**笛卡尔树--zccz14**

struct CartesianTreeNode {

int parent, left, right;};

void BuildFromArray(int \*value, int N, int \*index, CartesianTreeNode \*tree,

int \*stack) {

for (int i = **0**; i < N; i++) index[value[i]] = i; // 计算逆映射

for (int i = **0**; i < N; i++)// 初始化节点

tree[i].parent = tree[i].l eft = tree[i].right = **-1**;

int size = **0**; // 初始化清空栈

for (int i = **0**; i < N; i++) {

int nextSize = size;

// 维护单调栈

while (nextSize > **0** && index[stack[nextSize - **1**]] > index[i])

nextSize--;

// 下面两个 if 语句块的顺序可变

if (nextSize > **0**) { // 栈中有元素

int top = stack[nextSize - **1**]; // 当前元素为栈顶元素的右孩子

tree[i].parent = top; tree[top].right = i;}

if (nextSize < size) { // 弹过栈

int lastPop = stack[nextSize]; // 最后出栈的元素为当前元素的左孩子

tree[lastPop].parent = i; tree[i].left = lastPop;}

stack[nextSize++] = i; // 入栈

size = nextSize; // 更新栈大小 }}

**堆--zbh2047**

struct Heap{

T a[**1000002**];

int n;

Heap(){ clear(); }

void down(int i){

for (int j = i << **1**; j <= n; i = j, j <<= **1**){

if (a[j + **1**] < a[j])j++;

if (a[j] < a[i])swap(a[i], a[j]);

else break;}}

void build(T src[], int n){

memcpy(a + **1**, src, n\*sizeof(T));

this->n = n;

for (int i = n >> **1**; i; i--){

a[n + **1**] = a[n]; down(i);}}

void push(T value){

a[++n] = value;

for (int j = n, i = j >> **1**; i && a[j] < a[i]; j = i, i >>= **1**)

swap(a[i], a[j]);}

void pop(){

a[**1**] = a[n--]; down(**1**);}

T top()const{ return a[**1**]; }

bool empty()const{ return n > **0**; }

void clear(){ n = **0**; }};

**可合并堆--Const**

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

if (a==mem) return b;

if (b==mem) return a;

if (a->v<b->v) swap(a,b);

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

if (a->r->dis>a->l->dis) swap(a->l,a->r);

if (a->r==mem) a->dis=**0**;

else a->dis=a->r->dis**+1**;

return a;}

**线段树--igoodvegetablea**

void add(int l,int r,int nu,int x){

if(a[x].ll==l && a[x].rr==r){

a[x].add+=nu;a[x].mi+=nu;return;}

downdate(x);

int mid=(a[x].ll+a[x].rr)>>**1**;

if(mid<l)add(l,r,nu,a[x].rs);

else if(mid>=r)add(l,r,nu,a[x].ls);

else{add(l,mid,nu,a[x].ls);add(mid**+1**,r,nu,a[x].rs);}

update(x);}

int find(int l,int r,int x){

if(a[x].ll==l && a[x].rr==r)return a[x].mi;

downdate(x); update(x);

/\*下同正常\*/

**树链剖分(边)--zbh2047**

int dfs1(int i, int fa){

father[i] = fa;depth[i] = depth[fa] + **1**;tmp[i] = **-1**;

int ret = **0**, maxSize = **0**;

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

int t = v[i][j].first;

if (t == fa)continue;

int size = dfs1(t, i); ret += size;

if (size > maxSize){ maxSize = size;tmp[i] = j;}}

return ret + **1**;}

void dfs2(int i, int tp, int index){

top[i] = tp;id[i] = cnt;f[index] = cnt++;

if (tmp[i] != **-1**) dfs2(v[i][tmp[i]].first, tp, v[i][tmp[i]].second);

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

int t = v[i][j].first;

if (t != father[i] && j != tmp[i])dfs2(t, t, v[i][j].second);

}}

int queryTree(int s, int t){

int ret = **0**;

int top1 = top[s], top2 = top[t];

while (top1 != top2){

if (depth[top1] < depth[top2]){

ret += sum(id[t]) - sum(id[top2] - **1**);

t = father[top2]; top2 = top[t];}

else{

ret += sum(id[s]) - sum(id[top1] - **1**);

s = father[top1]; top1 = top[s];}}

if (s != t){

if (depth[s] > depth[t])swap(s, t);

ret += sum(id[t]) - sum(id[s]);}

return ret;}

void init(){

cnt = **0**; dfs1(**1**, **1**); dfs2(**1**, **1**, **0**);

for (int i = **1**; i < n; i++)tree[f[i]] = w[i];

build();}

**树链剖分(点)--zbh2047**

int dfs1(int i, int fa){

father[i] = fa; depth[i] = depth[fa] + **1**; son[i] = **0**;

int ret = **0**, maxSize = **0**;

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

int t = v[i][j];

if (t == fa)continue;

int size = dfs1(t, i); ret += size;

if (size > maxSize){ maxSize = size; son[i] = t;}}

return ret + **1**;}

void dfs2(int i, int tp){

top[i] = tp; id[i] = cnt++;

if (son[i])dfs2(son[i], tp);

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

int t = v[i][j];

if (t != father[i] && t != son[i])dfs2(t, t);}}

void init(){

cnt = **0**; depth[**1**] = **0**;

dfs1(**1**, **1**); dfs2(**1**, **1**);

for (treeLen = **1**; treeLen < n; treeLen \*= **2**);

memset(tree, **0**, sizeof(Tree) \* **2** \* treeLen);}

void pushDown(int i){

if (tree[i].set){

tree[**2** \* i].set = tree[**2** \* i + **1**].set = true;

tree[**2** \* i].delta = tree[**2** \* i + **1**].delta = tree[i].delta;

tree[i].delta = **0**; tree[i].set = false;}

else if (tree[i].delta){

tree[**2** \* i].delta += tree[i].delta;

tree[**2** \* i + **1**].delta += tree[i].delta;

tree[i].delta = **0**;}}

int queryL, queryR;

void addInternal(int i, int l, int len){

if (queryL <= l && queryR >= l + len){

tree[i].delta++; return;}

len >>= **1**; pushDown(i);

int mid = l + len;

if (mid > queryL)addInternal(**2** \* i, l, len);

if (mid < queryR)addInternal(**2** \* i + **1**, mid, len);}

inline void addValue(int l, int r){

queryL = l; queryR = r;

addInternal(**1**, **0**, treeLen);}

int addTree(int s, int t){

int ret = **0**;

int top1 = top[s], top2 = top[t];

while (top1 != top2){

if (depth[top1] < depth[top2]){

addValue(id[top2], id[t] + **1**);

t = father[top2]; top2 = top[t];}

else{

addValue(id[top1], id[s] + **1**);

s = father[top1]; top1 = top[s];}}

if (depth[s] > depth[t])swap(s, t);

addValue(id[s], id[t] + **1**); return ret;}

void process(int i){

if (tree[i].set){

if (tree[i].delta > tree[i].maxValue){

tree[i].maxValue = tree[i].delta;

tree[i].maxId = color;

}return;}

pushDown(i);

process(**2** \* i); process(**2** \* i + **1**);}

inline void pushDownColor(int i, int j){

if (tree[j].maxValue < tree[i].maxValue

|| (tree[j].maxValue == tree[i].maxValue && tree[i].maxId < tree[j].maxId)){

tree[j].maxValue = tree[i].maxValue;

tree[j].maxId = tree[i].maxId;}}

void getAns(int i){

if (i < treeLen){

pushDownColor(i, **2** \* i);

pushDownColor(i, **2** \* i + **1**);

getAns(**2** \* i); getAns(**2** \* i + **1**);}}

**Treap--igooevegetablea**

void update(int k){//更新结点信息

tr[k].size=tr[tr[k].l].size+tr[tr[k].r].size+tr[k].w;}

void rturn(int &k){

int t=tr[k].l;tr[k].l=tr[t].r;tr[t].r=k;

tr[t].size=tr[k].size;update(k);k=t;}

void lturn(int &k){

int t=tr[k].r;tr[k].r=tr[t].l;tr[t].l=k;

tr[t].size=tr[k].size;update(k);k=t;}

void insert(int &k,int x){

if(k==**0**){

size++;k=size;

tr[k].size=tr[k].w=**1**;tr[k].v=x;tr[k].rnd=rand();

return;}

tr[k].size++;

if(tr[k].v==x)tr[k].w++;

else if(x>tr[k].v){

insert(tr[k].r,x);

if(tr[tr[k].r].rnd<tr[k].rnd)lturn(k);}

else {

insert(tr[k].l,x);

if(tr[tr[k].l].rnd<tr[k].rnd)rturn(k);}}

void del(int &k,int x){

if(k==**0**)return;

if(tr[k].v==x){

if(tr[k].w>**1**){ tr[k].w--;tr[k].size--;return;}

if(tr[k].l\*tr[k].r==**0**)k=tr[k].l+tr[k].r;

else if(tr[tr[k].l].rnd<tr[tr[k].r].rnd) rturn(k),del(k,x);

else lturn(k),del(k,x);}

else if(x>tr[k].v) tr[k].size--,del(tr[k].r,x);

else tr[k].size--,del(tr[k].l,x);}

int query\_rank(int k,int x){

if(k==**0**)return **0**;

if(tr[k].v==x)return tr[tr[k].l].size**+1**;

else if(x>tr[k].v)

return tr[tr[k].l].size+tr[k].w+query\_rank(tr[k].r,x);

else return query\_rank(tr[k].l,x);}

int query\_num(int k,int x){

if(k==**0**)return **0**;

if(x<=tr[tr[k].l].size) return query\_num(tr[k].l,x);

else if(x>tr[tr[k].l].size+tr[k].w)

return query\_num(tr[k].r,x-tr[tr[k].l].size-tr[k].w);

else return tr[k].v;}

void query\_pro(int k,int x){

if(k==**0**)return;

if(tr[k].v<x){ ans=k;query\_pro(tr[k].r,x);}

else query\_pro(tr[k].l,x);}

void query\_sub(int k,int x){

if(k==**0**)return;

if(tr[k].v>x){ ans=k;query\_sub(tr[k].l,x);}

else query\_sub(tr[k].r,x);}

**Splay--igoodvegetablea**

struct tree{

int key,size,le,ri,add,rev,min,pre;}a[maxn];

void pushdown(int cur){

int ls=a[cur].le,rs=a[cur].ri;

if(a[cur].add>**0**){

a[ls].add+=a[cur].add;a[rs].add+=a[cur].add;

a[ls].key+=a[cur].add;a[rs].key+=a[cur].add;

a[ls].min+=a[cur].add;a[rs].min+=a[cur].add;

a[cur].add=**0**;}

if(a[cur].rev>**0**){

a[ls].rev^=**1**;a[rs].rev^=**1**;a[cur].le=rs;

a[cur].ri=ls;a[cur].rev=**0**;}}

void update(int cur){

int ls=a[cur].le,rs=a[cur].ri;

a[cur].size=a[ls].size+a[rs].size**+1**;

a[cur].min=a[cur].key;

if(ls&&a[ls].min<a[cur].min)a[cur].min=a[ls].min;

if(rs&&a[rs].min<a[cur].min)a[cur].min=a[rs].min;}

void leftrotate(int x){

int y=a[x].ri,p=a[x].pre;

a[x].ri=a[y].le;if(a[x].ri)a[a[x].ri].pre=x;

a[y].le=x;a[x].pre=y;a[y].pre=p;

if(!p)T=y;

else a[p].ri==x?a[p].ri=y:a[p].le=y;

update(x);}

void rightrotate(int x){

int y=a[x].le,p=a[x].pre;

a[x].le=a[y].ri;if(a[x].le)a[a[x].le].pre=x;

a[y].ri=x;a[x].pre=y;a[y].pre=p;

if(!p)T=y;

else a[p].ri==x?a[p].ri=y:a[p].le=y;

update(x);}

void splay(int x,int goal){

int y,z;

while(**1**){

if((y=a[x].pre)==goal)break;

if((z=a[y].pre)==goal)

a[y].ri==x?leftrotate(y):rightrotate(y);

else{

if(a[z].ri==y){

if(a[y].ri==x)leftrotate(z),leftrotate(y);

else rightrotate(y),leftrotate(z);

