板子

simpson

```
double work(double L,double R){
    double mid=(L+R)/2;
    return (f(L)+f(R)+f(mid)*4)*(R-L)/6;
}
double simpson(double L,double R){
    double mid=(L+R)/2;
    double x1=work(L,mid),x2=work(mid,R),x3=work(L,R);
    if (std::fabs(x1+x2-x3)<eps) return x1+x2;
    else return simpson(L,mid)+simpson(mid,R);
}
```

三分

```
while (fabs(lx - rx) > eps || fabs(ly - ry) > eps) {
    x1 = lx + (rx - lx) / 3; x2 = lx + (rx - lx) / 3 * 2;
    y1 = ly + (ry - ly) / 3; y2 = ly + (ry - ly) / 3 * 2;
    t1 = cal(x1, y1); t2 = cal(x2, y2);
    if (t1 > t2) {
        lx = x1, ly = y1;
    } else {
        rx = x2, ry = y2;
    }
}
```

随机增量

```cpp
const int maxn=1e5+100;
const double eps = 1e-8;
struct node {
    double x, y;
    node operator+(node a) {return (node){(a.x+x)/2, (a.y+y)/2};}
    node operator-(node a) {return (node){x-a.x, y-a.y};}
    node operator/(double a) {return (node){x/a, y/a};}
    node rev() {return (node) {y, -x};}
};
node p[maxn];
double dis(node a, node b) {
    return sqrt((a.x-b.x)*(a.x-b.x) + (a.y-b.y)*(a.y-b.y));
}
node center(node a, node b) {
    return (node) {(a.x+b.x)/2, (a.y+b.y)/2};
}
double sqr(double a) {
    return a * a;
}
node cal(double x, double y, double z, double u, double v, double w) {
    node t;
    t.x = (z*v-w*y)/(x*v-u*y); t.y = (z-t.x*x)/y;
    return t;
}
node center(node a, node b, node c) {
    double a1, a2, b1, b2, c1, c2;
    node ans;
    a1=2*(b.x-a.x),b1=2*(b.y-a.y),c1=sqr(b.x)-sqr(a.x)+sqr(b.y)-sqr(a.y);
    a2=2*(c.x-a.x),b2=2*(c.y-a.y),c2=sqr(c.x)-sqr(a.x)+sqr(c.y)-sqr(a.y);
    if(fabs(a1) < eps) {
        ans.y=c1/b1;
        ans.x=(c2-ans.y*b2)/a2;
    } else if(fabs(b1) < eps) {
        ans.x=c1/a1;
        ans.y=(c2-ans.x*a2)/b2;
    } else {
        ans.x=(c2*b1-c1*b2)/(a2*b1-a1*b2);
        ans.y=(c2*a1-c1*a2)/(b2*a1-b1*a2);
    }
    return ans;
}
int main() {
    int n; scanf("%d", &n);
    srand(time(0));
    for (int i = 1; i <= n; i++) {
        scanf("%lf%lf", &p[i].x, &p[i].y);
    }
    random_shuffle(p+1, p+1+n);
    node o = p[1];
    double r = 0;
    for (int i = 2; i <= n; i++) if (dis(o, p[i]) > r + eps) {
        o = center(p[1], p[i]); r = dis(o, p[i]);
        for (int j = 1; j < i; j++) if (dis(o, p[j]) > r + eps){
            o = center(p[i], p[j]); r = dis(o, p[j]);
            for (int k = 1; k < j; k++) if (dis(o, p[k]) > r + eps) {
                o = center(p[i], p[j], p[k]); r = dis(o, p[k]);
            }
        }
    }
    printf("%.10f\n%.10f %.10f", r, o.x, o.y);
    return 0;
}
```

高斯消元

```
const int maxn=1700;
int a[maxn][maxn];
int n, m;
int var, equ;
int ans[1700];
vector<int> free_x;
int g() {
    int mxr, col, k;
    for (k=0, col=0; k<equ && col<equ; k++, col++) {
        mxr=k;
        for (int i=k+1; i<equ; i++) {
            if (abs(a[mxr][col])<abs(a[i][col])) mxr=i;
        }
        if (a[mxr][col]==0) {
            free_x.push_back(col), k--;
            continue;
        }
        if (mxr!=k) {
            for (int j=col; j<var+1; j++) {
                swap(a[mxr][j], a[k][j]);
            }
        }
        for (int i=k+1; i<equ; i++) {
            if (a[i][col]!=0) {
                for (int j=col; j<var+1; j++) {
                    a[i][j]^=a[k][j];
                }
            }
        }
    }
    for (int i=k; i<equ; i++) {
        if (a[i][col]!=0) return -1;
    }
    if (k<var) return var-k;
    for (int i=var-1; i>=0; i--) {
        ans[i]=a[i][var];
        for (int j=i+1; j<var; j++) {
            ans[i]^=(a[i][j]&&ans[j]);
        }
    }
    return 0;
}
void init() {
    equ=m*n, var=m*n;
    free_x.clear();
    // for (int i=0; i<=equ; i++) a[i].reset();
    memset(ans, 0, sizeof(ans));
    memset(a, 0, sizeof(a));
    for (int i=0; i<n; i++) {
        for (int j=0; j<m; j++) {
            int t=i*m+j;
            a[t][t]=1;
            if (i>0) a[(i-1)*m+j][t]=1;
            if (i<n-1) a[(i+1)*m+j][t]=1;
            if (j>0) a[i*m+j-1][t]=1;
            if (j<m-1) a[i*m+j+1][t]=1;
        }
    }
}
```

