SYSU wangty6 Code Template(S2016-17)

Contents

1 code

2 DataStructure

2.1 BIT-sumseek

```
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <algorithm>
#include <iostream>
#include <string>
#include <ctime>
#include <map>
#include <vector>
#include <set>
#include <cmath>
#include <queue>
using namespace std;
#define rep(i, s, t) for (int i = s; i <= t; i++)
#define dwn(i, s, t) for (int i = s; i >= t; i--)
#define edg(i, x) for (int i = head[x]; ~ i; i = next[i])
#define ctn(i, x) for (i = x.begin(); i != x.end(); i++)
#define clr(x) memset ((x), 0, sizeof (x))
#define size(x) (int)x.size()
typedef long long LL;
int read()
{
    int 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;
}
void print(LL x) {
    static int a[24];int n = 0;
    while(x > 0) {
        a[n++] = x % 10;
        x /= 10;
    }
    if (n == 0) a[n++] = 0;
    while(n--) putchar('0' + a[n]);
    putchar('\n');
}
void from(const char *s) {
        freopen(s, "r", stdin);
}
const int N = 301000;
int a[N], b[N], s[N], ans[N], x, n, fa[N], L;
void update(int x, int val) {
        for (; x <= L; x += x & -x)
                s[x] += val;
}
```

```
int query(int x) {
        int z = 0;
        for (; x; x -= x & -x)
                z += s[x];
        return z;
}
int sumseek(int val) {
        int z = 0, sum = 0;
        for (int i = 18; i >= 0; i --) {
                z += (1<<i);
                if (z > L || sum + s[z] >= val) z -= (1<<i);
                else sum += s[z];
        }
        return z + 1;
}
int find(int x) {
        return fa[x] == x ? x : fa[x] = find(fa[x]);
}
int main() {
        from("a.in");
        n = read();
        L = read();L += n;
        rep(i, 1, L) fa[i] = i;
        rep(i, 1, n) {
                a[i] = read();
                int right = find(a[i]);
                update(right, 1);
                fa[right] = find(right + 1);
                //rep(i, 1, L) printf("%d", find(i));puts("");
                a[i] = query(a[i] - 1);
        }
        int res = 0;
        dwn(i, n, 1) {
                x = sumseek(a[i] + 1);
                update(x, -1);
                res = max(res, x);
                ans[x] = i;
        }
        printf("%d\n", res);
        rep(i, 1, res) printf("%d ", ans[i]);
        return 0;
}
```

2.2 bitofchairtree

```
#include <cstdio>
#include <iostream>
#include <cstring>
#include <algorithm>
#define mid (l + r >> 1)
using namespace std;
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;
}
```

2.3 chairtree

```
#include <cstdio>
#include <algorithm>
#include <cstring>
#include <iostream>
using namespace std;
#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;
}
```

2.4 HeavyLightDecomposition

```
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];
int a[N], b[N], c[N];
int 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 Cover(int x, int y, int v) {
    while(top[x] != top[y]) {
        if (d[top[x]] < d[top[y]])
            swap(x, y);
        seg.Cover(tid[top[x]], tid[x], v, 1, n, 1);
        x = p[top[x]];
    }
    if (d[x] > d[y])
        swap(x, y);
    seg.Cover(tid[son[x]], tid[y], v, 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] == 'C' && op[1] == 'o') {
            scanf("%d%d%d", &x, &y, &v);
            Cover(x, y, v);
        }
        if (op[0] == 'A') {
            scanf("%d%d%d", &x, &y, &v);
            Add(x, y, v);
        }
    }
    return 0;
}
```

2.5 LCT

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <iostream>
#include <climits>
#include <numeric>
#include <vector>
using namespace std;
#define rep(i, s, t) for (int i = s; i <= t; i ++)
typedef int int64;
const int LEN = 3001000;
```

