高精度类

```
const int LEN = 110, base = 10000;
struct Num {
    int s[LEN], len;
    Num() {
        len = 0;
        memset (s, 0, sizeof s);
    }
    Num(int x) {
        len = 1;
        memset (s, 0, sizeof s);
        s[1] = x;
    }
    int& operator[](int x) {
        return s[x];
    }
    int operator[](int x) const {
        return s[x];
    }
    void print() {
        printf("%d", s[len]);
        for (int i = len - 1; i >= 1; i --)
            printf("%04d", s[i]);
        printf("\n");
    }
}f[710][710];
Num operator+ (const Num& a, const Num& b) {
    Num c;
    c.len = max(a.len, b.len);
    for (int i = 1; i <= c.len; i ++) {
        c[i] += a[i] + b[i];
        c[i + 1] += c[i] / base;
        c[i] %= base;
    }
    if (c[c.len + 1]) c.len++;
    return c;
}
Num operator* (const Num& a, const Num& b) {
    Num c;
    c.len = a.len + b.len - 1;
    for (int i = 1; i <= a.len; i ++)
        for (int j = 1; j <= b.len; j ++) {
            c[i + j - 1] += a[i] * b[j];
            c[i + j] += c[i + j - 1] / base;
            c[i + j - 1] %= base;
        }
    if (c[c.len + 1]) c.len++;
    return c;
}
bool operator< (const Num& a, const Num& b) {
    if (a.len == b.len)
        for (int i = a.len; i >= 0; i --)
            if (a[i] != b[i])
                return a[i] < b[i];
    return a.len < b.len;
}
```

读入优化

```
ll read()
{
    ll x=0,f=1;char ch=getchar();
    while(ch<'0'||ch>'9'){if(ch=='-')f=-1;ch=getchar();}
    while(ch>='0'&&ch<='9'){x=x*10+ch-'0';ch=getchar();}
    return x*f;
}
```

2DRMQ

```
const int N = 310;
int f[N][N][9][9],  n, q, x1, x2, y1, y2, lx, ly, T;
int lg (int x) {
    if (x == 1)
        return 0;
    return lg (x >> 1) + 1;
}
int min (int a, int b, int c){
    return min (a, min (b, c));
}
int min (int a, int b, int c, int d){
    return min (a, min (b, min (c, d)));
}
inline void build (){
    for (int i = 0; (1 << i) <= n; i ++)
        for (int j = 0; (1 << j) <= n; j ++)
            if (i + j)
                for (int x = 1; x + (1 << i) - 1 <= n; x ++)
                    for (int y = 1; y + (1 << j) - 1 <= n; y ++) {
                        if (i)
                            f[x][y][i][j] =
min (f[x][y][i - 1][j], f[x + (1 << (i - 1))][y][i - 1][j]);
                        if (j)
                            f[x][y][i][j] =
min (f[x][y][i][j - 1], f[x][y + (1 << (j - 1))][i][j - 1]);
                    }
}

int main () {
    scanf ("%d", &T);
    for (int t = 1; t <= T; t ++)
    {
        scanf ("%d", &n);
        for (int i = 1; i <= n; i ++)
            for (int j = 1; j <= n; j ++)
                scanf ("%d", f[i][j]);
        build ();
        scanf ("%d", &q);
        for (int i = 1; i <= q; i ++)
        {
            scanf ("%d%d%d%d", &x1, &y1, &x2, &y2);
            lx = lg (x2 - x1 + 1);
            ly = lg (y2 - y1 + 1);
            printf ("%d\n",
min (f[x1][y1][lx][ly], f[x2 - (1 << lx) + 1][y1][lx][ly], f[x1][y2
- (1 << ly) + 1][lx][ly],
f[x2 - (1 << lx) + 1][y2 - (1 << ly) + 1][lx][ly]));
        }
    }
    return 0;
}
```

线段树

```
struct Seg{
#define lson idx << 1
#define rson idx << 1 | 1
#define N 101000 * 4
    int mx[N], add[N], cover[N];
    void pushup(int idx) {
        mx[idx] = max(mx[lson], mx[rson]);
    }
    void pushdown(int mid, int idx) {
        if (cover[idx] != -1) {
            cover[lson] = cover[rson] = cover[idx];
            add[lson] = add[rson] = 0;
            mx[lson] = mx[rson] = cover[idx];
            cover[idx] = -1;
        }
        if (add[idx]) {
            mx[lson] += add[idx];
            mx[rson] += add[idx];
            if (cover[lson] == -1) add[lson] += add[idx];
            else cover[lson] += add[idx];
            if (cover[rson] == -1) add[rson] += add[idx];
            else cover[rson] += add[idx];
            add[idx] = 0;
        }
    }
    void build(int l, int r, int idx, int* w) {
        cover[idx] = -1;
        if (l == r) {
            mx[idx] = w[l];
            return ;
        }
        int mid = (l + r) >> 1;
        build(l, mid, lson, w);
        build(mid + 1, r, rson, w);
        pushup(idx);
    }
    int query(int L, int R, int l, int r, int idx) {
        if (L <= l && r <= R) {
            return mx[idx];
        }
        int mid = (l + r) >> 1;
        int z = 0;
        pushdown(mid, idx);
        if (L <= mid)
            z = max(z, query(L, R, l, mid, lson));
        if (R > mid)
            z = max(z, query(L, R, mid + 1, r, rson));
        pushup(idx);
        return z;
    }
    void Add(int L, int R, int v, int l, int r, int idx) {
        if (L <= l && r <= R) {
            add[idx] += v;
            mx[idx] += v;
            return ;
        }
        int mid = (l + r) >> 1;
        pushdown(mid, idx);
        if (L <= mid)
            Add(L, R, v, l, mid, lson);
        if (R > mid)
            Add(L, R, v, mid + 1, r, rson);
        pushup(idx);
    }
    void Cover(int L, int R, int v, int l, int r, int idx) {
```

```
        if (L <= l && r <= R) {
            cover[idx] = v;
            add[idx] = 0;
            mx[idx] = v;
            return ;
        }
        int mid = (l + r) >> 1;
        pushdown(mid, idx);
        if (L <= mid)
            Cover(L, R, v, l, mid, lson);
        if (R > mid)
            Cover(L, R, v, mid + 1, r, rson);
        pushup(idx);
    }
}seg;
```

