Team Notebook

**A\*** (University of Engineering and Technology)

Contents

[1. Templates and misc 2](#_Toc58487075)

[Defines 2](#_Toc58487076)

[STLDS 2](#_Toc58487077)

[Settings 2](#_Toc58487078)

[2. Graphs 2](#_Toc58487079)

[Dinic flow 2](#_Toc58487080)

[Flow with demands 3](#_Toc58487081)

[Mincost – Maxflow 3](#_Toc58487082)

[Heavy-light decomposition 5](#_Toc58487083)

[Block-cut tree 6](#_Toc58487084)

[Kruscal tree 7](#_Toc58487085)

[Link-cut tree 8](#_Toc58487086)

[3. Geometry 9](#_Toc58487087)

[Line equation connecting two points 9](#_Toc58487088)

[Area of simpel polygon 10](#_Toc58487089)

[Lines intersection 10](#_Toc58487090)

[Circles intersection 10](#_Toc58487091)

[Convex hull using Graham’s scan 11](#_Toc58487092)

[Pick’s theorem 12](#_Toc58487093)

[Nearest pair of points 12](#_Toc58487094)

[4. Math 13](#_Toc58487095)

[Taylor series 13](#_Toc58487096)

[Gaussian elimination 13](#_Toc58487097)

[Polard Rho factorization 13](#_Toc58487098)

[Catalan number 14](#_Toc58487099)

[Fibonacci number 14](#_Toc58487100)

[Sum of divisors 15](#_Toc58487101)

[5. Strings 15](#_Toc58487102)

[Suffix array 15](#_Toc58487103)

[Aho-corasick 15](#_Toc58487104)

[6. Others 16](#_Toc58487105)

[Convex hull optimization 16](#_Toc58487106)

[Big integer 17](#_Toc58487107)

[Defines 18](#_Toc58487108)

# Templates and misc

## Macros

* #define in ({int x=0;int c=getchar(),n=0;for(;!isdigit(c);c=getchar()) n=(c=='-');for(;isdigit(c);c=getchar()) x=x\*10+c-'0';n?-x:x;})
* mt19937 rng(chrono::steady\_clock::now().time\_since\_epoch().count());
* 1.0\*clocks() / CLOCK\_PER\_SEC
* for(int i=bs.\_Find\_first();i< bs.size();i = bs.\_Find\_next(i))

## STLDS

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

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

using namespace \_\_gnu\_pbds;

typedef tree<int,null\_type,less<int>,rb\_tree\_tag,tree\_order\_statistics\_node\_update> ordered\_set;

* int can be replaced by types such as pair<int,int> for duplicate value storing.

## Settings

* Increase stack size:

-Wl,--stack,268435456

* Adding defines:

# Graphs

## Dinic flow

struct pack{

int u,v,c;

};

vector<int> ad[N];

vector<pack> e;

int n,m,s,t;

queue<int> q;

int d[N],ptr[N];

void add(int u,int v,int c){

ad[u].pb(e.size()); e.pb({u,v,c});

ad[v].pb(e.size()); e.pb({v,u,0});

}

int bfs(){

forinc(i,1,n) d[i]=-1;

queue<int> q; q.push(s); d[s]=0;

while(q.size()){

int u=q.front(); q.pop();

if(u==t) return 1;

forv(i,ad[u]) if(d[e[i].v]==-1 && e[i].c>0){

q.push(e[i].v);

d[e[i].v]=d[u]+1;

}

}

return 0;

}

int dfs(int u,int flow){

if(!flow) return 0;

if(u==t) return flow;

for(;ptr[u]<ad[u].size();ptr[u]++){

int i=ad[u][ptr[u]], v=e[i].v;

if(d[v]!=d[u]+1) continue;

int pushed=dfs(v,min(flow,e[i].c));

if(pushed){

e[i].c-=pushed;

e[i^1].c+=pushed;

return pushed;

}

}

return 0;

}

## Flow with demands

## Mincost – Maxflow

void minimize(int &x,const int &y) {

if (x>y) x=y;

