/\*\*\*\*\*\*\*\*\*\* suffix-depth array \*\*\*\*\*\*\*\*\*\*/

int sa[M],da[M];

struct word

{

int w;

word \*n;

word(){ };

word(int \_w):w(\_w),n(NULL){};

};

void sasort(char \*ch,int n)

{

word \*f[27],\*b[27],\*tf,\*tb;

FR(i,n)

ch[i]=ch[i]-'a'+1,

ch[n+i]=0;

for(int i=0;i<=26;++i)

f[i]=b[i]=NULL ;

tf = tb = new word(0);

FOR(i,1,n)

{

tb -> n = new word(i);

tb = tb->n;

};

BFR(i,n)

{

//dispute;

while( tf != NULL )

{

int now=ch[tf->w+i];

if(f[now]==NULL)

f[now]=b[now]= tf;

else

{

b[now]->n = tf ;

b[now] = b[now]->n ;

}

tf = tf->n ;

b[now]->n = NULL;

}

//merge

tf = tb = NULL ;

for(int j=0;j<=26;++j)

{

if(f[j]!=NULL)

if(tf == NULL)

{

tf = f[j];

tb = b[j];

}

else

{

tb->n = f[j];

tb = b[j];

};

f[j]=b[j]=NULL;

}

}

//to suffix array

int i=0;

while(tf != NULL)

{

sa[i++] = tf->w ;

tf = tf->n;

}

/\*FR(i,n)

PR(sa[i]);\*/

}

void satoda(char\*ch ,int n){

int r[M];

FR(i,n)

r[sa[i]]=i;

FR(i,n)

if(r[i]==0)

da[0]=0;

else

{

int len = i==0 ? 0 : MAX(0,da[r[i-1]]-1);

while(ch[i+len]==ch[sa[r[i]-1]+len])

++len ;

da[r[i]] = len ;

}

}

/\*\*\*\*\*\*\*\*\*\* math \*\*\*\*\*\*\*\*\*\*/

ll inv(ll a,ll p){return a==1 ? 1 : inv(p%a,p)\*(p-p/a)%MOD;}

ll check;

void exgcd(ll a,ll b,ll &x,ll &y){

if(b==0)

{

x=1;y=0;check=a;

return ;

}

exgcd(b,a%b,y,x);

y-=x\*(a/b);

}

ll findinv(ll a,ll p){

ll x,y;

exgcd( a , p , x , y );

if(check!=1)

return 0;

return (x+p)%p ;

}

11 31 61 101 151 211 281 661 911 1051 1481 1483 1487 1489 5651

5653 5657 5659 28657 42737 68111 141079 374321 514229 1046527

3626149 16769023 27644437 39916801 433494437 479001599 1073676287

1073807359 1000000007 1442968193 2147483647

/\*\*\*\*\*\*\*\*\*\* max\_flow \*\*\*\*\*\*\*\*\*\*/

struct E{

int go,val;

E(){};

E(int \_go,int \_val):go(\_go),val(\_val){};

};

struct flow{

static const int VM = 10000,VN=10000;

int n,nuse=0;

E e[VM];

VI v[VN];

void init(int \_n){

n = \_n ; nuse=0;

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

v[i] = VI();

}

void add(int a,int b,int val){

v[a].pb(nuse);e[nuse]=E(b,val);++nuse;

v[b].pb(nuse);e[nuse]=E(a, 0);++nuse;

}

int dis[VN];

int bfs(){

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

dis[i]=INT\_MAX;

dis[0]=0;

std::queue<int> q;

q.push(0);

while(q.size()){

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

for(int i=0;i<v[now].size();++i){

E &ee = e[ v[now][i] ] ;

if(ee.val && dis[now]+1 < dis[ee.go] ){

dis[ee.go] = dis[now]+1;

q.push(ee.go);

}

}

/\*for(int i=0;i<n;++i)

printf("%d ",dis[i]);

puts("");\*/

}

return dis[n-1] != INT\_MAX;

};

int ptr[VN];

int go(int to,int val) {

if(to == n-1)

return val;

for(int &i=ptr[to];i<v[to].size();++i){

E &ee =e[ v[to][i] ];

if(ee.val && dis[to]+1==dis[ee.go] ){

int plu = go(ee.go , std::min(val,ee.val) );

if(plu){

ee.val-=plu;

e[ v[to][i]^1 ].val +=plu;

return plu;