主席树

```
#define mid (((l) + (r)) >> 1)
const int N = 301000;
struct S {
    struct Q {
        Q *l, *r;
        int s, c;
    }key[N << 4];
    Q *root[N];
    Q *p;
    inline void init (int n) {
        p = key;
        root[0] = build(1, n);
    }
    inline Q* getnew (int _c) {
        return p->s = 1, p->c = _c, p ++;
    }
    inline Q* getnew (Q* a, Q* b) {
        return p->l = a, p->r = b, p->s = a->s + b->s, p->c = a->c
+ b->c, p ++;
    }
    inline Q* build (int l, int r) {
        if (l == r) return getnew (0);
        return getnew (build (l, (l + r) >> 1), build (((l + r) >>
1) + 1, r));
    }
    inline Q* inc (Q* t, int i) {
        if (t->s == 1) return getnew (t->c + 1);
        if (i <= t->l->s) return getnew (inc (t->l, i), t->r);
        else return getnew (t->l, inc (t->r, i - t->l->s));
    }
    inline int query (Q* a, Q* b, int k) {
        if (a->s == 1) return 1;
        int t = b->l->c - a->l->c;
        if (k <= t) return query (a->l, b->l, k);
        else return a->l->s + query (a->r, b->r, k - t);
    }
}seg;
int n, m, a[N], b[N], c[N], b_c;
int main() {
    freopen("in", "r", stdin);
    scanf("%d%d", &n, &m);
    for (int i = 1; i <= n; i ++)
        scanf("%d", &a[i]), b[++ b_c] = a[i];
    sort(b + 1, b + 1 + b_c);
    b_c = unique(b + 1, b + 1 + b_c) - (b + 1);
    for (int i = 1; i <= n; i ++)
        c[i] = lower_bound(b + 1, b + 1 + b_c, a[i]) - b;
    seg.init(b_c);
    for (int i = 1; i <= n; i ++)
        seg.root[i] = seg.inc(seg.root[i - 1], c[i]);
    for (int i = 1, x, y, k; i <= m; i ++) {
        scanf("%d%d%d", &x, &y, &k);
        printf("%d\n", b[seg.query(seg.root[x - 1], seg.root[y],
k)]);
    }
    return 0;
}
```

树状数组套主席树

```
#define mid (l + r >> 1)
const int N = 101000;
int n, m, a[N], b[N], b_c, sav[N];
struct S {
    struct Q {
        Q *l, *r;
        int s, c;
    }key[N << 5];
    Q* root[N];Q* p;
    Q *a[N], *b[N];
    int t1, t2;
    int sav[N];
    void init () {
        p = key;
        memset (sav, 0, sizeof sav);
    }
    Q* getnew (int _c) {
        return p->s = 1, p->c = _c, p ++;
    }
    Q* getnew (Q* a, Q* b) {
        return p->l = a, p->r = b, p->s = a->s + b->s, p->c = a->c
+ b->c, p ++;
    }
    Q* build (int l, int r) {
        if (l == r) return getnew (0);
        return getnew (build (l, mid), build (mid + 1, r));
    }
    Q* inc (Q* t, int i) {
        if (t->s == 1) return getnew (t->c + 1);
        if (i <= t->l->s) return getnew (inc (t->l, i), t->r);
        else return getnew (t->l, inc (t->r, i - t->l->s));
    }
    Q* dec (Q* t, int i) {
        if (t->s == 1) return getnew (t->c - 1);
        if (i <= t->l->s) return getnew (dec (t->l, i), t->r);
        else return getnew (t->l, dec (t->r, i - t->l->s));
    }
    int query (int k) {
        if (b[1]->s == 1) return 1;
        int t (0);
        for (int i = 1; i <= t1; i ++)
            t -= a[i]->l->c;
        for (int i = 1; i <= t2; i ++)
            t += b[i]->l->c;
        if (k <= t)
        {
            for (int i = 1; i <= t1; i ++)
                a[i] = a[i]->l;
            for (int i = 1; i <= t2; i ++)
                b[i] = b[i]->l;
            return query (k);
        }
        else
        {
            int tmp = b[1]->l->s;
            for (int i = 1; i <= t1; i ++)
                a[i] = a[i]->r;
            for (int i = 1; i <= t2; i ++)
                b[i] = b[i]->r;
            return tmp + query (k - t);
        }
    }
    void INC (int x, int v)
    {
        for (int i = x; i <= n; i += i & -i)
```

```
            root[i] = inc (root[i], v);
    }
    void DEC (int x, int v)
    {
        for (int i = x; i <= n; i += i & -i)
            root[i] = dec (root[i], v);
    }
}seg;
int l[N], r[N], k[N], x[N], v[N];
int main ()
{
    seg.init ();
    scanf ("%d%d", &n, &m);
    for (int i = 1; i <= n; i ++)
        scanf ("%d", &a[i]), b[i] = a[i];
    b_c = n;
    char s[10];
    for (int i = 1; i <= m; i ++)
    {
        scanf ("%s", s);
        if (s[0] == 'Q')
            scanf ("%d%d%d", &l[i], &r[i], &k[i]);
        if (s[0] == 'C')
            scanf ("%d%d", &x[i], &v[i]), b[++ b_c] = v[i];
    }
    sort (b + 1, b + 1 + b_c);
    b_c = unique (b + 1, b + 1 + b_c) - (b + 1);
    for (int i = 1; i <= n; i ++)
        a[i] = lower_bound (b + 1, b + 1 + b_c, a[i]) - b;
    for (int i = 1; i <= m; i ++)
        if (x[i] != 0)
            v[i] = lower_bound (b + 1, b + 1 + b_c, v[i]) - b;
    seg.root[0] = seg.build (1, b_c);
    for (int i = 1; i <= n; i ++)
        seg.root[i] = seg.root[0];
    for (int i = 1; i <= n; i ++)
        seg.INC (i, a[i]);
    for (int j = 1; j <= m; j ++)
    {
        seg.t1 = seg.t2 = 0;
        if (l[j] != 0)
        {
            for (int i = l[j] - 1; i; i -= i & -i)
                seg.a[++ seg.t1] = seg.root[i];
            for (int i = r[j]; i; i -= i & -i)
                seg.b[++ seg.t2] = seg.root[i];
            printf ("%d\n", b[seg.query (k[j])]);
        }
        if (x[j] != 0)
        {
            seg.DEC (x[j], a[x[j]]);
            a[x[j]] = v[j];
            seg.INC (x[j], a[x[j]]);
        }
    }
    return 0;
}
```