倍增求lca

```
int f[maxn][20];
int dep[maxn];
void dfs(int u, int dept, int fa) {
    f[u][0] = fa;
    dep[u] = dept;
    for (int i = 1; i < 20; i++) {
        if (f[u][i-1] < 0) f[u][i] = -1;
        else f[u][i] = f[f[u][i-1]][i-1];
    }
    for (int i = head[u]; i; i = v[i].nxt) {
        int to = v[i].u;
        if (to == fa) continue;
        dfs(to, dept + 1, u);
    }
}
int lca(int a, int b) {
    if (dep[a] < dep[b]) swap(a, b);
    for (int i = 0, d = dep[a] - dep[b]; d; i++, d >>= 1) {
        if (d & 1) a = f[a][i];
    }
    if (a == b) return a;
    for (int i = log(n) + 1; i >= 0; i--) {
        if (f[a][i] != f[b][i]) a = f[a][i], b = f[b][i];
    }
    return f[a][0];
}
```

On求图上所有点的最远点对

```
const int maxn=2e5 + 100;
struct node {
    int u, w, nxt;
}v[maxn << 2];
struct node2 {
    int pos;
    ll dis;
    bool operator < (const node2& tmp) const {
        return dis > tmp.dis;
    }
}da[maxn][4];
int n, m, cnt = 1;
int head[maxn];
void add(int a, int b, int c) {
    v[cnt] = (node) {b, c, head[a]}; head[a] = cnt++;
}
ll ans;
void dfs1(int u, int fa) {
    for (int i = head[u]; i; i = v[i].nxt) {
        int to = v[i].u;
        if (to == fa) continue;
        dfs1(to, u);
        da[u][3].dis = da[to][0].dis + v[i].w, da[u][3].pos = to;
        sort(da[u], da[u] + 4);
    }
    ans = max(ans, da[u][0].dis + da[u][1].dis * 2 + da[u][2].dis);
}
bool vis[maxn];
void dfs2(int u) {
    vis[u] = 1;
    for (int i = head[u]; i; i = v[i].nxt) {
        int to = v[i].u;
        if (!vis[to]) continue;
        if (u != da[to][0].pos) da[u][3].dis = da[to][0].dis + v[i].w;
        else da[u][3].dis = da[to][1].dis + v[i].w;
        da[u][3].pos = to;
        sort(da[u], da[u] + 4);
        break;
    }
    ans = max(ans, da[u][0].dis + da[u][1].dis * 2 + da[u][2].dis);
    for (int i = head[u]; i; i = v[i].nxt) {
        int to = v[i].u;
        if (!vis[to]) dfs2(to);
    }
}
```

树上主席树

```
const int maxn=1e5 + 100;
int da[maxn], ranks[maxn];
struct node {
    int l, r, sum;
} tr[maxn*20];
int root[maxn];
int Tcnt, total;
void ins(int num, int& x, int l, int r) {
    tr[++Tcnt] = tr[x]; x = Tcnt; tr[x].sum++;
    if (l == r) return;
    int mid = (l + r) >> 1;
    if (num <= mid) ins(num, tr[x].l, l, mid);
    else ins(num, tr[x].r, mid+1, r);
}
int query(int ss, int bs, int k, int l, int r) {
    if (l == r) return l;
    int d = tr[tr[bs].l].sum - tr[tr[ss].l].sum;
    int mid = (l + r) >> 1;
    if (k <= d) return query(tr[ss].l, tr[bs].l, k, l, mid);
    else return query(tr[ss].r, tr[bs].r, k - d, mid+1, r);
}
int f[maxn][20], dep[maxn];
int head[maxn], to[maxn<<1], nxt[maxn<<1], vcnt;
int n;
void add(int a, int b) {
    to[++vcnt] = b; nxt[vcnt] = head[a]; head[a] = vcnt;
}
void dfs(int u, int dept) {
    dep[u] = dept;
    for (int i = 1; i < 20; i++) {
        if (!f[u][i-1]) break;
        else f[u][i] = f[f[u][i-1]][i-1];
    }
    ins(ranks[u], root[u] = root[f[u][0]], 1, total);
    for (int i = head[u]; i; i = nxt[i]) {
        if (to[i] == f[u][0]) continue;
        f[to[i]][0] = u;
        dfs(to[i], dept+1);
    }
}
int logs[maxn];
int lca(int a, int b) {
    if (dep[a] < dep[b]) swap(a, b);
    for (int i = 0, d = dep[a] - dep[b]; d; d>>=1, i++) {
        if (d&1) a = f[a][i];
    }
    if (a == b) return a;
    for (int i = logs[n]; i >= 0; i--) {
        if (f[a][i] != f[b][i]) a = f[a][i], b = f[b][i];
    }
    return f[a][0];
}
int query2(int ss1, int ss2, int bs1, int bs2, int k, int l, int r) {
    if (l == r) return l; int mid = (l + r) >> 1;
    int d = tr[tr[bs1].l].sum + tr[tr[bs2].l].sum - tr[tr[ss1].l].sum - tr[tr[ss2].l].sum;
    if (k <= d) return query2(tr[ss1].l, tr[ss2].l, tr[bs1].l, tr[bs2].l, k, l, mid);
    else return query2(tr[ss1].r, tr[ss2].r, tr[bs1].r, tr[bs2].r, k - d, mid + 1, r);
}
#define pii pair<int, int>
int da2[maxn];

int main() {
```

```
    int m; scanf("%d%d", &n, &m);
    for (int i = 1, tmp; i <= n; i++) {
        scanf("%d", &tmp);
        da[i] = tmp; da2[i] =  tmp;
    }
    sort(da2 + 1, da2 + 1 + n);
    logs[0] = -1;
    for (int i = 1; i < maxn; i++) logs[i] = logs[i>>1]+1;
    total = unique(da2 + 1, da2 + 1 + n) - da2 - 1;
    for (int i = 1; i <= n; i++) ranks[i] = lower_bound(da2 + 1, da2 + 1 + total, da[i]) - da2;
    for (int i = 1, a, b; i < n; i++) {
        scanf("%d%d", &a, &b); add(a, b); add(b, a);
    }
    dfs(1, 1);
    int ans = 0, a, b, k, t;
    while (m--) {
        scanf("%d%d%d", &a, &b, &k);
        a ^= ans; t = lca(a, b);
        printf("%d", ans = da2[query2(root[t], root[f[t][0]], root[a], root[b], k, 1, total)]);
        if (m) printf("\n");
    }
    return 0;
}
```

