**Dp D&C + amortizado**

const int tam=100005;

ll a[tam];

ll cnt[tam];

const ll INF=1e16;

ll dp[25][tam];//G y pos

ll TOT=0;

int L=1,R;|

void add(int x){TOT+=cnt[x]++;}

void del(int x){TOT-=--cnt[x];}

ll query(int l,int r){

while(L>l) add(a[--L]);

while(R<r) add(a[++R]);

while(L<l) del(a[L++]);

while(R>r) del(a[R--]);

return TOT;

}

int solvedp(int g,int pos, int izq, int der){

int k=0;

dp[g][pos]=INF;

for(int i=izq;i<=min(der,pos-1);i++){

ll curr=dp[g-1][i]+query(i+1,pos);

if(curr<dp[g][pos]){

dp[g][pos]=curr;

k=i;

}

}

return k;

}

void solve(int g,int l, int r, int izq, int der){

if(l>r)return;

int mid=(l+r)/2;

int k=solvedp(g,mid,izq,der);

solve(g,l,mid-1,izq,k);

solve(g,mid+1,r,k,der);

}

int main(){

fast

fast

ll n,k;

cin>>n>>k;

ll acum=0;

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

cin>>a[i];

acum+=cnt[a[i]];cnt[a[i]]++;

dp[1][i]=acum;

}

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

for(int i=2;i<=k;i++){

solve(i,1,n,1,n);

}

cout<<dp[k][n]<<endl;

return 0;

}

**NCR modulo primo**

ll fact[5000005];

void init(){

fact[0]=1;

for(int i=1;i<=5000000;i++){

fact[i]=(fact[i-1]\*i)%MOD;

}

}

ll Pou(int a, int n){

if(n==0)return 1;

if(n%2==0){

ll A=Pou(a,n/2);

return (A\*A)%MOD;

}else{

ll A=Pou(a,n/2);

A=(A\*A)%MOD;

return (A\*a)%MOD;

}

}

ll nck(int n, int k){

if(n<k)return 0;

ll res=(fact[n]\*Pou((fact[k]\*fact[n-k])%MOD,MOD-2))%MOD;

return res;

}

int main()

{

ios::sync\_with\_stdio(0); cin.tie(0); cout.tie(0);

init();

return 0;

}

**NCR pascal**

ll pascal[5005][5005];

int main()

{

ios::sync\_with\_stdio(0); cin.tie(0); cout.tie(0);

pascal[0][0]=1;

for(int i=1;i<=5000;i++){

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

if(j==0 || j==i)pascal[i][j]=1;

else

pascal[i][j]=(pascal[i-1][j-1]+pascal[i-1][j])%MOD;

}

}

int n,k;

cout<<pascal[n][k]<<endl;

return 0;

}

**Punto con el mismo slope**

int dx=X-x;

int dy=Y-y;

int g=\_\_gcd(abs(dx),abs(dy));

pair<ii,int> meto={{x,y},h};

if(dx<0 || (dx==0 && dy<0)){

dx=-dx;

dy=-dy;

}

dx/=g;

dy/=g;

M[{dx,dy}].pb(meto);

----------------------------------------------------

int dx=X-x;

int dy=Y-y;

int g=\_\_gcd(abs(dx),abs(dy));

pair<ii,int> meto={{x,y},h};

dx/=g;

dy/=g;

M[{dx,dy}].pb(meto);

**LCA**

#include<bits/stdc++.h>

#define lcm(a,b) (a/\_\_gcd(a,b))\*b

#define fast ios\_base::sync\_with\_stdio(false);cin.tie(0);cout.tie(0);

using namespace std;

const int tam=1005;

int depth[tam];

int n;

int dp[tam][20];

vector<vi>G;

void init(int nodo, int p , int d){

depth[nodo]=d;

dp[nodo][0]=p;

for(auto it : G[nodo]){

if(it!=p){

init(it,nodo,d+1);

}

}

}

void initLCA(){

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

for(int l=0;l<20;l++){

dp[i][l]=-1;

}

}

init(1,-1,0);

for(int pot=1;pot<20;pot++){

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

if(dp[v][pot-1]==-1)continue;

dp[v][pot]=dp[dp[v][pot-1]][pot-1];