强连通分量

```
void DFS(int u) {
    dfn[u] = low[u] = ++ tmct;
    stk[++ top] = u;
    instk[u] = true;
    for (int i = head[u]; ~ i; i = next[i]) {
        int v = key[i];
        if (!dfn[v]) {
            DFS(v);
            low[u] = min(low[u], low[v]);
        } else if (instk[v]) {
            low[u] = min(low[u], dfn[v]);
        }
    }
    if (low[u] == dfn[u]) {
        ++ scc_c;
        int now;
        do {
            now = stk[top --];
            scc[now] = scc_c;
            instk[now] = false;
        } while(now != u);
    }
}
```

割点

```
inline void tj(int x){
    low[x] = dfn[x] = ++ tmct;
    v[x] = 0;
    for(int i = head[x]; ~ i; i = next[i])
        if(!dfn[key[i]]){
            tj(key[i]);
            low[x] = min(low[x], low[key[i]]);
            if(dfn[x] <= low[key[i]]) v[x] ++;
        } else
            low[x] = min(low[x], dfn[key[i]]);
}
    ans = 0;
    if(v[1] >= 2) ans ++;
    for(int i = 2; i <= n; i ++)
      if(v[i])
          ans ++;
```

割边

```
inline void tarjan (int u)
{
    dfn[u] = low[u] = ++ tmct;
    vs[u] = true;
    for (int i = head[u]; ~ i; i = next[i]) {
        int v = key[i];
        if (!dfn[v]) {
            p[v] = e[i];
            tarjan (v);
            low[u] = min (low[u], low[v]);
        } else if (vs[v] && p[u] != e[i])
            low[u] = min (low[u], dfn[v]);
    }
    if (p[u] && low[u] == dfn[u])
        ck[p[u]] = 1;
}
```

KM

```
const int N = 1010, INF = 0x3f3f3f3f;
int w[N][N], lx[N], ly[N], match[N], slack[N];
bool vx[N], vy[N];
bool dfs(int i) {
    vx[i] = true;
    for (int j = 0; j < n; j ++) {
        if (lx[i] + ly[j] > w[i][j]) {
            slack[j] = min(slack[j], lx[i] + ly[j] - w[i][j]);
        } else if (!vy[j]) {
            vy[j] = true;
        if (match[j] < 0 || dfs(match[j])) {
                match[j] = i;
                return true;
            }
        }
    }
    return false;
}
int km() {
    memset(match, -1, sizeof match);
    memset(ly, 0, sizeof ly);
    for (int i = 0; i < n; i ++)
        lx[i] = *max_element(w[i], w[i] + n);
    for (int i = 0; i < n; i ++) {
        while(1) {
            memset(vx, 0, sizeof vx);
            memset(vy, 0, sizeof vy);
            memset(slack, 0x3f, sizeof slack);
            if (dfs(i)) break;
            int d = 0x3f3f3f3f;
            for (int i = 0; i < n; i ++) {
                if (!vy[i]) d = min(d, slack[i]);
            }
            for (int i = 0; i < n; i ++) {
                if (vx[i]) lx[i] -= d;
                if (vy[i]) ly[i] += d;
            }
        }
    }
    int z = 0;
    for (int i = 0; i < n; i ++) {
        if (w[match[i]][i] == -INF) return -1;
        z += w[match[i]][i];
    }
    return z;
}
```

匈牙利

```
//优化：随机一个匹配增广；改邻接表
const int N = 1010;
bool vis[N], map[N][N];
int n, m, t, x, lnk[N];
bool DFS (int u) {
    for (int v = 1; v <= m; v ++)
        if (map[u][v] && !vis[v]) {
            vis[v] = true;
            if (lnk[v] == -1 || DFS (lnk[v])) {
                lnk[v] = u;
                return true;
            }
        }
    return false;
}
int hungary () {
    int ans (0);
    memset (lnk, -1, sizeof lnk);
    for (int i = 1; i <= n; i ++) {
        memset (vis, 0, sizeof vis);
        if (DFS (i))
            ans ++;
    }
    return ans;
}
    map[a][b] = true;
```