树剖

```cpp
const int maxn=3e5+100;
int head[maxn], to[maxn<<1], nxt[maxn<<1], vcnt;
void add(int a, int b) {
    to[++vcnt] = b; nxt[vcnt] = head[a]; head[a] = vcnt;
}

int son[maxn], sz[maxn], dep[maxn], fa[maxn];
int sum[maxn], id[maxn], top[maxn], idd;
void dfs1(int u, int f) {
    fa[u] = f; sz[u] = 1;
    for (int i = head[u]; i; i = nxt[i]) {
        if (to[i] == f) continue;
        dep[to[i]] = dep[u] + 1; dfs1(to[i], u);
        sz[u] += sz[to[i]];
        if (sz[to[i]] > sz[son[u]]) son[u] = to[i];
    }
}
void dfs2(int u, int tp) {
    top[u] = tp; id[u] = ++idd;
    if (son[u])
        dfs2(son[u], tp);
    for (int i = head[u]; i; i = nxt[i])
        if (to[i] != fa[u] && to[i] != son[u])
            dfs2(to[i], to[i]);
}
void update(int x, int y) {
    while (top[x] != top[y]) {
        if (dep[top[x]] < dep[top[y]]) swap(x, y);
        sum[id[top[x]]]++, sum[id[x] + 1]--; // 区间覆盖
        x = fa[top[x]];
    }
    if (dep[x] > dep[y]) swap(x, y);
    sum[id[x]]++, sum[id[y] + 1]--;
}
int da[maxn];
int main() {
    int n; scanf("%d", &n);
    for (int i = 1; i <= n; i++) scanf("%d", da + i);
    for (int i = 1, a, b; i < n; i++) {
        scanf("%d%d", &a, &b);
        add(a, b); add(b, a);
    }
    dfs1(1, -1); dfs2(1, 1);
    for (int i = 1; i < n; i++) update(da[i], da[i + 1]);
    for (int i = 1; i <= n; i++) sum[i] += sum[i - 1];
    for (int i = 1; i <= n; i++) {
        if (i == da[1]) printf("%d\n", sum[id[i]]);
        else printf("%d\n", sum[id[i]] - 1);
    }
    return 0;
}
```

模拟退火

```
double minn = 99999999999999999LL;
inline double dis(node& a, node& b) {
    return sqrt((a.x - b.x) * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));
}
inline double rands() {
    return (rand() % 1000) / 1000.0;
}
inline double judge(node& now) {
    double res = 0;
    for (int i = 1; i <= n; i++) res += dis(da[i], now) * da[i].p;
    if (res < minn) minn = res, ans = now;
    return res;
}
void cal() {
    double T = 1e5, dE;
    node now = ans, ne;
    while(T > 1e-3){
        ne.x = now.x + T * (rands() * 2 - 1);
        ne.y = now.y + T * (rands() * 2 - 1);
        dE = judge(now) - judge(ne);
        if (dE>0 || exp(dE/T)>rands()) now = ne;
        T *= 0.99;
    }
    for (int i = 1; i <= 1000; i++) {
        ne.x = ans.x + T * (rands() * 2 - 1);
        ne.y = ans.y + T * (rands() * 2 - 1);
        judge(ne);
    }
}
```

