[一. 数据结构： 1](#_Toc371519947)

[1.1 树状数组 1](#_Toc371519948)

[1.2 线段树框架 1](#_Toc371519949)

[1.3 Treap 2](#_Toc371519950)

[1.4 Splay 8](#_Toc371519951)

[1.5 LCT 12](#_Toc371519952)

[1.6 树链刨分 17](#_Toc371519953)

[1.7 Kd-tree 21](#_Toc371519954)

[二. 字符串 27](#_Toc371519955)

[2.1 Kmp&exKmp 27](#_Toc371519956)

[2.2 Aho-Corasick Automaton 28](#_Toc371519957)

[2.3 Hash 30](#_Toc371519958)

[2.4 Manacher 31](#_Toc371519959)

[2.5 Suffix\_Array o(nlgn) 34](#_Toc371519960)

[2.6 Suffix\_Arrar o(n) 35](#_Toc371519961)

[2.7 后缀自动机 37](#_Toc371519962)

[三. 数论 39](#_Toc371519963)

[3.1 线性素数筛及欧拉函数 39](#_Toc371519964)

[3.2 欧拉函数 39](#_Toc371519965)

[3.3 Lucas+C(n,m) 40](#_Toc371519966)

[3.4 质因数分解 41](#_Toc371519967)

[3.5 平方剩余 45](#_Toc371519968)

[3.6 原根 46](#_Toc371519969)

[3.7 Nim积 46](#_Toc371519970)

[四. 杂项 49](#_Toc371519971)

[4.1 FFT 49](#_Toc371519972)

[4.2 Tree count 51](#_Toc371519973)

[4.3 额外栈空间 53](#_Toc371519974)

[4.4 矩阵类~java 53](#_Toc371519975)

[4.5 分数类~java 54](#_Toc371519976)

[4.6 标准模板~java 56](#_Toc371519977)

[4.7 注意事项~java 56](#_Toc371519978)

[五. Reference 57](#_Toc371519979)

[5.1 <algorithm> 57](#_Toc371519980)

[5.2 <bitset> 57](#_Toc371519981)

[5.3 Container 57](#_Toc371519982)

[5.4 <ctime> 58](#_Toc371519983)

[5.5 <cctype> 58](#_Toc371519984)

[5.6 <cstdio> 58](#_Toc371519985)

[5.7 宏定义 59](#_Toc371519986)

[六. 数学知识 59](#_Toc371519987)

[6.1 组合数 59](#_Toc371519988)

[6.2 Stirling数 59](#_Toc371519989)

[6.3 Catalan数 61](#_Toc371519990)

[6.4 Bell数 64](#_Toc371519991)

# 数据结构：

## 树状数组

#define lowbit(x) ((x)&-(x))

#define sz 100010

int nt[sz];

void insert(int pos,int val)

{

for(pos+=3;pos<sz;pos+=lowbit(pos))nt[pos]+=val;

}

int query(int pos)

{

int ret=0;

for(pos+=3;pos>0;pos-=lowbit(pos))ret+=nt[pos];

return ret;

}

## 线段树框架

void insert(int L,int R,int mul,int add,int who=1,int l=1,int r=n)

{

if(L<=l&&R>=r)

{

update(who,l,r,mul,add);

return;

}

push\_down(who,l,r);

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

if(L<=mid)insert(L,R,mul,add,who<<1,l,mid);

if(R>mid)insert(L,R,mul,add,who<<1|1,mid+1,r);

push\_up(who);

}

int query(int L,int R,int who=1,int l=1,int r=n)

{

if(L<=l&&R>=r)

{

return nt[who];

}

int ret=0;

push\_down(who,l,r);

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

if(L<=mid)ret+=query(L,R,who<<1,l,mid);

if(R>mid)ret+=query(L,R,who<<1|1,mid+1,r);

return ret;

}

## Treap

可持久化：

#define sz 100100

struct node{int ch[2];int val,size;}nt[20000000];

void push\_up(int w){nt[w].size = nt[nt[w].ch[0]].size + 1 + nt[nt[w].ch[1]].size;}

void print(int w)

{

if(w == 0)return;

print(nt[w].ch[0]);

printf("%c",nt[w].val);

print(nt[w].ch[1]);

}

int tcnt = 0;

int get\_node(int val)

{

nt[tcnt].val = val;

nt[tcnt].size = 1;

return tcnt++;

}

int merge(int a,int b)

{

if(a==0||b==0)return a+b;

int r = 0;

//if(rand()%32768/32768.0 \* (nt[a].size + nt[b].size) < nt[a].size)

if(nt[a].size > nt[b].size)

{

r = get\_node(nt[a].val);

nt[r].ch[0] = nt[a].ch[0];

nt[r].ch[1] = merge(nt[a].ch[1],b);

}

else

{

r = get\_node(nt[b].val);

nt[r].ch[0] = merge(a,nt[b].ch[0]);

nt[r].ch[1] = nt[b].ch[1];

}

push\_up(r);

return r;

}

void split(int w,int lhas,int&a,int&b)

{

if(nt[w].size == lhas)a = w, b = 0;

else if(lhas == 0)a = 0, b = w;

else

{

int ret = get\_node(nt[w].val);

int lsz = nt[nt[w].ch[0]].size;

if(lhas <= nt[nt[w].ch[0]].size)

{

split(nt[w].ch[0],lhas,a,b);

nt[ret].ch[0] = b;

nt[ret].ch[1] = nt[w].ch[1];

b = ret;

}

else

{

lsz ++;

split(nt[w].ch[1],lhas - lsz,a,b);

nt[ret].ch[0] = nt[w].ch[0];

nt[ret].ch[1] = a;

a = ret;

}

push\_up(ret);

}

}

int build(char \*s,int n)

{

if(n<0)return 0;

int m = n/2;

int ret = get\_node(s[m]);

nt[ret].ch[0] = build(s,m-1);

nt[ret].ch[1] = build(s+m+1,n-m-1);

push\_up(ret);

return ret;

}

int tot;

void put(int w,int l,int r)

{

if(w == 0)return;

int lsz = nt[nt[w].ch[0]].size;

if(l <= lsz)put(nt[w].ch[0],l,r);

lsz ++;

if(l <= lsz && r>=lsz)

{

printf("%c",nt[w].val);

if(nt[w].val == 'c') tot ++;

}

if(r > lsz)put(nt[w].ch[1],l - lsz,r - lsz);

}

void init()

{

ver = 0;

tcnt = 1;

nt[0].size = 0;

root[0] = 0;

tot = 0;

}

/\*\*uva12538\*\*/

char s[200110];

int root[sz], ver = 0;

int main()

{

int n;scanf("%d",&n);

init();

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

{

int o;scanf("%d",&o);

if(o==1)

{

int p;scanf("%d%s",&p,&s); p -= tot;

int a,b;split(root[ver],p,a,b);

int z = build(s,strlen(s)-1);

root[++ver] = merge(merge(a,z),b);

}

else if(o==2)

{

int p,c;scanf("%d%d",&p,&c); p -= tot + 1; c -= tot;

int a,b,z;split(root[ver],p,a,b);

split(b,c,z,b);

root[++ver] = merge(a,b);

}

else

{

int v,p,c;scanf("%d%d%d",&v,&p,&c); v -= tot; p -= tot; c -= tot;

put(root[v],p,p+c-1);puts("");

}

}

}

正常版本：

#include<cstdio>

#include<algorithm>

using namespace std;

#define max\_N 200010

struct treap

{

int b,g;

int ch[2];

int has,size,ans,sz\_b,sz\_g;

}nt[max\_N];

#define ww nt[w]

#define ch0 nt[nt[w].ch[0]]

#define ch1 nt[nt[w].ch[1]]

int tcnt;

void init()

{

nt[0].sz\_b = nt[0].sz\_g = nt[0].size = 0;

nt[0].ans = 0;

tcnt = 1;

}

int init\_node(int b,int g)

{

if(b+g==0)return 0;

int w = tcnt++;

ww.size = 1;

ww.ch[0] = ww.ch[1] = 0;

ww.has = b + g;

ww.ans = 0;

ww.sz\_b = ww.b = b;

ww.sz\_g = ww.g = g;

return w;

}

void upd(int w)///b g

{

ww.size = ch0.size + 1 + ch1.size;

ww.has = ch0.has + ww.b + ww.g + ch1.has;

int b = ch0.sz\_b + ww.b;

int g = ch1.sz\_g + ww.g;

ww.ans = min(b,g);

ww.sz\_g = ch0.sz\_g + g - ww.ans;

ww.sz\_b = ch1.sz\_b + b - ww.ans;

ww.ans += ch0.ans + ch1.ans;

}

int merge(int a,int b)

{

if(a==0||b==0)return a+b;

int w;

if(rand()%32768/32768.0 \* (nt[a].size + nt[b].size) < nt[a].size)

{

w = a;

ww.ch[1] = merge(ww.ch[1],b);

}

else

{

w = b;

ww.ch[0] = merge(a,ww.ch[0]);

}

upd(w);

return w;

}

void split(int w,int lhas,int&a,int&m,int&b)

{

if(ww.has == lhas)a = w, m = 0, b = 0;

if(lhas == 0) a = 0, m = 0, b = w;

else

{

int lsz = ch0.has;

if(lhas <= lsz)

{

split(ww.ch[0],lhas,a,m,b);

ww.ch[0] = b;

b = w;

}

else

{

lsz += ww.b + ww.g;

if(lhas <= lsz)

{

a = ww.ch[0], b = ww.ch[1];

ww.ch[0] = ww.ch[1] = 0;

m = w;

}

else

{

split(ww.ch[1],lhas-lsz,a,m,b);

ww.ch[1] = a;

a = w;

}

}

upd(w);

}

}

int build(int w,int past,int b,int g)

{

if(w == 0)return init\_node(b,g);

if(ww.b&&b||ww.g&&g)

{

ww.b += b;

ww.g += g;

}

else

{

int bb = min(ww.b,past), gg = min(ww.g,past);

ww.ch[0] = init\_node(bb,gg);

ww.ch[1] = init\_node(ww.b-bb,ww.g-gg);

ww.b = b, ww.g = g;

}

upd(w);

return w;

}

void print(int w)

{

if(w == 0)return;

if(ww.b&&ww.g)printf("<%d>",w);

print(ww.ch[0]);

printf("%c%d ",ww.b==0?'g':'b',ww.b + ww.g);

print(ww.ch[1]);

}

/\*\*uva 12634\*\*/

int main()

{

int ti;scanf("%d",&ti);

for(int ca = 1; ca<=ti; ca++)

{

printf("Case %d:\n",ca);

int n;scanf("%d",&n);

init();

int root = init\_node(1,0);