}

class MCMF{

public:

struct edge{

int from,to,capa,flow,cost;

edge(){

from=to=capa=flow=cost=0;

}

edge(int u,int v,int ca,int co){

from=u,to=v,capa=ca,flow=0,cost=co;

}

int residual(void) const {

return capa-flow;

}

};

vector<vector<int>> g;

vector<edge> e;

vector<int> d,tr;

int n;

MCMF() {

n=0;

}

MCMF(int n) {

this->n=n;

e.clear();

g.resize(n+7);

d.resize(n+7);

tr.resize(n+7);

}

void addedge(int u,int v,int ca,int co) {

g[u].push\_back(e.size()); e.push\_back(edge(u,v,ca,co));

g[v].push\_back(e.size()); e.push\_back(edge(v,u,0,-co));

}

bool FordBellman(int s,int t){

forinc(i,1,n){

d[i]=INF;

tr[i]=-1;

}

vector<bool> inq(n+7);

queue<int> q;

q.push(s); d[s]=0; inq[s]=1;

while (!q.empty()) {

int u=q.front(); q.pop(); inq[u]=0;

forv(i,g[u]) if(e[i].residual()>0){

int v = e[i].to;

if(d[v] > d[u]+e[i].cost){

d[v] = d[u]+e[i].cost;

tr[v] = i;

if(!inq[v]){

q.push(v);

inq[v]=1;

}

}

}

}

return d[t]<INF;

}

ii getflow(int s,int t){

int totflow=0;

int totcost=0;

while(FordBellman(s,t)){

int delta=INF;

for (int u=t;u!=s;u=e[tr[u]].from) minimize(delta,e[tr[u]].residual());

for (int u=t;u!=s;u=e[tr[u]].from){

e[tr[u]].flow+=delta;

e[tr[u]^1].flow-=delta;

}

totflow+=delta;

totcost+=delta\*d[t];

}

return ii(totflow,totcost);

}

}T;

void process() {

int flow, cost; tie(flow,cost) = T.getflow(s,t);

forv(i,T.e)

if(i.flow > 0)

ad[i.from].push(i.to);

}

void loadgraph(){

T=MCMF(n+5);

T.addedge(s,u,cap,cost);

T.addedge(u,v,cap,cost);

T.addedge(v,t,cap,cost);

}

## Heavy-light decomposition

int it;

int sz[N],d[N],w[N],st[N],ed[N],par[N];

int f[N][22];

vector<int> ad[N];

void dfs1(int u,int p=-1){

sz[u]=1;

forinc(i,1,20) f[u][i]=f[f[u][i-1]][i-1];

forv(v,ad[u]) if(v!=p){

par[v]=u;

f[v][0]=u;

dfs1(v,u);

sz[u]+=sz[v];

d[u]=sz[d[u]]<sz[v] ? v : d[u];

}

}

void dfs2(int u,int p=-1){

st[u]=++it;

w[u]=w[u] ? w[u] : u;

if(d[u]) w[d[u]]=w[u], dfs2(d[u],u);

forv(v,ad[u]) if(v!=p && v!=d[u]) dfs2(v,u);

ed[u]=it;

}

int anc(int u,int v){

return st[u]<=st[v] && ed[u]>=ed[v] || !u;

}

int lca(int u,int v){

if(anc(u,v)) return u;

if(anc(v,u)) return v;

fordec(i,20,0) if(!anc(f[u][i],v)) u=f[u][i];

return f[u][0];

}

void upd(int u,int v){

while(!anc(w[u],v)){

(1,1,n,st[w[u]],st[u]);

u=par[w[u]];

}

while(w[v]!=w[u]){

(1,1,n,st[w[v]],st[v]);

v=par[w[v]];

}

(1,1,n,min(st[u],st[v]),max(st[u],st[v])); /// add 1 ?

}

int que(int u,int v){

int ret=0;

while(!anc(w[u],v)){

(1,1,n,st[w[u]],st[u]);

u=par[w[u]];

}

while(w[v]!=w[u]){

(1,1,n,st[w[v]],st[v]);

v=par[w[v]];

}

return ret+(1,1,n,min(st[u],st[v]),max(st[u],st[v])); /// add 1 ?