}

}

}

int LCA(int a, int b){

if(depth[a]>depth[b])swap(a,b);

int dif=depth[b]-depth[a];

for(int i=19;i>=0;i--){

if(dif&(1<<i)){

b=dp[b][i];

}

}

if(a==b)return a;

for(int i=19;i>=0;i--){

if(dp[a][i]!=dp[b][i]){

a=dp[a][i];b=dp[b][i];

}

}

return dp[a][0];

}

int main()

{

int t,a,b,m,caso=0;

cin>>t;

while(t--){

caso++;

cin>>n;

G.assign(n+1,vi());

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

cin>>m;

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

cin>>a;

G[i].pb(a);

G[a].pb(i);

}

}

initLCA();

return 0;

}

**ST**

void init(int nodo, int b, int e){

int L=2\*nodo+1,R=L+1,mid=(b+e)/2;

if(b==e){

T[nodo]=v[b];return;

}

init(L,b,mid);init(R,mid+1,e);

T[nodo]=min(T[L],T[R]);

}

void update(int pos,int val,int nodo,int b,int e){

int L=2\*nodo+1,R=2\*nodo+2,mid=(b+e)/2;

if(b==e && b==pos){

T[nodo]=val;

return;

}

if(pos>=b && pos<=mid){

update(pos,val,L,b,mid);

}else{

update(pos,val,R,mid+1,e);

}

T[nodo]=T[L]+T[R];

}

int query(int nodo, int b, int e, int izq, int der){

int L=2\*nodo+1,R=L+1,mid=(b+e)/2;

if(b>=izq && e<=der)return T[nodo];

if(der<=mid)return query(L,b,mid,izq,der);

if(izq>=mid+1)return query(R,mid+1,e,izq,der);

return min(query(L,b,mid,izq,der),query(R,mid+1,e,izq,der));

}

**Lazy Propagation**

ll T[4\*tam];

ll lazy[4\*tam];

vll v;

void init(int nodo, int b, int e){

int L=2\*nodo+1,R=L+1,mid=(b+e)/2;

lazy[nodo]=0;

if(e==b){

T[nodo]=v[b];

return;

}

init(L,b,mid);init(R,mid+1,e);

T[nodo]=min(T[L],T[R]);

}

void update(int nodo, int b, int e, int izq, int der, ll ad){

int L=2\*nodo+1,R=L+1,mid=(b+e)/2;

if(b>=izq && e<=der){

lazy[nodo]+=ad;

T[nodo]+=ad;

return;

}

if(der<=mid){

update(L,b,mid,izq,der,ad);

T[nodo]=min(T[L],T[R]);

return;

}

if(izq>=mid+1){

update(R,mid+1,e,izq,der,ad);

T[nodo]=min(T[L],T[R]);

return;

}

update(R,mid+1,e,izq,der,ad);

update(L,b,mid,izq,der,ad);

T[nodo]=min(T[L],T[R]);

}

int query(int nodo, int b, int e){

int L=2\*nodo+1,R=L+1,mid=(b+e)/2;

if(b==e){

return b;

}

lazy[L]+=lazy[nodo];

lazy[R]+=lazy[nodo];

T[L]+=lazy[nodo];

T[R]+=lazy[nodo];

lazy[nodo]=0;

ll res;

if(T[R]==0){

res=query(R,mid+1,e);

}else{

res=query(L,b,mid);

}

T[nodo]=min(T[L],T[R]);

return res;

}

**Factores Primos Criba**

vi v;

for(int i=0;i<=100000;i++){

v.pb(i);

}

for(int i=2;i\*i<=100000;i++){

for(int l=i\*i;l<=100000;l+=i){

if(v[l]==l){

v[l]=i;

}

}

}

int n,x;

cin>>n;

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

cin>>x;

while(1){

cout<<v[x]<<" ";

if(v[x]==x){

break;

}

x/=v[x];

}

cout<<endl;

}

**Matrix Exponentiation (fibo)**

//gcd(a,b)= ax + by

const ll MOD=1e9+7;

ll inv(ll a)

{

return fpow(a, MOD - 2);

}

struct Matrix{

ll a[2][2];