```
namespace LCT {
	struct Node {
		Node*p, *ch[2];
		int64 add, val;
		int size;
		bool isRoot;
		Node*fa;
		Node() {
			add = val = 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 pushup() {
			size = 1 + ch[0]->size + ch[1]->size;
		}
		void apply(int x) {
			add += x;
			val += x;
		}
		void pushdown();
		void setRoot(Node*f);
	} Tnull, *null = &Tnull;
	void Node::setRoot(Node*f) {
		fa = f;
		isRoot = true;
		p = null;
	}
	void Node::pushdown() {
		if (add) {
			rep(i, 0, 1)
				if (ch[i] != null)
					ch[i]->apply(add);
			add = 0;
		}
	}
	Node*make(int v) {
		Node* C = new Node();
		C->val = v;
		C->add = 0;
		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->pushdown();
		t->pushdown();
		bool d = t->d();
		p->p->sc(t, p->d());
		p->sc(t->ch[!d], d);
		t->sc(p, !d);
		p->pushup();
		if (p->isRoot) {
			p->isRoot = false;
			t->isRoot = true;
			t->fa = p->fa;
		}
	}
	void pushTo(Node*t) {
		static Node*stk[LEN];
		int top = 0;
		while (t != null) {
			stk[top++] = t;
			t = t->p;
		}
		for (int i = top - 1; i >= 0; --i)
			stk[i]->pushdown();
	}
	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->pushup();
	}
	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);
	}
	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->pushup();
		v->fa = null;
		v->isRoot = true;
		v->p = null;
	}
	void markPath(Node*u, Node*v, int x) {
		makeRoot(u);
		expose(v);
		splay(v);
		v->apply(x);
	}
```

```
}

struct Q {
        Q *suf, *go[26], *nxt;
        int val;
        LCT::Node* tree;
        Q() :
                        suf(0), val(0) {
                memset(go, 0, sizeof go);
                tree = LCT::make(0);
        }
}*root, *last;

void init() {
        root = last = new Q();
}

void extend(int w) {
        Q *p = last, *np = new Q();
        np->val = p->val + 1;
        while (p && !p->go[w])
                p->go[w] = np, p = p->suf;
        if (!p) {
                np->suf = root;
                LCT::addEdge(root->tree, np->tree);
        }
        else {
                Q *q = p->go[w];
                if (p->val + 1 == q->val) {
                        np->suf = q;
                        LCT::addEdge(q->tree, np->tree);
                } else {
                        Q *nq = new Q();
                        memcpy(nq->go, q->go, sizeof q->go);
                        nq->val = p->val + 1;
                        LCT::delEdge(q->suf->tree, q->tree);
                        nq->suf = q->suf; LCT::addEdge(q->suf->tree,
                            nq->tree);
                        q->suf = nq;  LCT::addEdge(nq->tree, q->tree);
                        np->suf = nq; LCT::addEdge(nq->tree, np->tree)
                            ;
                        LCT::pushTo(q->tree);
                        nq->tree->val = q->tree->val;
                        while (p && p->go[w] == q)
                                p->go[w] = nq, p = p->suf;
                }
        }
        last = np;
        markPath(root->tree, np->tree, 1);
        //for (; np; np = np->suf)
        //      np->size++;
}
int query(char *s) {
        Q* now = root;
        for (; *s; s ++) {
                now = now->go[*s - 'A'];
                if (now == 0) return 0;
        }
        LCT::pushTo(now->tree);
        return now->tree->val;
}
```

```
void Decode(char s[], int mask) {
        int ls = strlen(s);
        rep(i, 0, ls - 1) {
                mask = (mask * 131 + i) % ls;
                swap(s[i], s[mask]);
        }
}
int mask;
char a[LEN], s[LEN], op[10];
int n;
int main() {
        //freopen("b.in", "r", stdin);
        scanf("%d", &n);
        scanf("%s", a);
        init();
        int la = strlen(a);
        rep(i, 0, la - 1) {
                extend(a[i] - 'A');
        }
        rep(i, 1, n) {
                scanf("%s%s", op, s);
                int ls = strlen(s);
                Decode(s, mask);
                if (op[0] == 'Q') {
                        int tmp = query(s);
                        mask ^= tmp;
                        printf("%d\n", tmp);
                }
                if (op[0] == 'A') {
                        rep(i, 0, ls - 1)
                                extend(s[i] - 'A');
                }
        }
        return 0;
}
```

2.6 MonotonousQueue