}

## Block-cut tree

void tja(int u,int p=-1){

num[u]=low[u]=++it; st[++top]=u;

forv(v,ad[u]) if(v!=p){

if(!num[v]){

tja(v,u);

low[u]=min(low[u],low[v]);

if(num[u]<=low[v]){

scc++; int x;

do{

x=st[top--];

adj[scc].pb(x);

}

while(x!=v);

adj[u].pb(scc);

}

}

else low[u]=min(low[u],num[v]);

}

}

## ­­Kruscal tree

struct pack{

int u,v,c;

};

struct kruskal{

int dir, tim, it;

vector<pack> edg;

int id[N],st[N],ed[N],val[N],pos[N];

int f[N][22];

vector<int> ad[N];

int root(int x){

return id[x]==x? x : id[x]=root(id[x]);

}

void dfs(int u,int p=-1){

st[u]=++tim; pos[tim]=u;

forinc(i,1,20)

f[u][i]=f[f[u][i-1]][i-1];

forv(v,ad[u]) if(v!=p){

f[v][0]=u;

dfs(v,u);

}

ed[u]=tim;

}

ii que(int u,int x){

if((dir && u>x) || (!dir && u<x)) return {n+n,n+n};

if((dir && val[f[u][0]]>x) || (!dir && val[f[u][0]]<x)) return {st[u],ed[u]};

u=f[u][0];

fordec(i,20,0)

if((dir && val[f[u][i]]<=x) || (!dir && val[f[u][i]]>=x)) u=f[u][i];

return {st[u],ed[u]};

}

void build(int dire){ dir=dire;

sort(all(edg),[](pack i,pack j){

return i.c>j.c;

});

if(dir) reverse(all(edg));

int it=n-1;

forinc(i,1,n+n) id[i]=i;

forv(i,edg){

int u=i.u,

v=i.v,

c=i.c,

x=root(u),

y=root(v);

if(x==y) continue;

val[it+1]=c;

ad[it+1].pb(x);

ad[it+1].pb(y);

id[x]=id[y]=++it;

}

dfs(f[it][0]=it);

}

}

## Link-cut tree

int n,q;

int L[N],R[N],P[N],Z[N],F[N];

void lef(int x,int y){L[x]=y,P[y]=x;}

void rig(int x,int y){R[x]=y,P[y]=x;}

void disr(int x){if(R[x]) P[R[x]]=0,Z[R[x]]=x,R[x]=0;}

void disl(int x){if(L[x]) P[L[x]]=0,Z[L[x]]=x,L[x]=0;}

void down(int x){

if(!F[x]) return;

F[L[x]]^=1,F[R[x]]^=1,F[x]^=1;

swap(L[L[x]],R[L[x]]);

swap(L[R[x]],R[R[x]]);

}

void up(int x){

int y=P[x],z=P[y];

if(L[y]==x) lef(y,R[x]),rig(x,y);

else rig(y,L[x]),lef(x,y);

if(L[z]==y) lef(z,x);

else rig(z,x);

Z[x]=Z[y],Z[y]=0;

}

int st[N];

void splay(int x){

int y=x,top=0;

st[++top]=y;

while(P[y]) st[++top]=y=P[y];

while(top) down(st[top--]);

for(;;){

int y=P[x],z=P[y];

if(!y) return;

if(z) if((L[z]==y)==(L[y]==x)) up(y);

else up(x);

up(x);

}

}

void access(int x){

splay(x);

disr(x);

int z=x;

while(Z[x]){

int y=Z[x];

splay(y);

disr(y);

rig(y,x);

Z[x]=0;

x=y;

}

splay(z);

}

void m\_root(int x){

access(x);

F[x]^=1;

swap(L[x],R[x]);

}

int f\_root(int x){

access(x);

while(L[x]) x=L[x];

access(x);

return x;

}

void link(int x,int y){

m\_root(y);

Z[y]=x;

}

void cut(int x,int y){

m\_root(x);

access(y);