最短路 D

```
const int N = , M = ;
int key[M], next[M], len[M], head[N], cnt, d[N];
struct Q {
    int d, x;
}h[N];
inline void add (const int & x, const int & y, const int & w) {
    key[cnt] = y;
    next[cnt] = head[x];
    len[cnt] = w;
    head[x] = cnt ++;
}
inline bool cmp (const Q & a, const Q & b) {
    return a.d > b.d;
}
int dijk (int S, int T) {
    memset (d, 0x3f, sizeof d);
    p = 1;
    h[1].x = S;h[1].d = 0;
    d[S] = 0;
    while (p) {
        int u = h[1].x;
        pop_heap (h + 1, h + 1 + p, cmp);
        p --;
        for (int i = head[u]; ~ i; i = next[i])
            if (d[key[i]] > d[u] + len[i]) {
                d[key[i]] = d[u] + len[i];
                p ++;h[p].x = key[i];h[p].d = d[key[i]];
                push_heap (h + 1, h + 1 + p, cmp);
            }
    }
    return dis[T];
}
```

最大流 Dinic

```
int S, T;
const int N = 500, M = 501000, INF = 0x3f3f3f3f;
struct Flow {
    int key[M], next[M], head[N], f[M], cnt, q[N], d[N];
    void init() {
        cnt = 0;
        memset (head, -1, sizeof head);
    }
    inline void add (int x, int y, int F)
    {
        key[cnt] = y;
        next[cnt] = head[x];
        f[cnt] = F;
        head[x] = cnt ++;

        key[cnt] = x;
        next[cnt] = head[y];
        f[cnt] = 0;
        head[y] = cnt ++;
    }
    bool SPFA ()
    {
        memset (d, -1, sizeof d);
        int h = 1, t = 2;
        q[1] = S;
        d[S] = 0;
        while (h < t)
        {
            int u = q[h ++];
            for (int i = head[u]; ~ i; i = next[i])
                if (f[i] && d[key[i]] == -1)
                    d[key[i]] = d[u] + 1, q[t ++] = key[i];
        }
        return d[T] != -1;
    }
    int DFS (int a, int b)
    {
        if (a == T)
            return b;
        int t (0), r (0);
        for (int i = head[a]; ~ i && r < b; i = next[i])
            if (f[i] && d[key[i]] == d[a] + 1)
            {
                t = DFS (key[i], min (b - r, f[i]));
                f[i] -= t, r += t, f[i ^ 1] += t;
            }
        if (!r) d[a] = -1;
        return r;
    }
    int work() {
        int z(0);
        while(SPFA())
            z += DFS(S, INF);
        return z;
    }
}flow;
```

费用流 SPFA

```
const int N = 41000, M = N * 10, INF = 0x3f3f3f3f;
namespace Flow {
    int key[M], next[M], head[N], cnt, cost[M], f[M];
    int pe[N], pv[N], S, T;
    int dis[N], q[N];
    bool vis[N];
    void init(int s, int t) {
        S = s, T = t;
        cnt = 0;
        memset (head, -1, sizeof head);
    }
    void add(int x, int y, int w, int flow) {
        key[cnt] = y;
        next[cnt] = head[x];
        cost[cnt] = w;
        f[cnt] = flow;
        head[x] = cnt ++;

        key[cnt] = x;
        next[cnt] = head[y];
        cost[cnt] = -w;
        f[cnt] = 0;
        head[y] = cnt ++;
    }
    bool spfa() {
        memset (dis, 0x3f, sizeof dis);
        memset (vis, 0, sizeof vis);
        int h = 1, t = 2;
        q[1] = S;
        vis[S] = true;
        dis[S] = 0;
        while(h < t) {
            int u = q[h ++];
            vis[u] = false;
            for (int i = head[u]; ~ i; i = next[i]) {
                int v = key[i];
                if (dis[v] > dis[u] + cost[i] && f[i]) {
                    dis[v] = dis[u] + cost[i];
                    pv[v] = u;
                    pe[v] = i;
                    if (!vis[v]) {
                        vis[v] = true;
                        q[t ++] = v;
                    }
                }
            }
        }
        return dis[T] != INF;
    }
    int z() {
        int tmp = INF;
        for (int i = T; i != S; i = pv[i])
            tmp = min(tmp, f[pe[i]]);
        for (int i = T; i != S; i = pv[i])
            f[pe[i]] -= tmp, f[pe[i] ^ 1] += tmp;
        return dis[T] * tmp;
    }
    int work() {
        int ans = 0;
        while(spfa())
            ans += z();
        return ans;
    }
}
```