}

}

}

return 0;

}

int maxflow(){

int sum=0,tmp;

while(bfs()){

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

ptr[i]=0;

while( (tmp=go(0,INT\_MAX)) )

sum+=tmp;

};

return sum;

}

}f;

/\*\*\*\*\*\*\*\*\*\* BIT \*\*\*\*\*\*\*\*\*\*/

#define lowbit(a) (a&-a)

struct bits{

const int MM = 1000000;

ll arr[MM];

int n;//1<= <=n

void init(int \_n){

n=1;

while(\_n){

n<<=1;

\_n>>=1;

}

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

arr[i]=0;

}

ll ask(int i){//a<= <=b

if(i<1)

return 0;

ll sum=0;

while(i){

sum += arr[i];

i ^= lowbit(i);

}

return sum;

}

ll gsum(int a,int b){

return ask(b)-ask(a-1);

}

void modi(int i,int &v){

while(i<=n){

arr[i]+= v ;

i += lowbit(i);

}

}

}bit;

/\*\*\*\*\*\*\*\*\*\* prime \*\*\*\*\*\*\*\*\*\*/

struct prim{

const static int MM=100000;

int pr[MM/10],ps;

char c[MM+1];

void init(){

ps=0;

memset(c,1,sizeof(c));

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

if(c[i])pr[ps++]=i;

for(int j=0;j<ps && pr[j]\*i<=MM ; ++j){

c[ pr[j]\*i ] = 0 ;

if(i%pr[j]==0)

break;

};

}

}

int isprim(int &a){

//be careful at 0,1

int n = (int)sqrt(a);

for(int i=0;i<ps && pr[i]<=n ;++i)

if( a % pr[i] == 0)

return pr[i];

return a;

}

}p;

/\*\*\*\*\*\*\*\*\*\* treap \*\*\*\*\*\*\*\*\*\*/

#define M 500000

struct treap{

static treap mem[M],\*pmem;

treap \*l,\*r;

int pri,val;

int size,max;

lld sum;

treap(){};

treap(int \_val):l(NULL),r(NULL),pri(rand()),val(\_val),size(1),max(\_val),sum(\_val){}

}treap::mem[M],\*treap::pmem = treap::mem;

int size(treap \*a){return a ? a->size : 0 ;}

int max (treap \*a){return a ? a->max : 0 ;}

lld sum (treap \*a){return a ? a->sum : 0 ;}

void pull(treap \*a){

a->size = size(a->r) + size(a->l) + 1;

a->max = std::max ( std::max ( max (a->r) , max (a->l) ) , a->val );

a->sum = sum(a->r) + sum(a->l) + a->val;

}

treap\* merge (treap\* a,treap\* b){

if(!a || !b) return a? a:b ;

else if ( a->pri > b->pri){

a->r = merge( a->r , b);

pull(a);

return a;

}

else{

b->l = merge (a , b->l);

pull(b);

return b;

}

}

void split (treap\* t , int key ,treap\* &a,treap\*& b){

if(!t) a = b = NULL ;

else if(size(t->l) < key ){

a = t ;

split(t->r , key - 1 - size(t->l) , a->r , b);

pull(a);

}

else{

b = t ;

split(t->l , key , a , b->l );

pull(b);

}

}

void split(treap\*t , int key ,treap\* &a,treap\* &b){

if(!t) a=b=NULL;

else if( t->key < key){

a = t ;

split (t->r , key , a->r , b);

pull(a);

}

else{

b=t;

split (t->l , key , a , b->l);

pull(b);

}

}

/\*\*\*\*\*\*\*\*\*\* AVL \*\*\*\*\*\*\*\*\*\*/

struct avl{

avl \*l,\*r;int val,dep,ti;

avl(){};

avl(int \_val):val(\_val),dep(1),ti(1){l=r=NULL;};

};

int fdep(avl \*t){return t ? t->dep : 0;}

void rot\_L(avl \* &t){

avl \*k = t -> r ;

t -> r = k -> l ;

k -> l = t ;

t->dep = std::max(fdep(t->l),fdep(t->r) ) + 1 ;

t = k ; //cout << "L";