P[x]=Z[x]=L[y]=0;

}

# Geometry

## Line equation connecting two points

line cross(pt i,pt j){

int a=i.y-j.y;

int b=-i.x+j.x;

int c=-a\*i.x-b\*i.y;

return {a,b,c};

}

## Area of simple polygon

double area(const vector<point>& fig) {

double res = 0;

for (unsigned i = 0; i < fig.size(); i++) {

point p = i ? fig[i - 1] : fig.back();

point q = fig[i];

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

}

return fabs(res) / 2;

}

## Lines intersection

struct line{

int a,b,c;

};

int intersect(line i,line j,double &x,double &y){

int a1=i.a,b1=i.b,c1=i.c;

int a2=j.a,b2=j.b,c2=j.c;

if(a2\*b1==b2\*a1 && b2\*c1==b1\*c2) return 2;

if(a2\*b1==b2\*a1) return 0;

if(b1){

x=(b2\*c1-c2\*b1)/(a2\*b1-b2\*a1);

y=(-c1-a1\*x)/b1;

} else{

y=(a2\*c1-a1\*c2)/(b2\*a1-a2\*b1);

x=(-c1-b1\*y)/a1;

}

return 1;

}

## Circles intersection

struct circle{

double x,y,r;

};

pt central;

int giao(circle i,circle j){

line t=t={2\*(j.x-i.x),2\*(j.y-i.y),-(j.x-i.x)\*(i.x+j.x)-(j.y-i.y)\*(j.y+i.y)-i.r\*i.r+j.r\*j.r};

double a=t.a,b=t.b,c=t.c; c+=a\*i.x+b\*i.y;

double r=i.r;

double x0 = -a\*c/(a\*a+b\*b), y0 = -b\*c/(a\*a+b\*b);

if (c\*c > r\*r\*(a\*a+b\*b)+EPS) return 0;

else if (abs (c\*c - r\*r\*(a\*a+b\*b)) < EPS) {

central={x0+i.x,y0+i.y};

return 1;

}

else {

double d = r\*r - c\*c/(a\*a+b\*b);

double mult = sqrt (d / (a\*a+b\*b));

double ax, ay, bx, by;

ax = x0 + b \* mult;

bx = x0 - b \* mult;

ay = y0 - a \* mult;

by = y0 + a \* mult;

central={ax+i.x,ay+i.y};

return 2;

}

}

## Convex hull using Graham’s scan

bool cmp(pt a, pt b) {

return a.x < b.x || (a.x == b.x && a.y < b.y);

}

bool cw(pt a, pt b, pt c) {

return a.x\*(b.y-c.y)+b.x\*(c.y-a.y)+c.x\*(a.y-b.y) < 0;

}

bool ccw(pt a, pt b, pt c) {

return a.x\*(b.y-c.y)+b.x\*(c.y-a.y)+c.x\*(a.y-b.y) > 0;

}

void convex\_hull(vector<pt>& a) {

if (a.size() == 1)

return;

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

pt p1 = a[0], p2 = a.back();

vector<pt> up, down;

up.push\_back(p1);

down.push\_back(p1);

for (int i = 1; i < (int)a.size(); i++) {

if (i == a.size() - 1 || cw(p1, a[i], p2)) {

while (up.size() >= 2 && !cw(up[up.size()-2], up[up.size()-1], a[i]))

up.pop\_back();

up.push\_back(a[i]);

}

if (i == a.size() - 1 || ccw(p1, a[i], p2)) {

while(down.size() >= 2 && !ccw(down[down.size()-2], down[down.size()-1], a[i]))

down.pop\_back();

down.push\_back(a[i]);

}

}

a.clear();

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

a.push\_back(up[i]);

for (int i = down.size() - 2; i > 0; i--)

a.push\_back(down[i]);

}

## Pick’s theorem

Given a certain lattice polygon with non-zero area.

We denote its area by , the number of points with integer coordinates lying strictly inside the polygon by and the number of points on polygon sides by .