}else{

if(a[y].le==x)rightrotate(z),rightrotate(y);

else leftrotate(y),rightrotate(z);}}}

update(x);}

void rotateto(int k,int goal){

int i=T;

while(**1**){

pushdown(i);

if(a[a[i].le].size**+1**==k)break;

if(k<=a[a[i].le].size)i=a[i].le;

else k-=a[a[i].le].size**+1**,i=a[i].ri;}

splay(i,goal);}

void newnode(int &cur,int v){

cur=++node;

a[cur].min=a[cur].key=v;a[cur].size=**1**;

a[cur].le=a[cur].ri=a[cur].rev=a[cur].add=**0**;}

void build(int &cur,int x,int y,int p){

int mid=(x+y)>>**1**;newnode(cur,s[mid]);

a[cur].pre=p;if(x==y)return;

if(x<mid)build(a[cur].le,x,mid**-1**,cur);

if(y>mid)build(a[cur].ri,mid**+1**,y,cur);

update(cur);}

void init(int n){

int i;memset(s,**0**,sizeof(s));memset(a,**0**,sizeof(a));

for(i=**1**;i<=n;i++)scanf("%d",&s[i]);

T=node=**0**;build(T,**0**,n**+1**,**0**);}

void Add(int x,int y,int z){

int k;rotateto(x,**0**);rotateto(y**+2**,T);

k=a[a[T].ri].le;

a[k].add+=z;a[k].key+=z;a[k].min+=z;}

void Reverse(int x,int y){

int k;rotateto(x,**0**);rotateto(y**+2**,T);

k=a[a[T].ri].le; a[k].rev^=**1**;}

void Revolve(int x,int y,int z){

int k=z%(y-x**+1**),t;

if(k){rotateto(x,**0**);rotateto(y-k**+2**,T);

t=a[a[T].ri].le; a[a[T].ri].le=**0**;

update(a[T].ri);update(T);

rotateto(x+k,**0**);rotateto(x+k**+1**,T);

a[a[T].ri].le=t;a[t].pre=a[T].ri;

update(a[T].ri);update(T);}}

void Insert(int x,int y){

rotateto(x**+1**,**0**);rotateto(x**+2**,T);

newnode(a[a[T].ri].le,y);

a[a[a[T].ri].le].pre=a[T].ri;

update(a[T].ri);update(T);}

void Delete(int x){

rotateto(x,**0**);rotateto(x**+2**,T);

a[a[T].ri].le=**0**;

update(a[T].ri);update(T);}

void Min(int x,int y){

rotateto(x,**0**);rotateto(y**+2**,T);

printf("%d\n",a[a[a[T].ri].le].min);}

**NTT&&CRT---zbh2047**

inline int add(int a, int b){ return a + b - (a + b >= MOD ? MOD : **0**);}

inline int sub(int a, int b){ return a - b + (a - b < **0** ? MOD : **0**);}

inline int mul(int a, int b){ return (long long)a \* b % MOD;}

int power(int a, int b){

int ret = **1**;

for (int t = a; b; b >>= **1**){

if (b & **1**)ret = mul(ret, t);

t = mul(t, t);}

return ret;}

int cal\_root(int mod){

for (int i = **2**;; i++){

if (power(i, (mod - **1**) / **2**) == mod - **1**)return i;}}

void fft\_init(int n, int mod){

MOD = mod; bit = (int)log2(n - 0.5) + **2**; len = **1** << bit;

w[**0**][**0**] = power(cal\_root(mod), (mod - **1**) / len);

int i;

for (i = **1**; i < bit; i++)w[**0**][i] = mul(w[**0**][i - **1**], w[**0**][i - **1**]);

i--;w[**1**][i] = w[**0**][i];

for (i--; i >= **0**; i--) w[**1**][i] = mul(w[**1**][i + **1**], w[**0**][i]);}

void bitReverse(int a[]) {

for (int i = **1**, j = len / **2**; i < len - **1**; i++) {

if (i < j) swap(a[i], a[j]);

int k = len / **2**;

while (j >= k) { j -= k; k >>= **1**; }

if (j < k) j += k;}}

void fft\_main(int a[], bool reverse){

bitReverse(a);

for (int i = **1**, s = **1**; s < len; i++, s <<= **1**){

int step = w[reverse][bit - i];

for (int j = **0**; j < len; j += **2** \* s){

int cur = **1**;

for (int k = j; k < j + s; k++){

int u = a[k], t = mul(cur, a[k + s]);

a[k] = add(u, t); a[k + s] = sub(u, t);

cur = mul(cur, step);}}}

if (reverse){

int t = power(len, MOD - **2**);

for (int i = **0**; i < len; i++)a[i] = mul(a[i], t);}}

//确保数组中的数小于mod(mod<2^30)，数组需留足2^(logn向上取整+1)的空间，后面填充0

//并且mod为形如m\*2^k+1的素数，2^k>=2\*n

void fft(int a[], int b[], int n, int mod){

fft\_init(n, mod);

fft\_main(a, **0**); fft\_main(b, **0**);

for (int i = **0**; i < len; i++)a[i] = mul(a[i], b[i]);

fft\_main(a, **1**);}

//确保mod两两互质，retmod任意

void chineseRemainder(const int mod[], int \*a[], int ret[], int num, int n, int retMod){

int kk[**30**], mulMod[**30**][**30**], mulModr[**30**], mulretMod[**30**];

for (int i = **0**; i < num; i++){

MOD = mod[i]; mulMod[i][**0**] = **1**;

for (int j = **1**; j <= i; j++)

mulMod[i][j] = mul(mulMod[i][j - **1**], mod[j - **1**]);

mulModr[i] = power(mulMod[i][i], MOD - **2**);}

mulretMod[**0**] = **1**; MOD = retMod;

for (int i = **1**; i < num; i++)

mulretMod[i] = mul(mulretMod[i - **1**], mod[i - **1**]);

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

for (int j = **1**; j < num; j++){

MOD = mod[j];

int sum = a[**0**][i] % MOD;

for (int k = **1**; k < j; k++)

sum = add(sum, mul(mulMod[j][k], kk[k]));

kk[j] = mul(sub(a[j][i] % MOD, sum), mulModr[j]);}

MOD = retMod; ret[i] = a[**0**][i] % MOD;

for (int j = **1**; j < num; j++)

ret[i] = add(ret[i], mul(kk[j] % MOD, mulretMod[j]));}}

//附满足条件大整数：167772161, 469762049, 754974721

// MOD = mm[1];

int r = power(mm[0], mm[1] - 2);

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

ans[i] = mm[0] \* (long long)mul(r, sub(a[2][i + 50000], a[0][i + 50000])) + a[0][i + 50000];

**Math--igoodvegetablea**

LL det(LL a[][maxn], int n) {//求行列式值(整数版)

int i, j;

LL res = **1**;

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

for (j = i + **1**; j < n; j++) {

while (a[j][i]) {

LL f = a[i][i] / a[j][i];

for (int k = i; k < n; k++) a[i][k] -= f \* a[j][k];

for (int k = i; k < n; k++) swap(a[i][k], a[j][k]);

res = -res;}}

if (a[i][i] == **0**) return **0**;

res \*= a[i][i];}

return res < **0** ? -res : res;}

double FF(double x) {return 1.0;}

double simpson(double x,double y){

double z=x+(y-x)/2.0;

return (y-x)/6.0\*(FF(x)+FF(y)**+4**\*FF(z));}

double asr(double x,double y,double eeps,double A)//eeps为精度{

double z=x+(y-x)/2.0;

double L=simpson(x,z), R=simpson(z,y);

if(fabs(L+R-A)<=**15**\*eeps)return (L+R)+(L+R-A)/15.0;

else return asr(x,z,eeps/2.0,L)+asr(z,y,eeps/2.0,R);}

double simpson\_zsx(double x,double y,double eeps)//自适应辛普森主函数{

return asr(x,y,eeps,simpson(x,y));}

void gauss\_eli(struct Matrix& p,int n)//高斯消元{

int i,j,k,r;

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

r=i;

for(j=i**+1**;j<=n;j++)

if(fabs(p.a[j][i])>fabs(p.a[r][i]))r=j;

if(r!=i)for(j=**1**;j<=n**+1**;j++)swap(p.a[r][j],p.a[i][j]);

for(k=**1**;k<=i**-1**;k++){

if(p.a[i][k]==**0**)continue;

for(j=n**+1**;j>=k;j--)

p.a[i][j]-=p.a[k][j]/p.a[k][k]\*p.a[i][k];}}

for(i=n;i>=**1**;i--){

for(j=i**+1**;j<=n;j++)

p.a[i][n**+1**]-=p.a[j][n**+1**]\*p.a[i][j];

p.a[i][n**+1**]/=p.a[i][i];}}

void tgcd(LL a,LL b,LL& d,LL& x,LL& y) {//拓展欧几里德

if(!b){d=a;x=**1**;y=**0**;}

else{tgcd(b,a%b,d,y,x);y-=x\*(a/b);}}

int euler\_phi(int n)//求欧拉函数{

int m=(int)sqrt(n+0.5);

int ans=n;

for(int i=**2**;i<=m;i++)

if(n%i==**0**){

ans=ans/i\*(i**-1**);

while(n%i==**0**)n=n/i;}

if(n>**1**)ans=ans/n\*(n**-1**);

return ans;}

void phi\_table(int n)//欧拉函数表{

memset(phi,**0**,n**+1**); phi[**1**]=**1**;