2 sat

```
// A，B不能同时取,连边A→B’，B→A’
// A，B不能同时不取，A’→B，B’→A
// A，B要么都取，要么都不取,A→B，B→A，A’→B’，B’→A’
const int maxn=1e4+100;
int head[maxn], to[maxn*100], nxt[maxn*100], cnt;
void add(int a, int b) {
    to[cnt] = b, nxt[cnt] = head[a]; head[a] = cnt++;
}

char ans[maxn];
int dfn[maxn], low[maxn], vis[maxn], ins[maxn], dfns, Amount;
int belong[maxn], cnts[maxn];
stack<int> s;
void tarjan(int u) {
    vis[u] = ins[u] = 1; s.push(u);
    dfn[u] = low[u] = dfns++;
    for (int i = head[u]; ~i; i = nxt[i]) {
        if (!vis[to[i]]) tarjan(to[i]), low[u] = min(low[u], low[to[i]]);
        else if (ins[to[i]]) low[u] = min(low[u], dfn[to[i]]);
    }
    int x = 0;
    if (dfn[u] == low[u]) {
        Amount++;
        while (x != u) {
            x = s.top(); s.pop(); ins[x] = 0;
            belong[x] = Amount; cnts[Amount]++;
        }
    }
}
int another[maxn], color[maxn], ind[maxn];
vector<int> mp2[maxn];
void tp(int n) {
    for (int i=0; i<=n; i++) mp2[i].clear();
    memset(color, 0, sizeof(color));
    for (int i=1; i<=(n<<1); i++) {
        for (int j=head[i]; ~j; j=nxt[j]) {
            int t=to[j];
            if (belong[i]!=belong[t]) {
                mp2[belong[t]].push_back(belong[i]);
                ind[belong[i]]++;
            }
        }
    }
    for (int i=1; i<=n; i++) {
        another[belong[i]]=belong[i+n];
        another[belong[i+n]]=belong[i];
    }
    queue<int> q;
    for (int i=1; i<=Amount; i++) {
        if (ind[i]==0) q.push(i);
    }
    for (int nw; !q.empty(); q.pop()) {
        nw=q.front();
        if (color[nw]==0) {
            color[nw]='R'; color[another[nw]]='B';
        }
        for (int t:mp2[nw]) {
            if (!(--ind[t])) {
                q.push(t);
            }
        }
    }
```

```
}
void init() {
    memset(head, -1, sizeof(head));
    memset(vis, 0, sizeof(vis));
    memset(dfn, 0, sizeof(dfn));
    memset(low, 0, sizeof(low));
    memset(belong, 0, sizeof(belong));
    while (!s.empty()) ins[s.top()] = 0, s.pop();
    Amount = cnt = dfns = 0;
}
char s1[10], s2[10], s3[10];
int main() {
#ifdef LOCAL
    freopen("in.txt", "r", stdin);
    freopen("out.txt", "w", stdout);
#endif
    init();
    int n, k; scanf("%d%d", &k, &n);
    // for (int i=1; i<=k; i++) add(i, i+k), add(i+k, i);
    for (int i=0, a, b, c, _a, _b, _c; i<n; i++) {
        scanf("%d%s%d%s%d%s", &a, s1, &b, s2, &c, s3);
        if (s1[0]=='R') _a=a+k; else _a=a, a+=k;
        if (s2[0]=='R') _b=b+k; else _b=b, b+=k;
        if (s3[0]=='R') _c=c+k; else _c=c, c+=k;
        add(_a, b); add(_a, c);
        add(_b, a); add(_b, c);
        add(_c, a); add(_c, b);
    }
    for (int i=1; i<=(k<<1); i++) {
        if (!vis[i]) tarjan(i);
    }
    bool flag=false;
    for (int i=1; i<=k; i++) {
        if (belong[i]==belong[i+k]) {
            flag=true; break;
        }
    }
    if (flag) puts("-1");
    else {
        tp(k);
        for (int i=1; i<=k*2; i++) {
            if (color[belong[i]]=='R') {
                if (i<=k) ans[i]='R';
                else ans[i-k]='B';
            }
            // printf("%c", color[belong[i]]);
        }
        for (int i=1; i<=k; i++) printf("%c", ans[i]);
    }
    return 0;
}
```

```
const int maxn=1e4+100;
vector<int> mp[maxn];
bool vis[maxn];
stack<int> s;
bool dfs(int x) {
    if (vis[x^1]) return false;
    if (vis[x]) return true;
    vis[x]=1; s.push(x);
    for (int t:mp[x]) {
        if (!dfs(t)) return false;
    }
    return true;
}
bool twoSat(int n) {
    memset(vis, 0, sizeof(vis));
    for (int i=0; i<n; i+=2) {
        if (vis[i] || vis[i^1]) continue;
        while (s.size()) s.pop();
        if (!dfs(i)) {
            while (s.size()) vis[s.top()]=false, s.pop();
            if (!dfs(i^1)) return false;
        }
    }
    return true;
}
char s1[10], s2[10], s3[10];
int main() {
    int n, k; scanf("%d%d", &k, &n);
    for (int i=0, a, b, c, _a, _b, _c; i<n; i++) {
        scanf("%d%s%d%s%d%s", &a, s1, &b, s2, &c, s3);
        a--; a<<=1; if (s1[0]=='R') _a=a^1; else _a=a, a^=1;
        b--; b<<=1; if (s2[0]=='R') _b=b^1; else _b=b, b^=1;
        c--; c<<=1; if (s3[0]=='R') _c=c^1; else _c=c, c^=1;
        mp[_a].push_back(b); mp[_a].push_back(c);
        mp[_b].push_back(a); mp[_b].push_back(c);
        mp[_c].push_back(a); mp[_c].push_back(b);
    }
    if (twoSat(k*2)) {
        for (int i=0; i<2*k; i+=2) {
            printf("%c", "BR"[vis[i]]);
        }
    } else {
        puts("-1");
    }
}
```

tarjan缩点

```
for (int i = 1; i <= n; i++) {
    for (int j = head[i]; j; j = nxt[j]) {
        if (belong[i] != belong[to[j]] && !vis[to[j]]) {
            Add(belong[i], belong[to[j]]);
            in[belong[to[j]]]++; vis[belong[to[j]]] = 1;
        }
    }
    for (int j = head[i]; j; j = nxt[j]) {
        vis[belong[to[j]]] = 0;
    }
}
```

点分

```cpp
const int maxn=2e4 + 100;
int head[maxn];
struct node {
    int u, w, nxt;
} v[maxn << 1];
int cnt;
void add(int a, int b, int c) {
    v[++cnt] = (node) {b, c, head[a]}; head[a] = cnt;
}
bool vis[maxn];
int son[maxn], mSon[maxn], root;
int n, sum;
void getRt(int x, int fa) {
    son[x] = 1, mSon[x] = 0;
    for (int i = head[x]; i; i = v[i].nxt) {
        int to = v[i].u;
        if (vis[to] || to == fa) continue;
        getRt(to, x);
        son[x] += son[to];
        mSon[x] = max(mSon[x], son[to]);
    }
    mSon[x] = max(mSon[x], sum - son[x]);
    if (mSon[x] < mSon[root]) root = x;
}
int dep[maxn], ct[10];
void getDep(int u, int fa) {
    ct[dep[u]]++;
    for (int i = head[u]; i; i = v[i].nxt) {
        int to = v[i].u;
        if (to == fa || vis[to]) continue;
        dep[to] = (dep[u] + v[i].w) % 3;
        getDep(to, u);
    }
}
int cal(int x, int now) {
    ct[0] = ct[1] = ct[2] = 0;
    dep[x] = now; getDep(x, -1);
    return ct[0] * ct[0] + ct[1] * ct[2] * 2;
}
int ans;
void solve(int x) {
    ans += cal(x, 0); vis[x] = 1;
    for (int i = head[x]; i; i = v[i].nxt) {
        int to = v[i].u;
        if (vis[to]) continue;
        ans -= cal(to, v[i].w);
        root = 0; sum = son[to]; getRt(to, -1);
        solve(root);
    }
}
int main() {
    scanf("%d", &n);
    for (int i = 1, a, b, c; i < n; i++) {
        scanf("%d%d%d", &a, &b, &c); c %= 3;
        add(a, b, c); add(b, a, c);
    }
    mSon[0] = inf; sum = n;
    getRt(1, -1);
    solve(root);
    int gcd = __gcd(ans, n * n);
    printf("%d/%d\n", ans / gcd, n * n / gcd);
```

```
    return 0;
}
```