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

{

int x,y,z;scanf("%d%d%d",&x,&y,&z);

int b,g;

if(x == 0)b = z, g = 0;

else g = z , b = 0;

int l,m,r;

split(root,y,l,m,r);

m = build(m,y - nt[l].has,b,g);

root = merge(merge(l,m),r);

printf("%d\n",nt[root].ans);

//print(root);puts("");puts("-------------");

}

}

}

## Splay

from clj

const int MAX\_N = 50000 + 10;

const int INF = ~0U >> 1;

struct Node {

Node\*ch[2], \*p;

int size;

int b,g;

int rb,rg,p;

//bool isRoot;

Node() {

size = 0;

val = mx = -INF;

add = 0;

}

bool d() {

return this == p->ch[1];

}

void setc(Node\*c, int d) {

ch[d] = c;

c->p = this;

}

void addIt(int ad) {

add += ad;

mx += ad;

val += ad;

}

void revIt() {

rev ^= 1;

}

void relax();

void upd() {

size = ch[0]->size + ch[1]->size + 1;

mx = max(val, max(ch[0]->mx, ch[1]->mx));

}

} Tnull, \*null = &Tnull;

Node mem[MAX\_N], \*C = mem;

void Node::relax() {

if (add != 0) {

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

if (ch[i] != null)

ch[i]->addIt(add);

}

add = 0;

}

if (rev) {

swap(ch[0], ch[1]);

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

if (ch[i] != null)

ch[i]->revIt();

}

rev = 0;

}

}

Node\*make(int v) {

C->ch[0] = C->ch[1] = null;

C->size = 1;

C->val = v;

C->mx = v;

C->add = 0;

C->rev = 0;

return C++;

}

Node\*build(int l, int r) {

if (l >= r)

return null;

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

Node\*t = make(0);

t->setc(build(l, m), 0);

t->setc(build(m + 1, r), 1);

t->upd();

return t;

}

Node\*root;

Node\*rot(Node\*t) {

Node\*p = t->p;

p->relax();

t->relax();

int d = t->d();

p->p->setc(t, p->d());

p->setc(t->ch[!d], d);

t->setc(p, !d);

p->upd();

if (p == root)

root = t;

// if (p->isRoot) {

// p->isRoot = false;

// t->isRoot = true;

// }

}

void splay(Node\*t, Node\*f = null) {

while (t->p != f) {

if (t->p->p == f)

rot(t);

else

t->d() == t->p->d() ? (rot(t->p), rot(t)) : (rot(t), rot(t));

}

t->upd();

}

Node\* select(int k) {

for (Node\*t = root;;) {

t->relax();

int c = t->ch[0]->size;

if (k == c)

return t;

if (k > c)

k -= c + 1, t = t->ch[1];

else

t = t->ch[0];

}

}

Node\*&get(int l, int r) { //[l,r)

Node\*L = select(l - 1);

Node\*R = select(r);

splay(L);

splay(R, L);

return R->ch[0];

}

int n, m;

int main() {

cin >> n >> m;

root = build(0, n + 2);

root->p = null;

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

int k, l, r, v;

scanf("%d%d%d", &k, &l, &r);

Node\*&t = get(l, r + 1);

if (k == 1) {

scanf("%d", &v);

t->addIt(v);

splay(t);

} else if (k == 2) {

t->revIt();

splay(t);

} else {

printf("%d\n", t->mx);

}

}

}

## LCT

from clj havn’t changed

#include<cstdio>

#include<algorithm>

#include<cstring>

#include<vector>

#define REP(i,n) for(int i=0;i<n;++i)

using namespace std;

//Dynamic Tree

typedef long long int64;

const int MOD = 51061;

const int MAX\_N = int(1e5) + 10;

struct Mark {

int64 add, mul; //x\*mul+add

Mark(int64 add, int64 mul) {

this->add = add;

this->mul = mul;

}

Mark() {

mul = 1;

add = 0;

}

bool isId() {

return mul == 1 && add == 0;

}

};

Mark operator\*(Mark a, Mark b) {

return Mark((a.add \* b.mul + b.add) % MOD, a.mul \* b.mul % MOD);

}

struct Node {

Node\*p, \*ch[2];

bool rev;

Mark m;

int64 sum, val;

int size;

bool isRoot;

Node\*fa;

Node() {

sum = 0;

isRoot = 0;

size = 0;

}

void sc(Node\*c, int d) {

ch[d] = c;

c->p = this;

}

bool d() {

return this == p->ch[1];

}

void upd() {

sum = (val + ch[0]->sum + ch[1]->sum) % MOD;

size = 1 + ch[0]->size + ch[1]->size;

}

void apply(Mark a) {

m = m \* a;

sum = (sum \* a.mul + a.add \* size) % MOD;

val = (val \* a.mul + a.add) % MOD;

}

void revIt() {

rev ^= 1;

swap(ch[0], ch[1]);

}

void relax();

void setRoot(Node\*f);

} Tnull, \*null = &Tnull;

void Node::setRoot(Node\*f) {

fa = f;

isRoot = true;

p = null;

}

void Node::relax() {

if (!m.isId()) {

REP(i,2)

if (ch[i] != null)

ch[i]->apply(m);

m = Mark();

}

if (rev) {

REP(i,2)

if (ch[i] != null)

ch[i]->revIt();

rev = 0;

}

}

Node mem[MAX\_N], \*C = mem;

Node\*make(int v) {

C->sum = C->val = v;

C->rev = 0;

C->m = Mark();

C->ch[0] = C->ch[1] = null;

C->isRoot = true;

C->p = null;

C->fa = null;

return C++;

}

void rot(Node\*t) {

Node\*p = t->p;

p->relax();

t->relax();

bool d = t->d();

p->p->sc(t, p->d());

p->sc(t->ch[!d], d);

t->sc(p, !d);

p->upd();

if (p->isRoot) {

p->isRoot = false;

t->isRoot = true;

t->fa = p->fa;

}

}

void pushTo(Node\*t) {

static Node\*stk[MAX\_N];

int top = 0;

while (t != null) {

stk[top++] = t;

t = t->p;

}

for (int i = top - 1; i >= 0; --i)

stk[i]->relax();

}

void splay(Node\*u, Node\*f = null) {

pushTo(u);

while (u->p != f) {

if (u->p->p == f)

rot(u);

else

u->d() == u->p->d() ? (rot(u->p), rot(u)) : (rot(u), rot(u));

}

u->upd();

}

Node\*v[MAX\_N];

vector<int> E[MAX\_N];

int n, nQ;

int que[MAX\_N], fa[MAX\_N], qh = 0, qt = 0;

void bfs() {

que[qt++] = 0;

fa[0] = -1;

while (qh < qt) {

int u = que[qh++];

for (vector<int>::iterator e = E[u].begin(); e != E[u].end(); ++e)

if (\*e != fa[u])

fa[\*e] = u, v[\*e]->fa = v[u], que[qt++] = \*e;

}

}

Node\* expose(Node\*u) {

Node\*v;

for (v = null; u != null; v = u, u = u->fa) {

splay(u);

u->ch[1]->setRoot(u);

u->sc(v, 1);

v->fa = u;

}

return v;

}

void makeRoot(Node\*u) {

expose(u);

splay(u);

u->revIt();

}

void addEdge(Node\*u, Node\*v) {

makeRoot(v);

v->fa = u;

}

void delEdge(Node\*u, Node\*v) {

makeRoot(u);

expose(v);

splay(u);

u->sc(null, 1);

u->upd();

v->fa = null;

v->isRoot = true;

v->p = null;

}

void markPath(Node\*u, Node\*v, Mark m) {

makeRoot(u);

expose(v);

splay(v);

v->apply(m);

}

int queryPath(Node\*u, Node\*v) {

makeRoot(u);

expose(v);

splay(v);

return v->sum;

}

int main() {

scanf("%d%d", &n, &nQ);

REP(i,n-1) {

int u, v;

scanf("%d%d", &u, &v);

--u, --v;

E[u].push\_back(v);

E[v].push\_back(u);

}

REP(i,n)

v[i] = make(1);

bfs();

REP(w,nQ) {

char cmd;

scanf(" ");

scanf("%c", &cmd);

int i, j;

scanf("%d%d", &i, &j);

int v;

Node\*u = v[--i], \*v = v[--j];

if (cmd == '+') {

int c;

scanf("%d", &c);

markPath(u, v, Mark(c, 1));

} else if (cmd == '\*') {

int c;

scanf("%d", &c);

markPath(u, v, Mark(0, c));

} else if (cmd == '/') {

printf("%d\n", queryPath(u, v));

} else {

int k, l;

scanf("%d%d", &k, &l);

delEdge(u, v);

addEdge(::v[--k], ::v[--l]);

}

}

}

## 树链刨分

#include<cstdio>

#include<vector>

#include<cstring>

#include<algorithm>

using namespace std;

#define sz 10010

vector<pair<int, int> >links[sz];

int val[sz],gid[sz],n;

struct node

{

int has,deep;

int top,fa,id;

}nt[sz];

struct segtree

{

int sgt[sz<<2],f[sz],cnt;

void push\_up(int who)

{

sgt[who] = max(sgt[who<<1],sgt[who<<1|1]);

}

int query(int L,int R,int who=1,int l=1,int r=n)

{

if(L<=l&&R>=r)

{

return sgt[who];

}

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

int ret = 0;

if(L<=mid)ret = max(ret, query(L,R,who<<1,l,mid));

if(R>mid)ret = max(ret, query(L,R,who<<1|1,mid+1,r));

return ret;

}

void change(int pos,int val,int who=1,int l=1,int r=n)

{

if(l==r)

{

sgt[who]=val;

return;

}

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

if(pos<=mid)change(pos,val,who<<1,l,mid);

else change(pos,val,who<<1|1,mid+1,r);

push\_up(who);

}

void build(int who=1,int l=1,int r=n)

{

if(l==r)

{

sgt[who]=f[l];

return;

}

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

build(who<<1,l,mid);

build(who<<1|1,mid+1,r);

push\_up(who);

}

void print(int who=1,int l=1,int r=n)

{

printf("%d %d %d %d\n",who,l,r,sgt[who]);

if(l==r)return;

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

print(who<<1,l,mid);

print(who<<1|1,mid+1,r);

}

inline int append(int val)

{

f[++cnt] = val;

return cnt;

}

}sg;

#define A first

#define B second

void pdfs(int who, int from)

{

nt[who].deep = nt[from].deep + 1;

nt[who].fa = from;

nt[who].has = 1;

for(int i = 0; i < links[who].size(); i ++)

{

int to = links[who][i].B;

if(to==from)continue;

pdfs(to,who);

nt[who].has += nt[to].has;

}

}

void dfs(int who,int top,int id)

{

nt[who].top = top;

nt[who].id = gid[id] = sg.append(val[id]);

int mto = -1;

int iid = -1;

for(int i = 0; i < links[who].size(); i ++)