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

if(phi[i])continue;

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

if(!phi[j])phi[j]=j;

phi[j]=phi[j]/i\*(i**-1**);}}}

LL inv(LL a,LL n)//a关于n的逆元{

LL d,x,y;

tgcd(a,n,d,x,y);

return d==**1**?(x+n)%n:**-1**;}

// x mod m0=a0,x mod m =a,noSolution return 0

//初始可令 m0 = 1 ,a0 = 0布尔值返回是否有解 m0,m可以不互质

//若有多个方程，做多次此剩余定理 m0，a0返回答案

bool \_china(LL &m0,LL &a0,LL m,LL a) {

LL g,x,y,c=abs(a-a0);

tgcd(m0,m,g,x,y);

if ( c % g ) return **0**;

x\*=(a-a0)/g; x%=m/g;

a0=x\*m0+a0; m0\*=m/g; a0%=m0;

if(a0<**0**) a0+=m0;

return **1**;}

LL china(int n,int\* a,int\* m) {//中国剩余定理

LL M=**1**,d,y,x=**0**;

for(int i=**0**;i<n;i++)M\*=m[i];

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

LL w=M/m[i];

tgcd(m[i],w,d,d,y); x=(x+y\*w\*a[i])%M;}

return (x+M)%M;}

int log\_mod(int a,int b,int n) {//求解模方程a^x=b(mod n),n为素数,无解返回-1

int m,v,e=**1**,i;

m=(int)sqrt(n+0.5); v=inv(pow\_mod(a,m,n),n);

map<int,int> x; x[**1**]=**0**;

for(i=**1**;i<m;i++){

e=(e\*a)%n;

if(!x.count(e))x[e]=i;}

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

if(x.count(b))return (i\*m+x[b]);

b=(b\*v)%n;}

return **-1**;}

**Euler-Prime--zbh2047**

void calPrime(){

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

if (!minFactor[i]){

prime[primeNum++] = i;

minFactor[i] = primeNum;}

for (int j = **1**; j <= minFactor[i]; j++){

int t = i \* prime[j - **1**];

if (t >= MAXN)break;

minFactor[t] = j;}}}

void calPhi(){

phi[**1**] = **1**;

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

if (!minFactor[i]){

prime[primeNum++] = i;

minFactor[i] = primeNum; phi[i] = i - **1**;}

for (int j = **1**;; j++){

int t = i \* prime[j - **1**];

if (t >= MAXN)break;

minFactor[t] = j;

if (j == minFactor[i]){

phi[t] = phi[i] \* prime[j - **1**];break;}

phi[t] = phi[i] \* (prime[j - **1**] - **1**);}}}

**Miller-Rabin&&Pollard--zbh2047**

const UInt base1[] = { **2**, **7**, **61**, **0** };

const UInt base2[] = { **2**, **325**, **9375**, **28178**, **450775**, **9780504**, **1795265022**, **0** };

const UInt prime[] = { **2**, **3**, **5**, **7**, **11**, **13**, **17**, **19**, **23**, **29**, **31**, **37**, **41**, **43**, **47**, **53** };

template <typename T>

inline T add(T a, T b, T mod){

return a + b - (a + b >= mod ? mod : **0**);}

inline UInt mul(UInt a, UInt b, UInt mod){

return (ULL)a \* b % mod;}

ULL mul(ULL a, ULL b, ULL mod){

ULL ret = **0**;

for (ULL t = a; b; b >>= **1**){

if (b & **1**)ret = add(ret, t, mod);

t <<= **1**;

if (t >= mod)t -= mod; }

return ret;}

template <typename T>

T power(T a, T b, T mod){

T ret = **1**;

for (T t = a; b; b >>= **1**){

if (b & **1**)ret = mul(ret, t, mod);

t = mul(t, t, mod); }

return ret;}

//n为小于2^63的非1奇数，正确性100%

template <typename T>

bool millerRabin(T n){

int s = **0**; T r = n;

for (r--; !(r & **1**); r >>= **1**)s++;

for (const UInt \*base = typeid(T) == typeid(UInt) ? base1 : base2; \*base; base++){

T t = power(\*base % n, r, n);

if (t == **0** || t == **1** || t == n - **1**)continue;

for (int j = **1**; j < s; j++){

t = mul(t, t, n);

if (t == **1**)return false;

if (t == n - **1**)break;}

if (t != n - **1**)return false;}

return true;}

template <typename T>

bool checkPrime(T n){

if (n == **1**)return false;

for (int i = **0**; i < sizeof(prime) / sizeof(int); i++){

if (n % prime[i] == **0**)return n == prime[i];}

return millerRabin(n);}

template <typename T>

T gcd(T x, T y){ return y ? gcd(y, x % y) : x;}

template <typename T>

T pollard(T n){

if (millerRabin(n))return n;

while (**1**){

T x = rand() % n, y = x, c = rand() % (n - **1**) + **1**;

for (UInt i = **1**, j = **2**;; i++){

if (i == j){ j \*= **2**; y = x; }

x = add(mul(x, x, n), c, n);

T d = gcd(x - y + n, n);

if (d != **1**){

if (d != n)return d;

break;}}}}

ULL factor[**64**];

int factorNum;

void calFactorInternal(ULL n){

ULL d;

d = n >> **32** ? pollard(n) : pollard((UInt)n);

if (d == n){ factor[factorNum++] = d; return; }

calFactorInternal(d); calFactorInternal(n / d);}

void calFactor(ULL n){

factorNum = **0**;

for (int i = **0**; i < sizeof(prime) / sizeof(int); i++){

while (n % prime[i] == **0**){

n /= prime[i]; factor[factorNum++] = prime[i];}}

if (n != **1**)calFactorInternal(n);

sort(factor, factor + factorNum);}

**AC自动机--zbh2047**

void init(){

cnt = **1**;memset(trie, **0**, **2** \* sizeof(Trie));

trie[**0**].fail = **1**;}

inline int convert(char ch){ return ch - 'a'; }

void insert(char \*s){

int cur = **0**;

for (int i = **0**; s[i]; i++){

int &pos = trie[cur].next[convert(s[i])];

if (!pos){

pos = ++cnt;

memset(&trie[cnt], **0**, sizeof(Trie));}

cur = pos;}

trie[cur].num++;}

void makeFail(){

queue<int> q; q.push(**0**);

while (!q.empty()){

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

for (int i = **0**; i < LETTER; i++){

int &cur = trie[t].next[i];

if (cur){q.push(cur);

trie[cur].fail = trie[trie[t].fail].next[i];

trie[cur].match = trie[cur].num ? cur : trie[trie[cur].fail].match;}

else cur = trie[trie[t].fail].next[i];}}}

int search(char \*s){

int ret = **0**, cur = **0**;

for (int i = **0**; s[i]; i++){

cur = trie[cur].next[convert(s[i])];

for (int temp = trie[cur].match; temp; temp = trie[trie[temp].fail].match)

ret += trie[temp].num;}

return ret;}

**KMP--Const**

void get\_nxt(){

int n=strlen(target\_string);

nxt[**0**]=**0**;nxt[**1**]=**0**;

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

int j=nxt[i];

while(j&&target\_string[i]!=target\_string[j]) j=nxt[j];

nxt[i**+1**]=target\_string[i]==target\_string[j]?j**+1**:**0**;}}

int kmp(){

int n=strlen(origin\_string),m=strlen(target\_string);

int j=**0**,cnt=**0**;

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

while(j&&origin\_string[i]!=target\_string[j]) j=nxt[j];

if (origin\_string[i]==target\_string[j]) j++;

if (j==m) {cnt++;j=nxt[j];}}

return cnt;}

**LCS--zbh2047**

int maxCommonSubstring(char \*s1, char \*s2){

init();suffixAutomation(s1);

int match = **0**, ret = **0**, cur = **0**;

for (int i = **0**; s2[i]; i++){

char c = convert(s2[i]);

if (!st[cur].next[c]){

while (cur != **-1** && !st[cur].next[c])

cur = st[cur].link;

if (cur == **-1**){ match = cur = **0**; continue; }

match = st[cur].len;}

cur = st[cur].next[c]; ret = max(ret, ++match);}

return ret;}

**后缀数组--zbh2047**

int r[**2**][MAXN];

int tmp[MAXN], sa[MAXN];//存储第i小的字符位置

int \*rk;//存储位置i上的字符是第几小的

int suffixArray(unsigned char \*s){

int len = **0**, m = **0**;

int \*r1 = r[**0**], \*r2 = r[**1**];

memset(bucket, **0**, sizeof(int)\*CHAR);

while (bucket[s[len]]++, s[len++]);

for (int i = **1**; i < CHAR; i++) bucket[i] += bucket[i - **1**];

for (int i = len - **1**; i >= **0**; i--) sa[--bucket[s[i]]] = i;

for (int i = **0**, k = **0**; i < CHAR; i++){

if (bucket[i]>k){ k = bucket[i]; m++; }

bucket[i] = m;}

for (int i = **0**; i < len; i++) r1[i] = bucket[s[i]];

for (int i = **1**; m < len; i \*= **2**){

for (int j = **0**; j < i; j++) tmp[j] = len - i + j;

for (int j = i, k = **0**; j < len; k++){

if (sa[k] >= i)tmp[j++] = sa[k] - i;}

memset(bucket, **0**, sizeof(int)\*m);

for (int j = **0**; j < len; j++) bucket[r1[j]]++;

for (int j = **1**; j < m; j++) bucket[j] += bucket[j - **1**];

for (int j = len - **1**; j >= **0**; j--)

sa[--bucket[r1[tmp[j]]]] = tmp[j];

r2[sa[**0**]] = **0**; m = **0**;

for (int j = **1**; j < len; j++){

if (r1[sa[j]] != r1[sa[j - **1**]] || r1[sa[j] + i] != r1[sa[j - **1**] + i])m++;

r2[sa[j]] = m;}

int \*t = r1; r1 = r2; r2 = t;

m++;}

rk = r1;

return len;}

int h[MAXN];

int getH(unsigned char \*s){

int len = suffixArray(s) - **1**;

for (int i = **0**, k = **0**; i < len; i++){

int j = sa[rk[i] - **1**];