```
struct T {
        int pos, val;
        T(){}
        T(int pos, int val):pos(pos),val(val){}
        bool operator > (const T& a) const {
                return val > a.val;
        }
}q[6001000];
struct Q {
        int h, t;
        void init() {
                h = 1, t = 1;
        }
        void insert(const T& x) {
                while(t > h && x > q[t - 1]) t--;
                q[t++] = x;
        }
        void pop(int pos) {
                while(t > h && q[h].pos < pos) h++;
        }
        int get() {
                return q[h].val;
```

```
        }
}que;
```

2.7 pb-ds-heap

```
#include<iostream>
#include<cstdio>
#include<cstring>
#include<ext/pb_ds/priority_queue.hpp>
#define ll long long
#define pa pair<ll,int>
#define llinf 9000000000000000000LL
using namespace std;
using namespace __gnu_pbds;
typedef __gnu_pbds::priority_queue<pa,greater<pa>,pairing_heap_tag >
    heap;
int n,m,cnt,last[1000005];
int T,rxa,rxc,rya,ryc,rp;
heap::point_iterator id[1000005];
int x,y,z;
ll dis[1000005];
struct data{int to,next,v;}e[10000005];
inline int read()
{
    int 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;
}
void insert(int u,int v,int w)
{
    e[++cnt].to=v;e[cnt].next=last[u];last[u]=cnt;e[cnt].v=w;
}
void dijkstra()
{
    heap q;
    for(int i=1;i<=n;i++)dis[i]=llinf;
    dis[1]=0;id[1]=q.push(make_pair(0,1));
    while(!q.empty())
    {
        int now=q.top().second;q.pop();
        for(int i=last[now];i;i=e[i].next)
            if(e[i].v+dis[now]<dis[e[i].to])
            {
                dis[e[i].to]=e[i].v+dis[now];
                if(id[e[i].to]!=0)
                    q.modify(id[e[i].to],make_pair(dis[e[i].to],e[i].
                        to));
                else id[e[i].to]=q.push(make_pair(dis[e[i].to],e[i].to
                    ));
            }
    }
}
int main()
{
    n=read();m=read();
    T=read();rxa=read();rxc=read();rya=read();ryc=read();rp=read();
    int a,b;
    for(int i=1;i<=T;i++)
    {
        x=((ll)x*rxa+rxc)%rp;
        y=((ll)y*rya+ryc)%rp;
        a=min(x%n+1,y%n+1);
        b=max(y%n+1,y%n+1);
        insert(a,b,100000000-100*a);
    }
    for(int i=1;i<=m-T;i++)
    {
        x=read(),y=read(),z=read();
        insert(x,y,z);
    }
    dijkstra();
    printf("%lld",dis[n]);
    return 0;
}
```

2.8 pd-ds-tree

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <ext/pb_ds/assoc_container.hpp> // Common file
#include <ext/pb_ds/tree_policy.hpp> // Including
    tree_order_statistics_node_update
#include <ext/pb_ds/detail/standard_policies.hpp>
using namespace std;
using namespace __gnu_pbds;
const int N = 101000;
tree<int, null_type, less<int>, rb_tree_tag,
    tree_order_statistics_node_update> st[N], ed[N];
int t[N], x[N], T, a[N], b[N], b_c;
int main ()
{
    int n, m;
    scanf ("%d", &n);
    for (int i = 1; i <= n; i ++)
        scanf ("%d%d%d", &a[i], &t[i], &x[i]);

    b_c = n;
    for (int i = 1; i <= n; i ++)
        b[i] = x[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 ++)
        x[i] = lower_bound (b + 1, b + 1 + b_c, x[i]) - b;

    for (int i = 1; i <= n; i ++) {
        if (a[i] == 1) {
                st[x[i]].insert(t[i]);
                }
        if (a[i] == 2) {
                ed[x[i]].insert(t[i]);
        }
        if (a[i] == 3) {
                printf("%d\n",
                        st[x[i]].order_of_key(t[i])
                          - ed[x[i]].order_of_key(t[i]));
        }
    }
```

```
        return 0;
}
```