Matrix operator \*(Matrix A){

Matrix res = {{{0ll,0ll},{0ll,0ll}}};

for(int i=0;i<2;i++){

for(int l=0;l<2;l++){

for(int j=0;j<2;j++){

res.a[i][l]+=a[i][j]\*A.a[j][l];

res.a[i][l]%=MOD;

}

}

}

return res;

}

};

Matrix pou(Matrix a,ll n){

if(n==0){

return Matrix{{{1,0},{0,1}}};

}

if(n%2)return a\*pou(a,n-1);

Matrix res=pou(a,n/2);

return res\*res;

}

int main()

{

ll n;

cin>>n;

if(n==0){

cout<<0<<endl;

return 0;

}

Matrix aux=pou(Matrix{{{0ll,1ll},{1ll,1ll}}},n-1);

cout<<aux.a[1][1]<<endl;

return 0;

}

SOLO GRAFOS

**Centroid Descomposition**

// a+b=2\*(a&b) + (a^b)

using namespace std;

const int tam=200005;

vi G[tam];

int sz[tam];

bool vis[tam];

int cant[tam];

int max\_depth;

ll res=0;

int n,k;

int centroid(int nodo, int ant, int desired){

for(auto it : G[nodo]){

if(it==ant or vis[it])continue;

if(sz[it]>=desired)return centroid(it,nodo,desired);

}

return nodo;

}

void initsz(int nodo, int ant){

sz[nodo]=1;

for(auto it : G[nodo]){

if(it==ant or vis[it])continue;

initsz(it,nodo);

sz[nodo]+=sz[it];

}

}

void update(int nodo, int ant,int depth){

cant[depth]++;

for(auto it : G[nodo]){

if(it==ant or vis[it])continue;

update(it,nodo,depth+1);

}

}

void cal(int nodo,int ant, int depth){

if(depth>k)return;

max\_depth=max(max\_depth,depth);

res+=cant[k-depth];

for(auto it : G[nodo]){

if(it==ant or vis[it])continue;

cal(it,nodo,depth+1);

}

}

void solve(int nodo){

//aqui le paso cualquier nodo del subarbol

initsz(nodo,-1);

cant[0]=1;

max\_depth=0;

int centro=centroid(nodo,-1,sz[nodo]/2);

vis[centro]=1;

for(auto it : G[centro]){

if(vis[it])continue;

cal(it,centro,1);

update(it,centro,1);

}

for(int i=0;i<=max\_depth+2;i++)cant[i]=0;//Esto es super importante si no TLE

for(auto it : G[centro]){

if(!vis[it])solve(it);

}

}

int main(){

fast

cin>>n>>k;

int a,b;

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

cin>>a>>b;

G[b].pb(a);

G[a].pb(b);

}

solve(1);

cout<<res<<endl;

return 0;

}

//Given a tree of n nodes, your task is to count the number of distinct paths that consist of exactly k edges.

**Descomposición por puentes**

//la idea es convertir el grafo en un arbol donde cada nodo sea un conjunto

//en cada conjunto puedo ir de un nodo a otro pasando por la arista que quiero, checar descomposicion por orejas

vector<vi>G,G2,MG;//grafo normal, grafo sin puentes, grafo entre colores

int num=0;

vi vis,arc;

set<pair<int,int> >st;//puentes

set<pair<int,int> >rosa;//aristas con rosas

vi color;//color de cada nodo

vector<bool>check;// este color puede conseguir una rosa ?

int C=0;

void dfs(int nodo){

color[nodo]=C;

for(auto it : G2[nodo]){

if(rosa.find({nodo,it})!=rosa.end()){

//si hay alguna rosa en este conjunto siempre es posible tomarla y sesguir pq no hay puentes

check[C]=true;

}

if(st.find({nodo,it})!=st.end()){

continue;

}

if(color[it]==-1){

dfs(it);

}

}

}

set<pair<int,int> >ROSA;//rosas que hay en puentes que unen colores

void cal(int nodo, int padre){

for(auto it : MG[nodo]){

if(it==padre)continue;

check[it]=(check[nodo] or check[it]);//si ya consegui la rosa le paso

if(ROSA.find({nodo,it})!=ROSA.end()){

//si en el puente hay rosa tcs messirve

check[it]=true;