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

h[rk[i] - **1**] = k;

if (k)k--;}

return len;}

**后缀树--zbh2047**

void sortState()//后缀树层序遍历{

int size = st[last].len;

memset(bucket, **0**, sizeof(int)\*(size + **1**));

for (int i = **0**; i <= cnt; i++) bucket[st[i].len]++;

for (int i = **1**; i <= size; i++) bucket[i] += bucket[i - **1**];

for (int i = cnt; i; i--) order[--bucket[st[i].len]] = i;}

void suffixTree(char \*s){

int len = strlen(s);

init();

memset(leave, **0**, len \* sizeof(bool) \* **2**);

for (int i = len - **1**; i >= **0**; i--){

add(s[i]); id[i] = last;

leave[last] = true;}

for (int i = **0**; i <= cnt; i++) tree[i].clear();

for (int i = cnt; i; i--) tree[st[i].link].push\_back(i);}

**后缀自动机--zbh2047**

void init(){

last = cnt = **0**;

st[cnt].len = **0**; st[cnt].link = **-1**;

memset(st[**0**].next, **0**, sizeof(st[**0**].next));}

inline int convert(char ch){ return ch - 'a'; }

void add(char c){

c = convert(c);

int cur = ++cnt, i;

st[cur].len = st[last].len + **1**;

memset(st[cur].next, **0**, sizeof(st[cur].next));

for (i = last; i != **-1** && !st[i].next[c]; i = st[i].link)

st[i].next[c] = cur;

if (i == **-1**)st[cur].link = **0**;

else{int j = st[i].next[c];

if (st[i].len + **1** == st[j].len)

st[cur].link = j;

else{int copy = ++cnt;

st[copy].len = st[i].len + **1**;

memcpy(st[copy].next, st[j].next, sizeof(st[j].next));

st[copy].link = st[j].link;

for (; i != **-1** && st[i].next[c] == j; i = st[i].link)

st[i].next[c] = copy;

st[j].link = st[cur].link = copy;}}

last = cur;}

void suffixAutomation(char \*s){

init();

for (int i = **0**; s[i]; i++)add(s[i]);}

void suffixTree(char \*s){

init();

for (int i = strlen(s) - **1**; i >= **0**; i--){

add(s[i]); tree[i] = last;}}

**manachur--igoodvegetablea**

for(i=**1**;s2[i]!='\0';i++){

p[i]=mx>i?min(p[**2**\*id-i],mx-i):**1**;

while(s2[i+p[i]]==s2[i-p[i]])p[i]++;

if(i+p[i]>mx){ mx=i+p[i];id=i;}}

**回文树--Const**

void init(){

memset(mem,**0**,sizeof(mem));

headf=mem;last=heads=mem**+1**;

headf->fail=heads;heads->len=**-1**;

tot=**1**;now=**0**;}

void add(int x,int p){

node\* cur=last;

for (;s[p-cur->len**-1**]!=s[p];cur=cur->fail);

if (!cur->next[x]){

node\* ths=&mem[++tot];

last=cur->next[x]=ths; ths->len=cur->len**+2**;

if (cur==heads) ths->fail=headf;

else{

for (cur=cur->fail;s[p-cur->len**-1**]!=s[p];cur=cur->fail);

ths->fail=cur->next[x];}

ths->sum=ths->fail->sum**+1**;}

else last=cur->next[x];}

**ISAP--Const**

void bfs(){

queue<int> q;

while(!q.empty()) q.pop();

for (int i=**1**;i<=n;i++) d[i]=maxn**-1**; //由初始下标决定01

d[T]=**0**;q.push(T);

while(!q.empty()){

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

for (edge\* it=head[u];it;it=it->next){

edge \*now=rever(it);

if (now->v==**0**||d[now->s]<n) continue;

d[now->s]=d[u]**+1**;

q.push(now->s);}}

memset(numbs,**0**,sizeof(numbs));

for (int i=**1**;i<=n;i++) numbs[d[i]]++;}//由初始下标决定01

int isap(){

int flow=**0**;for (int i=**1**;i<=n;i++) cur[i]=head[i];//由初始下标决定01

int u=S;

while(d[S]<n){

if (u==T) {

int augflow=**2147483647**;

for (int i=S;i!=T;i=cur[i]->t)

augflow=min(augflow,cur[i]->v);

for (int i=S;i!=T;i=cur[i]->t){

cur[i]->v-=augflow;

rever(cur[i])->v+=augflow;}

flow+=augflow;u=S;}

edge \*e;

for (e=cur[u];e;e=e->next)

if (e->v&&d[u]==(d[e->t]**+1**)) break;

if (e){

cur[u]=e; revpath[e->t]=rever(e);

u=e->t;}

else{

numbs[d[u]]--;

if (numbs[d[u]]==**0**) break;

cur[u]=head[u];

int mindist=n;

for (edge\* it=head[u];it;it=it->next)

if (it->v) mindist=min(mindist,d[it->t]);

d[u]=mindist**+1**; numbs[d[u]]++;

if (u!=S) u=revpath[u]->t;}}

return flow;}

**DINIC--Const**

bool bfs(){

for (int i=**0**;i<=n;i++) dis[i]=INF;

q.push(S);dis[S]=**0**;

while(!q.empty()){

for (edge \*it=head[q.front()];it;it=it->next)

if (it->v&&dis[q.front()]+it->c<dis[it->t]){

dis[it->t]=dis[q.front()]+it->c;

prev[it->t]=it; q.push(it->t);}

q.pop();}

return (dis[T]!=INF);}

int cost=**0**;

int dinic(){

int flow=**0**;

while(bfs()){

int augflow=INF,tmpcost=**0**;

for (edge\* it=prev[T];it;it=prev[it->s]){

augflow=min(augflow,it->v);

tmpcost+=it->c;}

for (edge\* it=prev[T];it;it=prev[it->s]){

it->v-=augflow;

rever(it)->v+=augflow;}

flow+=augflow;cost+=augflow\*tmpcost;}

return flow;}

**K度限制生成树--igoodvegetablea**

(**1**)先求出最小m度限制生成树:

原图中去掉和V0相连的所有边(可以先存两个图,建议一个邻接矩阵,一个邻接表,用方便枚举边的邻接表来构造新图);

得到m个连通分量,则这m个连通分量必须通过v0来连接;

则在图G的所有生成树中dT(v0)>=m;

则当k<m时,问题无解;

对每个连通分量求一次最小生成树;

对于每个连通分量V’,用一条与V0直接连接的最小的边把它与V0点连接起来,使其整体成为一个生成树;

就得到了一个m度限制生成树,即为最小m度限制生成树;

(**2**)由最小m度限制生成树得到最小m**+1**度限制生成树;

连接和V0相邻的点v,则可以知道一定会有一个环出现(因为原来是一个生成树);

只要找到这个环上的最大权边(不能与v0点直接相连)并删除,就可以得到一个m**+1**度限制生成树;

枚举所有和V0相邻点v,找到替换后,增加权值最小的一次替换(如果找不到这样的边,就说明已经求出);

就可以求得m**+1**度限制生成树;

如果每添加一条边,都需要对环上的边一一枚举,时间复杂度将比较高;

用动态规划解决;

设dp(v)为路径v0—v上与v0无关联且权值最大的边;

定义father(v)为v的父结点,由此可以得到状态转移方程:

dp(v)=max(dp(father(v)),ω(father(v),v));

边界条件为dp[v0]=-∞(因为每次寻找的是最大边,所以-∞不会被考虑),dp[v’]=-∞|(v0,v’)∈E(T);

(**3**)当dT(v0)=k时停止(即当V0的度为k的时候停止),但不一定k的时候最优;

算法实现：

并查集+kruskal;

首先,每个连通分量的的最小生成树可以直接用一个循环,循环着Kruskal求出;

这里利用了联通分量间的独立性,对每个连通分量分别求最小生成树,和放在一起求,毫不影响;

而且kruskral算法保证了各连通分量边的有序性;

找最小边的时候,可以用动态规划,也可以这么做：

先走一个循环,但我们需要逆过来加边,将与v0关联的所有边从小到达排序;

然后将各连通分量连接起来,利用并查集可以保证每个连通分量只有一条边与v0相连;

由于边已经从小到达排序,故与每个连通分量相连的边就是每个连通分量与v0相连中的最小边;

然后求m**+1**度的最小生成树时,可以直接用DFS,最小生成树要一直求到k度,然后从中找出一个最优值;

**次小生成树--igoodvegetablea**

for(i=**1**;i<=m;i++){

scan3(j,k,l);

cost[j][k]=cost[k][j]=l;}

vis[**1**]=true;a[k=**1**]=**1**;

for(i=**2**;i<=n;i++){maxd[i][**1**]=maxd[**1**][i]=lowcost[i]=cost[**1**][i];fat[i]=**1**;}

for(i=**1**;i<=n;i++)maxd[i][i]=cost[i][i]=**0**;

for(u=**1**,i=**1**;i<=n**-1**;i++){

mini=inf,v=**-1**;

for(j=**1**;j<=n;j++)

if(!vis[j] && lowcost[j]<mini)

{mini=lowcost[j];v=j;}

vis[v]=true; ans+=mini;

for(j=**1**;j<=k;j++)

maxd[a[j]][v]=maxd[v][a[j]]=max(mini,maxd[fat[v]][a[j]]);

a[++k]=v;

for(j=**1**;j<=n;j++)

if(!vis[j] && cost[v][j]<lowcost[j])

{lowcost[j]=cost[v][j];fat[j]=v;}}

mini=inf;

for(i=**1**;i<=n**-1**;i++)

for(j=i**+1**;j<=n;j++){

if(fat[i]==j || fat[j]==i || cost[i][j]==inf)continue;

mini=min(mini,cost[i][j]-maxd[i][j]);}

if(mini==**0**)printf("Not Unique!\n");else printf("%d\n",ans);

**最小树形图--igoodvegetablea**

type ZLEdmonds(int n,int m,int root){//自环在输入建图时直接忽略，如需加入，可另存

type tot=0.0;

//判断是否有树

while(true){

for(int i=**1**;i<=n;i++)in[i]=inf;

for(int i=**1**;i<=m;i++){

int u=edg[i].from;

int v=edg[i].to;

if(edg[i].cost<in[v] && u!=v){pre[v]=u;in[v]=edg[i].cost;}}

for(int i=**1**;i<=n;i++)if(i!=root && in[i]==inf)return **-1**;

//找环

int cnt=**1**;memset(id,**0**,sizeof(id));

memset(vis,**0**,sizeof(vis)); in[root]=**0**;

for(int i=**1**;i<=n;i++){//标记每个环

tot+=in[i];

int v=i;

while(vis[v]!=i && id[v]==**0** && v!=root)

{vis[v]=i;v=pre[v];}

if(v!=root && id[v]==**0**){//缩点

for(int u=pre[v];u!=v;u=pre[u])id[u]=cnt;

id[v]=cnt++;}}

if(cnt==**1**)break;

for(int i=**1**;i<=n;i++)if(id[i]==**0**)id[i]=cnt++;