2.9 SegTree-Search

```
#include <iostream>
#include <cstdio>
#include <cstring>
#include <cstdlib>
#include <vector>
#define lc idx << 1
#define rc idx << 1 | 1
#define lson l, mid, lc
#define rson mid + 1, r, rc
using namespace std;
const int N = 401000;
typedef long long LL;
LL sum[N << 2];
int cov[N << 2], mn[N << 2];
void pushup(int idx) {
        sum[idx] = sum[lc] + sum[rc];
        mn[idx] = mn[lc];
}
void pushdown(int l, int r, int mid, int idx) {
        if (cov[idx] != 0) {
                cov[lc] = cov[rc] = cov[idx];
                mn[lc] = cov[idx];
                mn[rc] = cov[idx];
                sum[lc] = (LL)cov[idx] * (mid - l + 1);
                sum[rc] = (LL)cov[idx] * (r - mid);
                cov[idx] = 0;
        }
}
void build(int l, int r, int idx) {
        if (l == r) {
                sum[idx] = l;
                mn[idx] = l;
                return ;
        }
        int mid = (l + r) >> 1;
        build(lson);
        build(rson);
        pushup(idx);
}
void update(int L, int R, int val, int l, int r, int idx) {
        if (L > R) return ;
        if (L <= l && r <= R) {
                cov[idx] = val;
                mn[idx] = val;
                sum[idx] = (LL)val * (r - l + 1);
                return ;
        }
        int mid = (l + r) >> 1;
        pushdown(l, r, mid, idx);
        if (L <= mid) update(L, R, val, lson);
        if (R > mid) update(L, R, val, rson);
        pushup(idx);
}
int left(int L, int R, int val, int l, int r, int idx) {
        if (l == r) {
                if (mn[idx] < val) return l;
                else return 0;
        }
        int mid = (l + r) >> 1;
        pushdown(l, r, mid, idx);
        if (R <= mid) {
                return left(L, R, val, lson);
        }
        else if (L > mid) {
                return left(L, R, val, rson);
        }
        else {
                if (mn[rc] < val) return left(L, R, val, rson);
                else return left(L, R, val, lson);
        }
}
```