{

int to = links[who][i].B;

if(to==nt[who].fa)continue;

if(mto==-1||nt[to].has>nt[mto].has)mto = to, iid = links[who][i].A;

}

if(mto!=-1)dfs(mto,top,iid);

for(int i = 0; i < links[who].size(); i ++)

{

int to = links[who][i].B;

if(to==nt[who].fa||to==mto)continue;

dfs(to,to,links[who][i].A);

}

}

void change(int pos,int val)

{

sg.change(gid[pos],val);

}

int query(int a,int b)

{

int ret = 0;

int ta = nt[a].top, tb = nt[b].top;

while(ta!=tb)

{

//printf("%d %d %d %d\n",a,b,ta,tb);

if(nt[ta].deep<nt[tb].deep)swap(a,b),swap(ta,tb);

ret = max(ret, sg.query(nt[ta].id,nt[a].id));

//printf("zz%d %d %d\n",ta,a,sg.query(nt[ta].id,nt[a].id));

a = nt[ta].fa;

ta = nt[a].top;

}

if(a!=b)

{

if(nt[a].deep>nt[b].deep)swap(a,b);

ret = max(ret, sg.query(nt[a].id+1,nt[b].id));

//printf("uu%d %d %d\n",a,b,sg.query(nt[a].id+1,nt[a].id));

}

return ret;

}

/\*\*http://www.spoj.com/problems/QTREE/\*\*\*/

int main()

{

int ti;scanf("%d",&ti);

for(int ca = 1; ca <= ti; ca ++)

{

scanf("%d",&n);

for(int i = 1; i <=n; i ++)links[i].clear();

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

{

int a,b,c;scanf("%d%d%d",&a,&b,&c);

links[a].push\_back(make\_pair(i,b));

links[b].push\_back(make\_pair(i,a));

val[i] = c;

}

sg.cnt = 0;

pdfs(1, 0);

dfs(1,1,0);

sg.build();

while(1)

{

char tmp[13];scanf("%s",tmp);

if(tmp[0]=='D')break;

else

{

int a,b;scanf("%d%d",&a,&b);

if(tmp[0]=='Q')

{

printf("%d\n",query(a,b));

}

else

{

change(a,b);

}

}

}

}

}

## Kd-tree

from c\_loud26

#define eps 1e-8

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

#define INF 0x3fffffff

#define M 5 //坐标最大维数

#define N 50005 //总点数

int K; //坐标维数

struct Point {

int x[M];

double dis(Point& t) {

double ans = 0;

rep(i, 0, K)

ans += (double)(x[i] - t.x[i]) \* (x[i] - t.x[i]);

return sqrt(ans);

}

};

struct Node {

Point p;

Node \*left, \*right;

int lx[M], mx[M], cnt, split;

bool flag;

void init(Point& t, int s) {

p = t, split = s;

rep(i, 0, K)

mx[i] = INF, lx[i] = -INF;

cnt = 1;

left = right = 0;

flag = false;

}

void initLeft() {

rep(i, 0, K) {

left->mx[i] = mx[i];

left->lx[i] = lx[i];

}

left->mx[split] = p.x[split];

}

void initRight() {

rep(i, 0, K) {

right->mx[i] = mx[i];

right->lx[i] = lx[i];

}

right->lx[split] = p.x[split];

}

};

struct Com {

Node\* v;

double r;

Com() {}

Com(Node\* \_v, double \_r) {

v = \_v, r = \_r;

}

bool operator <(const Com &cur)const {

return r < cur.r;

}

};

Point p[N];

Node nodes[N], \*pstart, \*cur, \*root;

priority\_queue<Com> pq;

void QuickSort(int ii, int first, int end) {

int i = first, j = end;

Point tmp = p[first];

while (i < j) {

while (i < j && p[j].x[ii] >= tmp.x[ii])

j--;

p[i] = p[j];

while (i < j && p[i].x[ii] <= tmp.x[ii])

i++;

p[j] = p[i];

}

p[i] = tmp;

if (first < i - 1)

QuickSort(ii, first, i - 1);

if (end > i + 1)

QuickSort(ii, i + 1, end);

}

Node \*buildkdtree(int f, int t, int depth) {

if (t - f == 0)

return 0;

else if (t - f == 1)

{

pstart->init(p[f], depth);

return pstart++;

}

int mid = (f + t - 1) >> 1;

QuickSort(depth, f, t - 1);

pstart->init(p[mid], depth);

Node \*v = pstart++;

v->left = buildkdtree(f, mid, (depth + 1) % K);

v->right = buildkdtree(mid + 1, t, (depth + 1) % K);

return v;

}

int BuildSqr(Node \*v) {

if (!v) return 0;

if (v->left) v->initLeft();

if (v->right) v->initRight();

v->cnt = BuildSqr(v->left) + BuildSqr(v->right) + 1;

return v->cnt;

}

void insert(Node \*v, Point &t)

{

if(!v) {

pstart->init(t, 0);

root = pstart++;

return ;

}

if(t.x[v->split] < v->p.x[v->split]) {

if(v->left) insert(v->left, t);

else {

pstart->init(t, (v->split + 1) % K);

v->left = pstart++;

v->initLeft();

}

v->cnt++;

}

else {

if(v->right) insert(v->right, t);

else {

pstart->init(t, (v->split + 1) % K);

v->right = pstart++;

v->initRight();

}

v->cnt++;

}

}

Node\* SearchKNode(Node\* v, Point t, int k) {

if (!v || v->cnt < k)

return 0;

int dp = v->split;

if (t.x[dp] < v->p.x[dp] && v->left && v->left->cnt >= k)

return SearchKNode(v->left, t, k);

else if (t.x[dp] >= v->p.x[dp] && v->right && v->right->cnt >= k)

return SearchKNode(v->right, t, k);

else {

v->flag = 1;

cur = v;

return v;

}

}

void SearchMinr(Node\* v, Point t) {

if (!v) return;

pq.push(Com(v, v->p.dis(t)));

SearchMinr(v->left, t);

SearchMinr(v->right, t);

}

double searchKR(Point t, int k) {

Node\* v = SearchKNode(root, t, k);

if (!v) return -1.0;

SearchMinr(v, t);

while (pq.size() > k) pq.pop();

if (pq.size() == 0) return 0.0;

else return pq.top().r;

}

bool check(Node \*v, Point t, double r) {

Point now;

rep(i, 0, K) {

if (t.x[i] > v->mx[i])

now.x[i] = v->mx[i];

else if(t.x[i] < v->lx[i])

now.x[i] = v->lx[i];

else

now.x[i] = t.x[i];

}

return now.dis(t) < eps + r;

}

void KFind(int k, Point t, double &r, Node\* v) {

if (v->flag) return;

double dd = v->p.dis(t);

if (dd + eps < r) {

pq.push(Com(v, dd));

pq.pop();

r = pq.top().r;

}

if (v->left && check(v->left, t, r))

KFind(k, t, r, v->left);

if (v->right && check(v->right, t, r))

KFind(k, t, r, v->right);

}

void print(Node \*t) {

printf("%d", t->p.x[0]);

rep(i, 1, K)

printf(" %d", t->p.x[i]);

printf("\n");

}

//查找距离点t最近的k个点, 结果放在pq中

void KNN(Point t, int k) {

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

double r = searchKR(t, k);

KFind(k, t, r, root);

cur->flag = 0;

}

Node \*sta[N];

void printResult(int k)

{

printf("the closest %d points are:\n", k);

int top = 0;

while (!pq.empty())

sta[top++] = pq.top().v, pq.pop();

while(top > 0)

{

top--;

printf("%d", sta[top]->p.x[0]);

rep(i, 1, K)

printf(" %d", sta[top]->p.x[i]);

printf("\n");

}

}

int main() {

int T, mm, n, k;

while (scanf("%d%d", &n, &k) == 2) {

pstart = nodes;

K = k;

root = 0;

rep(i, 0, n) {

rep(j, 0, k) {

scanf("%d", &p[i].x[j]);

}

}

root = buildkdtree(0, n, 0);

BuildSqr(root);

Point aim;

scanf("%d", &T);

while (T--) {

rep(i, 0, k)

scanf("%d", &aim.x[i]);

scanf("%d", &mm);

KNN(aim, mm);

printResult(mm);

}

}

return 0;

}

# 字符串

## Kmp&exKmp

//kmp

int get\_next(char s[],int fail[])

{

int i=0,k=-1;

fail[0]=-1;

while(s[i])//i<n

{

if(k==-1||s[i]==s[k])fail[++i]=++k;

else k=fail[k];

}

return i;//len

}

int check(char str[],char ss[],int fail[],int m)

{

int ret = 0;

for(int i=0,j=0;str[i];i++)

{

while(j>=0&&str[i]!=ss[j])j=fail[j];

j++;

if(j == m)

{

ret ++;

j = fail[j];

}

}

return ret;

}

//exKmp

void exkmp(char ss[],int next[],int len)

{

int i=1,j=0,w;

next[0]=len;

while(i<len)

{

if(i+j<len&&ss[j]==ss[i+j])j++;

else

{

if(j==0)

{

next[i++]=0;

continue;

}

for(w=0;j>w&&j-w!=next[w];w++)

{

next[i+w]=min(j-w,next[w]);

}

i+=w;

j-=w;

}

}

}

## Aho-Corasick Automaton

#define sz 500010

#define son\_num 26

int ha[128],scnt;

void hash\_init()

{

memset(ha,-1,sizeof(ha));

scnt=0;

}

#define sha(c) son[((ha[c]==-1)?(ha[c]=scnt++):ha[c])]

struct tree{int fail,has,mark,son[son\_num];}nt[sz];

int tcnt;

int get\_node()

{

nt[tcnt].has=0;

nt[tcnt].mark=0;

memset(nt[tcnt].son,0,sizeof(nt[tcnt].son));

return tcnt++;

}

void insert(char \*s)

{

int now=0;

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

{

int &t = nt[now].sha(s[i]);

now=t==0?t=get\_node():t;

}

nt[now].has++;

}

int que[sz];

void build()

{

que[0]=0;

nt[0].fail=0;

for(int l=0,r=1;l<r;)

{

int w=que[l++];

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

{

int &t=nt[w].son[i];

if(t==0)

{

t=nt[nt[w].fail].son[i];//%tcnt+tcnt;

}

else

{

que[r++]=nt[w].son[i];

nt[t].fail=w>0?nt[nt[w].fail].son[i]:w;//%tcnt;

}

}

}

}

void ac\_init()

{

tcnt=0;

get\_node();

hash\_init();

}

char ss[1000010];

int main()

{

int ti;

scanf("%d",&ti);

while(ti--)

{

int n;

scanf("%d",&n);

ac\_init();