//建立新图

for(int i=**1**;i<=m;i++){

int u=edg[i].from,v=edg[i].to;

edg[i].from=id[u]; edg[i].to=id[v];

if(id[u]!=id[v])edg[i].cost-=in[v];}

n=cnt**-1**; root=id[root];}

return tot;}

**欧拉回路--igoodvegeteblea**

void dfs(int x){

stac[++sta]=x;

for(unsigned int i=**0**;i<g[x].size();i++){

if(used[g[x][i]])continue;

used[g[x][i]]=true; dfs(other(g[x][i],x));

break;}}

void Fleury(int x){

sta=**1**;stac[sta]=x;

while(sta>=**1**){

x=stac[sta];

bool f=false;

for(unsigned int i=**0**;i<g[x].size();i++){

if(!used[g[x][i]]){f=true;break;}}

if(!f)printf("%d ",stac[sta--]);else {

sta--; dfs(stac[sta**+1**]);}}}

**最大团—igoodvegetableabool**

void dfs( int u, int pos ){

int i, j;

for( i = u**+1**; i <= n; i++){

if( cnt[i]+pos <= ans ) return **0**;

if( a[u][i] ){

// 与目前团中元素比较，取 Non-N(i)

for( j = **0**; j < pos; j++ ) if( !a[i][ vis[j] ] ) break;

if( j == pos ){ // 若为空，则皆与 i 相邻，则此时将i加入到 最大团中

vis[pos] = i;if( dfs( i, pos**+1** ) ) return **1**;}}}

if( pos > ans ){

for( i = **0**; i < pos; i++ )

group[i] = vis[i]; // 最大团 元素

ans = pos; return **1**;}

return **0**;}

void maxclique(){

ans=**-1**;

for(int i=n;i>**0**;i--){vis[**0**]=i; dfs(i,**1**); cnt[i]=ans;}}

**01分数规划-igoodvegetablea**

while(true){

ans=l;for(i=**1**;i<=n;i++)d[i]=hei(a[i]-ans\*b[i],i);

sort(d**+1**,d+n**+1**);

double fz,fm;

fz=fm=0.0;

for(i=**1**;i<=m;i++){fz+=a[d[i].pos]; fm+=b[d[i].pos];}

l=fz/fm;

if(fabs(ans-l)<eps)break;}

**K短路--igoodvegetablea**

struct node{

int v,c;

node(int v,int c):v(v),c(c){}

inline bool operator<(const node &b) const{//用于优先队列先出的条件

return c+dis[v]>b.c+dis[b.v];}};

vector<node> map1[MAXN];//用于dijkstra算法

vector<node> map2[MAXN];//用于A\_star算法

void dijkstra(){

int i,find[MAXN],v;

for(i=**1**;i<=n;i++)dis[i]=INF;

memset(find,**0**,sizeof(find));

priority\_queue<node> heap;

dis[t]=**0**;heap.push(node(t,**0**));

while(!heap.empty()){

v=heap.top().v; heap.pop();

if(find[v])continue;

find[v]=**1**;

for(i=**0**;i<map1[v].size();i++)

if(!find[map1[v][i].v] && dis[v]+map1[v][i].c<dis[map1[v][i].v]){

dis[map1[v][i].v]=dis[v]+map1[v][i].c;

heap.push(node(map1[v][i].v,dis[map1[v][i].v]));}}}

int A\_star(){

int i,cnt[MAXN],v,g;

if(dis[s]==INF)return **-1**;

priority\_queue<node> heap;

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

heap.push(node(s,**0**));//0是g（x）

while(!heap.empty()){

v=heap.top().v;g=heap.top().c;

heap.pop();cnt[v]++;

if(cnt[t]==k)return g;

if(cnt[v]>k)continue;

for(i=**0**;i<map2[v].size();i++)

heap.push(node(map2[v][i].v,g+map2[v][i].c));}

return **-1**;}

for(i=**0**;i<m;i++){//存储方式

cin>>u>>v>>c;

map2[u].push\_back(node(v,c));

map1[v].push\_back(node(u,c));}//反向储存求各节点到目标节点的最短距离

**Tarjan--igoodvegetablea**

int dfs(int u,int fat){

int lowu,lowv;

lowu=pre[u]=++dfs\_clock;int child=**0**;

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

int v=g[u][i];

if(!pre[v]){

child++;lowv=dfs(v,u);lowu=min(lowu,lowv);

if(lowv>pre[u])p.push(edge(min(u,v),max(u,v)));

if(lowv>=pre[u])iscut[u]=true;

}else if(v!=fat) lowu=min(lowu,pre[v]);}

if(fat==**-1** && child<=**1**)iscut[u]=false;

return lowu;}

**点-双连通分量--igoodvegetablea**

int dfs(int u,int fa){

int lowu=pre[u]=++dfs\_clock;

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

int side=g[u][i];int v=other(side,u);

if(!pre[v]){

s[++s[**0**]]=side;int lowv=dfs(v,u);

lowu=min(lowu,lowv);

if(lowv>=pre[u]){

bcc\_cnt++;

for(;;){

int x=s[s[**0**]];s[**0**]--;bcc[x]=bcc\_cnt;

p[bcc\_cnt]=min(p[bcc\_cnt],x);if(x==side)break;}}

}else if(pre[v]<pre[u] && v!=fa){

s[++s[**0**]]=side;lowu=min(lowu,pre[v]);}}

return lowu;}

**SCC--igoodvegetablea**

void dfs(int u){

pre[u]=low[u]=++dfs\_clock;s.push(u);

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

int v=g[u][i];

if(!pre[v]){

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

}else if(!sccno[v])low[u]=min(low[u],pre[v]);}

if(low[u]==pre[u]){scc\_cnt++;

for(;;){

int x=s.top();s.pop();

sccno[x]=scc\_cnt;if(x==u)break;}}}

**大整数**

class Number :public vector<UInt>{

bool flag;

Number(UInt value){ flag = 0; if (value)push\_back(value); }

static ULL link(ULL x, UInt y){ return (x << 31) | y; }

Number(){ flag = 0; }};

Number::Number(const char \*s){

if (s[0] == '-'){ flag = 1; s++; }

else flag = 0;

vector<char> str(strlen(s));

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

str[i] = s[str.size() - i - 1] - '0';

while (str.size()){

ULL sum = 0;

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

sum = sum \* 10 + str[i];

str[i] = (char)(sum >> 31);

sum &= ~(1 << 31);}

push\_back((UInt)sum);

while (str.size() && !str.back())str.pop\_back();}

if (!back())pop\_back();}

Number::Number(int value){

if (value)push\_back(value > 0 ? value : -value);

flag = value < 0;}

Number::Number(long long value){

flag = value < 0;

if (flag)value = -value;

while (value){

push\_back(value & ~(1 << 31));value >>= 31;}}

void Number::convert10(char \*s){

if (flag)\*s++ = '-';

vector<UInt> copy = \*this; UInt len = 0;

while (copy.size()){

ULL sum = 0;

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

sum = link(sum, copy[i]);

copy[i] = (UInt)(sum / 10);sum %= 10;}

s[len++] = (char)sum + '0';

if (!copy.back())copy.pop\_back(); }

if (len == 0)s[len++] = '0';

reverse(s, s + len);s[len] = 0;}

int Number::cmp(const Number& num)const{

if (size() != num.size())return size() < num.size() ? -1 : 1;

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

if ((\*this)[i] != num[i])return (\*this)[i] < num[i] ? -1 : 1;}

return 0;}

bool Number::operator == (const Number& num)const{

return flag == num.flag && !cmp(num);}

bool Number::operator < (const Number& num)const{

if (flag != num.flag)return flag;

return flag ? cmp(num) > 0 : cmp(num) < 0;}

bool Number::operator <= (const Number& num)const{

if (flag != num.flag)return flag;

return flag ? cmp(num) >= 0 : cmp(num) <= 0;}

//为提高效率确保\*this位数>=num位数

void Number::add(const Number& num){

UInt f = 0, i = 0;

if (size() < num.size())resize(num.size());

for (; i < num.size(); i++){

(\*this)[i] += num[i] + f; f = (\*this)[i] >> 31;

if (f)(\*this)[i] ^= 1 << 31;}

push\_back(0);

for (; f; i++){

f = ++(\*this)[i] >> 31;

if (f)(\*this)[i] ^= 1 << 31;}

if (!back())pop\_back();}

//确保\*this>=num

void Number::sub(const Number& num){

UInt f = 0, i = 0;

for (; i < num.size(); i++){

(\*this)[i] -= num[i] + f; f = (\*this)[i] >> 31;

if (f)(\*this)[i] ^= 1 << 31;}

for (; f; i++){

f = --(\*this)[i] >> 31;

if (f)(\*this)[i] ^= 1 << 31;}

while (size() && !back())pop\_back();}

Number Number::operator + (const Number& num)const{

Number ret;

if (flag == num.flag){

if (size() < num.size()){ ret = num; ret.add(\*this); }

else{ ret = \*this; ret.add(num); }

ret.flag = flag;}

else{

int t = cmp(num);

if (t < 0){

ret = num; ret.sub(\*this);

ret.flag = num.flag;}

else if (t > 0){

ret = \*this; ret.sub(num);

ret.flag = flag;}}

return ret;}

Number Number::operator - (const Number& num)const{

Number ret;

if (flag != num.flag){

if (size() < num.size()){ ret = num; ret.add(\*this); }

else{ ret = \*this; ret.add(num); }

ret.flag = flag;}

else{

int t = cmp(num);

if (t < 0){

ret = num; ret.sub(\*this);

ret.flag = !flag;}

else if (t > 0){

ret = \*this; ret.sub(num);

ret.flag = flag;}}

return ret;}

//为提高效率确保\*this位数>=num位数

Number Number::mul(const Number& num)const{

if (num.empty() || empty())return 0;

Number ret;

ret.resize(size() + num.size(), 0);

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

ULL sum = 0;

for (UInt j = 0; j < size(); j++){

sum += (ULL)num[i] \* (\*this)[j];

ret[i + j] += sum & ~(1 << 31); sum >>= 31;

if (ret[i + j] & (1 << 31)) { sum++; ret[i + j] ^= 1 << 31; }}

ret[i + size()] += (UInt)sum;}

for (UInt i = size(); i < ret.size(); i++){

if (ret[i] & (1 << 31)){ret[i] ^= 1 << 31;ret[i + 1]++;}}

if (!ret.back())ret.pop\_back();

return ret;}

Number Number::operator \* (const Number& num)const{

Number ret = size() < num.size() ? num.mul(\*this) : mul(num);

if (ret.size())ret.flag = flag ^ num.flag;

return ret;}

Number Number::div(const Number& num, Number& mod)const{

const UInt aSize = size(), bSize = num.size();

if (aSize < bSize){ mod = \*this; return 0; }

Number ret;

ret.resize(aSize - bSize + 1);

mod.assign(begin() + aSize - bSize + 1, end());

ULL y = num.back();

int bit = 0;

for (int i = 16; i; i >>= 1){

if (y >> (bit + i))bit += i;}

y = (y << (31 - bit)) + (num[bSize - 2] >> bit) + 1;

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

mod.shl31(1);