树链剖分

```
const int N = 500000;
int p[N], s[N], d[N], tid[N], top[N], son[N], key[N], next[N], len[N],
head[N], cnt, tid_c, w[N], a[N], b[N], c[N], n;
inline void add (int x, int y, int w) {
    key[cnt] = y;
    next[cnt] = head[x];
    len[cnt] = w;
    head[x] = cnt ++;
}
void D1 (int x, int fa) {
    p[x] = fa;
    d[x] = d[fa] + 1;
    s[x] = 1;
    int t1 (0), t2 (0);
    for (int i = head[x]; ~ i; i = next[i])
    {
        if (key[i] == fa) continue;
        D1 (key[i], x);
        s[x] += s[key[i]];
        if (s[key[i]] > t1)
            t1 = s[key[i]], t2 = key[i];
    }
    son[x] = t2;
}
void D2 (int x, int TOP) {
    tid[x] = ++ tid_c;
    top[x] = TOP;
    if (son[x]) D2 (son[x], TOP);
    for (int i = head[x]; ~ i; i = next[i])
    {
        if (key[i] == p[x] || key[i] == son[x]) continue;
        D2 (key[i], key[i]);
    }
}
void D3 (int x, int fa) {
    for (int i = head[x]; ~ i; i = next[i])
    {
        if (key[i] == fa) continue;
        D3 (key[i], x);
        w[tid[key[i]]] = len[i];
    }
}
int ask (int x, int y) {
    int z (0);
    while (top[x] != top[y])
    {
        if (d[top[x]] < d[top[y]])
            swap (x, y);
        z = max (z, seg.query (tid[top[x]], tid[x], 1, n, 1));
        x = p[top[x]];
    }
    if (d[x] > d[y])
        swap (x, y);
    return max (z, seg.query (tid[son[x]], tid[y], 1, n, 1));
}
void Add(int x, int y, int v) {
    while(top[x] != top[y]) {
        if (d[top[x]] < d[top[y]])
            swap(x, y);
        seg.Add(tid[top[x]], tid[x], v, 1, n, 1);
        x = p[top[x]];
    }
    if (d[x] > d[y])
        swap(x, y);
    seg.Add(tid[son[x]], tid[y], v, 1, n, 1);
```

```
}
int main () {
    freopen("in", "r", stdin);
    tid_c = 0;cnt = 0;
    memset (head, -1, sizeof head);
    scanf ("%d", &n);
    for (int i = 1; i <= n - 1; i ++)
        scanf ("%d%d%d", &a[i], &b[i], &c[i]), add (a[i], b[i], c[i]),
add (b[i], a[i], c[i]);
    w[1] = 0;d[1] = 0;
    D1 (1, 1);D2 (1, 1);D3 (1, 1);
    seg.build(1, n, 1, w);
    char op[15];
    while(scanf("%s", op), op[0] != 'S') {
        int x, y, v;
        if (op[0] == 'M') {
            scanf("%d%d", &x, &y);
            printf("%d\n", ask(x, y));
        }
        if (op[0] == 'A') {
            scanf("%d%d%d", &x, &y, &v);
            Add(x, y, v);
        }
    }
    return 0;
}
```

LCA

```
inline void DFS (int x, int fa)
{
    f[x][0] = fa; d[x] = d[fa] + 1;
    for (int i = head[x]; ~ i; i = next[i])
        if (key[i] != fa)
            DFS (key[i], x);
}
inline int lca (int x, int y)
{
    if (d[x] < d[y])
        swap (x, y);
    for (int j = 20; j >= 0; j --)
        if (d[f[x][j]] >= d[y])
            x = f[x][j];
    if (x == y)
        return x;
    for (int j = 20; j >= 0; j --)
        if (f[x][j] != f[y][j])
            x = f[x][j], y = f[y][j];
    return f[x][0];
}
for(int j = 1; j <= 20; j++)
   for(int i = 1; i <= n; i++)
      f[i][j] = f[f[i][j-1]][j-1];
```

树分治

```
const int N = 50010;
int key[N], next[N], head[N], len[N], cnt, n, k;
void add(int x, int y, int w) {
    key[cnt] = y;
    next[cnt] = head[x];
    len[cnt] = w;
    head[x] = cnt ++;
}
int sum, root, son[N], f[N];
bool vis[N];
void findroot(int u, int fa) {
    son[u] = 1;
    f[u] = 0;
    for (int i = head[u]; ~i; i = next[i]) {
        int v = key[i];
        if (vis[v] || fa == v) continue;
        findroot(v, u);
        son[u] += son[v];
        f[u] = max(f[u], son[v]);
    }
    f[u] = max(f[u], sum - son[u]);
    if (f[u] < f[root]) root = u;
}
int a[N], a_c, dis[N];
void getdep(int u, int fa) {
    a[++a_c] = dis[u];
    for (int i = head[u]; ~i; i = next[i]) {
        int v = key[i];
        if (vis[v] || v == fa) continue;
        dis[v] = dis[u] + len[i];
        getdep(v, u);
    }
}
int calc(int u, int st) {
    dis[u] = st;
    a_c = 0;
    getdep(u, 0);
    sort(a + 1, a + 1 + a_c);
    int z = 0;
    for (int l = 1, r = a_c; l < r;) {
        if (a[l] + a[r] <= k) z += r - l, l ++;
        else r --;
    }
    return z;
}
int ans;
void work(int u) {
    ans += calc(u, 0);
    vis[u] = true;
    for (int i = head[u]; ~i; i = next[i]) {
        int v = key[i];
        if (vis[v]) continue;
        ans -= calc(v, len[i]);
        sum = son[v];
        root = 0;
        findroot(v, root);
        work(root);
    }
}
        f[0] = 0x3f3f3f3f;
        sum = n;
        root = 0;
        findroot(1, 0);
        work(1);
```

BSGS

```
//求解 A^x=B(mod C)(C 是质数)
//当 A=g 且(B,C)=1 时 x 为指标
int BSGS(int a, int b, int p) {
    a %= p, b %= p;
    if (b == 1) return 0;
    int cnt = 0;
    LL t = 1;
    for (int g = gcd(a, p); g != 1; g = gcd(a, p)) {
        if (b % g) return -1;
        p /= g, b /= g, t = t * a / g % p;
        ++cnt;
        if (b == t) return cnt;
    }
    hash.clear();
    int m = int(sqrt(1.0 * p) + 0.5);
    LL base = b;
    for (int i = 0; i < m; i ++) {
        hash.add(base, i);
        base = base * a % p;
    }
    base = pw(a, m, p);
    LL now = t;
    for (int i = 1; i <= m + 1; ++i) {
        now = now * base % p;
        if (hash.find(now))
            return i * m - hash.get(now) + cnt;
    }
    return -1;
}
```