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

{

scanf("%s",ss);

insert(ss);

}

build();

scanf("%s",ss);

int now=0;

for(int i=0;ss[i];i++)

{

now=nt[now].sha(ss[i]);

nt[now].mark=1;

}

int ans=0;

for(int i=tcnt-1;i>0;i--)

{

if(nt[que[i]].mark==0)continue;

nt[nt[que[i]].fail].mark=1;

ans+=nt[que[i]].has;

}

printf("%d\n",ans);

}

}

## Hash

struct mhash

{

int val[sz],mm[sz],hash[200];

long long mul,mod;

void init(long long \_mul,long long \_mod)

{

mul = \_mul, mod = \_mod;

mm[0] = 1;

for(int i = 1; i < sz; i ++)mm[i] = mm[i-1]\*mul%mod;

for(int i = 0; i < 26; i ++)hash[i+'a'] = i+1;

}

void init(int vv[])

{

for(int i = 0; i < 26; i ++)hash[i+'a'] = vv[i];

}

void init(char ss[])

{

val[0] = 0;

for(int i = 0; ss[i]; i ++)

val[i+1] = (val[i]\*mul + hash[ss[i]])%mod;

}

int get(int f,int t)///[f,f + t)

{

return (val[f+t] + (mod - val[f])\*mm[t]%mod)%mod;

}

int get\_c(int f,int t)///[f, t]

{

return (val[t+1] + (mod - val[f])\*mm[t-f+1])%mod;

}

}ha;

## Manacher

struct manacher

{

char s[sz<<1],\*pre;

int pl[sz<<1],n,nn;

///abcba\0 -> $#a#b#c#b#a#\0

int exec(char \*\_ss)

{

pre = \_ss;

s[0]='$',s[1]='#',nn=2;

for(n=0;pre[n];n++)s[nn++]=pre[n],s[nn++]='#';s[nn]=0;

for(int i=1,f=pl[0]=0; i < nn;i++)

{

if(pl[f]+f>=i&&pl[f]-i+f!=pl[2\*f-i])

{

pl[i]=min(pl[f]-i+f,pl[2\*f-i]);

}

else

{

int &t=pl[i]=(pl[f]+f<i)?0:pl[2\*f-i];

while(s[i-t-1]==s[i+t+1])t++;

if(i+t>f+pl[f])f=i;

}

}

return n;

}

//hash声明 亦可用suffix\_array

mhash ha[2];

map<pii,int>lst[sz<<1];

map<int,int>ans[sz<<1];

manacher()

{

ha[0].init(31,1000000007);

ha[1].init(37,1000000013);

}

inline void add(int f,int t,int add)

{

pii val = make\_pair(ha[0].get(f,t),ha[1].get(f,t));

if(lst[t].find(val)==lst[t].end())

lst[t][val] = f;

ans[t][lst[t][val]]+=add;

}

void deal()

{

for(int i = 0; i <= n; i ++)lst[i].clear(),ans[i].clear();

ha[0].init(pre),ha[1].init(pre);//hash初始化

for(int i = 1; i < nn; i ++)if(pl[i])

add((i-1-pl[i])/2, pl[i], 1);

for(int i = n; i > 2; i --)

for(map<int,int>::iterator it=ans[i].begin();

it!=ans[i].end(); it ++)

add(it->A+1, i - 2, it->B);

}

void print(int f,int t)

{

for(int i = f; i <= t; i ++)

{

printf("%c",pre[i]);

}

printf("\n");

}

}manc;

/\*\*\*zoj3661\*\*\*/

void solve()

{

int n,m;scanf("%d%d",&n,&m);

static char ss[sz];

scanf("%s",ss);

manc.exec(ss);

manc.deal();

int vv[26];

for(int w = 0; w < m; w ++)

{

long long k;scanf("%lld",&k);

for(int j = 0; j < 26; j ++)scanf("%d",vv+j);

ha.init(vv);ha.init(ss);

vector<pair<int,int> >ans;

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

for(map<int,int>::iterator it = manc.ans[i].begin();

it != manc.ans[i].end(); it ++)

{

//printf("%d ",it->B);manc.print(it->A.A,it->A.B);

ans.push\_back(make\_pair(ha.get(it->A,(i+1)/2),it->B));

//printf("%d %d\n",ans.back().A,ans.back().B);

}

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

for(vector<pii>::iterator it=ans.begin();

it!=ans.end(); it++)

{

k -= it->B;

if(k<=0)

{

printf("%d\n",it->A);

break;

}

}

if(k>0)while(1);

}

printf("\n");

}

int main()

{

ha.init(26, 777777777);

int ti;scanf("%d",&ti);

for(int ca = 1; ca <= ti; ca++)

{

solve();

}

}

另一种suffix\_array 的方法：

suffix\_array sa;

map<int,int>ans[sz];

inline void add(int f,int t,int add)

{

int l = 0, r = sa.rank[f];

while(l+1<r)

{

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

if(sa.lcp(mid,sa.rank[f])<t)l = mid;

else r = mid;

}

f = sa.sa[r];

ans[t][f]+=add;

}

对lcp的修改：

inline int lcp(int a, int b)

{

if(a == b)return n;

if(a > b) swap(a, b);

int t = log[b - a];

return min(best[t][a + 1] , best[t][b - (1<<t) + 1]);

}

## Suffix\_Array o(nlgn)

int sa[sz],r[sz],h[sz];

void get\_suffix(char s[],int sa[],int r[],int h[],int len)

{

static int tmp[sz],top[sz];

int n=len;

int i,j,p,na=n<256?256:n;

memset(top,0,na\*sizeof(int));

for(i=0;i<n;i++)top[r[i]=s[i]&0xff]++;

for(i=0;i<na;i++)top[i]+=top[i-1];

for(i=0;i<n;i++)sa[--top[r[i]]]=i;

for(p=1;p<n;p<<=1)

{

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

{

j=sa[i]-p;

if(j<0)j+=n;

tmp[top[r[j]]++]=j;

}

sa[tmp[top[0]=0]]=j=0;

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

{

if(r[tmp[i]]!=r[tmp[i-1]]||r[tmp[i]+p]!=r[tmp[i-1]+p])

top[++j]=i;

sa[tmp[i]]=j;

}

memcpy(r,sa,n\*sizeof(int));

memcpy(sa,tmp,n\*sizeof(int));

if(j>=n-1)break;

}

//height=lcp(sa[i-1],sa[i]);

for(j=r[h[i=p=0]=0];i<n-1;i++,p++)

{

while(p>=0&&s[i]!=s[sa[j-1]+p])

{

h[j]=p--;

j=r[sa[j]+1];

}

}

}

## Suffix\_Arrar o(n)

#include<cstdio>

#include<cstring>

#include<algorithm>

using namespace std;

#define M 17

#define N (1<<M)

//sa数组从sa[1]到sa[n],存储的是0到n-1的排列

//sa[i]记录的是排名为i的后缀的起始位置

//rank数组从rank[0]到rank[n-1],存储的是1到n的排列

//rank[i]记录的是以i为起点的后缀的排名

struct suffix\_array

{

//dc3要求sa数组和r数组要开3\*N

int rank[N], sa[3\*N], init[3\*N], high[N], n;

int buc[N], m, wv[N], i, j ,k;

#define F(x) ((x)/3+((x)%3==1?0:tb))

#define G(x) ((x)<tb?(x)\*3+1:((x)-tb)\*3+2)

#define cmp1(r,a,b) (r[a]==r[b]&&r[a+1]==r[b+1]&&r[a+2]==r[b+2])

#define cmp3(r,a,b) (r[a]<r[b]||r[a]==r[b]&&wv[a+1]<wv[b+1])

#define cmp2(k,r,a,b) (k==2?(r[a]<r[b]||r[a]==r[b]&&cmp3(r,a+1,b+1)):cmp3(r,a,b))

inline void sort(int \*r, int \*a, int \*b, int n, int m)

{

for(i = 0; i < n; i++) wv[i] = r[a[i]];

for(i = 0; i < m; i++) buc[i] = 0;

for(i = 0; i < n; i++) buc[wv[i]]++;

for(i = 1; i < m; i++) buc[i] += buc[i-1];

for(i = n - 1; i >= 0; i--) b[--buc[wv[i]]] = a[i];

return;

}

inline void suffix\_dc3(int \*r, int \*sa, int n, int m)

{

int \*rn = r + n;

int \*san = sa + n, ta = 0, tb = (n + 1) / 3, tbc = 0, p, \*wa = rank ,\*wb = high;

r[n] = r[n+1] = 0;

for(i = 0; i < n; i++) if(i % 3 != 0) wa[tbc++] = i;

sort(r + 2, wa, wb, tbc, m);

sort(r + 1, wb, wa, tbc, m);

sort(r , wa, wb, tbc, m);

for(p = 1, rn[F(wb[0])] = 0, i = 1; i < tbc; i++)

rn[F(wb[i])] = cmp1(r, wb[i-1], wb[i]) ? p - 1 : p++;

if(p < tbc) suffix\_dc3(rn, san, tbc, p);

else for(i = 0; i < tbc; i++) san[rn[i]] = i;

for(i = 0; i < tbc; i++) if(san[i] < tb) wb[ta++] = san[i] \* 3;

if(n % 3 == 1) wb[ta++] = n - 1;

sort(r, wb, wa, ta, m);

for(i = 0; i < tbc; i++) wv[wb[i] = G(san[i])] = i;

for(i = 0, j = 0, p = 0; i < ta && j < tbc; p++)

sa[p] = cmp2(wb[j] % 3, r, wa[i], wb[j]) ? wa[i++] : wb[j++];

for(; i < ta; sa[p++] = wa[i++]);

for(; j < tbc; sa[p++] = wb[j++]);

}

inline int exec(char \*in)

{

for(int &p=n=m=0; in[p]; p++)

{

init[p] = in[p];

m = max(m, init[p] + 1);

}

init[n] = 0;

suffix\_dc3(init, sa, n + 1, m);

//后缀数组关键是求出high,所以求sa的时候顺便把rank和high求出来

for(i = 1; i <= n ; i++) rank[sa[i]] = i;

for(i = 0, k = 0; i < n ; high[rank[i++]] = k)

for(k?k--:0,j=sa[rank[i]-1]; init[i+k] == init[j+k] ; k++);

//当需要反复询问两个后缀的最长公共前缀时用到RMQ

for(i = 1; i <= n ; i ++) best[0][i] = high[i];

for(i = 1; i <= log[n] ; i ++)

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

best[i][j] = min(best[i-1][j] , best[i-1][j+(1<<i>>1)]);

return n;

}

int log[N],best[M][N];//RMQ变量

//调用格式lcp(sa[i],sa[j])//询问a,b后缀的最长公共前缀

//注意a==b

inline int lcp(int a, int b)