## Nearest pair of points

pt e[N];

double dist(pt a, pt b){

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

}

struct comp{

bool operator() (int i,int j) const{

return e[i].y < e[j].y;

}

};

double findNearest(vector<pt> a){

int n = a.size();

forinc(i,1,n) e[i] = a[i-1];

auto cmp=[&](int i,int j){

return e[i].y<e[j].y;

};

sort(e+1,e+n+1,[](pt i,pt j){

return make\_pair(i.x,i.y) < make\_pair(j.x,j.y);

});

set<int,comp> st;

set<int,comp>::iterator it;

double ret = dist(e[1], e[2]);

int j = 1;

forinc(i,1,n){

while(j<=n && e[i].x - e[j].x > sqrt(ret) + EPS) st.erase(j++);

e[0].y = e[i].y - (int)sqrt(ret) - 2;

it = st.lower\_bound(0);

while(it!=st.end()){

int id = \*it;

if(fabs(e[id].y - (e[i].y + sqrt(ret))) < EPS) break;

ret = min(ret, dist(e[id],e[i]));

it++;

}

st.insert(i);

}

return sqrt(ret);

}

# Math

## Taylor series



## Gaussian elimination

forinc(i,1,k){

int j=1;

while(j<=k && !a[i][j]) j++;

if(j>k) continue;

forinc(t,1,k) if(t!=i){

if(a[t][j]){

int rat=a[i][j]\*pw(a[t][j],M-2)%M;

forinc(l,1,k+1) a[t][l]=a[t][l]\*rat%M;

forinc(l,1,k+1) a[t][l]=(a[i][l]-a[t][l]+M)%M;

}

}

}

## Polard Rho factorization

long long mult(long long a, long long b, long long mod) {

return (\_\_int128)a \* b % mod;

}

long long f(long long x, long long c, long long mod) {

return (mult(x, x, mod) + c) % mod;

}

long long rho(long long n, long long x0=2, long long c=1) {

long long x = x0;

long long y = x0;

long long g = 1;

while (g == 1) {

x = f(x, c, n);

y = f(y, c, n);

y = f(y, c, n);

g = gcd(abs(x - y), n);

}

return g;

}

long long mult(long long a, long long b, long long mod) {

long long result = 0;

while (b) {

if (b & 1)

result = (result + a) % mod;

a = (a + a) % mod;

b >>= 1;

}

return result;

}

long long brent(long long n, long long x0=2, long long c=1) {

long long x = x0;

long long g = 1;

long long q = 1;

long long xs, y;

int m = 128;

int l = 1;

while (g == 1) {

y = x;

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

x = f(x, c, n);

int k = 0;

while (k < l && g == 1) {

xs = x;

for (int i = 0; i < m && i < l - k; i++) {

x = f(x, c, n);

q = mult(q, abs(y - x), n);

}

g = gcd(q, n);

k += m;

}

l \*= 2;

}

if (g == n) {

do {

xs = f(xs, c, n);

g = gcd(abs(xs - y), n);

} while (g == 1);

}

return g;

}

## Catalan number

## Fibonacci number

## Sum of divisors

# Strings

## Suffix array

int gap;

int pos[N],sa[N],tmp[N],lcp[N];

bool cmp(int i,int j){

if(pos[i]!=pos[j]) return pos[i]<pos[j];

i+=gap,j+=gap;

return i<=n && j<=n ? pos[i]<pos[j] : i>j;

}

void build\_sfa(string s,int n){

for(int i=1;i<=n;++i) pos[i]=s[i]-'A',sa[i]=i;

for(gap=1;;gap\*=2){

sort(sa+1,sa+n+1,cmp);

for(int i=1;i<n;++i) tmp[i+1]=tmp[i]+cmp(sa[i],sa[i+1]);

for(int i=1;i<=n;++i) pos[sa[i]]=tmp[i]+1;

if(tmp[n]==n-1) break;

}

for(int i=1,k=0;i<=n;++i) if(pos[i]!=n){

for(int j=sa[pos[i]+1];s[i+k]==s[j+k];) ++k;

lcp[pos[i]]=k;

if(k) --k;