匈牙利算法

```
// 二分图最大独立集(点数-最大匹配数)
bool vis[10101];
int fa[maxn];
int dfs(int u) {
    if (vis[u]) return 0;
    vis[u] = 1;
    for (int i = head[u]; i; i = nxt[i]) {
        if (!fa[to[i]] || dfs(fa[to[i]])) {
            fa[to[i]] = u;
            return 1;
        }
    }
    return 0;
}
for (int i = 1; i <= n; i++) {
    memset(vis, 0, sizeof(vis));
    ans+=dfs(i)
}
```

极角排序+叉积

```
const int maxn=4000;
struct node {
    ll x, y;
}da[maxn], da2[maxn];
ll cal(node a, node b) {
    return a.x * b.y - b.x * a.y;
}
ll cmp1(node a, node b) {
    return cal(a, b) > 0;
}
ll cmp2(node a, node b) {
    if (a.y != b.y) return a.y < b.y;
    return a.x < b.x;
}
int main() {
    int n; scanf("%d", &n);
    for (int i = 1; i <= n; i++) {
        scanf(lld lld, &da[i].x, &da[i].y);
    }
    sort(da+1, da+1+n, cmp2);
    ll ans = 0;
    for (int i = 1; i <= n; i++) {
        ll sumx = 0, sumy = 0;
        int cnt = 0;
        for (int j = i + 1; j <= n; j++) {
            sumx += da2[++cnt].x = da[j].x - da[i].x;
            sumy += da2[cnt].y = da[j].y - da[i].y;
        }
        sort(da2+1, da2+1+cnt, cmp1);
        for (int j = 1; j <= cnt; j++) {
            sumx -= da2[j].x;
            sumy -= da2[j].y;
            ans += (da2[j].x * sumy - da2[j].y * sumx);
        }
    }
    printf(lld "." lld, ans>>1, (ans&1)*5);
    return 0;
}
```

逆元

```
void exgcd(ll a, ll b, ll& d, ll&x, ll& y) {
    if (!b) {
        d = a, x = 1, y = 0;
    } else {
        exgcd(b, a%b, d, y, x);
        y -= x * (a / b);
    }
}
ll inv(ll a, ll mod) {
    ll x, y, d;
    exgcd(a, mod, d, x, y);
    return (x%mod+mod)%mod;
}
// 线性求逆元
ll invs[maxn];
void getInv(int mod) {
    invs[1] = 1;
    for (int i = 2; i < maxn; i++) {
        invs[i] = 1LL * invs[mod%i] * (mod-mod/i) % mod;
    }
}
```

矩阵快速幂

```
const int maxn=1e6+100;
ll mod;
struct mat {
    ll a[2][2];
    mat(){memset(a, 0, sizeof(a));}
    mat operator* (const mat& A) const {
        mat ans;
        for (int i=0; i<2; i++)
            for (int j=0; j<2; j++)
                for (int k=0; k<2; k++)
                    ans.a[i][j]+=a[i][k]*A.a[k][j]%mod;

        return ans;
    }
    void init() {
        for (int i=0; i<2; i++) a[i][i]=1;
    }
};
inline mat qpow(mat A, int b) {
    mat ans; ans.init();
    for (; b; b>>=1, A=A*A) {
        if (b&1) ans=ans*A;
    }
    return ans;
}
mat qpow(mat A, char* b, int len) {
    mat ans; ans.init();
    for (int i=len; i; i--) {
        int cnt=b[i]-'0';
        ans=ans*qpow(A, cnt);
        A=qpow(A, 10);
    }
    return ans;
}
char s[maxn];
int main() {
    ll x0, x1, a, b;
    scanf("%lld%lld%lld%lld%s%lld", &x0, &x1, &a, &b, s+1, &mod);
    int len=strlen(s+1);
    s[len]--;
    for (int i=len; s[i]<'0' && i; i--) s[i-1]--, s[i]+=10;
    mat A; A.a[0][0]=a; A.a[0][1]=b; A.a[1][0]=1; A.a[1][1]=0;
    A=qpow(A, s, len);
    ll ans=(A.a[0][0]*x1%mod+A.a[0][1]*x0%mod)%mod;
    printf("%lld\n", ans);
    return 0;
}
```

dinic

```
const int maxn=6e4;
int head[maxn], to[maxn<<6], nxt[maxn<<6], w[maxn<<6], vcnt;
void add(int a, int b, int c) {
    to[vcnt] = b; w[vcnt] = c; nxt[vcnt] = head[a]; head[a] = vcnt++;
}
int start, stop;
int dep[maxn];
bool bfs() {
    memset(dep, -1, sizeof(dep));
    queue<int> q; q.push(start);
    dep[start] = 0;
    while(!q.empty()){
        int x = q.front(); q.pop();
        for (int i = head[x]; ~i; i = nxt[i]) {
            if (w[i] && dep[to[i]] == -1) {
                q.push(to[i]);
                dep[to[i]] = dep[x] + 1;
            }
        }
    }
    return dep[stop] != -1;
}
int dfs(int u, int flow) {
    if (u == stop) return flow;
    int res = 0;
    for (int i = head[u]; ~i; i = nxt[i]) {
        int t = to[i];
        if (w[i] && dep[t] == dep[u] + 1) {
            int f = dfs(t, min(flow, w[i]));
            flow -= f; w[i] -= f; w[i^1] += f;
            res += f;
            if (!flow) break;
        }
    }
    if (!res) dep[u] = -1;
    return res;
}
int dinic() {
    int ans = 0;
    while (bfs()) {
        ans += dfs(start, inf);
    }
    return ans;
}
void init() {
    memset(head, -1, sizeof(head));
    vcnt = 0;
}
```