UInt oldSize = mod.size(); mod.resize(bSize + 1);

mod[0] = (\*this)[i];

ULL x = link(mod[bSize], mod[bSize - 1]);

x = (x << (31 - bit)) | (mod[bSize - 2] >> bit);

if (!oldSize && mod[0])oldSize++;

mod.resize(oldSize);

if (ret[i] = (UInt)(x / y))mod.sub(num.mul(ret[i]));

if (mod.cmp(num) >= 0) { mod.sub(num); ret[i]++; }}

if (!ret.back())ret.pop\_back();

return ret;}

Number Number::divInt(UInt num, UInt& mod)const{

Number ret;

ret.resize(size()); ULL sum = 0;

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

sum = link(sum, (\*this)[i]);

ret[i] = (UInt)(sum / num); sum %= num; }

if (ret.size() && !ret.back())ret.pop\_back();

mod = (UInt)sum;return ret;}

Number Number::operator / (const Number& num)const{

UInt t;

Number ret = num.size() == 1 ? divInt(num[0], t) : div(num, Number());

if (ret.size())ret.flag = flag ^ num.flag;

return ret;}

Number Number::operator % (const Number& num)const{

Number ret;

if (num.size() == 1){

UInt t;

divInt(num[0], t);ret = t; }

else div(num, ret);

if (ret.size())ret.flag = flag;

return ret;}

void Number::shl31(UInt num){

if (empty())return;

UInt oldSize = size();

resize(oldSize + num);

for (int i = oldSize - 1; i >= 0; i--)(\*this)[i + num] = (\*this)[i];

for (int i = num - 1; i >= 0; i--)(\*this)[i] = 0;}

**卡特兰数**

求n个结点能构成不同二叉数的数目。

凸n边形三角剖分数

在圆上选择2n个点，将这些点成对连接起来使得所得到的n条线段不相交的方法数。

有n个左括号和n个右括号排列，试求两两匹配的方案数。

N个不同元素按一定的顺序入栈，求不同的出栈序列数目

对于一个n\*n的正方形网格，每次我们能向右或者向上移动一格，那么从左下角到右上角的所有在副对角线右下方的路径总数

2n个人排队买票，票价为50元，其中n个人各手持一张50元钞票，n个人各手持一张100元钞票，除此之外大家身上没有任何其他的钱币，并且初始时候售票窗口没有钱，问有多少种排队的情况数能够让大家都买到票。

答案：Catakan(n)\*A(n,n)^2

**Polya定理**

涉及到组合数学的问题，首先是群的概念：设G是一个集合，\*是G上的二元运算，如果(G,\*)满足下面的条件：

封闭性：对于任何a,b∈G,有a\*b∈G;结合律：对任何a,b,c∈G有(a\*b)\*c=a\*(b\*c);单位元：存在e∈G,使得对所有的a∈G,都有a\*e=e\*a=a;逆元：对于每个元素a∈G,存在x∈G,使得a\*x=x\*a=e,这个时候记x为a-1，称为a的逆元，那么则称(G,\*)为一个群。

例：G={0,1,2,3,4....n-1}那么它在mod n加法下是一个群。

群元素的个数有限，称为有限群，且其中元素的个数称为阶，记为|G|,群元素的个数无限，称为无限群。

若对于群元素中的任意两个元素a,b都有ab=ba那么称G为交换群，简称Abel群。

置换：设X为一个有限集，π是X到X的一个--变换，那么称π是X上的一个置换。

例：设X={1,2,3,4....n},设π是X的一个变换，满足π：1->a1,2->a2,......n->an,其中a1,a2...an是X的一个排列，则称π是X上的一个置换。

可将π记为   1     2   ......   n

            a1   a2   ......a n

同一置换用这样的表示法有n!种，但其对应的关系不变。

假设循环π只这样一个置换，满足π：a1->a2,a2->a3,....ak->a1,但是对于其他元素保持不变，即：a->a,

可将π记为   a1     a2   ......   ak

            a2   a3   ......  a1

称为k阶循环，K为循环长度。

每个置换都可以写成若干个互不相交的循环的乘积，且表示是唯一的.

如   1   2  3   4  5  6

       2   4   5  1  3  6    ，则可以表示为(124)(35)(6),置换的循环节数是上面的循环个数，上面的例题的循环节数为3.

定义：设G是有限集X上的置换群，点a,b∈X称为"等价"的，当且仅当，存在π∈G使得π(a)=b，记为a~b，这种等价条件下，X的元素形成的等价类称为G的轨道，它是集X的一个子集，G的任意两个不同的轨道之交是空集，所以置换群G的轨道全体是集合X的一个划分，构成若干个等价类，等价类的个数记为L。

**Zk (K不动置换类)：**设G是1…n的置换群。若K是1…n中某个元素，G中使K保持不变的置换的全体，记以Zk，叫做G中使K保持不动的置换类，简称K不动置换类。

**Ek(等价类)：**设G是1…n的置换群。若K是1…n中某个元素，K在G作用下的轨迹，记作Ek。即K在G的作用下所能变化成的所有元素的集合。.

这个时候有：|**Ek**|\*|**Zk**|=|G|成立(k=1,2,.....n)。

C(π)：对于一个置换π∈G,及a∈X，若π(a)=a，则称a为π的不动点。π的不动点的全体记为C(π)。例如π=(123)(3)(45)(6)(7)，X={1,2,3,4,5,6,7};那么C(π)={3,6,7}共3个元素。

**Burnside引理**：L=1/|G|\*(**Z1**+**Z2**+**Z3**+**Z4**+**......Zk**)=1/|G|\*(C(π1)+C(π2)+C(π3)+.....+C(πn))(其中k∈X,π∈G)。

**Polya定理**：设G={π1，π2，π3........πn}是X={a1，a2，a3.......an}上一个置换群，用m中颜色对X中的元素进行涂色，那么不同的涂色方案数为：1/|G|\*(mC(π1)+mC(π2)+mC(π3)+...+mC(πk)). 其中C(πk)为置换πk的循环节的个数。

polya定理求循环节个数代码模板：

const int MAX=1001;

#define CLR(arr,val) memset(arr,val,sizeof(arr))

int n,perm[MAX],visit[MAX];//sum求循环节个数,Perm用来存储置换,即一个排列

int gcd(int n,int m){return m==0?n:gcd(m,n%m);}

void Polya(){

int pos,sum=0; CLR(visit,0);

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

if(!visit[i]){

sum++;pos=i;

for(int j=0;!visit[perm[pos]];j++){

pos=perm[pos]; visit[pos]=1; }}

return sum; }

一般可以证明：当只有旋转的时候(顺时针或逆时针)，对于一个有n个字符的环，可顺时针或逆时针旋转几个位置，由于至少有n个置换，但是假设我顺时针旋转k个位置，他就等同于逆时针转动n-k个位置，假设一个置换为:G={π0，π1，π2，π3，π4，...，πn-1}，这个时候可以证明逆时针旋转k个位置时πk的循环节的个数为Gcd(n,k)，且每个循环的长度为L=n/gcd(n,i)。

例题1：[NYOJ 280(LK的项链)](http://acm.nyist.net/JudgeOnline/problem.php?pid=280)，LK的男朋友送给LK一盒有红、蓝、绿三种颜色的珠子，每种颜色珠子的个数都大于24，现在LK想用这一盒珠子穿出一条项链，项链上的珠子个数为n（0<=n<=24）,请你帮她计算一下一共可以用这一盒珠子可以穿出多少条不同的项链。通过旋转、翻转达到同一种状态的被认为是相同的项链。

涉及到旋转和翻转，上面已经说了旋转的情况，下面说下翻转的规律。

当n为奇数的时候，这个时候只有一种形式，假设经过某个顶点i与中心的连线为轴的翻转πi，共有n个，置换πi的形式如下，i保持不变：

πi：i->i，i+1->i-1，i+2->i-2，i+3->i-3.................i+n-1->(i-(n-1)+n)%n。

这个时候由对称性知，加上顶点i共有n个循环节数为(n+1)/2的循环群 。

当n为偶数时，有两种形式：

(1)、经过某个顶点与中心的连线为轴的翻转，有n/2个，这个时候和第一种为奇数的时候一样。

(2)、以顶点i和i+1的中点与中心的连线为轴翻转，共有n/2个：

πi:i->i+1，i-1->i+2，i-2->i+3，.................(i-j+n)%n->(i+j+1)%n。

这个时候共有n/2个循环节数(n+2)/2的循环群，和n/2个循环节数n/2的循环群。要特别注意0的情况，输出0即可。且由于对于输入不同的num均有2\*num中置换，所以结果应该是/(2\*num)。

**计算几何**

#define EPS 1e-10

#define INF 1e10

inline bool EQUAL(double t1, double t2){return t1 - t2 < EPS && t1 - t2 > -EPS;}

inline bool LESS(double t1, double t2){ return t1 <= t2 - EPS;}

inline bool LESS\_EQUAL(double t1, double t2){ return t1 < t2 + EPS;}

inline int SGN(double t){ return LESS(t, 0) ? -1 : LESS(0, t) ? 1 : 0;}

class Point{

double x, y;

Point(){}

Point(double x, double y) :x(x), y(y){}

bool operator == (const Point& p)const{ return EQUAL(x, p.x) && EQUAL(y, p.y);}

bool operator < (const Point& p)const{

return LESS\_EQUAL(x, p.x) && (LESS(x, p.x) || LESS(y, p.y));}

Point operator + (const Point& p)const{return Point(x + p.x, y + p.y);}

Point operator - (const Point& p)const{return Point(x - p.x, y - p.y);}

double operator \* (const Point& p)const{return x\*p.y - y\*p.x;}

Point operator \* (double value)const{return Point(x\*value, y\*value);}

Point operator / (double value)const{return Point(x / value, y / value);}

double dot(const Point& p)const{return x\*p.x + y\*p.y; }

double r2()const{ return x\*x + y\*y; }

double r()const{ return sqrt(r2()); }

double dis2(const Point& p)const{ return (\*this - p).r2();}

double dis(const Point& p)const{return (\*this - p).r();}

bool onLine(const Point& p1, const Point& p2)const{

return EQUAL((\*this - p1)\*(\*this - p2), 0);}

bool onLineSeg(const Point& p1, const Point& p2)const{

return onLine(p1, p2) && inRect(p1, p2);} //include extream points

double lineRelation(const Point& p1, const Point& p2)const{

Point t = p2 - p1;

return t.dot(\*this - p1) / t.r2();}

//ret 0, \*this=p1; ret 1,\*this=p2;ret (0,1), \*this is interior to p1p2

Point footPoint(const Point& p1, const Point& p2)const{

double r = lineRelation(p1, p2);

return p1 + (p2 - p1)\*r;}

double lineDis(const Point& p1, const Point& p2)const{

return abs((p1 - \*this)\*(p2 - \*this)) / p1.dis(p2);}

Point mirror(Point& p1, Point& p2){

Point foot = footPoint(p1, p2);return foot \* 2 - \*this;}

Point rotate(double angle)const{

Point f(sin(angle), cos(angle));

return Point(\*this \* f, dot(f));}

Point rotate90()const{return Point(-y, x);}

double cosAngle(const Point& p1, const Point& p2)const{

Point t1 = \*this - p1, t2 = \*this - p2;

return t1.dot(t2) / sqrt(t1.r2()\*t2.r2());}

double sinAngle(const Point& p1, const Point& p2)const{

Point t1 = \*this - p1, t2 = \*this - p2;

return t1 \* t2 / sqrt(t1.r2()\*t2.r2());}

double tanAngle(const Point& o = Point(0, 0))const{

if (EQUAL(x, o.x))return y - o.y >= 0 ? INF : -INF;

return (y - o.y) / (x - o.x);}

double angle(const Point& p1, const Point& p2)const{

return acos(cosAngle(p1, p2));}

double angle(const Point& o = Point(0, 0))const{

return atan2(y - o.y, x - o.x);}

//left return 1, right return -1, on line return 0.