{

a = rank[a], b = rank[b];

if(a > b) swap(a, b);

int t = log[b - a];

return min(best[t][a + 1] , best[t][b - (1<<t) + 1]);

}

suffix\_array()

{

log[0] = -1;

for(i = 1; i < N ; i ++)log[i] = (i&(i-1)) ? log[i-1]:log[i-1]+1;

}

}sa;

## 后缀自动机

const int MAX\_N = 1000000 + 10;

struct State {

State\*suf, \*go[26], \*nxt;

int val, cnt;

State() :

suf(0), val(0) {

memset(go, 0, sizeof go);

}

}\*root, \*last;

State statePool[MAX\_N \* 2], \*cur;

State\*first[MAX\_N] = { };

void init() {

cur = statePool;

root = last = cur++;

}

void extend(int w) {

State\*p = last, \*np = cur++;

np->val = p->val + 1;

np->cnt = 1;

while (p && !p->go[w])

p->go[w] = np, p = p->suf;

if (!p)

np->suf = root;

else {

State\*q = p->go[w];

if (p->val + 1 == q->val) {

np->suf = q;

} else {

State\*nq = cur++;

memcpy(nq->go, q->go, sizeof q->go);

nq->val = p->val + 1;

nq->suf = q->suf;

q->suf = nq;

np->suf = nq;

while (p && p->go[w] == q)

p->go[w] = nq, p = p->suf;

}

}

last = np;

}

int main() {

string str;

cin >> str;

init();

int L = str.size();

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

extend(str[i] - 'a');

}

for (State\*i = statePool; i != cur; ++i)

i->nxt = first[i->val], first[i->val] = i;

for (int it = L; it >= 0; --it) {

for (State\*i = first[it]; i; i = i->nxt)

if (i->suf)

i->suf->cnt += i->cnt;

}

// cout << root->go[0]->go[0]->cnt << endl;

return 0;

}

# 数论

## 线性素数筛及欧拉函数

#define size 10000000

bool isp[size];

int prime[size], lp, phi[size];

void make\_prime\_and\_phi()

{

int i, j;

memset(isp, 0, sizeof(isp));

isp[0] = isp[1] = 1;

lp = 0;

phi[1] = 1;

for (i = 2; i < size; ++i)

{

if (isp[i]==0) {

prime[lp++] = i;

phi[i] = i - 1;

}

for (j = 0; j < lp && prime[j] \* i < size; ++j)

{

isp[prime[j] \* i] = 1;

if (i % prime[j] == 0) {

phi[i \* prime[j]] = prime[j] \* phi[i];

break;

} else {

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

}

}

}

}

## 欧拉函数

// test: phi(846720)=193536 ??

// return Euler's totient(n), O(sqrt(n))

int euler\_phi(int n)

{

int phi = n, p;

for (p = 2; p \* p <= n; ++p)

if (n % p == 0)

{

phi = phi / p \* (p - 1);

while (n % p == 0)

n /= p;

}

if (n > 1)

phi = phi / n \* (n - 1);

return phi;

}

## Lucas+C(n,m)

#include<cstdio>

int powm(int a,int p,int mod)

{

int ret=1;

while(p>0)

{

if(p&1)ret=a\*ret%mod;

a=a\*a%mod;

p>>=1;

}

return ret;

}

struct lucas

{

int \*jie,\*inv,mod;

void init(int \_mod)

{

mod = \_mod;

jie = new int[mod], inv = new int[mod];

jie[0]=1;

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

{

jie[i]=jie[i-1]\*i%mod;

inv[i]=powm(jie[i],mod-2,mod);

}

}

~lucas()

{

delete jie, inv;

}

int c(int n,int m)

{

int ret=1;

while(n>0)

{

int nn=n%mod,mm=m%mod;

if(mm>nn)return 0;

ret=ret\*jie[nn]%mod\*inv[mm]%mod\*inv[nn-mm]%mod;

n/=mod;

m/=mod;

}

return ret;

}

};

int main()

{

lucas \_23;\_23.init(23);

printf("%d ",\_23.c(19,9));//10

}

## 质因数分解

#include <cstdio>

#include <cstring>

#include <iostream>

#include <cmath>

#include <deque>

using namespace std;

typedef long long ll;

ll plusmod(ll a,ll b,ll p)

{

a+=b;

if(a>=p)a-=p;

return a;

}

ll mulmod(ll a,ll b,ll p)

{

ll y = (double)a\*b/p;

a= a\*b-y\*p;

if(a<0)a = a+p;

return a;

}

ll powm(ll a,ll p, ll mod)

{

ll ans = 1;

for(; p; p>>=1)

{

if(p & 1)ans=mulmod(ans, a, mod);

a=mulmod(a, a, mod);

}

return ans;

}

ll gcd(ll a, ll b)//a,b>=0

{

for(ll tmp; b; tmp=a,a=b,b=tmp%b);

return a;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\

http://en.wikipedia.org/wiki/Miller\_rabin

n < 1,373,653, a = {2,3};

n < 341,550,071,728,321, a = {2,3,5,7,11,13,17};

n < 9,080,191, a = {31,73};

n < 170,584,961, a = {350,3958281543};

n < 4,759,123,141, a = {2,7,61};

n < 75,792,980,677, a = {2,379215,457083754};

n < 21,652,684,502,221 a = {2,1215,34862,574237825};

n < 1LL<<64, a = {2,325,9375,28178,450775,9780504,1795265022};

if n is of long long, mulmod is needed.

test:

2047, 1373653, 25326001, 3215031751, 2152302898747,

3474749660383, 341550071728321, 341550071728321,

3825123056546413051, 382512305654641305(all are not prime)

\\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

ll need[]= {2,325,9375,28178,450775,9780504,1795265022};

const int needsz=7;

bool miller\_rabin(ll n)//n==1 useless here;

{

ll m,a;

int s,i,j;

if(n==2)return 1;

for(m=n-1,s=0; (m & 1)==0; m>>=1)s++;

if(s==0)return 0;

for(i = 0; i < needsz; i++)

{

a = powm(need[i],m,n);

if(a == 1) continue;

if(a == 0&&need[i]>=n)continue;

for(j = 1;a != n-1; j++)

{

if(a == 1||j > s)return 0;

a=mulmod(a,a,n);

}

}

return 1;

}#define f(x,c,mod) plusmod(mulmod(x, x, mod), c, mod)

ll pollard\_rho(ll n)

{

ll x,y,i,d;

for(i=1,d=n; d==n; i++)

for(x=2,y=2,d=1; d==1;)

{

x=f(x,i,n);

y=f(f(y,i,n),i,n);

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

}

return d;

}

ll pf[310],has[310];//3 times of length of pfs.

int factor(ll n)//factors in pf.

{

int cnt=0,l=1,r=1;

if(!(n&1))

{

pf[cnt]=2;

has[cnt]=0;

while(!(n&1))

{

has[cnt]++;

n>>=1;

}

cnt ++;

}

pf[r++]=n;

while(l<r)

{

n=pf[l++];

if(n==1)continue;

if(miller\_rabin(n))

{

pf[cnt]=n;

has[cnt]=1;

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

{

while(pf[i]%n==0)

{

has[cnt]++;

pf[i]/=n;

}

}

cnt ++;

}

else

{

pf[r++]=pollard\_rho(n);

pf[r++]=n/pf[r-1];

}

}

return cnt;

}

int main()

{

int t;

scanf("%d", &t);

while(t--)

{

ll n,ret;

scanf("%I64d", &n);

int cc = factor(n);

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

{

printf("%I64d %d\n",pf[i],has[i]);

}

}

}

中国剩余定理

//伪代码

BigInt[] congruence(BigInt[] A, BigInt[] B, BigInt[] M)

{

BigInt x = 0, m = 1;

for ( int i = 0; i < A.length; i++)

{

BigInt a = A[i] \* m, b = B[i] - A[i] \* x, d = a.gcd(M[i]);

if (b % d != 0) return null ;

x += m \* (b / d \* (a / d).modInv(M[i] / d) % (M[i] / d));

m \*= M[i] / d;

}

return new BigInt[] {x % m, m };

}

//java:

BigInteger[] china(BigInteger []aa,BigInteger[]bb,BigInteger[]mm)

{

BigInteger X=BigInteger. ZERO,M=BigInteger.ONE;

for(int i=0; i<aa.length; i++)

{

BigInteger a = aa[i].multiply(M),

b = bb[i].subtract(aa[i].multiply(X)),

d = a.gcd(mm[i]);

if(b.remainder(d).compareTo(BigInteger.ZERO)!=0)

return null ;

X=X.add(M.multiply(b.divide(d)

.multiply(a.divide(d).modInverse(mm[i].divide(d)))

.remainder(mm[i].divide(d))));

M=M.multiply(mm[i].divide(d));

}

return new BigInteger[] {X.remainder(M),M};

}

// x = cc[i] ( %mm[i] )

int deal(int n, int cc[], int mm[], ll&C, ll&M)//返回0则无解

{

ll x ,y, d, tt, m;

M = mm[0], C = cc[0];

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

{

m=mm[i];tt=cc[i]-C;

d=exgcd(M,m,x,y);

if(tt%d)return 0;

m/=d;

C+=tt/d\*x%m\*M;

M\*=m;

if(C<0)C+=M;

}

return 1;

}

## 平方剩余

from fduacm

///求解平方剩余，先要特判勒让得符号为-1与p=2的情况。

if(p%4==3)ans=powmod(n,(p+1)/4,p);

else

{

q=p-1,s=0;

while(q%2==0)q/=2,s++;

for(w=1; powmod(w,(p-1)/2,p)==1; w++);

r=powmod(n,(q+1)/2,p);

v=powmod(w,q,p);

\_n=powmod(n,p-2,p);

while(1)

{

for(i=0,t=r\*r%p\*\_n%p; t!=1; t=t\*t%p,i++);

if(i==0)

{

ans=r;

break;

}

r=r\*powmod(v,1LL<<((int)(s-i-1)),p)%p;

}

}

## 原根

If the multiplicative order of a number m modulo n is equal to fai(n), then it is a primitive root. In fact the converse is true: If m is a primitive root modulo n then the multiplicative order of m is fai(n). We can use this to test for primitive roots.

First, compute fai(n). Then determine the different prime factors of fai(n), say p(1),...,p(k). Now, for every element m, compute m^(fai(n)/p(i))%n for i=1,2,...,k using a fast algorithm for modular exponentiation such as exponentiation by squaring. A number m for which these k results are all different from 1 is a primitive root.

The number of primitive roots modulo n, if there are any, is equal to fai(fai(n)).

## Nim积

x ⊗ y = g(x, y) = mex{g(x, b) ⊕ g(a, y) ⊕ g(a, b) : 0 ≤ a < x, 0 ≤ b < y}.

