0;  
}

树状背包dp

#include<bits/stdc++.h>  
#define ll long long  
using namespace std;  
vector<vector<ll>> son(320);  
vector<ll> score(320);  
ll n,m;  
ll dp[305][305];  
ll dfs(ll root) {  
 dp[root][1]=score[root];  
 ll p=1;  
 for(auto i:son[root]) {  
 ll siz=dfs(i);  
 for(ll j=min(p,m+1);j>=1;j--) {  
 for(ll k=0;k<=siz&&k+j<=m+1;k++)  
 dp[root][j+k]=max(dp[root][j+k],dp[root][j]+dp[i][k]);  
 }  
 p+=siz;  
 }  
 return p;  
}  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 cin>>n>>m;  
 for(ll i=0;i<n;i++) {  
 ll x;  
 cin>>x>>score[i+1];  
 son[x].emplace\_back(i+1);  
 }  
 dfs(0);  
 cout<<dp[0][m+1];  
 return 0;  
}

树状数组

#include<bits/stdc++.h>//用差分可以做区间和  
#define ll long long  
using namespace std;  
ll n,w;  
ll c[1000006];  
ll lowbit(ll x) {//最后1和后面的0组成的数  
 return x & -x;  
}  
ll getsum(ll x) { // a[1]..a[x]的和  
 ll ans = 0;  
 while (x > 0) {  
 ans = ans + c[x];  
 x = x - lowbit(x);  
 }  
 return ans;  
}  
void add(ll x, ll k) {  
 while (x <= n) { // 不能越界  
 c[x] = c[x] + k;  
 x = x + lowbit(x);  
 }  
}  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 cin>>n>>w;  
 while(w--){  
 char op;  
 ll a,b;  
 cin>>op>>a>>b;  
 if(op=='x')  
 add(a,b);  
 else  
 cout<<getsum(b)-getsum(a-1)<<'\n';  
 }  
 return 0;  
}

桥

#include<bits/stdc++.h>  
#define ll long long  
#define endl "\n"  
using namespace std;  
const ll N=1e5+5;  
ll n,m,dfncnt;  
vector<vector<ll>> e;  
vector<ll> low,dfn,f;  
ll ans;  
void init(ll x){  
 ans=x\*(x-1)/2;  
 low.assign(x+1,0),dfn.assign(x+1,0),f.assign(x+1,0);  
 e.assign(x+1,vector<ll>(0));  
 dfncnt=0;  
}  
void tarjan(ll u,ll fa){  
 f[u]=fa;  
 low[u]=dfn[u]=++dfncnt;  
 for(auto i:e[u]){  
 if(i==fa)continue;  
 if(dfn[i]!=0){  
 low[u]=min(low[u],dfn[i]);  
 continue;  
 }  
 tarjan(i,u);  
 ll num=dfncnt-dfn[i]+1;  
 low[u]=min(low[u],low[i]);  
 if(low[i]>dfn[u])  
 ans=min(ans,num\*(num-1)/2+(n-num)\*(n-num-1)/2);  
 }  
}  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 ll t;  
 cin>>t;  
 while(t--){  
 cin>>n>>m;  
 init(n);  
 for(ll i=0;i<m;i++){  
 ll u,v;  
 cin>>u>>v;  
 e[u].emplace\_back(v),e[v].emplace\_back(u);  
 }  
 tarjan(1,0);  
 cout<<ans<<endl;  
 }  
 return 0;  
}

欧拉函数

#include<bits/stdc++.h>  
#define ll long long  
#define endl "\n"  
using namespace std;  
//求特定数欧拉  
ll euler\_phi(ll n) {  
 ll ans = n;  
 for (ll i = 2; i \* i <= n; i++)  
 if (n % i == 0) {  
 ans = ans / i \* (i - 1);  
 while (n % i == 0) n /= i;  
 }  
 if (n > 1) ans = ans / n \* (n - 1);  
 return ans;  
}  
  
//线性求欧拉  
const ll N=40005;  
vector<ll> pri;  
bool not\_prime[N];  
ll phi[N];  
void pre(ll n) {  
 phi[1] = 1;  
 for (ll i = 2; i <= n; i++) {  
 if (!not\_prime[i]) {  
 pri.emplace\_back(i);  
 phi[i] = i - 1;  
 }  
 for (ll pri\_j : pri) {  
 if (i \* pri\_j > n) break;  
 not\_prime[i \* pri\_j] = true;  
 if (i % pri\_j == 0) {  
 phi[i \* pri\_j] = phi[i] \* pri\_j;  
 break;  
 }  
 phi[i \* pri\_j] = phi[i] \* phi[pri\_j];  
 }  
 }  
}  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 ll n;  
 cin>>n;  
 pre(n);  
 ll ans=3;  
 for(ll i=2;i<=n-1;i++)  
 ans+=phi[i]\*2;  
 if(n==1)  
 cout<<0;  
 else  
 cout<<ans;  
 return 0;  
}