线性筛法

```
bool vis[N];
int pr[N];
void getpr() {
    int N = 100000;
    memset (vis, 0, sizeof vis);
    int cnt = 0;
    for (int i = 2; i <= N; i ++) {
        if (!vis[i])
            pr[++ cnt] = i;
        for (int j = 1; j <= cnt; j ++) {
            if (i * pr[j] > N) break;
            vis[i * pr[j]] = true;
            if (i % pr[j] == 0) break;
        }
    }
}
```

原根 素因子分解

```
//此方法要求 n 为素数
//若 a 模 m 的阶等于 φ(m)，则称 a 为模 m 的一个原根
//2,4,p^a,2(p^a)有原根
struct ROOT {
    int n, a_c;
    int a[N];
    void divide(int n) {
        for (int i = 1; pr[i] * pr[i] <= n; i ++) {
            if (n % pr[i] != 0) continue;
            a[++ a_c] = pr[i];
            while(n % pr[i] == 0) n /= pr[i];
        }
        if (n != 1) a[++ a_c] = n;
    }
    bool ck(int x) {
        for (int i = 1; i <= a_c; i ++)
            if (pw(x, n / a[i], n) == 1)
                return false;
        return true;
    }
    int get(int _n) {
        a_c = 0;
        n = _n;
        divide(n - 1);
        int i;
        for (i = 2; ; i ++) {
            if (ck(i))
                return i;
        }
    }
}root;
```

二次剩余

```
typedef long long LL;
LL D, P;
struct Q{
    LL a, b;
    Q(LL _a = 0, LL _b = 0) : a(_a), b(_b) {}
    Q operator* (const Q& p) {
        return Q((a * p.a % P + b * p.b % P * D % P) % P, (a * p.b %
P + p.a * b % P) % P);
    }
};
LL qk(LL x, LL k) {
    LL z(1);
    for (; k; k >>= 1) {
        if (k & 1) z = z * x % P;
        x = x * x % P;
    }
    return z;
}
LL qk(Q x, LL k) {
    Q z(1, 0);
    for (; k; k >>= 1) {
        if (k & 1) z = z * x;
        x = x * x;
    }
    return z.a;
}
LL L(LL a) {
    return qk(a, (P - 1) / 2) == 1;
}
LL solve(LL n) {
    //P == 2 special judge
    if (P == 2) {
```

```
        if (n == 1) return 1;
        else return -1;
    }
    if (!L(n)) return -1;
    LL a;
    while(1) {
        a = rand() % P;
        D = ((a * a - n) % P + P) % P;
        if (!L(D)) break;
    }
    return qk(Q(a, 1), (P + 1) / 2);
}
int main() {
    srand(time(0));
    int T;
    scanf("%d", &T);
    while(T --) {
        int a, n, t;
        scanf("%d%d", &a, &n);
        a %= n;
        P = n;
        t = solve(a);
        if (t == -1)
            puts("No root");
        else {
            if (t == n - t)
                printf("%d\n", t);
            else
                printf("%d %d\n", min(t, n - t), max(t, n - t));
        }
    }
    return 0;
}
```

中国剩余定理

```
//求解 x=a[i](mod m[i])
LL china(int n, int *a, int *m) {
    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];
        gcd(m[i], w, d, d, y);
        x = (x + y * w * a[i]) % M;
    }
    return (x + M) % M;
}
```

莫比乌斯函数

```
void getmu() {
    const int N = 50000;
    mu[1] = 1;
    int cnt = 0;
    for (int i = 2; i <= N; i ++) {
        if (!vis[i]) {
            pr[++ cnt] = i;
            mu[i] = -1;
        }
        for (int j = 1; j <= cnt; j ++) {
            if (i * pr[j] > N) break;
            vis[i * pr[j]] = true;
            if (i % pr[j] == 0) {
                mu[i * pr[j]] = 0;
                break;
            } else
                mu[i * pr[j]] = -mu[i];
        }
    }
}
```

欧拉函数

```
void getphi() {
    const int N = 4000000;
    phi[1] = 1;
    int cnt = 0;
    for (int i = 2; i <= N; i ++) {
        if (!vis[i]) {
            pr[++ cnt] = i;
            phi[i] = i - 1;
        }
    for (int j = 1; j <= cnt; j ++) {
        if (i * pr[j] > N) break;
            vis[i * pr[j]] = true;
            if (i % pr[j] == 0) {
                phi[i * pr[j]] = phi[i] * pr[j];
                break;
            } else
                phi[i * pr[j]] = phi[i] * (pr[j] - 1);
            }
    }
}
```

狄利克雷卷积

```
for (int i = 1; i * i <= n; i ++)
     for (int j = i; i * j <= n; j ++)
          if (i == j) h[i * j] += i * phi[i];
          else h[i * j] += i * phi[j] + j * phi[i];
```

数论零件

```
LL gcd(LL x, LL y) {
    LL t;
     while(y) {
         t = y;
         y = x % y;
         x = t;
     }
     return x;
}
LL qk(LL x, LL n) {
    LL ans = 1;
    for (; n; n >>= 1)
    {
            if (n & 1)
                    ans = ans * x % P;
            x = x * x % P;
    }
    return ans;
}
LL multi(LL x, LL n) {
    if (n > x) swap(x, n);
    LL ans = 0;
    for (; n; n >>= 1)
    {
            if (n & 1)
                    ans = (ans + x) % P;
            x = (x + x) % P;
    }
    return ans;
}
void gcd(LL a, LL b, LL& d, LL& x, LL& y) {
    if (!b) { d = a; x = 1; y = 0; }
    else { gcd(b, a % b, d, y, x); y -= x * (a / b); }
}
LL inv(LL a, LL n) {
    LL d, x, y;
    gcd(a, n, d, x, y);
    return d == 1 ? (x + n) % n : -1;
}
```