}

}

## Aho-corasick

/// cây aho đánh số từ 1

int add(string &s){

int i=1;

forv(j,s){

int t=j-'a';

if(!nxt[i][t]) nxt[i][t]=++it;

i=nxt[i][t];

}

return i;

}

void build\_aho(){

queue<int> q;

forinc(i,0,25){

if(nxt[1][i]){

kmp[nxt[1][i]]=1;

q.push(nxt[1][i]);

} else{

nxt[1][i]=1;

}

}

while(q.size()){

int u=q.front(); q.pop();

forinc(i,0,25) if(nxt[u][i]){

int v=nxt[u][i];

int t=kmp[u];

while(!nxt[t][i]) t=kmp[t];

t=nxt[t][i];

kmp[v]=t;

q.push(v);

}

}

forinc(i,2,it) ad[kmp[i]].push\_back(i);

}

# Others

## Convex hull optimization

int mode;

struct line{

mutable int a,b;

mutable double k;

bool operator<(const line &p)const{return mode?k<p.k:a>p.a;}

};

struct hull{

multiset<line> st;

double I(int x,int y){return 1.0\*x/y;}

int bad(auto x,auto y){

if(y==st.end()) return x->k=INF,0;

if(x->a==y->a) x->k=(x->b<=y->b?INF:-INF);

else x->k=I(x->b-y->b,y->a-x->a);

return x->k>=y->k;

}

void add(int a,int b){

auto z=st.insert({a,b,0}),x=z++,y=x;

while(bad(x,z)) z=st.erase(z);

if((y=x)!=st.begin()&&bad(--x,y)) bad(x,st.erase(y));

while((y=x)!=st.begin()&&bad(--x,y)) bad(x,st.erase(y));

}

int que(int x){

mode=1;

auto pos=st.lower\_bound({0,0,x});

mode=0;

return pos->a\*x+pos->b;

}

};

## Big integer

void stab(vector<int> &a,vector<int> &b){

while(a.size()<b.size()) a.push\_back(0);

while(a.size()>b.size()) b.push\_back(0);

}

bool operator > (vector<int> a,vector<int> b){

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

fordec(i,a.size()-1,0) if(a[i]!=b[i]) return a[i]>b[i];

return false;

}

vector<int> operator +(vector<int> a,vector<int> b){

stab(a,b);

int n=a.size();

vector<int> c(n);

int r=0;

forinc(i,0,n-1) {

c[i]=a[i]+b[i]+r;

r=(c[i]>9);

c[i]%=10;

}

if(r) c.push\_back(1);

return c;

}

vector<int> operator - (vector<int> a,vector<int> b){

if(b>a) swap(a,b),cout<<"-";

stab(a,b);

int n=a.size();

vector<int> c(n);

int r=0;

forinc(i,0,n-1) {

c[i]=(10+a[i]-b[i]-r)%10;

r=(a[i]-b[i]-r<0);

}

while(c.size()>1&&!c.back()) c.pop\_back();

return c;

}

vector<int> operator \* (vector<int> a,vector<int> b) {

stab(a,b);

int n=a.size();

vector<int> c(2\*n);

int r=0;

forinc(i,0,n-1){

r=0;

forinc(j,0,n-1){

c[i+j]+=a[i]\*b[j]+r;

r=c[i+j]/10;

c[i+j]%=10;

}

c[i+n]+=r;

}

while(c.size()>1&&!c.back()) c.pop\_back();

return c;

}

istream& operator >> (istream& cin,vector<int> &a){

string x; cin >> x; int na=x.size();

a.resize(na);

forinc(i,0,na-1) a[i]=x[na-i-1]-'0';

return cin;

}

ostream& operator << (ostream& cout, const vector<int> &a) {

fordec(i,a.size()-1,0) cout<<a[i];

return cout;

}

## Defines

* NHBinh

#include <bits/stdc++.h>

using namespace std;