#include <cstdio>

#include <cstring>

#include <cmath>

#include <queue>

#include <stack>

#include <vector>

#include <string>

#include <map>

#include <iostream>

#include <algorithm>

using namespace std;

typedef long long ll;

const int inf=(1<<30);

const int N=1005;

const int pw[]={2,4,16,256,65536}; //2^(2^a)

//----------------------------------------------------------------

int n,m;

int sg[20][20];

int nim\_multi\_power(int x,int y) { //确保x是一个2的幂

if(x<16) return sg[x][y];

int a=0; //找到 2^(2^a) <= x < 2^(2^(a+1))

while(pw[a+1]<=x) a++;

int M=pw[a]; //M=2^(2^a);

int p=x/M; //x=p\*M;

int s=y/M,t=y-s\*M; //y=s\*M+t;

int d1=nim\_multi\_power(p,s);

int d2=nim\_multi\_power(p,t);

return (d1^d2)\*M^nim\_multi\_power(M/2,d1);

}

int nim\_multi(int x,int y) {

if(x<y) return nim\_multi(y,x);

if(x<16) return sg[x][y];

int a=0; //找到 2^(2^a) <= x < 2^(2^(a+1))

while(pw[a+1]<=x) a++;

int M=pw[a]; //M=2^(2^a);

int p=x/M,q=x-p\*M; //x=p\*M+q;

int s=y/M,t=y-s\*M; //y=s\*M+t;

int c1=nim\_multi(p,s);

int c2=nim\_multi(p,t)^nim\_multi(q,s);

int c3=nim\_multi(q,t);

return (c1^c2)\*M^c3^nim\_multi\_power(M/2,c1);

}

void init\_nim() { //打16\*16的表

memset(sg,-1,sizeof(sg));

bool hash[1005];

for(int i=0;i<16;i++) {

for(int j=0;j<16;j++) {

memset(hash,0,sizeof(hash));

for(int k1=1;k1<=i;k1++)

for(int k2=1;k2<=j;k2++) {

if(k1==i) {

if(k2==j) hash[0]=1;

else hash[sg[k1][k2]]=1;

}else {

if(k2==j) hash[sg[k1][k2]]=1;

else hash[sg[k1][j]^sg[i][k2]^sg[k1][k2]]=1;

}

}

for(int k1=0;;k1++)

if(!hash[k1]) { sg[i][j]=k1; break; }

}

}

}

/\*\*\*hdu3404\*\*\*/

int main()

{

int i,j,t,cas=0;

int x,y;

init\_nim();

scanf("%d",&t);

while(t--) {

scanf("%d",&n);

int nim=0;

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

scanf("%d%d",&x,&y);

nim^=nim\_multi(x,y);

}

if(nim) puts("Have a try, lxhgww.");

else puts("Don't waste your time.");

}

return 0;

}

/\*

Nim积的性质: (+表示Nim和运算,x表示Nim积运算)

1.X x 2^(2^a) = 2^(2^a) \* X

2.X x Y < 2^(2^a)

3.2^(2^a) x 2^(2^a) = (3/2) \* 2^(2^a)

运算规则:类似四则运算的加和乘.

\*/

# 杂项

## FFT

#include<algorithm>

#include<cstdio>

#include<cstring>

#include<cmath>

using namespace std;

const double PI = acos(-1);

const int MAXL = (1 << 18)+100;

struct cp

{

double re,im;

cp(double re=0,double im=0):re(re),im(im){}

}ya[MAXL],yb[MAXL];

int id=0;

template<typename T>void X(cp to[],T f[],int n,int m=1,int d=0)

{

if (m == n)to[d] = f[id++];

else X(to,f,n,m<<1,d),X(to,f,n,m<<1,d+m);

}

void FFT(cp P[],int n,int oper)

{

for (int d = 0;(1 << d) < n;++d)

{

int m = (1 << d);

double p0 = PI / m \* oper;

double sinp0 = sin(p0), cosp0 = cos(p0);

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

{

double sinp = 0, cosp = 1;

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

{

double ta = cosp \* P[i+j+m].re - sinp\*P[i+j+m].im;

double tb = cosp \* P[i+j+m].im + sinp\*P[i+j+m].re;

P[i+j+m].re = P[i+j].re - ta;

P[i+j+m].im = P[i+j].im - tb;

P[i+j].re += ta;

P[i+j].im += tb;

double tsinp = sinp;

sinp = sinp \* cosp0 + cosp \* sinp0;

cosp = cosp \* cosp0 -tsinp \* sinp0;

}

}

}

}

///na大于a的最大非零数

int convolution(int \*a,int na,int \*b,int nb,int \*r)

{

int n,i;

double x,y;

for(n=1;n<(na+nb);n<<=1);

id = 0;X(ya,a,n);FFT(ya,n,1);

id = 0;X(yb,b,n);FFT(yb,n,1);

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

{

x = ya[i].re \* yb[i].re - ya[i].im \* yb[i].im;

y = ya[i].re \* yb[i].im + ya[i].im \* yb[i].re;

yb[i] = cp(x,y);

}

id = 0;X(ya,yb,n);FFT(ya,n,-1);

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

r[i] = (int)(ya[i].re / n + 0.5);

return n;//length of r

}

/\*\*hdu 4609\*\*/

int a[MAXL],c[MAXL];

int main()

{

int ti;scanf("%d",&ti);

for(int ca=1;ca<=ti;ca++)

{

int n;scanf("%d",&n);

memset(a,0,sizeof(a));

int mm=0;

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

{

int tmp;

scanf("%d",&tmp);

mm=max(tmp,mm);

a[tmp]++;

}

mm++;

convolution(a,mm,a,mm,c);

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

{

c[i+i]-=a[i];

}

double ans=0;

double has=n;

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

{

has-=a[i-1];

ans+=c[i]/2\*has;

}

double ss=1.\*n\*(n-1)\*(n-2)/6.;

printf("%.7f\n",1.-ans/ss);

}

}

## Tree count

///1. Counting the number of labeled unrooted trees with n nodes

///2. Counting the number of labeled rooted trees with n nodes

///3. Counting the number of unlabeled rooted trees with n nodes

///4. Counting the number of unlabeled unrooted trees with n nodes

///Calculate the answer modulo p.

int a[r],s[r][r],k,n,p;

int pow(int a,int b,int p)

{

int ans=1;

a%=p;

while(b>0)

{

if(b&1)

ans=ans\*a%p;

a=a\*a%p;

b/=2;

}

return ans;

}

void extended\_gcd(int a,int b,int c,int &x,int &y)

{

if(b==0)x=c/a,y=0;

else

{

extended\_gcd(b,a%b,c,y,x);

y-=a/b\*x;

}

}

int main()

{

int tmp,\_\_;

while(scanf("%d%d%d",&k,&n,&p)!=-1)

{

switch(k)

{

case 1:

printf("%d\n",pow(n,n-2,p));

break;

case 2:

printf("%d\n",pow(n,n-1,p));

break;

default:

a[1]=1;

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

{

a[N+1]=0;

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

{

if(N-i<i) s[N][i]=a[N+1-i];

else s[N][i]=(s[N-i][i]+a[N+1-i])%p;

a[N+1]+=s[N][i]\*i%p\*a[i]%p;

}

\_\_=a[N+1]%p;

extended\_gcd(N,p,\_\_,a[N+1],tmp);

if(a[N+1]>=0) a[N+1]%=p;

else a[N+1]=p-(-a[N+1])%p;

}

if(k==4)

{

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

if((n&1)==0) a[n]+=a[n/2]\*(a[n/2]+1)/2;

}

printf("%d\n",a[n]%p);

break;

}

}

}

## 额外栈空间

int size = 10 << 20; // 256MB

char \*p = (char\*)malloc(size) + size;

\_\_asm\_\_("movl %0, %%esp\n" :: "r"(p) );

## 矩阵类~java

class Mat {

static final int sz = 4;

public BigInteger[][] a;

public Mat() {

a = new BigInteger[sz][sz];

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

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

a[i][j] = BigInteger.valueOf(0);

}

}

public Mat mul(Mat b) {

Mat ret = new Mat();

for (int k = 0; k < sz; k++)

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

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

ret.a[i][j] = ret.a[i][j].add(a[i][k].multiply(b.a[k][j]));

}

return ret;

}

public Mat pow(int p) {

Mat ret = I, che = this;

while (p > 0) {

if ((p & 1) > 0) {

ret = ret.mul(che);

}

p >>= 1;

che = che.mul(che);

}

return ret;

}

public String toString() {

String ret = "";

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

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

ret += a[i][j].toString() + " ";

ret += '\n';

}

return ret;

}

static public Mat I = new Mat();

static {

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

I.a[i][i] = BigInteger.ONE;

}

}

}

## 分数类~java

class Rat implements Comparable<Rat> {

BigInteger num, div;

public Rat() {

this.num = BigInteger.ZERO;

this.div = BigInteger.ONE;

}

public Rat(BigInteger num, BigInteger div) {

BigInteger g = num.gcd(div);

num = num.divide(g);

div = div.divide(g);

if (div.compareTo(BigInteger.ZERO) < 0) {

num = num.negate();

div = div.negate();

}

this.num = num;

this.div = div;

}

static Rat valueOf(long num) {

return new Rat(BigInteger.valueOf(num), BigInteger.ONE);

}

static Rat valueOf(long num, long div) {

return new Rat(BigInteger.valueOf(num), BigInteger.valueOf(div));

}

public Rat mul(Rat b) {

return new Rat(num.multiply(b.num), div.multiply(b.div));

}

public Rat add(Rat b) {

return new Rat(num.multiply(b.div).add(b.num.multiply(div)),

div.multiply(b.div));

}

public Rat sub(Rat b) {

return new Rat(num.multiply(b.div).subtract(b.num.multiply(div)),

div.multiply(b.div));

}

public Rat div(Rat b) {

return new Rat(num.multiply(b.div), div.multiply(b.num));

}

public int compareTo(Rat b) {

return num.multiply(b.div).compareTo(div.multiply(b.num));

}

public double doubleValue() {

return num.doubleValue() / div.doubleValue();

}

public String toString() {

return num.toString() + "/" + div.toString();

}

static String test(){

Rat a[] = {Rat.valueOf(1, -1),Rat.valueOf(2, -2),Rat.valueOf(2,-1)};

Arrays.sort(a);

StringBuilder r = new StringBuilder();

r.append(a[0]);

r.append(" ");

r.append(a[1]);

r.append(" ");

r.append(a[2]);

return r.toString();

}

}

## 标准模板~java

public class Main {

static {

try {

// System.setIn(new FileInputStream("1000.in"));

// System.setOut(new PrintStream(new FileOutputStream("1000.out")));