2.10 SegTree

```
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;
```

2.11 Splay-BZOJ1588

```
#include<cstdio>
#include<iostream>
#include<algorithm>
using namespace std;
const int MAX_N = 1001000 + 10;
const int INF = ~0U >> 2;
struct Node {
        Node*ch[2], *p;
        int size, val, gcd;
        Node() {
                size = 0;
                val = gcd = 0;
        }
        bool d() {
                return this == p->ch[1];
        }
        void setc(Node*c, int d) {
                ch[d] = c;
                c->p = this;
        }
        void relax();
        void upd() {
                size = ch[0]->size + ch[1]->size + 1;
                gcd = __gcd(ch[0]->gcd, __gcd(ch[1]->gcd, val));
        }
} Tnull, *null = &Tnull;
Node mem[MAX_N], *C = mem;

void Node::relax() {
}

Node*make(int v) {
        C->ch[0] = C->ch[1] = null;
        C->size = 1;
        C->val = v;
        C->gcd = v;
        return C++;
}

Node*root;

Node*rot(Node*t) {
        Node*p = t->p;
        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;
}

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* insert(Node* t, int val) {
        bool d = val > t->val;
        if (t->ch[d] == null) {
                Node* p = make(val);
                t->setc(p, d);
                return p;
        }
        return insert(t->ch[d], val);
}

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* find(Node* t, int val) {
    bool d = val > t->val;
    if (val == t->val) return t;
    return find(t->ch[d], val);
}
int mx(Node* x) {
    if (x == null) return -INF;
    for (Node* t = x; ; t = t->ch[1])
        if (t->ch[1] == null) return t->val;
}
int mn(Node* x) {
    if (x == null) return INF;
    for (Node* t = x; ; t = t->ch[0])
        if (t->ch[0] == null) return t->val;
}
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];
}
void travel(Node* t) {
    if (t == null) return ;
    travel(t->ch[0]);
    printf("%d ", t->val);
    travel(t->ch[1]);
}
void ins(int val) {
    if (root == null)
        root = make(val), root->p = null;
    else
        splay(insert(root, val));
}
int n, m;
char s[10];
int x;
int main() {
    freopen("a.in", "r", stdin);
    scanf("%d", &n);
    root = null;
    scanf("%d", &x);
    ins(x);
    int ans = x;
    for (int i = 2; i <= n; i ++) {
        if (scanf("%d", &x) == -1) x = 0;
        ins(x);
        ans += min(mn(root->ch[1]) - x, x - mx(root->ch[0]));
    }
    printf("%d\n", ans);
}
```

2.12 Splay-Remove

```
#include<cstdio>
#include<iostream>
#include<algorithm>
using namespace std;
const int MAX_N = 1001000 + 10;
const int INF = ~0U >> 1;
struct Node {
    Node*ch[2], *p;
    int size, val, gcd;
    Node() {
        size = 0;
        val = gcd = 0;
    }
    bool d() {
        return this == p->ch[1];
    }
    void setc(Node*c, int d) {
        ch[d] = c;
        c->p = this;
    }
    void relax();
    void upd() {
        size = ch[0]->size + ch[1]->size + 1;
        gcd = __gcd(ch[0]->gcd, __gcd(ch[1]->gcd, val));
    }
} Tnull, *null = &Tnull;
Node mem[MAX_N], *C = mem;
void Node::relax() {
}
Node*make(int v) {
    C->ch[0] = C->ch[1] = null;
    C->size = 1;
    C->val = v;
    C->gcd = v;
    return C++;
}
Node*root;
Node*rot(Node*t) {
    Node*p = t->p;
    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;
}
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* insert(Node* t, int val) {
```

```
        for (; ; ) {
                bool d = val > t->val;
                if (t->ch[d] == null) {
                        Node* p = make(val);
                        t->setc(p, d);
                        return p;
                }
                t = t->ch[d];
        }
}
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* find(Node* t, int val) {
        bool d = val > t->val;
        if (val == t->val) return t;
        return find(t->ch[d], val);
}
Node* mx(Node* t) {
        for (; ; t = t->ch[1])
                if (t->ch[1] == null) return t;
}
Node* mn(Node* t) {
        for (; ; t = t->ch[0])
                if (t->ch[0] == null) return t;
}
Node* remove(Node* t) {
        splay(t);
        if (root->ch[0] != null) {
                splay(mx(root->ch[0]), root);
                root->ch[0]->setc(root->ch[1], 1);
                root = root->ch[0];
                root->p = null;
                root->upd();
        } else {
                root = root->ch[1];
                root->p = null;
        }
}
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];
}
void travel(Node* t) {
        if (t == null) return ;
        travel(t->ch[0]);
        printf("%d ", t->val);
        travel(t->ch[1]);
}
```

```
void ins(int val) {
        if (root == null)
                root = make(val), root->p = null;
        else
                splay(insert(root, val));
}
void rm(int val) {
        remove(find(root, val));
}


int n, m;
char s[10];
int x;
int main() {
        scanf("%d", &n);
        root = null;
        for (int i = 1; i <= n; ++i) {
                scanf("%s%d", s, &x);
                if (s[0] == '+') {
                        ins(x);
                } else {
                        rm(x);
                }
                if (root != null) printf("%d\n", root->gcd);
                else printf("1\n");
        }
}
```