矩阵快速幂

#include<bits/stdc++.h>  
#define ll long long  
#define endl "\n"  
using namespace std;  
const ll N=105,mod=1e9+7;  
ll n,K;  
vector<vector<ll>> mul(vector<vector<ll>> &a,vector<vector<ll>> &b){  
 vector<vector<ll>> c(N,vector<ll>(N,0));  
 for(ll i=1;i<=n;i++){  
 for(ll k=1;k<=n;k++){  
 if(a[i][k]==0)continue;  
 for(ll j=1;j<=n;j++){  
 c[i][j]+=a[i][k]\*b[k][j];  
 c[i][j]%=mod;  
 }  
 }  
 }  
 return c;  
}  
vector<vector<ll>> kpow(vector<vector<ll>> &a,ll y){  
 vector<vector<ll>> ans,x=a;  
 while((y&1)!=1){  
 x=mul(x,x);  
 y>>=1;  
 }  
 ans=x;  
 x=mul(x,x);  
 y>>=1;  
 while(y){  
 if(y&1)ans=mul(ans,x);  
 x=mul(x,x);  
 y>>=1;  
 }  
 return ans;  
}  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 cin>>n>>K;  
 if(K==0){  
 for(ll i=1;i<=n;i++){  
 for(ll j=1;j<=n;j++){  
 if(i==j)cout<<1<<' ';  
 else  
 cout<<0<<' ';  
 }  
 cout<<endl;  
 }  
 }  
 else{  
 vector<vector<ll>> mat(N,vector<ll>(N));  
 for(ll i=1;i<=n;i++){  
 for(ll j=1;j<=n;j++){  
 cin>>mat[i][j];  
 }  
 }  
 vector<vector<ll>> ans(N,vector<ll>(N));  
 ans= kpow(mat,K);  
 for(ll i=1;i<=n;i++){  
 for(ll j=1;j<=n;j++)  
 cout<<ans[i][j]<<' ';  
 cout<<endl;  
 }  
 }  
 return 0;  
}

线段树

#include<bits/stdc++.h>  
#define ll long long  
#define endl "\n"  
using namespace std;  
template <typename T>  
class Segtree{  
 vector<T> tree,lazy1,lazy2;  
 vector<T> \*arr;  
 ll n,root,n4,end;  
 void maintain(ll cl,ll cr,ll p){  
 ll cm=cl+((cr-cl)>>1);  
 if(cl!=cr){  
 lazy2[p\*2]\*=lazy2[p];  
 lazy2[p\*2+1]\*=lazy2[p];  
 lazy1[p\*2]\*=lazy2[p];  
 lazy1[p\*2+1]\*=lazy2[p];  
 tree[p\*2]\*=lazy2[p];  
 tree[p\*2+1]\*=lazy2[p];  
 lazy2[p]=1;  
 lazy1[p\*2]+=lazy1[p];  
 lazy1[2\*p+1]+=lazy1[p];  
 tree[2\*p]+=lazy1[p]\*(cm-cl+1);  
 tree[2\*p+1]+=lazy1[p]\*(cr-cm);  
 lazy1[p]=0;  
 }  
 }  
 T range\_sum(ll l,ll r,ll cl,ll cr,ll p){  
 if(l<=cl&&cr<=r)return tree[p];  
 ll m=cl+((cr-cl)>>1);  
 T sum=0;  
 maintain(cl,cr,p);  
 if(l<=m) sum+= range\_sum(l,r,cl,m,p\*2);  
 if(r>m) sum+= range\_sum(l,r,m+1,cr,p\*2+1);  
 return sum;  
 }  
 void range\_mul(ll l,ll r,T val,ll cl,ll cr,ll p){  
 if(l<=cl&&cr<=r){  
 lazy2[p]\*=val;  
 lazy1[p]\*=val;  
 tree[p]\*=val;  
 return;  
 }  
 ll m=cl+((cr-cl)>>1);  
 maintain(cl,cr,p);  
 if(l<=m)range\_mul(l,r,val,cl,m,p\*2);  
 if(r>m)range\_mul(l,r,val,m+1,cr,p\*2+1);  
 tree[p]=tree[2\*p]+tree[2\*p+1];  
 }  
 void range\_add(ll l,ll r,T val,ll cl,ll cr,ll p){  
 if(l<=cl&&cr<=r){  
 lazy1[p]+=val;  
 tree[p]+=(cr-cl+1)\*val;  
 return;  
 }  
 ll m=cl+((cr-cl)>>1);  
 maintain(cl,cr,p);  
 if(l<=m)range\_add(l,r,val,cl,m,p\*2);  
 if(r>m)range\_add(l,r,val,m+1,cr,p\*2+1);  
 tree[p]=tree[2\*p]+tree[2\*p+1];  
 }  
 void build(ll s,ll t,ll p){  
 if(s==t){  
 tree[p]=(\*arr)[s];  
 return;  
 }  
 ll m=s+((t-s)>>1);  
 build(s,m,p\*2),build(m+1,t,p\*2+1);  
 tree[p]=tree[p\*2]+tree[p\*2+1];  
 }  
public:  
 explicit Segtree<T>(vector<T> v){  
 n=v.size();  
 n4=4\*n;  
 tree=vector<T>(n4,0);  
 lazy1=vector<T>(n4,0);  
 lazy2=vector<T>(n4,1);  
 arr=&v;  
 end=n-1;  
 root=1;  
 build(0,end,1);  
 arr= nullptr;  
 }  
 T range\_sum(int l, int r) { return range\_sum(l, r, 0, end, root); }  
  