int direction(const Point& p1, const Point& p2)const{

return SGN(x\*(p1.y - p2.y) + p1.x\*(p2.y - y) + p2.x\*(y - p1.y));}

bool inRect(const Point& p1, const Point& p2)const{

return LESS\_EQUAL((p1.x - x)\*(p2.x - x), 0)

&& LESS\_EQUAL((p1.y - y)\*(p2.y - y), 0);}

int inPolygon(const Point\* p, int n)const;

int inConvex(const Point\* p, int n)const;

int inCircle(const Point& o, double r)const{

double dist = dis2(o);return SGN(r\*r - dist);}};

double Point::lineSegDis(const Point& p1, const Point& p2, Point& ret)const{

double r = lineRelation(p1, p2);

if (LESS\_EQUAL(r, 0))ret = p1;

else if (LESS\_EQUAL(1, r))ret = p2;

else ret = footPoint(p1, p2);

return dis(ret);}

//input lineNum+1 points

double Point::lineSegArrayDis(const Point\* p, int lineNum, Point& ret)const{

Point tp;double td, mind = INF;

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

td = lineSegDis(p[i], p[i + 1], tp);

if (LESS(td, mind)){mind = td; ret = tp;}}

return mind;}

//donnot include extream points, and donnot include coincidence.

inline bool lineSegLineSegIntersect(const Point& p1, const Point& p2, const Point& q1, const Point& q2){

Point pq1 = p1 - q1, p12 = p2 - p1, q12 = q2 - q1;

return SGN(pq1\*q12)\*SGN((p2 - q1)\*q12)<0&&SGN(pq1\*p12)\*SGN((p1 - q2)\*p12)<0;}

//include extream points and coincidence.

inline bool lineSegLineSegIntersect2(const Point& p1, const Point& p2, const Point& q1, const Point& q2){

if (!(LESS\_EQUAL(min(q1.x, q2.x), max(p1.x, p2.x))

&& LESS\_EQUAL(min(p1.x, p2.x), max(q1.x, q2.x))

&& LESS\_EQUAL(min(q1.y, q2.y), max(p1.y, p2.y))

&& LESS\_EQUAL(min(p1.y, p2.y), max(q1.y, q2.y))))return false;

Point pq1 = p1 - q1, p12 = p2 - p1, q12 = q2 - q1;

return SGN(pq1\*q12)\*SGN((p2 - q1)\*q12)<=0&&SGN(pq1\*p12)\*SGN((p1-q2)\*p12)<=0;}

//donot include extream points, and donot include coincidence.

inline bool lineLineSegIntersect(const Point& l1, const Point& l2, const Point& p1, const Point& p2){

Point line = l2 - l1;return SGN((p1 - l1)\*line)\*SGN((p2 - l1)\*line) < 0;}

//donnot include coincidence.

inline bool lineLineIntersect(const Point& p1, const Point& p2, const Point& q1, const Point& q2){

return !EQUAL((p2 - p1)\*(q2 - q1), 0);}

inline Point lineLineIntersectPoint(const Point& p1, const Point& p2, const Point& q1, const Point& q2){

Point q12 = q2 - q1;double k = (p2 - p1)\*q12;

if (EQUAL(k, 0))return Point(INF\*INF, INF\*INF);

double r = ((q1 - p1)\*q12) / k;return p1 + (p2 - p1) \* r;}

Point circumcenter(const Point& p1, const Point& p2, const Point& p3){

Point t1 = (p1 + p2)\*0.5, t2, t3 = (p2 + p3)\*0.5, t4;

t2 = t1 + (p1 - p2).rotate90();t4 = t3 + (p2 - p3).rotate90();

return lineLineIntersectPoint(t1, t2, t3, t4);}

Point incenter(const Point& p1, const Point& p2, const Point& p3){

double r12 = p1.dis(p2), r23 = p2.dis(p3), r31 = p3.dis(p1);

Point t1 = (p2\*r31 + p3\*r12)/(r12 + r31), t2 = (p1\*r23 + p3\*r12)/(r12 + r23);

return lineLineIntersectPoint(p1, t1, p2, t2);}

Point prepencenter(const Point& p1, const Point& p2, const Point& p3){

Point t1 = p1 + (p2 - p3).rotate90();

Point t2 = p2 + (p1 - p3).rotate90();

return lineLineIntersectPoint(p1, t1, p2, t2);}

inline Point barycenter(const Point& p1, const Point& p2, const Point& p3){

return (p1 + p2 + p3) / 3;}

inline double apothem(const Point& p1, const Point& p2, const Point& p3){

Point p12 = p2 - p1, p13 = p3 - p1, p23 = p3 - p2;

return abs(p12\*p23) / (p12.r() + p13.r() + p23.r());}

inline double circumradius(const Point& p1, const Point& p2, const Point& p3){

Point p12 = p2 - p1, p13 = p3 - p1, p23 = p3 - p2;

return sqrt(p12.r2()\*p23.r2()\*p13.r2()) / (2 \* abs(p12\*p23));}

int getPolygonDirection(const Point\* p, int n){

int index = 0;

for (int i = 1; i < n; i++){if (p[i] < p[index])index = i;}

return p[index].direction(p[index + 1 < n ? index + 1 : 0],

p[index - 1 >= 0 ? index - 1 : n - 1]);}

bool checkConvex(const Point\* p, int n){

int direction = p[0].direction(p[n - 1], p[1]);

if (direction == 0)return false;

if (p[n - 1].direction(p[n - 2], p[0]) != direction)return false;

for (int i = n - 2; i > 0; i--){

if (p[i].direction(p[i - 1], p[i + 1]) != direction)return false;}

return true;}

double polygonArea(const Point\* p, int n){

double area = 0;

for (int i = n - 2; i > 0; i--)

area += p[i].y \*(p[i - 1].x - p[i + 1].x);

area += p[0].y\*(p[n - 1].x - p[1].x);

area += p[n - 1].y\*(p[n - 2].x - p[0].x);

return area / 2;}

int Point::inPolygon(const Point\* p, int n)const{

int i, j = n - 1, odd = -1;

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

if (LESS(p[i].y, y) != LESS(p[j].y, y)){

double tx = (y - p[j].y)/(p[i].y - p[j].y)\*(p[i].x - p[j].x) + p[j].x;

if (LESS\_EQUAL(tx, x)){

if (LESS(tx, x))odd = -odd;

else return 0; }}

else if (onLineSeg(p[i], p[j]))return 0; }

return odd;}

int Point::inConvex(const Point\* p, int n)const{

int \_direction = p[1].direction(p[2], p[0]);

if (direction(p[0], p[1]) != \_direction){

if (onLineSeg(p[0], p[1]))return 0;

return -1; }

if (direction(p[n - 1], p[0]) != \_direction){

if (onLineSeg(p[n - 1], p[0]))return 0;

return -1; }

int left = 2, right = n - 1;

while (left < right){

int mid = (left + right) >> 1;

if (direction(p[0], p[mid]) == \_direction)left = mid + 1;

else right = mid; }

int ret = direction(p[left - 1], p[left]);

return ret == \_direction ? 1 : ret == 0 ? 0 : -1;}

bool lineSegInPolygon(const Point& p1, const Point& p2, const Point\* p, int n){

bool flag = false; Point minPoint;

switch (p1.inPolygon(p, n)){

case -1:return false;

case 0:flag = true; }

switch (p2.inPolygon(p, n)){

case -1:return false;

case 1:flag = false; }

if (flag)minPoint = max(p1, p2);

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

if (p[i].onLineSeg(p1, p2) && !(p[i] == p1 || p[i] == p2)){

if (p[i > 0 ? i - 1 : n - 1].direction(

p1, p2) \* p[i + 1 < n ? i + 1 : 0].direction(p1, p2) < 0) return false;

if (flag && p[i] < minPoint)minPoint = p[i]; }

else if (lineSegLineSegIntersect(p[i], p[j], p1, p2)) return false; }

if (flag){

const Point& t = min(p1, p2);

Point mid = (t + minPoint)\*0.5;

if (mid.inPolygon(p, n) == -1)return false; }

return true;}

Point gravityCenter(const Point\* p, int n){

if (n < 3){

if (n == 1)return p[0];

else return (p[0] + p[1])\*0.5; }

double area = 0;

Point ret(0, 0);

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

double t = p[i] \* p[j]; area += t;

ret.x += (p[i].x + p[j].x)\*t; ret.y += (p[i].y + p[j].y)\*t; }

return ret / (3 \* area);}

//ret[n] must be available to visit.

int convexHullSorted(const Point\* p, int n, Point\* ret){

int j = 0;

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

while (j >= 2 && p[i].direction(ret[j - 2], ret[j - 1]) != 1)j--;

ret[j++] = p[i]; }

int mid = j + 1;

for (int i = n - 2; i >= 0; i--){

while (j >= mid && p[i].direction(ret[j - 2], ret[j - 1]) != 1)j--;

ret[j++] = p[i]; }

return j - 1;}

int halfPlainIntersectInternal(vector<pair<double, const Point\*>>& v, int n, Point\* ret)

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

v[i].first = v[i].second[1].angle(v[i].second[0]);

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

vector<const Point\*> line(n);

vector<Point> point(n);

int first = 0, last = 0;

line[0] = v[0].second;