2.13 Splay-RevFlag

```
#include<cstdio>
#include<iostream>
#include<algorithm>
using namespace std;
const int MAX_N = 50000 + 10;
const int INF = ~0U >> 1;
struct Node {
        Node*ch[2], *p;
        int size, val, mx;
        int add;
        bool rev;
        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;
}

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);
                }
        }
}
```

2.14 Splay-Sequence-SplitMerge

2.15 Splay-SumSearch-RandomSplay

C++

```
#include <cstdio>
#include <iostream>
#include <queue>
#include <cstdlib>
#include <ctime>
using namespace std;
#define rep(i,s,t) for(int i=s;i<=t;i++)
#define dwn(i,s,t) for(int i=s;i>=t;i--)
typedef long long LL;
struct Node {
        Node*ch[2], *p;
        int size, val;
        LL sum;
        Node() {
                size = 0;
                val = sum = 0;
        }
        bool d() {
                return this == p->ch[1];
        }
        void setc(Node*c, int d) {
                ch[d] = c;
                c->p = this;
        }
        void relax();
        void upd() {
                size = ch[0]->size + ch[1]->size + 1;
                sum = ch[0]->sum + ch[1]->sum + val;
        }
} Tnull, *null = &Tnull;
Node mem[1001000], *C = mem;
void Node::relax() {
}
Node*make(int v) {
        C->ch[0] = C->ch[1] = null;
        C->size = 1;
        C->val = v;
        C->sum = v;
        return C++;
}
Node*root;
Node*rot(Node*t) {
        Node*p = t->p;
        bool 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;
}
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();
}
void random_spaly() {
        if (C - mem < 10) return ;
        int t = rand() % (C - mem - 1) + 1;
        splay(mem + t);
}
Node* insert(Node* t, int val) {
        for (; ; ) {
                bool d = val > t->val;
                if (t->ch[d] == null) {
                        Node* p = make(val);
                        t->setc(p, d);
                        return p;
                }
                t = t->ch[d];
        }
}
int select(LL k) {
        int z = 0;
        for (Node*t = root;;) {
                t->relax();
                if (t == null)
                        return z;
                if (k >= t->sum)
                        return z + t->size;
                LL c = t->ch[0]->sum + t->val;
                if (k == c)
                        return z + t->ch[0]->size + 1;
                if (k > c)
                        k -= c, z += t->ch[0]->size + 1, t = t->ch[1];
                else
                        t = t->ch[0];
        }
}
void ins(int val) {
        if (root == null)
                root = make(val), root->p = null;
        else
                splay(insert(root, val));
}
int read()
{
    int 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;
}
const int N = 1001000;
int n, T, K, a[N], b[N], c[N], ans;
priority_queue<int> pq;
int main() {
        srand(time(0));
//      freopen("c.in", "r", stdin);
        root = null;
        scanf("%d%d%d", &n, &T, &K);
        rep(i, 1, n) a[i] = read();
        rep(i, 1, n) b[i] = read();
```

```
        rep(i, 1, n) c[i] = read();
        LL sum = 0;
        bool flag = false;
        rep(i, 1, n) {
                if (c[i] == 1) {
                        pq.push(b[i]);
                        sum += b[i];
                        if (pq.size() > K) {
                                int x = pq.top();
                                pq.pop();
                                sum -= x;
                                ins(x);
                        }
                } else
                        ins(b[i]);
                if (pq.size() == K) {
                        if ((LL)T >= a[i] + sum)
                                flag = true, ans = max(ans, K + select
                                    (T - a[i] - sum));
                }
                random_spaly();
        }
        printf("%d\n", flag ? ans : -1);
        return 0;
}
```