} catch (Throwable e) {

}

}

static Scanner in = new Scanner(new BufferedInputStream(System.in));

static PrintWriter out = new PrintWriter(new BufferedOutputStream(

System.out));

void solve() {

int val = in.nextInt();

out.println(val);

}

public static void main(String[] args) {

new Main().solve();

out.flush();

}

}

## 注意事项~java

///Decimai舍入

public class Main {

public static void main(String []args){

BigDecimal a = BigDecimal.TEN;

BigDecimal b = BigDecimal.valueOf(3);

a = a.divide(b, 3, BigDecimal.ROUND\_HALF\_EVEN);

System.out.println(a);

}

}

///排序，见分数类(Comparable,arrays.sort)

# Reference

## <algorithm>

void swap (T& a, T& b);

void fill (it first,it last,const T& val);

void rotate (it first,it middle,it last);

//1,2,3,4,5,6,7,8,9

//rotate(a,a+3,a+9)

//4,5,6,7,8,9,1,2,3

Void reverse(it first,it last);

void random\_shuffle (it first,it last);

void nth\_element (it first,it nth,it last);

it lower\_bound (it first,it last,const T& val);

it upper\_bound (it first,it last,const T& val);

it merge (it first1,it last1,it first2,it last2,it result);

it min\_element (it first, it last);

it max\_element (it first, it last);

bool next\_permutation (it first,it last);

bool prev\_permutation (it first,it last);

## <bitset>

bool all();//Test if all are 1

bool any();//Returns whether any is 1

size\_t count();//the number of 1

bitset::operator[]

bitset& reset();//all bit

bitset& reset(size\_t pos);// single bit

bitset& set();//all bit

bitset& set(size\_t pos);// single bit

bitset& flip();//all bit

bitset& flip(size\_t pos);// single bit

operator : &,^,|,<<,>>,~

## Container

<string>

string substr (size\_t pos = 0, size\_t len = npos) const;

<map>

Complexity on sway(map a,map b) is Constant. ///交换指针的10倍时间~1000000次~0.064s

<set>

Complexity on sway(set a,set b) is Constant.///交换指针的10倍时间~1000000次~0.085s

size\_type count (const value\_type& val) const;///multiset中不要用

<vector> ~o(n):

iterator insert (iterator position, const value\_type& val); ///在pos前插入

void insert (iterator position, size\_type n, const value\_type& val);

void insert (iterator position, InputIterator first, InputIterator last);

iterator erase (iterator position);

iterator erase (iterator first, iterator last);

## <ctime>

int t = clock();

printf("%f\n",clock()\*1./CLOCKS\_PER\_SEC);

## <cctype>

isalnum/// isalpha || isdigit

isalpha///isupper || islower

iscntrl

isdigit///0~9

isgraph/// isprint without ' '

islower///a~z

isprint/// an ASCII code greater than 0x1f (US), except 0x7f (DEL).

ispunct/// isgraph without isalnum

isspace/// ' ','\t','\n','\v','\f','\r'

isupper///A~Z

isxdigit///0 1 2 3 4 5 6 7 8 9 a b c d e f A B C D E F

## <cstdio>

fflush(FILE \* stream);

FILE \* fopen ( const char \* filename, const char \* mode );

## 宏定义

#define lowbit(x) ((x)&-(x))

#out(x) cerr << #x << x << endl

#define tr(it , container) \

for(typeof(container.begin()) it = container.begin(); it != container.end(); it++)

# 数学知识

## 组合数

***C(n,m)= C(n,n-m)= C(n-1,m-1)+C(n-1,m)***

***k \* C(n, k) = n \* C(n-1, k-1);***

***C(n, 0) + C(n, 2) + ... = C(n, 1) + C(n, 3) + ...***

***1 \* C(n, 1) + 2 \* C(n, 2) + ... + n \* C(n, n) = n \* 2^(n-1)***

## Stirling数

只准备讨论其中的第二类Stirling数,至于第一类的Stirling数只准备给出它的定义和递推关系。

【定义】称S(n,0),S(n,1), …,S(n,n)为第一类Stirling数。

显然有S(n+1,k)=S(n,k-1)-nS(n,k) (2-10-5)

【定义】n个有区别的球放到m个相同的盒子中,要求无一空盒,其不同的方案数用S(n,m) 表示,称为第二类Stirling数。

【例题】

例如红,黄,蓝,白四种颜色的球,放到两个无区别的盒子里,不允许有空盒,其方案有如下七种：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
|  | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 第1盒子 | r | y | b | w | ry | rb | rw |
| 第2盒子 | ybw | rbw | ryw | ryb | bw | yw | Yb |

其中r表红球,y表黄球,b表蓝球,w表白球,

∴S(4,2)=7。

【定理】

第二类StirlingS(n,k) 有下列性质:

***(a) S(n,0)=0,***

***(b) S(n,1)=1,***

***(c) S(n,2)=2n-1-1,***

***(d) S(n,n-1)=C(n,2)***

***(e) S(n,n)=1。***

【证明】

公式(a),(b),(e)是显然的,只证(c),(d)。

(c)设有n个不相同的球,,...,从中取出球,其余的n-1个球,每个都有与同盒,或不与同盒两种选择.但必须排除一种情况,即全体与同盒,因这时另一盒将是空盒。故实际上只有-1 种方案, 即S(n,2)=-1

(d)n个球放到n-1个盒子里,不允许有一空盒,故必有一盒有两个球,从n个有区别的球中取2个共有C(n,2)种组合方案。

【定理】

第二类Stirling数满足下面的递推关系，

S(n,m)=mS(n-1,m)+S(n-1,m-1),(n≥1,m≥1)　　　(2-10-6)

【证明】

设有n个有区别的球,,...,从中取一个球设为。把n个球放到m个盒子无一空盒的方案的全体可分为两类：

（a）独占一盒，其方案数显然为S(n-1,m-1)

（b）不独占一盒，这相当于先将剩下的n-1个球放到m个盒子，不允许空盒，共有S(n-1,m)种不同方案，然后将球放进其中一盒，从乘法法则得不独占一盒的方案数应为mS(n-1,m)。

根据加法法则有 S(n,m)=S(n-1,m-1)+mS(n-1,m)。

上面证明递推公式(2-10-6)的过程，也就是给出构造所有方案的办法。例如今将红、黄、蓝、白、绿五个球放到无区别的两个盒子里。

S(5,2)=2S(4,2)+S(4,1)=2×7+1=15。

故共有15种不同的方案。

先把绿球取走，余下的四个球放到两个盒子的方案已见前面的例子。和前面一样用 r，y，b，w分别表示红，黄，蓝，白球，绿球用g表示，故得表 2-10-1。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| g不独占一盒 | | | | g独占一盒 | |
| 第1盒子 | 第2盒子 | 第1盒子 | 第2盒子 | 第1盒子 | 第2盒子 |
| rg yg bg wg ryg rbg rwg | ybw rbw ryw ryb bw yw yb | r y b w ry rb rw | ybwg rbwg rywg rybg bwg ywg ybg | g | rybw |

n个球放到m个盒子里，依球和盒子是否有区别？是否允许空盒？共有=8 种状态。其方案计数下面分别列出。

　　· n个球有区别，m个盒子有区别，有空盒时方案计数为mn

　　· n个球有区别，m个盒子有区别，无空盒时方案计数为m!S(n,m)

　　· n个球有区别，m个盒子无区别，有空盒时方案计数为

　　· n个球有区别，m个盒子无区别，无空盒时方案计数为S(n,m)

　　· n个球无区别，m个盒子有区别，有空盒时方案计数为C(n+m-1,n)

　　· n个球无区别，m个盒子有区别，无空盒时方案计数为

　　　　C(m+(n-m)-1,n-m)=C(n-1,n-m)=C(n-1,m-1)。

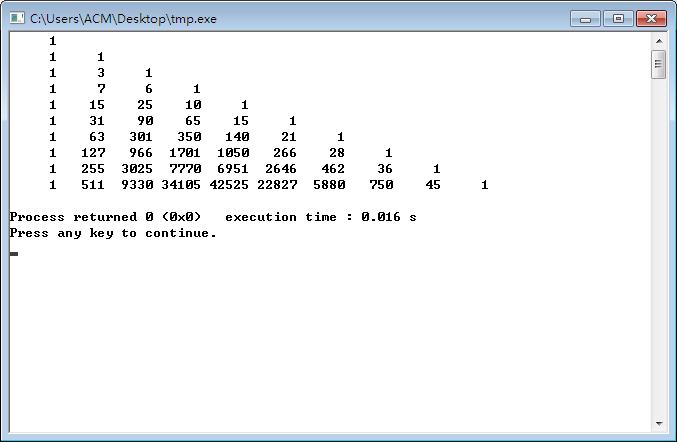
　　· n个球无区别，m个盒子无区别，有空盒时方案计数为

　　　　 的xn项系数。

　　· n个球无区别，m个盒子无区别，无空盒时方案计数为

　　　　 的xn项系数。

利用公式S(n,m)=S(n-1,m-1)+mS(n-1,m),S(n,1)=S(n,n)=1，还可以如Pascal三角形一样得到表2-10-3。



## Catalan数

卡特兰数

　　卡特兰数前几项为 : 1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, 16796, 58786, 208012, 742900, 2674440, 9694845, 35357670, 129644790, 477638700, 1767263190, 6564120420, 24466267020, 91482563640, 343059613650, 1289904147324, 4861946401452, ...

英文名

　　Catalan number

原理

　　令h(0)=1,h(1)=1，catalan数满足递推式[1]：

　　h(n)= h(0)\*h(n-1)+h(1)\*h(n-2) + ... + h(n-1)h(0) (n>=2)

　　例如：h(2)=h(0)\*h(1)+h(1)\*h(0)=1\*1+1\*1=2

　　h(3)=h(0)\*h(2)+h(1)\*h(1)+h(2)\*h(0)=1\*2+1\*1+2\*1=5

　　另类递推式[2]：

***h(n)=h(n-1)\*(4\*n-2)/(n+1);***

　　递推关系的解为：

***h(n)=C(2n,n)/(n+1) (n=1,2,3,...)***

　　递推关系的另类解为：

***h(n)=c(2n,n)-c(2n,n+1)(n=1,2,3,...)***

**卡特兰数的应用**

　　实质上都是递推等式的应用

**括号化**

　　矩阵链乘： P=a1×a2×a3×……×an，依据乘法结合律，不改变其顺序，只用括号表示成对的乘积，试问有几种括号化的方案？(h(n)种)

**出栈次序**

　　一个[栈](http://baike.baidu.com/view/38877.htm)(无穷大)的[进栈](http://baike.baidu.com/view/346788.htm)序列为1，2，3，…，n，有多少个不同的[出栈](http://baike.baidu.com/view/346791.htm)序列?