线性基

```
for(int i=1;i<=n;i++)
for(int j=64;j>=1;j--)
{
    if(a[i]>>(j-1)&1)
    {
        if(!lb[j]){lb[j]=a[i];break;}
        else a[i]^=lb[j];
    }
}
```

矩阵乘法

```
struct M {
    long long s[170][170];
    M () { memset (s, 0, sizeof s); }
}x, ans;
inline M operator * (const M & a, const M & b)
{
    M c;
    for (int i = 0; i <= cnt; i ++)
        for (int j = 0; j <= cnt; j ++)
                for (int k = 0; k <= cnt; k ++)
                      c.s[i][j] += a.s[i][k] * b.s[k][j],
c.s[i][j] %= p;
        return c;
}
```

高斯消元

```
void gauss(int n, double a[N][N]) {
     int i, j, k, r;
     for (i = 0; i < n; i ++) {
          r = i;
          for (j = i + 1; j < n; j ++)
               if (fabs(a[j][i]) > fabs(a[r][i])) r = j;
          if (r != i) for (j = 0; j <= n; j ++) swap(a[r][j], a[i][j]);
          for (k = i + 1; k < n; k ++) {
               double f = a[k][i] / a[i][i];
               for (j = i; j <= n; j ++) a[k][j] -= f * a[i][j];
          }
     }
     for (i = n - 1; i >= 0; i --) {
          for (j = i + 1; j < n; j ++)
               a[i][n] -= a[j][n] * a[i][j];
          a[i][n] /= a[i][i];
     }
}
```

挂链 Hash

```
const int N = 40000, M = 100000, HEAD = 39997;
struct HASH {
    int cnt, head[N], next[M], len[M], key[M];
    HASH() {
        clear();
    }
    inline void clear() {
        memset (head, -1, sizeof head);
        cnt = 0;
    }
    inline void ADD(int x, int y, int w) {
        key[cnt] = y;
        next[cnt] = head[x];
        len[cnt] = w;
        head[x] = cnt ++;
    }
    inline int GETHEAD(int idx) {
        return idx % HEAD;
    }
    inline void add(int idx, int val) {
        int h = GETHEAD(idx);
        ADD(h, idx, val);
    }
    bool find(int idx) {
        int h = GETHEAD(idx);
        for (int i = head[h]; ~ i; i = next[i])
            if (key[i] == idx)
                return true;
        return false;
    }
    int get(int idx) {
        int h = GETHEAD(idx);
        for (int i = head[h]; ~ i; i = next[i])
            if (key[i] == idx)
                return len[i];
    }
};
```

手工堆

```
struct H {
    Q a[N];
    int size;
    inline int L (int x) { return x << 1; }
    inline int R (int x) { return x << 1 | 1; }
    void push (const Q & x) {
        a[++ size] = x;
        int t = size;
        while (t != 1 && a[t] > a[t >> 1])
            swap (a[t], a[t >> 1]), t >>= 1;
    }
    void dele (int t) {
        swap (a[t], a[size --]);
        int r = t;
        while (1) {
            if (L (t) <= size && a[L (t)] > a[r]) r = L (t);
            if (R (t) <= size && a[R (t)] > a[r]) r = R (t);
            if (r == t) break;
            swap (a[t], a[r]);t = r;
        }
    }
    Q top () {}
    void pop () {}
    void clear () {}
}h;
```

AC 自动机

```
int trie[170][4], map[255], fail[150], q[2000000], cnt;
bool v[170];
inline void add (char * s)
{
    int p = 0;
    for (; * s; s ++)
    {
        if (trie[p][map[*s]] == 0)
            trie[p][map[*s]] = ++ cnt;
        p = trie[p][map[*s]];
    }
    v[p] = true;
}
inline void build ()
{
    int h = 1, t = 2;
    while (h < t)
    {
        int u = q[h ++];
        for (int i = 0; i < 4; i ++)
            if (trie[u][i])
            {
                if (u)
fail[trie[u][i]]=trie[fail[u]][i],        v[trie[u][i]]        |=
v[trie[fail[u]][i]];
q[t ++] = trie[u][i];
            }
            else
                trie[u][i] = trie[fail[u]][i];
    }
}
```

KMP

```
inline void getp(){
    p[1] = 0;
    int j = 0;
    for(int i = 2; i <= lb; i ++){
        while(j > 0 && b[j + 1] != b[i])
            j = p[j];
        if(b[j + 1] == b[i])
            j = j + 1;
        p[i] = j;
    }
}
inline int KMP(){
    int ans(0), j = 0;
    for(int i = 1; i <= la; i ++){
        while(j > 0 && b[j + 1] != a[i])
            j = p[j];
        if(a[i] == b[j + 1])
            j = j + 1;
        if(j == lb)
            ans ++, j = p[j];
    }
    return ans;
}
```

最小生成树 P

```
const int MN = 0x3f3f3f3f, N = ;
int n;
int map[N][N], dis[N];
bool vis[N];

int prim()
{
    memset(vis, 0, sizeof vis);
    int res = 0;
    for(int i = 1; i <= n; i ++)
        dis[i] = MN;
    for(int i = 1; i <= n; i ++)
    {
        int t1 = 1, t2 = MN;
        for(int j = 1; j <= n; j ++)
            if(!vis[j] && dis[j] < t2)
                t1 = j, t2 = dis[j];
        vis[t1] = true;
        res += t2;
        for(int j = 1; j <= n; j ++)
            dis[j] = min(dis[j], map[t1][j]);
    }
    return res - MN;
}
```