2.16 zkw-segtree

```
#include <stdio.h>

const int MAXN = 10010;
int n, q;
int a[MAXN];
struct tree
{
        struct S
        {
                int len, lc, rc, count;
                S(): len(0), lc(0), rc(0), count(0) {}
                S(bool x): len(1), lc(x), rc(x), count(x) {}
                S(const S &L, const S &R):
                        len(L.len + R.len), lc(L.lc), rc(R.rc),
                        count(L.count + R.count + L.rc * R.lc)
                {
                        if (lc == L.len) lc += R.lc;
                        if (rc == R.len) rc += L.rc;
                }
        }T[MAXN * 2];

        void build(int x)
        {
                for (int i = 0; i < n; ++i) T[i + n] = S(a[i] % x ==
                    0);
                for (int i = n - 1; i; --i) T[i] = S(T[i<<1], T[i
                    <<1|1]);
        }

        void update(int i, bool x)
        {
                T[i += n] = x;
                while (i >>= 1) T[i] = S(T[i<<1], T[i<<1|1]);
        }

        int query(int l, int r)
        {
                S L, R;
                for (l += n, r += n; l <= r; l >>= 1, r >>= 1) {
                        if ( l&1) L = S(L, T[l++]);
                        if (~r&1) R = S(T[r--], R);
                }
                return S(L, R).count;
        }
}T[101];

int tot;
int prime[101];
int mu[101];
bool vis[101];

int mm[101];

void init()
{
        for (int i = 0; i < n; ++i) scanf("%d", &a[i]);
        for (int i = 1; i <= 100; ++i) T[i].build(i);
}

void solve()
{
        while (q--) {
                char op[5];
                scanf("%s", op);
                if (op[0] == 'M') {
                        int p, v;
                        scanf("%d%d", &p, &v);
                        for (int i = 1; i <= 100; ++i) {
                                T[i].update(p, v % i == 0);
                        }
                        a[p] = v;
                } else {
                        int l, r;
                        scanf("%d%d", &l, &r);
                        for (int i = 1; i <= 100; ++i) {
                                mm[i] = T[i].query(l, r - 1);
                        }
                        int ans = 0;
                        for(int i = 100; i >= 1; i --){
                                for (int j = i + i; j <= 100; j += i)
                                        mm[i] = mm[i] - mm[j];
                                if (mm[i]) ans++;
                        }
                        printf("%d\n", ans);
                }
        }
}

int main()
{
        //freopen("a.in", "r", stdin);
        while (scanf("%d%d", &n, &q) != EOF) {
                init();
```

```
                solve();
        }
}
```

3 DynamicProgramming

3.1 MultiPackage

```
#include <cstdio>
#include <cstring>
const int N = 1001000, M = 1001000;
int n, m, v[N], c[N], f[M];
void init() {
        memset (f, 0, sizeof f);
}
void solve() {

        for (int i = 1; i <= n; i ++)
                scanf("%d", &v[i]);
        for (int i = 1; i <= n; i ++)
                scanf("%d", &c[i]);
        //w[i]
        f[0] = 1;
        for (int i = 1; i <= n; i ++)
                for (int d = 0; d < v[i]; d ++) {
                        que.init();
                        for (int k = 0; k <= (m - d) / v[i]; k ++) {
                                que.insert(T(k, f[d + k * v[i]]));
                                que.pop(k - c[i]);
                                f[d + k * v[i]] = que.get();
                        }
                }
        int ans = 0;
        for (int i = 1; i <= m; i ++)
                ans += f[i];
        printf("%d\n", ans);
}
int main() {
        while(scanf("%d%d", &n, &m), n + m) {
                init();
                solve();
        }
        return 0;
}
```