spfa 费用流

```
const int maxn=2000;
const int maxm=3000100;
int start, stop;
int head[maxn], nxt[maxm], to[maxm], f[maxm], w[maxm], fr[maxm], vcnt;
void _add(int a, int b, int c, int ww) {
    to[vcnt]=b; f[vcnt]=c; w[vcnt]=ww; nxt[vcnt]=head[a]; fr[vcnt] = a; head[a]=vcnt++;
}
void add(int a, int b, int c, int ww) {
    _add(a, b, c, ww);
    _add(b, a, 0, -ww);
}
int dis[maxn], from[maxn];
bool inq[maxn];
bool spfa() {
    memset(dis, 0x3f, sizeof(dis));
    memset(inq, 0, sizeof(inq));
    deque<int> q;
    q.push_back(0); dis[0]=0; inq[0]=1;
    while(!q.empty()){
        int x = q.front(); q.pop_front(); inq[x] = 0;
        for (int i=head[x]; ~i; i=nxt[i]) {
            if (f[i] && dis[to[i]] > dis[x]+w[i]) {
                dis[to[i]] = dis[x] + w[i];
                from[to[i]] = i;
                if (!inq[to[i]]) {
                    inq[to[i]] = 1;
                    if (q.size() && dis[to[i]] < dis[q.front()]) {
                        q.push_front(to[i]);
                    } else {
                        q.push_back(to[i]);
                    }
                }
            }
        }
    }
    return dis[stop] != inf;
}
int ans;
void mcf() {
    int x = inf;
    for (int i = stop; i; i=fr[from[i]]) {
        x = min(x, f[from[i]]);
    }
    for (int i = stop; i; i=fr[from[i]]) {
        ans += x * w[from[i]];
        f[from[i]] -= x; f[from[i]^1] += x;
    }
}
while(spfa()){
    mcf();
}
```

线性基

```
for (int i=0; i<n; i++) {
    for (int j=64; ~j; j--) {
        if ((da[i].x>>j)&1) {
            if (p[j]) da[i].x^=p[j];
            else {
                p[j]=da[i].x; break;
            }
        }
    }
}
```

线性基的交

```
uint x[32], y[32];
void merge(int rt, int a, int b) {
    for (int i=0; i<32; i++) x[i]=y[i]=tr[a][i];
    for (int i=0; i<32; i++) {
        uint u=tr[b][i], k=0;
        if (!u) continue;
        for (int j=31; ~j; j--) {
            if ((u>>j)&1) {
                if (x[j]) u^=x[j], k^=y[j];
                else {
                    x[j]=u, y[j]=k; break;
                }
            }
        }
        if (!u) ins(tr[rt], k);
    }
}
```

rope

```
const int maxn=2e6+100;
char str1[maxn], str2[maxn];
rope<char> da, revda, tmp;
int main() {
    int n, x, now=0, len; scanf("%d", &n);
    char opt[50];
    while (n--) {
        scanf("%s", opt);
        len=da.length();
        if (opt[0]=='I') {
            scanf("%d%*c", &x);
            for (int i=0; i<x; i++) {
                str1[i]=getchar();
                str2[x-i-1]=str1[i];
            }
            str1[x]=str2[x]=0;
            da.insert(now, str1);
            revda.insert(len-now, str2);
        } else if (opt[0]=='M') {
            scanf("%d", &x); now=x;
        } else if (opt[0]=='D') {
            scanf("%d", &x);
            da.erase(now, x);
            revda.erase(len-now-x, x);
        } else if (opt[0]=='R') {
            scanf("%d", &x);
            tmp=da.substr(now, x);
            da=da.substr(0, now)+revda.substr(len-now-x, x)+da.substr(now+x, len-now-x);
            revda=revda.substr(0, len-now-x)+tmp+revda.substr(len-now, now);
        } else if (opt[0]=='G') {
            printf("%c\n", da[now]);
        } else if (opt[0]=='P') {
            now--;
        } else if (opt[0]=='N') {
            now++;
        }
    }
    return 0;
}
```

后缀自动机

```
// 广义sam，last置为1，然后插入新串， ed置为串id
const int maxn=5e5+100;
int n;
struct sam {
//点i ii上面表示子串的数量为len[fa[i]]-len[i]
//在parents上，父亲是儿子上的子串的公共后缀
//父亲上的子串出现次数，是儿子上的子串出现次数之和
//在一个点上的子串，短的为长的的后缀
//len数组表示的是这个节上的子串最长长度
    int len[maxn<<1], fa[maxn<<1], son[maxn<<1][26];
    int ed[maxn<<1];
    int sz, last;
    void init() {sz=last=1;}
    void ins(char c) {
        int s=c-'a';
        int p=last, np=++sz;
        last=np; ed[np]=1; len[np]=len[p]+1;
        for (; p && !son[p][s]; p=fa[p]) son[p][s]=np;
        if (!p) fa[np]=1;
        else {
            int q=son[p][s];
            if (len[p]+1==len[q]) fa[np]=q;
            else {
                int nq=++sz; len[nq]=len[p]+1;
                memcpy(son[nq], son[q], sizeof(son[q]));
                fa[nq]=fa[q]; fa[q]=fa[np]=nq;
                for (; q==son[p][s] && p; p=fa[p]) son[p][s]=nq;
            }
        }
    }
    void Ins(char *str) {
        init();
        for (int i=0; str[i]; i++) {
            ins(str[i]);
        }
    }
    int x[maxn<<1], y[maxn<<1];
    void msort() {
        for (int i=1; i<=sz; i++) x[len[i]]++;
        for (int i=1; i<=sz; i++) x[i]+=x[i-1];
        for (int i=1; i<=sz; i++) y[x[len[i]]--]=i;
    }
    void solve() {
        ll ans1=1LL*n*(n+1)/2*(n-1), ans2=0;
        for (int i=sz; i; i--) {
            int tp = y[i];
            ans2+=1LL*ed[fa[tp]]*len[fa[tp]]*ed[tp];
            ed[fa[tp]]+=ed[tp];
        }
        printf("%lld", ans1-ans2*2);
    }
    ll count() {
        ll ans=0;
        for (int i=2; i<=sz; i++) ans+=len[i]-len[fa[i]];
        return ans;
    }
};
```