}

void rot\_R(avl \*&t){

avl \*k = t -> l ;

t -> l = k -> r ;

k -> r = t ;

t->dep = std::max(fdep(t->l),fdep(t->r) ) + 1 ;

t = k ; //cout << "R" ;

}

void bal(avl \*&t){

if(!t)return ;

if(fdep(t->l) >= fdep(t->r)+2 ){

if( fdep(t->l->l) < fdep(t->l->r))

rot\_L(t->l);

rot\_R(t);

}

else if(fdep(t->l)+2 <= fdep(t->r) ){

if( fdep(t->r->l) > fdep(t->r->r) )

rot\_R(t->r);

rot\_L(t);

}

//pull

t->dep = std::max(fdep(t->l),fdep(t->r) ) + 1 ;

}

void insert(avl \* &t,int &val){

if(!t) t= new avl(val);

else if(val < t->val)

insert(t->l , val);

else if(val > t->val)

insert(t->r , val);

else

t->ti++;

bal(t);

}

/\*\*\*\*\*\*\*\*\*\* disjointset \*\*\*\*\*\*\*\*\*\*/

struct disjointset

{ static const int MM=10100;

int s[MM],n,group;

void init(int \_n){

group = n = \_n ;

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

s[i] = i ;

};

inline int find(int a){

return s[a]==a ? a : ( s[a] = find(s[a]) );

}

int unin(int a,int b){

if(find(a)!=find(b)){

s[find(a)] = find(b);

--group;

return 1;

}

else

return 0;

}

}disj;

/\*\*\*\*\*\*\*\*\*\* pack \*\*\*\*\*\*\*\*\*\*/

struct package{

int pack[M],n;

void init(int \_n){

n = \_n;

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

pack[i]=0;

}

void zero(int cost,int val){

for(int i=n-cost;i>=0;--i)

if(pack[i+cost] < pack[i]+val )

pack[i+cost] = pack[i]+val;

}

void mult(int cost,int val){

for(int i=cost;i<=n;++i)

if(pack[i] < pack[i-cost]+val)

pack[i] = pack[i-cost]+val;

}

void lim(int cost,int val,int t){

int now=1;

while(t>=now){

zero(cost\*now,val\*now);

t-=now;

now<<=1;

}

if(t)

zero(cost\*t,val\*t);

}

int out(){return pack[n];}

}pack;

/\*\*\*\*\*\*\*\*\*\* segment\_tree \*\*\*\*\*\*\*\*\*\*/

struct seg{

int max;ll sum,add;seg \*l,\*r;

seg():max(0),sum(0),add(0){}

void push(){

if(add){

l->add +=add;r->add +=add;

add=0;

}

}

void pull(){

max = std::max( l->max , r->max );

sum = l->sum + r->sum ;

}

};

struct {

int o,n;//<=n

seg \*root;

void init(int \_o , int \_n){

n = \_n; o = \_o;

root = build(o,n);

}

int ask(int x,int y){

return ask(root,o,n,x,y);}

void fix(int i,int v){

return fix(root,o,n,i,v);}

int ori[M];

seg\* build(int l,int r){//l<=x<=r

seg\* now = new seg();

if(l==r){

now->max = ori[l];

return now;

}

int mid = (l+r)>>1;

now->l = build( l,mid);

now->r = build(mid+1, r);

//now->pull();

return now;

}

int ask(seg\* a,int l,int r,int &x,int &y){

if(l>y || r<x)

return -2147483648;

if(x<=l && r<=y)

return a->max ;

a->push();

int mid = (l+r)>>1;

return std::max( ask(a->l, l,mid,x,y) ,

ask(a->r,mid+1, r,x,y) );

}

void fix(seg \*a,int l,int r,int &i,int &v){

if(l==r){

a->max+=v; //ori[l]+=v;

return ;

}

a->push();

int mid = (l+r)>>1;

if(mid >=i)

fix(a->l, l,mid,i,v);

else

fix(a->r,mid+1, r,i,v);

a->pull();

}

}tree;