}

cal(it,nodo);

}

}

void tarjan(int nodo, int padre){

num++;

vis[nodo]=arc[nodo]=num;

for(auto it : G[nodo]){

if(it==padre)continue;

if(vis[it]){

arc[nodo]=min(arc[nodo],vis[it]);

}else{

tarjan(it,nodo);

arc[nodo]=min(arc[nodo],arc[it]);

if(arc[it]>vis[nodo]){

st.insert({nodo,it});

st.insert({it,nodo});

}

}

}

}

int main()

{

int n,m,a,b,c;

cin>>n>>m;

vector<pair<pair<int,int>,int> >E;color.assign(n+2,-1);

arc.resize(n+1);vis.assign(n+1,0);

G.assign(n+1,vi());G2.assign(n+1,vi());MG.assign(n+1,vi());

check.assign(n+1,false);

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

cin>>a>>b>>c;

if(c==1){

rosa.insert({a,b});rosa.insert({b,a});

}

G[a].pb(b);G[b].pb(a);

E.pb({{a,b},c});

}

tarjan(1,-1);

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

a=E[i].F.F,b=E[i].F.S,c=E[i].S;

if(st.find({a,b})==st.end() && st.find({b,a})==st.end()){

//entro solo si no es un puente

G2[a].pb(b);G2[b].pb(a);

}

}

//este forsito para colorear los nodos

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

if(color[i]==-1){

C++;

dfs(i);

}

}

//este forsito para unir colores con puentes :)

for(auto it : st){

a=color[it.F],b=color[it.S];//me interesa el color no el nodo siempre se conectan entre diferentes colores

if(rosa.find({it.F,it.S})!=rosa.end()){

ROSA.insert({a,b});ROSA.insert({b,a});

}

MG[a].pb(b);

}

int X,Y;

cin>>X>>Y;

cal(color[X],-1);//en un arbol solo hay un camino entre dos nodos

if(check[color[Y]]){

//si se puede conseguir la rosa

cout<<"YES"<<endl;

}else{

cout<<"NO"<<endl;

}

return 0;

}

**Dijkstra**

void dijkstra(int nodo){

priority\_queue<pair<int,int> >q;

q.push(mp(0,nodo));

peso[nodo]=0;

while(q.size()>0){

int u=q.top().S;

q.pop();

if(vis[u]==true){

continue;

}

vis[u]=true;

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

int hijo=G[u][i].F;

if(peso[u]+G[u][i].S<peso[hijo]){

peso[hijo]=peso[u]+G[u][i].S;

q.push(mp(-peso[hijo],hijo));

}

}

}

}

**Encontrar ciclos negativos**

ll n,m,a,b,w;

cin>>n>>m;

vector<pair<ll,pair<ll,ll> > >E;

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

cin>>a>>b>>w;

E.pb({a,{b,w}});

}

vll dis(n+1,INF);

dis[1]=0;

bool ok;

vi p(n+1,-1);

int in=0;

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

ok=true;

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

int a=E[l].F,b=E[l].S.F;w=E[l].S.S;

if(dis[b]>dis[a]+w){

ok=false;

dis[b]=dis[a]+w;

p[b]=a;

in=b;

}

}

}

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

in=p[in];

}

//este for para econtrar un cilo pq tal vez estoy en la rama

if(ok){

cout<<"NO"<<endl;

}else{

cout<<"YES"<<endl;

vi res;

int u=in;

while(1){

if(u==in && res.size()>1)break;

res.pb(u);

u=p[u];

}

res.pb(in);

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

cout<<res[i]<<" ";

}

}

**Euler path no dirigido ciclo**

const int tam=100005;

int n,m;//empieza y termina en 1 y pasa por todas las edges

set<int> G[tam]; //el indice de la arista

vi res;

int vis=0;

bool check(){

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

if(G[i].size()%2){

return false;

}

}

return true;

}

void go(int nodo){

while(G[nodo].size()){

vis++;

int to=\*G[nodo].begin();

G[nodo].erase(G[nodo].begin());

auto it=G[to].find(nodo);

G[to].erase(it);

go(to);

}

res.pb(nodo);

}

int main(){

int a,b;

cin>>n>>m;

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

cin>>a>>b;