回文树

```
// fail[i]表示以此节点位结尾，较短的回文串
// num表示以节点i表示的最长回文串的最右端点为回文串结尾的回文串个数
// cnt表示以此节点结尾本质不同的回文串数量，在count()执行后才为正确结果
// 节点数量-2就是本质不同串数量
const int N=6e5;
struct Pal {
    int nxt[N][26], fail[N], cnt[N], num[N], len[N], S[N], last, n, p;
    int newNode(int l) {
        memset(nxt[p], 0, sizeof(nxt[p]));
        cnt[p]=num[p]=0; len[p]=l;
        return p++;
    }
    void init() {
        p=0; newNode(0); newNode(-1);
        last=0; n=0; fail[0]=1; S[0]=-1;
    }
    int getFail(int x) {
        for (; S[n-len[x]-1]!=S[n]; x=fail[x]);
        return x;
    }
    void add(int c) {
        c-='a';
        S[++n]=c;
        int cur=getFail(last);
        if (!nxt[cur][c]) {
            int now=newNode(len[cur]+2);
            fail[now]=nxt[getFail(fail[cur])][c];
            nxt[cur][c]=now;
            num[now]=num[fail[now]]+1;
        }
        last=nxt[cur][c];
        cnt[last]++;
    }
    void count() {
        for (int i=p-1; ~i; i--) cnt[fail[i]]+=cnt[i];
    }
}pal;
```

支配树

```
// DAG上点对的LCA
const int maxn=1e5+10;
struct tree{
    vector<int> G[maxn], E[maxn];
    int ind[maxn], dep[maxn];
    int f[maxn][20];
    int n, root, tot, a[maxn];
    void init(int _n) {
        n=_n;
        for (int i=0; i<=n; i++) G[i].clear(), ind[i]=0, dep[i]=0, E[i].clear();
        root=_n+1; tot=0;
    }
    void add(int a, int b) {
        G[a].push_back(b); E[b].push_back(a);
        ind[a]++;
    }
    void TP() {
        queue<int> q;
        for (int i=1; i<=n; i++) if (ind[i]==0) q.push(i), E[root].push_back(i), G[i].push_back(root);
        while (!q.empty()) {
            int u=q.front(); q.pop();
            a[++tot]=u;
            for (int a:E[u])
                if ((--ind[a])==0) q.push(a);
        }
    }
    int lca(int a, int b) {
        if (dep[a]>dep[b]) swap(a, b);
        for (int i=19; ~i; i--) if (dep[b]>dep[a] && dep[f[b][i]]>=dep[a]) b=f[b][i];
        for (int i=19; ~i; i--) if (f[a][i]!=f[b][i]) a=f[a][i], b=f[b][i];
        return a==b ? a : f[a][0];
    }
    void pre() {
        dep[root]=1;
        for (int i=1; i<=n; i++) {
            int u=a[i], fa=-1;
            for (int v:G[u])
                fa = (fa==-1 ? v : lca(fa, v));
            dep[u]=dep[fa]+1; f[u][0]=fa;
            for (int i=1; i<20; i++) f[u][i]=f[f[u][i-1]][i-1];
        }
    }
}T;
```

二次同余方程

```
const int mod=1e9+9;
ll w;
void exgcd(ll a, ll b, ll& d, ll&x, ll& y) {
    if (!b) {
        d = a, x = 1, y = 0;
    } else {
        exgcd(b, a%b, d, y, x);
        y -= x * (a / b);
    }
}
ll inv(ll a, ll mod) {
    ll x, y, d;
    exgcd(a, mod, d, x, y);
    return (x%mod+mod)%mod;
}
ll qpow(ll a, ll b, ll md) {
    ll ans=1;
    for (a%=md; b; b>>=1, a=a*a%md) {
        if (b&1) ans=ans*a%md;
    }
    return ans;
}
struct node {
    ll p, d;
};
node mul(node a, node b, ll md) {
    node ans;
    ans.p=(a.p*b.p%md + a.d*b.d%md*w%md)%md;
    ans.d=(a.p*b.d%md + a.d*b.p%md)%md;
    return ans;
}
node qpow(node a, ll b, ll md) {
    node ans=node{1, 0};
    for (; b; b>>=1, a=mul(a, a, md)) {
        if (b&1) ans=mul(ans, a, md);
    }
    return ans;
}
ll solve(ll x, ll md) {
    if (md==2) return 1;
    if (qpow(x, (md-1)>>1, md)+1==md) return -1;
    ll a, t;
    for (a=1; a<md; a++) {
        t=a*a - x;
        w=(t+md)%md;
        if (qpow(w, (md-1)>>1, md)+1==md) break;
    }
    node tmp=node{a, 1};
    tmp=qpow(tmp, (md+1)>>1, md);
    return tmp.p;
    // sqrt(x)==tmp.p(%mod)
}
```

连分数

```
// a1/b1 <= den/num <= a2/b2
void continue_frac(ll a1, ll b1, ll a2, ll b2, ll& num, ll& den) {
    ll c1=a1/b1, d1=a1%b1;
    ll c2=a2/b2, d2=a2%b2;
    if (c1!=c2 || !d1 || !d2) {
        den=1, num=min(c1, c2)+1;
    } else {
        continue_frac(b1, d1, b2, d2, den, num);
        num+=c1*den;
    }
}
```

SG

```
int getSg(int n) {
    if (n == 1) return sg[1] = 0;
    if (sg[n] != -1) return sg[n];
    set<int> vis;
    for (int i = 1; i < n; i++) {
        int cnt = n / i, res = n % i;
        if (cnt % 2 == 0) {
            if (res) vis.insert(getSg(res));
            else vis.insert(0);
        } else {
            if (res)
                vis.insert(getSg(i) ^ getSg(res));
            else
                vis.insert(getSg(i));
        }
    }
    for (int i = 0; ; i++) if (!vis.count(i)) return sg[n] = i;
}
sg==0,先手必败
```