最小生成树 K

```
int n, m, p[N];
struct Q {
    int a, b, c;
}e[M];
inline int F (int x) { return p[x] == x ? x : p[x] = F (p[x]); }
inline bool cmp (const Q & a, const Q & b) { return a.c < b.c; }

int main ()
{
    scanf ("%d%d", &n, &m);
    for (int i = 1; i <= m; i ++)
        scanf ("%d%d%d", &e[i].a, &e[i].b, &e[i].c);
    for (int i = 1; i <= n; i ++)
        p[i] = i;
    sort (e + 1, e + 1 + m, cmp);
    for (int i = 1; i <= m; i ++)
        if (F (e[i].a) != F (e[i].b))
            p[p[e[i].a]] = p[e[i].b], ;
    return 0;
}
```

几何基本

```
#define V P
const double eps = 1e-6;
inline int dcmp (double x) {
    return x < -eps ? -1 : x > eps;
}
struct P {
    double x, y;
    void scan() {
        scanf("%lf%lf", &x, &y);
    }
    P(double _x = 0, double _y = 0) : x(_x), y(_y) { }
    V operator + (V a) const {
        return V(x + a.x, y + a.y);
    }
    V operator - (V a) const {
        return V(x - a.x, y - a.y);
    }
    V operator * (double p) const {
        return V(p * x, p * y);
    }
    V operator / (double p) const {
        return V(x / p, y / p);
    }
    bool operator < (P a) const {
        return x < a.x || (dcmp(x - a.x) == 0 && y < a.y);
    }
    bool operator == (P a) const {
        return dcmp(x - a.x) == 0 && dcmp(y - a.y) == 0;
    }
};
inline double dot(V a, V b) {
    return a.x * b.x + a.y * b.y;
}
inline double len(V a) {
    return sqrt(dot(a, a));
}
inline double dis(P a, P b) {
    return len(b - a);
}
inline double ang(V a, V b) {
    return acos(dot(a, b) / len(a) / len(b));
}
inline double cross(V a, V b) {
    return a.x * b.y - a.y * b.x;
}
inline double area(P a, P b, P c) {
    return cross(b - a, c - a);
}
V rot(V a, double p) {
    return V(a.x * cos(p) - a.y * sin(p), a.x * sin(p) + a.y * cos(p));
}
V normal(V a) {
    double L = len(a);
    return V(-a.y / L, a.x / L);
}
P inter(P p, V v, P q, V w) {
    V u = p - q;
    double t = cross(w, u) / cross(v, w);
    return p + v * t;
}
double dis(P p, P a, P b) {
    V v1 = b - a, v2 = p - a;
    return fabs(cross(v1, v2)) / len(v1);
}
double dis2(P p, P a, P b) {
```

```
    if (a == b) return len(p - a);
    V v1 = b - a, v2 = p - a, v3 = p - b;
    if (dcmp(dot(v1, v2)) < 0) return len(v2);
    else if (dcmp(dot(v1, v3)) > 0) return len(v3);
    else return fabs(cross(v1, v2)) / len(v1);
}
P proj(P p, P a, P b) {
    V v = b - a;
    return a + v * (dot(v, p - a) / dot(v, v));
}
bool isInter(P a1, P a2, P b1, P b2) {
    double c1 = cross(a2 - a1, b1 - a1), c2 = cross(a2 - a1, b2 -
    a1),
           c3 = cross(b2 - b1, a1 - b1), c4 = cross(b2 - b1, a2 -
        b1);
    return dcmp(c1) * dcmp(c2) < 0 && dcmp(c3) * dcmp(c4) < 0;
}
bool onSeg(P p, P a1, P a2) {
    return dcmp(cross(a1 - p, a2 - p)) == 0 && dcmp(dot(a1 - p, a2
    - p)) < 0;
}
double area(P* p, int n) {
    double s = 0;
    p[n] = p[0];
    for (int i = 1; i < n; i ++)
        s += cross(p[i] - p[0], p[i + 1] - p[0]);
    return s / 2;
}
```

凸包

```
int graham(P* p, int n, P* ch) {
    sort(p, p + n);
    int m = 0;
    for (int i = 0; i < n; i ++) {
        while (m > 1 && cross(ch[m - 1] - ch[m - 2], p[i] - ch[m
        - 2]) <= 0) m --;
        ch[m ++] = p[i];
    }
    int k = m;
    for (int i = n - 2; i >= 0; i --) {
        while (m > k && cross(ch[m - 1] - ch[m - 2], p[i] - ch[m
        - 2]) <= 0) m --;
        ch[m ++] = p[i];
    }
    if (n > 1) m --;
    return m;
}
```

象限极角排序

```
inline int get(P a) {
     if( a.x > 0 && a.y >= 0) return 1;
     if( a.x <= 0 && a.y > 0) return 2;
     if( a.x < 0 && a.y <= 0) return 3;
     if( a.x >= 0 && a.y < 0) return 4;
     return 0;
}
inline bool cmp (V a, V b) {
     return get(a) < get(b) || (get(a) == get(b) && dcmp( cross(a,
     b) ) >0);
}
inline bool cmp2 (L a, L b) {
     return get(a.v) < get(b.v) || (get(a.v) == get(b.v) &&
     dcmp( cross(a.v, b.v) ) >0);
}
```

旋转卡壳

```
int rt(P* p, int n) {
    int z = 0;
    p[n + 1] = p[1];
    for (int i = 1, j = 2, r = 2, l = n; i <= n; i ++) {
        while (cross(p[i + 1] - p[i], p[j] - p[i]) <
cross(p[i + 1] - p[i], p[j + 1] - p[i]))
            j = j == n ? 1 : j + 1;
        while (dot(p[i + 1] - p[i], p[r] - p[i]) <
dot(p[i + 1] - p[i], p[r + 1] - p[i]))
            r = r == n ? 1 : r + 1;
        z = max(z, max(dis(p[i], p[j]), dis(p[i + 1], p[j + 1])));
    }
    return z;
}
```