G[a].insert(b);G[b].insert(a);

}

if(!check()){

cout<<"IMPOSSIBLE"<<endl;

return 0;

}

go(1);

if(vis!=m){

cout<<"IMPOSSIBLE"<<endl;

return 0;

}

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

cout<<res[i]<<" ";

}

return 0;

}

**Euler path (de A a B)**

**//de 1 a n grafo dirigido**

const int tam=100005;

set<int>G[tam];

vi res;

void go(int nodo){

while(G[nodo].size()>0){

auto it = G[nodo].begin();

G[nodo].erase(it);

go(\*it);

}

res.pb(nodo);

}

int main(){

int n,m,a,b;

cin>>n>>m;

vi in(n+1,0);

vi out(n+1,0);

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

cin>>a>>b;

G[a].insert(b);

out[a]++;in[b]++;

}

int be=-1;

int en=-1;

int cant=0;

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

if(in[i]==out[i]){

cant++;continue;

}

if(out[i]-1==in[i]){

be=i;

continue;

}

if(in[i]-1==out[i]){

en=i;

continue;

}

}

go(1);

if(be!=1 or en!=n or cant!= n-2 or res.size()-1 != m){

cout<<"IMPOSSIBLE"<<endl;

return 0;

}

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

if(res.back()!=n){

cout<<"IMPOSSIBLE"<<endl;

return 0;

}

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

cout<<res[i]<<" ";

}

return 0;

}

**Hopcroft-Karp Matching + BS**

int n;

int m;

const int tam=20000;

vector<pair<int,ii> >E;//son las aristas

vector<vi>G;

int matcha[tam];

int matchb[tam];

vi unmatch;

vi dis;

bool bfs(){

queue<int>q;

dis.assign(n+1,0);

for(auto it : unmatch){

dis[it]=1;

q.push(it);

}

bool ok=false;

while(!q.empty()){

int nodo=q.front();

q.pop();

for(auto it : G[nodo]){

if(!matchb[it]){

ok=1;

continue;

}

if(dis[matchb[it]])continue;

dis[matchb[it]]=dis[nodo]+1;

q.push(matchb[it]);

}

}

return ok;

}

bool dfs(int nodo,int w){

for(auto it : G[nodo]){

if(!matchb[it]){

matchb[it]=nodo;

matcha[nodo]=it;

return true;

}

}

for(auto it : G[nodo]){

if(dis[matchb[it]]==w+1){

dis[matchb[it]]=-1;

if(dfs(matchb[it],w+1)){

matchb[it]=nodo;

matcha[nodo]=it;

return true;

}

}

}

return false;

}

bool check(int x){

memset(matcha,0,sizeof(matcha));

memset(matchb,0,sizeof(matchb));

unmatch.clear();

for(int i=1;i<=n;i++)unmatch.pb(i);

G.assign(n+1,vi());

int cant=0;

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

int a=E[i].F,b=E[i].S.F,c=E[i].S.S;

if(c<=x){

G[a].pb(b);

}

}

//ya esta creado el grafito

while(bfs()){

//cout<<"lo nuestro se rompio y rompi la foto"<<endl;

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

if(dfs(unmatch[i],1)){

swap(unmatch[i],unmatch.back());

unmatch.pop\_back();

cant++;

}

}

}

return cant==n;

}

int main()

{

fast

int a,b,c;

cin>>n>>m;

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

cin>>a>>b>>c;

E.pb({a,{b,c}});

}

b=1;

int e=1e9,res=-1;

while(b<=e){

int mid=(b+e)/2;

if(check(mid)){

e=mid-1;

res=mid;

}else{

b=mid+1;

}

}

cout<<res<<endl;

return 0;

}

**Kosaraju scc**

const int tam=200005;

vi G[tam],G1[tam];

vector<bool>vis;

vi v;

int scc=0;

void dfs0(int nodo){

vis[nodo]=1;

for(auto it : G[nodo]){

if(!vis[it])dfs0(it);

}

v.pb(nodo);

}

void dfs1(int nodo){

vis[nodo]=1;

for(auto it : G1[nodo]){

if(!vis[it])dfs1(it);

}

}

int main()

{

int n,m,a,b;

cin>>n>>m;

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

cin>>a>>b;