输入挂

```
char buf[100000],*p1=buf,*p2=buf;
inline char nc(){
    return p1==p2&&(p2=(p1=buf)+fread(buf,1,100000,stdin),p1==p2)?EOF:*p1++;
}
inline bool rea(int & x){
    char c=nc();x=0;
    if(c==EOF) return false;
    for(;!isdigit(c);c=nc());
    for(;isdigit(c);x=x*10+c-'0',c=nc());
    return true;
}

namespace IO {
    const int maxn=4096;
    char buf[maxn], t[50];
    int bn=maxn, bi=maxn;
    int read(char* s) {
        while (bn) {
            for (; bi<bn && buf[bi]<=' '; bi++);
            if (bi<bn) break;
            bn=fread(buf, 1, maxn, stdin);
            bi=0;
        }
        int sn=0;
        while (bn) {
            for (; bi<bn && buf[bi]>' '; bi++) s[sn++]=buf[bi];
            if (bi<bn) break;
            bn=fread(buf, 1, maxn, stdin);
            bi=0;
        }
        s[sn]=0;
        return sn;
    }
    bool read(int& x) {
        if (!read(t)) return 0;
        x=atoi(t);
        return 1;
    }
}
```

原根

> 在$(a,m)=1$时，定义a对模m的指数 $\delta m(a)$为使$a^d \equiv 1 \pmod m$ 成立的最小的正整数d
> 若 $\delta m(a)=\varphi(m)$, 则称a是模m的原根
> 素数p的原根个数为 $\varphi(p-1)$
> 指标:当确定模p和原根g时，定义a的指标为$I(a)$为使$g^d \equiv a \pmod p$成立的最小的正整数d
> 指标性质: $I(ab) \equiv I(a)+I(b) \pmod{p-1}$, $I(a^k) \equiv kI(a) \pmod{p-1}$
> 找m的原根：将$\varphi(m)$因数分解，即$\varphi(m)=p_1^{k_1}p_2^{k_2}\ldots\ldots p_n^{k_n}$ 如果一个数g是模m的原根，则其必然满足对于任意一个$1 \le i \le n, g^{\frac{\varphi(m)}{p_i}} \neq 1(mod\ m)$

```cpp
// 素数的原根
const int maxn=1e6+100;
int qpow(ll a, int b, int mod) {
    ll ans=1;
    for (; b; b>>=1, a=a*a%mod) {
        if (b&1) ans=ans*a%mod;
    }
    return ans%mod;
}
int getG(int n) {
    if (n==2) return 1;
    vector<int> p;
    int phi=n-1;
    for (int i=2; i*i<=phi; i++) {
        if (phi%i==0) {
            while (phi%i==0) phi/=i;
            p.push_back(i);
        }
    }
    if (phi>1) p.push_back(phi);
    for (int i=2; i<n; i++) {
        bool flag=1;
        for (int j:p) {
            if (qpow(i, (n-1)/j, n)==1) {
                flag=0; break;
            }
        }
        if (flag) return i;
    }
    return -1;
}
int x[maxn], I[maxn];
int cnt[maxn];
int main() {
    int t; cin >> t;
    for (int n; t--; ) {
        cin >> n;
        memset(cnt, 0, n<<2);
        int g=getG(n);
        I[0]=1; x[0]=1;
        for (int i=1; i<n; i++) {
            x[i]=1LL*x[i-1]*g%n;
            I[x[i]]=i;
        }
        for (int i=1; i<n; i++) {
            cnt[x[1LL*I[i]*i%(n-1)]]++;
        }
        ll ans=0;
        for (int i=1; i+i<=n; i++) {
            ans+=1LL*cnt[i]*cnt[n-i];
        }
        cout << ans << "\n";
    }
    return 0;
}
```

蔡勒公式

```
int getWeek(int y, int m, int d) {
    if (m<3) m+=12, y--;
    int c=y/100; y%=100;
    return ((y+y/4+c/4-2*c+26*(m+1)/10+d-1)%7+7)%7;
}
```

公式

$\sum_{m|n}\varphi(m)=n$
$\varphi^2(n)=\varphi(n^2)$
$1^2+2^2+3^2+...+n^2=n(n+1)(2n+1)\div 6$
$1^3+2^3+3^3+...+n^3=n^2(n+1)^2\div 4$
$\sum_{i\|n}\mu(i)=[n==1]$
$\sum_i^n[gcd(i,n)=1]=n*phi(n)\div 2$
$\sum_{i=1}^n(i*[gcd(n,i)=1])=i*\phi(i)\div 2+[i=1]$
$F(n)=\sum_{d\|n}f(d)=>f(n)=\sum_{d\|n}F(d)*\mu(\frac{n}{d})$
$lucas(n,m,p)=C(n\%p,m\%p)*lucas(n/p,m/p,p)$
$C_n^m\%p=\prod C_{n_i}^{m_i}\%p,m=\sum m_ip^i,n=\sum n_ip^i$
$h(n)=h(n-1)*(4*n-2)/(n+1)$
$h(n)=C_{2n}^n/(n+1)$
$h(n)=C_{2n}^n-C_{2n}^{n+1}$
$gcd(Fib[a],Fib[b])=Fib[gcd(a,b)]$
$gcd(a^i-1,a^j-1)=a^{gcd(i,j)}-1$
球缺：用平面截取球的一部分
球缺面积 $=2\cdot\pi h$
球缺体积 $=\pi\cdot h^2\cdot(R-\frac{h}{3})$
$w=\left(y+\lfloor\frac{y}{4}\rfloor+\lfloor\frac{c}{4}\rfloor-2c+\lfloor\frac{26(m+1)}{10}\rfloor+d-1\right)\bmod 7$
w：星期;c：年份前两位数;y：年份后两位数;m：月,某年的1、2月要看作上一年的13、14月来计算;d：日