 void range\_add(int l, int r, int val) { range\_add(l, r, val, 0, end, root); }  
 void range\_mul(int l, int r, int val) { range\_mul(l, r, val, 0, end, root); }  
};  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 ll n,q;  
 cin>>n>>q;  
 vector<ll> a(n+1);  
 for(ll i=1;i<=n;i++)  
 cin>>a[i];  
 Segtree<ll> tr(a);  
 for(ll i=0;i<q;i++){  
 ll op;  
 cin>>op;  
 if(op==1){  
 ll x,y,k;  
 cin>>x>>y>>k;  
 tr.range\_mul(x,y,k);  
 }  
 else if(op==2){  
 ll x,y,k;  
 cin>>x>>y>>k;  
 tr.range\_add(x,y,k);  
 }  
 else{  
 ll x,y;  
 cin>>x>>y;  
 ll ans=tr.range\_sum(x,y);  
 cout<<ans<<endl;  
 }  
 }  
 return 0;  
}

随机数

#include<bits/stdc++.h>  
#define ll long long  
#define endl "\n"  
using namespace std;  
const ll mod=1e15+7;  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 ll n,q;  
 cin>>n>>q;  
 vector<ll> a(n+1,0),b(n+1,0);  
 for(ll i=1;i<=n;i++) cin>>a[i];  
 for(ll i=1;i<=n;i++) cin>>b[i];  
 vector<ll> ran(n+1);  
 mt19937\_64 myrand(time(nullptr));  
 for(ll i=1;i<=n;i++)  
 ran[i]=myrand()%mod;  
 for(ll i=1;i<=n;i++){  
 a[i]=ran[a[i]];  
 b[i]=ran[b[i]];  
 a[i]+=a[i-1];  
 b[i]+=b[i-1];  
 }  
 while(q--){  
 ll l,r,L,R;  
 cin>>l>>r>>L>>R;  
 cout<<((a[r]-a[l-1]==b[R]-b[L-1])?"Yes\n":"No\n");  
 }  
 return 0;  
}

高斯消元

#include<bits/stdc++.h>  
#define ll long long  
#define endl "\n"  
using namespace std;  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr),cout.tie(nullptr);  
 ll n;  
 cin>>n;  
 vector<vector<double>> fc(n+1,vector<double>(n+2));  
 for(ll i=1;i<=n;i++){  
 for(ll j=1;j<=n+1;j++)  
 cin>>fc[i][j];  
 }  
 vector<double> ans(n+1);  
 for(ll i=1;i<=n;i++){  
 ll r=i;  
 for(ll j=i+1;j<=n;j++){  
 if(llabs(fc[r][i])<llabs(fc[j][i]))  
 r=j;  
 }  
 if(llabs(fc[r][i]==0)){  
 cout<<"No Solution";  
 return 0;  
 }  
 if(i!=r)  
 swap(fc[i],fc[r]);  
 double div=fc[i][i];  
 for(ll j=i;j<=n+1;j++)  
 fc[i][j]/=div;  
 for(ll j=i+1;j<=n;j++){  
 div=fc[j][i];  
 for(ll k=i;k<=n+1;k++)  
 fc[j][k]-=fc[i][k]\*div;  
 }  
 }  
 ans[n]=fc[n][n+1];  
 for(ll i=n-1;i>0;i--){  
 ans[i]=fc[i][n+1];  
 for(ll j=i+1;j<=n;j++)  
 ans[i]-=fc[i][j]\*ans[j];  
 }  
 for(ll i=1;i<=n;i++){  
 cout<<fixed<<setprecision(2)<<ans[i]<<endl;  
 }  
 return 0;  
}