G[a].pb(b);G1[b].pb(a);

}

vis.assign(n+1,false);

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

if(!vis[i])dfs0(i);

}

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

vis.assign(n+1,false);

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

if(!vis[v[i]]){

scc++;

dfs1(v[i]);

}

}

return 0;

}

**Puentes**

set<pair<int,int> >st;

vi arc,vis;

vector<vi>G;

int num=0;

void dfs(int nodo, int ant){

num++;

vis[nodo]=arc[nodo]=num;

for(auto it : G[nodo]){

if(it==ant)continue;

if(vis[it]){

arc[nodo]=min(arc[nodo],vis[it]);

}else{

dfs(it,nodo);

arc[nodo]=min(arc[nodo],arc[it]);

if(arc[it]>vis[nodo]){//no se pone mayor o igual por el primer if tcs si hay un puente hacia nodo es de más abajo

st.insert({nodo,it});

}

}

}

}

int main()

{

int n,m,a,b;

cin>>n>>m;

arc.resize(n+1);vis.resize(n+1);

G.assign(n+1,vi());

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

cin>>a>>b;

G[a].pb(b);G[b].pb(a);

}

dfs(1,-1);

cout<<st.size()<<endl;

for(auto it : st){

cout<<it.F<<" "<<it.S<<endl;

}

return 0;

}

**Puntos de articulación**

vector<vi>G;

vi vis,arc;

vector<bool>check;

int num=0;

void dfs(int nodo, int ant){

num++;

vis[nodo]=arc[nodo]=num;

int hijos=0;

for(auto it : G[nodo]){

if(ant==it)continue;

if(vis[it]){

arc[nodo]=min(arc[nodo],vis[it]);

}else{

hijos++;

dfs(it,nodo);

arc[nodo]=min(arc[nodo],arc[it]);

if(ant!=-1 && arc[it]>=vis[nodo]){

check[nodo]=1;

}

}

}

if(ant==-1 && hijos>1){

check[nodo]=1;

}

}

int main()

{

int n,m,a,b;

cin>>n>>m;

arc.resize(n+1);vis.resize(n+1);

check.assign(n+1,false);

G.assign(n+1,vi());

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

cin>>a>>b;

G[a].pb(b);

G[b].pb(a);

}

dfs(1,-1);

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

cout<<check[i]<<" ";

}

return 0;

}

SOLO STRINGS

**LCP**

int main(){

string s;

cin>>s;s.pb(char(36));

vector<pair<char,int> >a;//caracter e indice

int n=s.size();

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

a.pb({s[i],i});

}

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

vi p;//p son los indices en ordenados, la respuesta

for(int i=0;i<n;i++)p.pb(a[i].S);

int c[n];

c[p[0]]=0;//c[i] es la posicion de s[i] si ordeno los sufijos

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

if(a[i].F==a[i-1].F){

c[p[i]]=c[p[i-1]];

}else{

c[p[i]]=c[p[i-1]]+1;

}

}

int k=1;//es el tamaño curr

while(k <n){

vector<pair<pair<int,int>,int> >A;//valores de las dos mitades e indice

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

A.pb({{c[i],c[(i+k)%n]},i});

}

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

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

p[i]=A[i].S;

}

c[p[0]]=0;

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

if(A[i].F==A[i-1].F){

c[p[i]]=c[p[i-1]];

}else{

c[p[i]]=c[p[i-1]]+1;

}

}

k\*=2;

}

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

cout<<p[i]<<" ";

//cout<<p[i]<<" "<<s.substr(p[i],n-p[i])<<endl;

}

cout<<endl;

vi lcp(n);

k=0;

for(int i=0;i<n-1;i++){

int pi=c[i];

int j=p[pi-1];

while(s[i+k]==s[j+k])k++;

lcp[pi]=k;

k=max(k-1,0);

}

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

cout<<lcp[i]<<" ";

}

return 0;

}

**Longest common subsequence**

string s,ss;

cin>>s>>ss;

int n=s.size(),m=ss.size();

int M[n+1][m+1];

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

M[i][0]=0;

}

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

M[0][l]=0;

}

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

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

if(s[i-1]==ss[l-1]){

M[i][l]=1+M[i-1][l-1];