#define in ({int x=0;int c=getchar(),n=0;for(;!isdigit(c);c=getchar()) n=(c=='-');for(;isdigit(c);c=getchar()) x=x\*10+c-'0';n?-x:x;})

mt19937 rng(chrono::steady\_clock::now().time\_since\_epoch().count());

int rnd(int l,int r){return l+rng()%(r-l+1);}

#define fasty ios\_base::sync\_with\_stdio(0),cin.tie(0);

#define forinc(a,b,c) for(int a=b,\_c=c;a<=\_c;++a)

#define fordec(a,b,c) for(int a=b,\_c=c;a>=\_c;--a)

#define forv(a,b) for(auto&a:b)

#define fi first

#define se second

#define pb push\_back

#define ii pair<int,int>

#define mt make\_tuple

#define all(a) a.begin(),a.end()

#define reset(f, x) memset(f, x, sizeof(f))

#define gg exit(0);

main(){

#define task "TASK"

if(fopen(task".inp","r")){

freopen(task".inp","r",stdin);

//freopen(task".out","w",stdout);

}

}

* DQTDuong

#include<bits/stdc++.h>

using namespace std;

typedef long long ll;

typedef unsigned long long ull;

#define int long long

int hmt() {int x=0;int c=getchar(),n=0;for(;!isdigit(c);c=getchar()) n=(c=='-');for(;isdigit(c);c=getchar()) x=x\*10+c-'0';if(n) x=-x;return x;}

#define in hmt()

#define ins ({string x;char c=getchar();for(;c==' '||c=='\n';c=getchar());for(;c!=' '&&c!='\n';c=getchar()) x+=c;x;})

#define forinc(i,a,b) for(int i=a,\_b=b;i<=\_b;++i)

#define fordec(i,a,b) for(int i=a;i>=b;--i)

#define forb(i,BS) for(int i=BS.\_Find\_first();i< BS.size();i = BS.\_Find\_next(i))

#define forv(a,b) for(auto &a:b)

#define pb push\_back

#define pii pair<int,int>

#define fi first

#define se second

#define all(a) a.begin(),a.end()

#define reset(f,x) memset(f,x,sizeof(f))

#define bit(x,i) ((x>>(i-1))&1)

#define onbit(x,i) (x|(1<<(i-1)))

#define offbit(x,i) (x&~(1<<(i-1)))

* PXHuy

#include<bits/stdc++.h>

#define int ll

//{ @pxh612

using namespace std;

template<typename A,typename B>bool Min(A &a,B b){if(a<=b) return 0;a=b;return 1;}

template<typename A,typename B>bool Max(A &a,B b){if(a>=b) return 0;a=b;return 1;}

#define in ({ll x=0;int o=0,c=char(),t=0;for(;!isdigit(c)&&t<1000;c=getchar(),t++) o=c=='-';for(;isdigit(c);c=getchar()) x=x\*10+c-'0';o?-x:x;})

#define inchar ({char c=getchar();while(c==' '||c=='\n') c=getchar();c;})

#define false(x) if(!(x))

#define FOR(i,a,b) for(int i=a,ENDFOR=b;i<=ENDFOR;i++)

#define ROF(i,a,b) for(int i=b,ENDFOR=a;ENDFOR<=i;i--)

#define RR(x,a,b) {cout<<#x<<": ";FOR(\_,a,b) cout<<x[\_]<<" ";cout<<"\n";}

#define rr(x) " "<<#x<<'='<<(x)<<" "

#define VEC(i,a) for(auto&i:a)

#define pb push\_back

#define rb(x) (int)x.size()-1

#define bit(x,i) ((x>>(i-1))&1ll)

#define on(x,i) (x|(1ll<<(i-1)))

#define off(x,i) (x&~(1ll<<(i-1)))

#define mu(x) (1ll<<x)

#define bitnum(x) \_\_builtin\_popcountll(x)

#define segg(a,b) (abs(a-b)+1)

#define midd(a,b) (a+(b-(a))/2)

#define mems(x,a) memset(x,a,sizeof x)

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

#define shut(b) freopen(b,"w",stdout)

#define ll long long

#define db double

#define pp pair<int,int>

#define x first

#define y second

//}/////////////////////////////////////////////////////}