/\*\*\*\*\*\*\*\*\*\* convex \*\*\*\*\*\*\*\*\*\*/

struct xy{

typedef int POI; // to change [double , ll , int]

POI x,y;

xy(){};

xy(POI \_x,POI \_y):x(\_x),y(\_y){};

xy operator + (const xy &b)const{

return xy(x+b.x,y+b.y);}

xy operator - (const xy &b)const{//ab = a-b

return xy(x-b.x,y-b.y);}

POI operator \* (const xy &b)const{

return x\*b.x + y\*b.y ;}

POI operator / (const xy &b)const{

return x\*b.y - y\*b.x; }

};

typedef std::vector<xy> VXY ;

struct convexhull{

double area2(VXY &v){

if(v.size()<=2)

return 0;

double sum=0;

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

sum += v[i-1]/v[i];

sum += v[v.size()-1]/v[0];

return std::fabs(sum/2);

}

VXY convex(xy \*arr,int n){

if(n<3)

puts("ERROR");

std::sort(arr,arr+n);

//down

VXY v;

v.pb(arr[0]);

v.pb(arr[1]);

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

while( v.size()>=2 &&

(v[v.size()-1] -v[v.size()-2] ) / ( arr[i]-v.back()) <= 0 )

v.pop\_back();

v.pb(arr[i]);

}

//up

v.pb(arr[n-2]);

int k = v.size();

for(int i=n-3;i>=0;--i){

while( v.size()>=k &&

(v[v.size()-1] -v[v.size()-2] ) / ( arr[i]-v.back()) <= 0 )

v.pop\_back();

v.pb(arr[i]);

}

return v;//the last one is the first one

}

}con;

/\*\*\*\*\*\*\*\*\*\* dijkstra \*\*\*\*\*\*\*\*\*\*/

struct xy{

int f,t,v;

};

struct dijkstra{

typedef std::pair<int,int> pii ;

static const int MM = 100000;

std::vector< pii > v[MM];

int ver[MM],use[MM];

int n,m;

void init(int \_n){

n=\_n;

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

v[i] = std::vector< pii >();

}

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

v[x].pb ( mp(y,val) );

v[y].pb ( mp(x,val) );

}

void go(int s=0){

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

ver[i]=INT\_MAX,use[i]=0;

ver[s]=0;

std::priority\_queue< pii ,

std::vector< pii > ,

std::greater< pii > > q;

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

while(q.size()) {

int now =q.top().sd;q.pop();

if(use[now])continue;

use[now]=1;

for(auto i:v[now])

if(ver[i.ft] > ver[now]+i.sd){

ver[i.ft] = ver[now]+i.sd ;

q.push( mp(ver[i.ft],i.ft) );

}

}

}

}dij;

/\*\*\*\*\*\*\*\*\*\* matrix \*\*\*\*\*\*\*\*\*\*/

#define matM 2

struct mat{

ll m[matM][matM];

int r,c;

/\*r=2 c=3

xxx

xxx\*/

mat(){};

mat(int \_r,int \_c):r(\_r),c(\_c){memset(m,0,sizeof(m));};

};

mat operator \* (mat a,mat b)

{

if(a.c != b.r)

puts("error");

mat c(a.r,b.c);

for(int i=0;i<c.r;++i)

for(int j=0;j<c.c;++j)

for(int k=0;k<a.c;++k)

c.m[i][j] = ( c.m[i][j] + a.m[i][k]\*b.m[k][j]%MOD ) %MOD;

return c;

}

struct fastmat{

mat pow(mat b,int time){

if(b.c != b.r)

puts("error");

mat now(b.c,b.r);

for(int i=0;i<b.r;++i)

now.m[i][i] = 1 ;

while(time){

if(time&1)

now = now \* b ;

b = b\*b;

time>>=1;

}

return now;

}

mat init(){

mat c(matM,1);

c.m[0][0] = 1;//f1

c.m[1][0] = 1;//f0

return c;

}

mat gobase(){

mat c(matM,matM);

c.m[0][0] = c.m[0][1] = c.m[1][0]= 1;

return c;

}

ll want(int n){

mat t = pow(gobase(),n) \* init(); //f0 = f1 = 1 , fn=fn-1 + fn-2

return t.m[1][0];// 0th = f0

}

}fm;