高精度

#include<bits/stdc++.h>  
#define ll long long  
using namespace std;  
bool check(vector<ll> n) {  
 if(n.size()==1&&n[0]==1)  
 return false;  
 return true;  
}  
vector<ll> mul(vector<ll> &a,ll b) {  
 vector<ll> c;  
 for(ll i=0,t=0;i<a.size()||t;i++) {  
 if(i<a.size())  
 t+=a[i]\*b;  
 c.emplace\_back(t%(ll)(1e16));  
 t/=1e16;  
 }  
 while(c.size()>1&&c.back()==0)  
 c.pop\_back();  
 return c;  
}  
vector<ll> div(vector<ll> &a,ll b) {  
 vector<ll> c;  
 ll r=0;  
 for(ll i=a.size()-1;i>=0;i--) {  
 r=r\*1e16+a[i];  
 c.emplace\_back(r/b);  
 r%=b;  
 }  
 reverse(c.begin(),c.end());  
 while(c.size()>1&&c.back()==0)  
 c.pop\_back();  
 return c;  
}  
vector<ll> add(vector<ll> &a,ll b) {  
 vector<ll> c;  
 a[0]+=b;  
 for(ll i=0,t=0;i<a.size()||t;i++) {  
 t+=a[i];  
 c.emplace\_back(t%(ll)1e16);  
 t/=1e16;  
 }  
 while(c.size()>1&&c.back()==0)  
 c.pop\_back();  
 return c;  
}  
void show(vector<ll> &a) {  
 cout<<a.back();  
 for(ll i=a.size()-2;i>=0;i--)  
 cout<<setw(16)<<setfill('0')<<a[i];  
 cout<<endl;  
}  
vector<ll> max(vector<ll> &a,vector<ll> &b) {  
 if(a.size()>b.size())  
 return a;  
 if(a.size()<b.size())  
 return b;  
 for(ll i=a.size()-1;i>=0;i--) {  
 if(a[i]>b[i])  
 return a;  
 if(a[i]<b[i])  
 return b;  
 }  
 return a;  
}  
int main() {  
 ios::sync\_with\_stdio(false),cin.tie(nullptr);  
 string s;  
 cin>>s;  
 vector<ll> n;  
 for(ll i=s.size()-1;i>=0;i-=16) {  
 ll st=max((ll)0,i-16+1),len=i-st+1;  
 n.emplace\_back(stoi(s.substr(st,len)));  
 }  
 ll cnt=0;  
 while(check(n)) {  
 if(n[0]%2==0)  
 n=div(n,2);  
 else {  
 n=mul(n,3);  
 n=add(n,1);  
 }  
 cnt++;  
 }  
 cout<<cnt;  
 return 0;  
}

字符串最小表示

#include <bits/stdc++.h>  
  
#define endl '\n'  
#define ll long long  
using namespace std;  
vector<ll> getnex(string s){  
 ll n=s.size();  
 vector<ll> nex(n+1,0);  
 for(ll i=2,j=0;i<=n;i++){  
 while(j&&s[i-1]!=s[j])  
 j=nex[j];  
 if(s[i-1]==s[j])  
 j++;  
 nex[i]=j;  
 }  
 return nex;  
}  
//字符串最小表示法  
string getmin(string s) {  
 ll n = s.size();  
 string sec = s + s;  
 int k = 0, i = 0, j = 1;  
 while (k < n && i < n && j < n) {  
 if (sec[(i + k) % n] == sec[(j + k) % n]) {  
 k++;  
 } else {  
 sec[(i + k) % n] > sec[(j + k) % n] ? i = i + k + 1 : j = j + k + 1;  
 if (i == j) i++;  
 k = 0;  
 }  
 }  
 i = min(i, j);  
 return sec.substr(i, n);  
}  
void solve() {  
 string s1,s2;  
 cin>>s1>>s2;  
 vector<ll> nex;  
 nex=getnex(s2);  
 s1=s1+s1;  
 ll n=s1.size(),m=s2.size();  
 for(ll i=1,j=0;i<=n;i++){  
 while(j&&s1[i-1]!=s2[j])  
 j=nex[j];  
 if(s1[i-1]==s2[j])  
 j++;  
 if(j==m){  
 cout<<"Yes"<<endl;  
 string tmp= getmin(s2);  
 cout<<tmp<<endl;  
 return;  
 }  
 }  
 cout<<"No"<<endl;  
}  
  
int main() {  
 ios::sync\_with\_stdio(false), cin.tie(nullptr), cout.tie(nullptr);  
 ll t;  
 t=1;  
 while (t--)  
 solve();  
 return 0;  
}