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

while (first<last&&point[last-1].direction(v[i].second[0],v[i].second[1])==-1)

last--;

while (first<last&&point[first].direction(v[i].second[0],v[i].second[1])==-1)

first++;

line[++last] = v[i].second;

if (!lineLineIntersect(line[last-1][0],line[last-1][1],

line[last][0],line[last][1])){

last--;

if (v[i].second[0].direction(line[last][0], line[last][1]) == 1)

line[last] = v[i].second; }

if (first<last) point[last - 1] = lineLineIntersectPoint(

line[last - 1][0], line[last - 1][1], line[last][0], line[last][1]); }

while (first<last&&point[last-1].direction(line[first][0], line[first][1])==-1)

last--;

if (last - first <= 1) return 0;

point[last] = lineLineIntersectPoint(line[first][0], line[first][1],

line[last][0], line[last][1]);

int num = unique(&\*point.begin()+first, &\*point.begin()+last+1) - &point[first];

while (num>1 && point[first] == point[first + num - 1])num--;

memcpy(ret, &point[first], sizeof(Point)\*num);

return num;}

int halfPlainIntersect(const Point(\*p)[2], int n, Point\* ret){

vector<pair<double, const Point\*>> v(n + 4);

Point ext[4][2] ={{Point(-INF,-INF),Point(INF,-INF)},{Point(INF,-INF),Point(INF, INF)},{Point(INF,INF),Point(-INF,INF)},{Point(-INF,INF),Point(-INF,-INF)}};

for (int i = 0; i < 4; i++)v[i].second = ext[i];

for (int i = 0; i < n; i++)v[i + 4].second = p[i];

return halfPlainIntersectInternal(v, n + 4, ret);}

//These points must be put counter-clockwise.

int polygonKernel(const Point\* p, int n, Point\* ret){

vector<pair<double, const Point\*>> v(n);

Point ext[2] = { p[n - 1], p[0] };

v[0].second = ext;

for (int i = 1; i < n; i++) v[i].second = &p[i - 1];

return halfPlainIntersectInternal(v, n, ret);}

//These points must be put counter-clockwise.

//Ensure p[n] exists and p[n]=p[0].

double convexDiameter(const Point\* p, int n, Point& ret1, Point& ret2){

double ret = 0;

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

double t1 = (p[i + 1] - p[i]) \* (p[j] - p[i]), t2;

for (; LESS(t1, t2 = (p[i + 1] - p[i]) \* (p[j + 1] - p[i])); t1 = t2){

if (++j == n)j = 0; }

double td2 = p[i].dis2(p[j]);

if (ret < td2){ ret = td2; ret1 = p[i]; ret2 = p[j]; }

td2 = p[i + 1].dis2(p[j]);

if (ret < td2){ ret = td2; ret1 = p[i + 1]; ret2 = p[j]; }}

return sqrt(ret);}

// These points must be put counter - clockwise.

//Ensure p[n] exists and p[n]=p[0].

double convexWidth(const Point\* p, int n){

double ret = INF;

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

double t1 = (p[i + 1] - p[i]) \* (p[j] - p[i]), t2;

for (; LESS(t1, t2 = (p[i + 1] - p[i]) \* (p[j + 1] - p[i])); t1 = t2){

if (++j == n)j = 0; }

ret = min(ret, t1 / p[i].dis(p[i + 1])); }

return ret;}

//include extream points.

bool circleLineSegIntersect(const Point& o,double r,const Point& p1,const Point& p2){

int t1 = p1.inCircle(o, r), t2 = p2.inCircle(o, r);

if (t1 >= 0 || t2 >= 0) return t1 != 1 || t2 != 1;

double t = o.lineRelation(p1, p2);

if (t >= 1 || t <= 0)return false;

Point foot = p1 + (p2 - p1)\*t;

return foot.inCircle(o, r) >= 0;}

//ret1 is near p1,ret2 is near p2.

void circleLineIntersect(const Point& o, double r, const Point& p1, const Point& p2, Point& ret1, Point& ret2){

Point foot = o.footPoint(p1, p2);

double t = r\*r - o.dis2(foot);

t = LESS\_EQUAL(t, 0) ? 0 : sqrt(t / p1.dis2(p2));

ret1 = foot + (p1 - p2)\*t; ret2 = foot \* 2 - ret1;}

void circleCircleIntersect(const Point& o1, double r1, const Point& o2, double r2, Point& ret1, Point& ret2){

double d2 = o1.dis2(o2);

double t1 = (r1\*r1 - r2\*r2) / (2 \* d2) + 0.5;

double t2 = r1\*r1 / d2 - t1\*t1;

t2 = LESS\_EQUAL(t2, 0) ? 0 : sqrt(t2);

Point foot = o1 + (o2 - o1)\*t1;

ret1 = foot + (o2 - o1).rotate90()\*t2; ret2 = foot \* 2 - ret1;}

void Point::pointcut(const Point& o, double r, Point& ret1, Point& ret2)const{

double t1 = r\*r / dis2(o);

Point foot = o + (\*this - o)\*t1;

double t2 = t1 - t1\*t1;

t2 = LESS\_EQUAL(t2, 0) ? 0 : sqrt(t2);

ret1 = foot + (\*this - o).rotate90() \* t2; ret2 = foot \* 2 - ret1;}

//ret[0] and ret[2] are on circle o1,ret[1] and ret[3] are on circle o2.

void circleCirclePointcutOuter(const Point& o1, double r1, const Point& o2, double r2, Point\* ret){

Point o12 = o2 - o1;

double d12 = o12.r2(), r = (r1 - r2) / d12;

Point foot1 = o1 + o12\*(r\*r1), foot2 = o2 + o12\*(r\*r2);

double t = 1 / d12 - r\*r;

t = LESS\_EQUAL(t, 0) ? 0 : sqrt(t);

Point line = o12.rotate90();

ret[0] = foot1 + line\*(t\*r1); ret[1] = foot2 + line\*(t\*r2);

ret[2] = foot1 \* 2 - ret[0]; ret[3] = foot2 \* 2 - ret[1];}

void circleCirclePointcutInner(const Point& o1, double r1, const Point& o2, double r2, Point\* ret){

Point o12 = o2 - o1;

double d12 = o12.r2(), r = (r1 + r2) / d12;

Point foot1 = o1 + o12\*(r\*r1), foot2 = o2 - o12\*(r\*r2);

double t = 1 / d12 - r\*r;

t = LESS\_EQUAL(t, 0) ? 0 : sqrt(t);

Point line = o12.rotate90();

ret[0] = foot1 + line\*(t\*r1); ret[1] = foot2 - line\*(t\*r2);

ret[2] = foot1 \* 2 - ret[0]; ret[3] = foot2 \* 2 - ret[1];}

Point Point::nearnestPoint(const Point& o, double r)const{

Point p = \*this - o;

double d = p.r();

if (EQUAL(d, 0))return o;

return o + p \* (r / d);}

//Upset the order before using this function.

double minCoveringCircle(const Point\* p, int n, Point& ret)

{

if (n == 1){ ret = p[0]; return 0; }

double r2 = p[0].dis2(p[1]) / 4;

ret = (p[0] + p[1]) \* 0.5;

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

if (LESS(r2, ret.dis2(p[i]))){

ret = (p[0] + p[i]) \* 0.5; r2 = p[0].dis2(p[i]) / 4;

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

if (LESS(r2, ret.dis2(p[j]))){

ret = (p[i] + p[j]) \* 0.5; r2 = p[i].dis2(p[j]) / 4;

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

if (LESS(r2, ret.dis2(p[k]))){

ret = circumcenter(p[i], p[j], p[k]);

r2 = ret.dis2(p[k]); }}}}}}

return sqrt(r2);}

int unitCoveringCircle(const Point\* p, int n, double r){

int ret = 0;

vector<pair<double, bool>> v;

v.reserve(2 \* n);

double t = r\*r \* 4;

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

v.clear();

int value = 0;

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

if (LESS\_EQUAL(p[i].dis2(p[j]), t) && i != j){

double a = p[j].angle(p[i]);

double b = acos(p[i].dis(p[j]) / r / 2);

double t1 = a - b, t2 = a + b;

if (t1 < -PI / 2){

t1 += 2 \* PI;

if (t2 < -PI / 2)t2 += 2 \* PI;

else value++;}

v.push\_back(make\_pair(t1, true));

v.push\_back(make\_pair(t2, false)); }}

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

if (value > ret)ret = value;

for (unsigned int j = 0; j < v.size(); j++){

if (v[j].second){

value++;

if (value > ret)ret = value;

}else value--;}}

return ret + 1;}

double circlePolygonAreaIntersect(const Point& o, double r, const Point\* p, int n){

double area = 0;

Point p1, p2;

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

int f1 = p[i].inCircle(o, r), f2 = p[j].inCircle(o, r);

if (f1 >= 0 && f2 >= 0)area += (o - p[i])\*(o - p[j]);

else if (f1 >= 0 && f2 < 0){

circleLineIntersect(o, r, p[i], p[j], p1, p2);

area += (o - p[i])\*(o - p2);

area += asin(o.sinAngle(p2, p[j])) \* r \* r; }

else if (f1 < 0 && f2 >= 0){

circleLineIntersect(o, r, p[i], p[j], p1, p2);

area += (o - p1)\*(o - p[j]);

area += asin(o.sinAngle(p[i], p1)) \* r \* r; }

else if (circleLineSegIntersect(o, r, p[i], p[j])){

circleLineIntersect(o, r, p[i], p[j], p1, p2);

area += (o - p1)\*(o - p2);

area += (asin(o.sinAngle(p[i], p1)) + asin(o.sinAngle(p2, p[j]))) \* r \* r;}

else area += asin(o.sinAngle(p[i], p[j])) \* r \* r; }

return abs(area / 2);}

*////*

Tags 二分 离线 倒跑 并查集 DFS BFS 贪心 DP 递推 莫队

前缀和 快速幂 倍增！ 差分数列

3.1415926535897932384626433832795; 2.718281828459045235360287471352;

i 插入模式

esc 正常模式

:e xxx.cpp 编辑xxx.cpp

:w 保存

:q 退出

:wq 保存并退出

ggvG 全选

y 复制

p 粘贴

u 撤销

"+y 复制到外部

"+p 从外部粘贴

正常模式下按v 再按ijkl可选定指定区域

（vim的剪切板和外部的剪切板貌似是分开的？）

（在vimrc正常工作的情况下）

F9 编译

F5 运行

:split xxx.txt 横屏打开新文件 xxx.txt

:vsplit xxx.txt 竖屏打开新文件 xxx.txt

Ctrl+ww 移动到下一个窗口

**inline** void read(int &x) {

char ch=getchar();

**while**(ch<'0'||ch>'9') ch=getchar();x=0;

**while**(ch<='9'&&ch>='0'){

x=x\*10+ch-'0';ch=getchar();}}

map <F9> :! g++ % -o %< -g -lm -Wall && size %<.exe <CR>

map <F5> :! gdb %< <CR>

set cindent

set smartindent

set autoindent

set number

set ruler

set mouse=a

set bs=2

set tabstop=4

set softtabstop=4

set shiftwidth=4

set autoread

"set expandtab

syntax on

colo evening