}else{

M[i][l]=max(M[i-1][l],M[i][l-1]);

}

}

}

cout<<M[n][m]<<endl;

**Longest common subsequence**

Cuando h[j]-h[i-1] queda p^i

const int tam=1000005;

const int p=31;

const int MOD=1e9+9;

ll H[tam];

ll P[tam];

ll n,m;

string a,b;

void init(){

P[0]=1;

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

P[i]=(p\*P[i-1])%MOD;

}

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

ll val=a[i]-'a'+1;

if(i){

H[i]=(H[i-1]+val\*P[i])%MOD;

}else{

H[i]=(P[i]\*val)%MOD;

}

}

}

int main(){

while(cin>>m){

cin>>b>>a;

n=a.size();

if(m>n){cout<<endl;continue;}

init();

ll A=0,B=0;

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

int val=b[i]-'a'+1;

B=(B+val\*P[i])%MOD;

}

B=(B\*P[n])%MOD;

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

if(i+m>n)break;

A=H[i+m-1];

if(i)A=(A-H[i-1]+MOD)%MOD;

A=(A\*P[n-i])%MOD;

if(A==B)cout<<i<<endl;

//cout<<A<<" "<<B<<endl;

}

}

return 0;

}

**Longest increasing subsequence**

int n,x;

n=v.size();

multiset<int>st;

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

x=v[i];

auto it=st.lower\_bound(x);

if(it==st.end()){

st.insert(x);

}else{

st.erase(it);

st.insert(x);

}

}

cout<<st.size()<<endl;

**Extraaaaaaa**

**Moebius (?)**

//FACILITO

//si m es el mayor numero de mi arreglo tcs res[m]=cal(m), res[m-1]=cal(m-1)-sumatoria(cal(multiplos(m-1) )

//cal(m)=crear la respuesta con multiplos de m

**DP auto**

int dp[10002][2005];

//dp[i][j]=el rango minimo si ya recorri las primeras i posiciones y estoy parado a j del borde izquierdo

int a[10005];

int main(){

int t,n,x;

cin>>t;

while(t--){

cin>>n;

for(int i=1;i<=n;i++)cin>>a[i];

for(int i=0;i<2005;i++)dp[0][i]=i+i;

int res=1e9;

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

for(int l=0;l<2005;l++){

dp[i][l]=1e9;

if(l-a[i]>=0)dp[i][l]=max(l,dp[i-1][l-a[i]]);

if(l+a[i]<2003)dp[i][l]=min(dp[i][l],dp[i-1][l+a[i]]);

}

}

for(int i=0;i<2000;i++)res=min(res,dp[n][i]);

cout<<res<<endl;

}

return 0;

}

//tienes te mueves a[i] hacia la izq o der, hallar la minima diferencia entre el borde izq y der

**BIT en 2D**

You are given an n×n grid representing the map of a forest. Each square is either empty or has a tree. Your task is to process q queries of the following types:

Change the state (empty/tree) of a square.

How many trees are inside a rectangle in the forest?

const int tam=1005;

int n,q;

int T[tam][tam];

void update(int x, int y, int val){

x++;y++;

for(;x<tam;x+=x&-x){

for(int l=y;l<tam;l+=l&-l)T[x][l]+=val;

}

}

int query(int x, int y){

x++;y++;

int res=0;

for(;x>0;x-=x&-x){

for(int l=y;l>0;l-=l&-l)res+=T[x][l];

}

return res;

}

int main()

{

cin>>n>>q;

string s;

vector<string>M;

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

cin>>s;

M.pb(s);

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

if(s[l]=='\*'){

update(i,l,1);

}

}

}

while(q--){

int c,x1,x2,y1,y2;

cin>>c;

if(c==1){

cin>>x1>>y1;

x1--;y1--;

if(M[x1][y1]=='\*'){

M[x1][y1]='.';

update(x1,y1,-1);

}else{

M[x1][y1]='\*';

update(x1,y1,1);

}

}else{

cin>>x1>>y1>>x2>>y2;

x1--;y1--;x2--;y2--;

cout<<query(x2,y2)-query(x2,y1-1)-query(x1-1,y2)+query(x1-1,y1-1)<<endl;

}

}

return 0;

}