　　常规分析

　　首先，我们设f（n）=序列个数为n的出栈序列种数。同时，我们假定第一个出栈的序数是k。

　　第一个出栈的序数k将1~n的序列分成两个序列，其中一个是1~k-1，序列个数为k-1，另外一个是k+1~n，序列个数是n-k。

　　此时，我们若把k视为确定一个序数，那么根据乘法原理，f（n）的问题就等价于——序列个数为k-1的出栈序列种数乘以序列个数为n - k的出栈序列种数，即选择k这个序数的f（n）=f（k-1）×f（n-k）。而k可以选1到n，所以再根据加法原理，将k取不同值的序列种数相加，得到的总序列种数为：f（n）=f（0）f（n-1）+f（1）f（n-2）+……+f（n-1）f（0）。

　　看到此处，再看看卡特兰数的递推式，答案不言而喻，即为f（n）=h（n）= C（2n,n）/（n+1）= c（2n,n）-c（2n,n+1）（n=1，2，3，……）。

　　最后，令f（0）=1，f（1）=1。

　　非常规分析

　　对于每一个数来说，必须进栈一次、出栈一次。我们把进栈设为状态‘1’，出栈设为状态‘0’。n个数的所有状态对应n个1和n个0组成的2n位二进制数。由于等待入栈的操作数按照1‥n的顺序排列、入栈的操作数b大于等于出栈的操作数a(a≤b)，因此输出序列的总数目=由左而右扫描由n个1和n个0组成的2n位二进制数，1的累计数不小于0的累计数的方案种数。

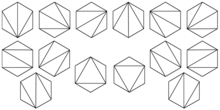
　　在2n位二进制数中填入n个1的方案数为c(2n,n),不填1的其余n位自动填0。从中减去不符合要求（由左而右扫描，0的累计数大于1的累计数）的方案数即为所求。

　　不符合要求的数的特征是由左而右扫描时，必然在某一奇数位2m+1位上首先出现m+1个0的累计数和m个1的累计数，此后的2(n-m)-1位上有n-m个 1和n-m-1个0。如若把后面这2(n-m)-1位上的0和1互换，使之成为n-m个0和n-m-1个1，结果得1个由n+1个0和n-1个1组成的2n位数，即一个不合要求的数对应于一个由n+1个0和n-1个1组成的排列。

　　反过来，任何一个由n+1个0和n-1个1组成的2n位二进制数，由于0的个数多2个，2n为偶数，故必在某一个奇数位上出现0的累计数超过1的累计数。同样在后面部分0和1互换，使之成为由n个0和n个1组成的2n位数，即n+1个0和n-1个1组成的2n位数必对应一个不符合要求的数。

　　因而不合要求的2n位数与n+1个0，n－1个1组成的排列一一对应。

　　显然，不符合要求的方案数为c(2n,n+1)。由此得出输出序列的总数目***=c(2n,n)-c(2n,n+1)=c(2n,n)/(n+1)=h(n+1)。***



**买票找零**

　　有2n个人排成一行进入剧场。入场费5元。其中只有n个人有一张5元钞票，另外n人只有10元钞票，剧院无其它钞票，问有多少中方法使得只要有10元的人买票，售票处就有5元的钞票找零？(将持5元者到达视作将5元入栈，持10元者到达视作使栈中某5元出栈)

**凸多边形三角划分**

　　在一个[凸多边形](http://baike.baidu.com/view/363944.htm)中，通过若干条互不相交的对角线，把这个多边形划分成了若干个[三角形](http://baike.baidu.com/view/5670.htm)。现在的任务是键盘上输入凸多边形的边数n，求不同划分的方案数f（n）。比如当n=6时，f（6）=14。

　　分析

　　如果纯粹从f（4）=2，f（5）=5，f（6）=14，……，f（n）=n慢慢去归纳，恐怕很难找到问题的递推式，我们必须从一般情况出发去找规律。

　　因为凸多边形的任意一条边必定属于某一个三角形，所以我们以某一条边为基准，以这条边的两个顶点为起点P1和终点Pn（P即Point），将该凸多边形的顶点依序标记为P1、P2、……、Pn，再在该凸多边形中找任意一个不属于这两个点的顶点Pk（2<=k<=n-1），来构成一个三角形，用这个三角形把一个凸多边形划分成两个凸多边形，其中一个凸多边形，是由P1，P2，……，Pk构成的凸k边形（顶点数即是边数），另一个凸多边形，是由Pk，Pk+1，……，Pn构成的凸n-k+1边形。

　　此时，我们若把Pk视为确定一点，那么根据乘法原理，f（n）的问题就等价于——凸k多边形的划分方案数乘以凸n-k+1多边形的划分方案数，即选择Pk这个顶点的f（n）=f（k）×f（n-k+1）。而k可以选2到n-1，所以再根据加法原理，将k取不同值的划分方案相加，得到的总方案数为：f（n）=f（2）f（n-2+1）+f（3）f（n-3+1）+……+f（n-1）f（2）。看到此处，再看看卡特兰数的递推式，答案不言而喻，即为f（n）=h（n-1） （n=2，3，4，……）。

　　最后，令f（2）=1，f（3）=1。

　　此处f（2）=1和f（3）=1的具体缘由须参考详尽的“卡特兰数”，也许可从凸四边形f（4）=f（2）f（3）+ f（3）f（2）=2×f（2）f（3）倒推，四边形的划分方案不用规律推导都可以知道是2，那么2×f（2）f（3）=2，则f（2）f（3）=1，又f（2）和f（3）若存在的话一定是整数，则f（2）=1，f（3）=1。（因为我没研究过卡特兰数的由来，此处仅作刘抟羽的臆测）。

　　类似问题

　　一位大城市的律师在她住所以北n个街区和以东n个街区处工作。每天她走2n个街区去上班。如果她从不穿越（但可以碰到）从家到办公室的对角线，那么有多少条可能的道路？

　　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？

**给定节点组成二叉树**

　　给定N个[节点](http://baike.baidu.com/view/47398.htm)，能构成多少种不同的[二叉树](http://baike.baidu.com/view/88806.htm)？

　　（能构成h（N）个）

　　（这个公式的下标是从h(0)=1开始的）

**卡特兰数的扩展**

　　对于在n位的2进制中，有m个0，其余为1的catalan数为：C（n,m）-C(n,m-1)。证明可以参考标准catalan数的证明。

　　问题1的描述：有n个1和m个-1（n>=m），共n+m个数排成一列，满足对所有0<=k<=n+m的前k个数的部分和Sk > 0的排列数。 问题等价为在一个格点阵列中，从（0，0）点走到（n，m）点且不经过对角线x==y的方法数（x > y）。

*考虑情况I*：第一步走到（0，1），这样从（0，1）走到（n，m）无论如何也要经过x==y的点，这样的方法数为(( n+m-1,m-1 ));

*考虑情况II*：第一步走到（1，0），又有两种可能：

　　a . 不经过x==y的点；（所要求的情况）

　　b . 经过x==y的点，我们构造情况II.b和情况I的一一映射，说明II.b和I的方法数是一样的。设第一次经过x==y的点是（x1，y1），将（0，0）到（x1，y1）的路径沿对角线翻折，于是唯一对应情况I的一种路径；对于情况I的一条路径，假设其与对角线的第一个焦点是（x2，y2），将（0，0）和（x2，y2）之间的路径沿对角线翻折，唯一对应情况II.b的一条路径。

　　问题的解就是总的路径数 ((n+m, m)) - 情况I的路径数 - 情况II.b的路径数。

***((n+m , m)) - 2\*((n+m-1, m-1))***

**或：*((n+m-1 , m)) - ((n+m-1 , m-1))***

　　问题2的描述：有n个1和m个-1（n>=m），共n+m个数排成一列，满足对所有0<=k<=n+m的前k个数的部分和Sk >= 0的排列数。（和问题1不同之处在于此处部分和可以为0，这也是更常见的情况） 问题等价为在一个格点阵列中，从（0，0）点走到（n，m）点且不穿过对角线x==y的方法数（可以走到x==y的点）。

　　把（n，m）点变换到（n+1，m）点，问题变成了问题1。

　　方法数为：

***((n+m+1, m)) - 2\*((n+m+1-1, m-1))***

**或：*((n+m+1-1, m)) - ((n+m+1-1, m-1))***

## Bell数

贝尔数以埃里克•坦普尔•贝尔(Eric Temple Bell)为名，是组合数学中的一组整数数列，开首是（OEIS的A000110数列）：

1, 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597, 27644437, 190899322, 1382958545, 10480142147, 82864869804, 682076806159, 5832742205057, 51724158235372, 474869816156751……

Bn是基数为n的集合的划分方法的数目。集合S的一个划分是定义为S的两两不相交的非空子集的族，它们的并是S。例如B3 = 5因为3个元素的集合{a, b, c}有5种不同的划分方法。

贝尔数适合递推公式：

它们也适合“Dobinski公式”：

期望值为1的泊松分数的n次矩。

它们也适合“Touchard同余”：若p是任意质数，那么

每个贝尔数都是"第二类Stirling数"的和

Stirling数S（n, k）是把基数为n的集划分为正好k个非空集的方法的数目。

把任一[概率分布](http://zh.wikipedia.org/wiki/%E6%A6%82%E7%8E%87%E5%88%86%E5%B8%83)的n次[矩](http://zh.wikipedia.org/wiki/%E7%9F%A9)以首n个[累积量](http://zh.wikipedia.org/wiki/%E7%B4%AF%E7%A9%8D%E9%87%8F)表示的多项式，其系数和正是第n个贝尔数。这种数划分的方法不像用Stirling数那个方法粗糙。

贝尔数的[指数母函数](http://zh.wikipedia.org/wiki/%E6%AF%8D%E5%87%BD%E6%95%B0)是：

用以下方法建构一个三角矩阵：

第一行第一项是1

对于n>1，第n行第一项等同第n-1行最后一项。

对于m,n>1，第n行第m项等于它左边和左上方的两个数之和。

\begin{array}{cccccccccccccccccc}
1 \\
1 & & 2 & & & & & & &\\
2 & & 3 & & 5 & & & & & &\\
5 & & 7 & & 10 & & 15 & & & & &\\
15 & & 20 & & 27 & & 37 & & 52 & & & &\\
52 & & 67 & & 87 & & 114 & & 151 & & 203 & & &\\
203 & & 255 & & 322 & & 409 & & 523 & & 674 & & 877 & &\\
877 & & 1080 & & 1335 & & 1657 & & 2066 & & 2589 & & 3263 & & 4140 &\\
& & & & &\vdots & & & & \vdots & & & & \vdots& & & & \\
\end{array}

每行首项是贝尔数。每行之和是第二类Stirling数。