4 Geometry

4.1 circlescan

```
/**
 * Copyright ? 2016 Authors. All rights reserved.
 *
 * FileName: L.cpp
 * Author: SYSU_Team3 <msy5>
 * Date: 2016-04-13
 */
#include <bits/stdc++.h>

using namespace std;

#define rep(i,n) for (int i = 0; i < (n); ++i)
#define For(i,s,t) for (int i = (s); i <= (t); ++i)
#define foreach(i,c) for (__typeof(c.begin()) i = c.begin(); i != c.
    end(); ++i)

typedef long long LL;

const int inf = 0x3f3f3f3f;
const LL infLL = 0x3f3f3f3f3f3f3f3fLL;
const int N = 201000;
double Time;
struct Q {
        int x, y, r;
        double getX(bool flag) {
                if (flag) return x + r;
                else return x - r;
        }
        double getY(bool flag) {
                double ret = sqrt(1.0 * r * r - (Time - x) * (Time - x
                    ));
                if (flag) return   y + ret;
                else return y - ret;
        }
        void scan() {
                scanf("%d%d%d", &x, &y, &r);
        }
}p[N];
struct E {
        int x, y, id;
        bool flag;
        E() {}
        E(int x, int y, int id, bool flag) : x(x), y(y), id(id), flag(
            flag) {}
        bool operator < (const E& a) const {
                return x == a.x ? y < a.y : x < a.x;
        }
}event[N];
struct P {
        int id;
        bool flag;
        P () {}
        P (int id, bool flag) : id(id), flag(flag) {}
        bool operator < (const P& a) const {
                double y1 = p[id].getY(flag);
                double y2 = p[a.id].getY(a.flag);
                return y1 < y2 || y1 == y2 && flag < a.flag;
        }
        bool operator == (const P& a) const {
                return id == a.id;
        }
};
set<P> ss;
set<P>::iterator up, dn;
int n, fa[N], key[N], nxt[N], head[N], cnt;
void add(int x, int y) {
        key[cnt] = y;
```

```
        nxt[cnt] = head[x];
        head[x] = cnt++;
        fa[y] = x;
}
void init() {
        ss.clear();
        cnt = 0;
        memset (head, -1, sizeof head);
}
int sg(int u) {
        if (head[u] == -1) return 0;
        int t = 0;
        for (int i = head[u]; ~ i; i = nxt[i]) {
                int v = key[i];
                t = t ^ (sg(v) + 1);
        }
        return t;
}
void solve() {
        scanf("%d", &n);
        for (int i = 1; i <= n; i ++) {
                p[i].scan();
                event[i] = E(p[i].getX(0), p[i].y, i, 0);
                event[n + i] = E(p[i].getX(1), p[i].y, i, 1);
        }
        sort(event + 1, event + 1 + n + n);
        for (int i = 1; i <= n; i ++)
                fa[i] = i;
        for (int i = 1; i <= n + n; i ++) {
                Time = event[i].x;
                if (event[i].flag == 0) {
                        if (ss.empty()) {
                                ss.insert(P(event[i].id, 0));
                                ss.insert(P(event[i].id, 1));
                                add(0, event[i].id);
                                continue;
                        }
                        up = ss.upper_bound(P(event[i].id, 1));
                        dn = ss.lower_bound(P(event[i].id, 0));
                        if (dn == ss.begin() || up == ss.end())
                                add(0, event[i].id);
                        else {
                                dn--;
                                int t1 = up->id;
                                int t2 = dn->id;
                                if (t1 == t2)
                                        add(t1, event[i].id);
                                else if (fa[t1] == fa[t2])
                                        add(fa[t1], event[i].id);
                                else if (fa[t1] == t2)
                                        add(t2, event[i].id);
                                else if (fa[t2] == t1)
                                        add(t1, event[i].id);
                                else
                                        add(0, event[i].id);
                        }
                        ss.insert(P(event[i].id, 0));
                        ss.insert(P(event[i].id, 1));
                } else {
                        ss.erase(ss.find(P(event[i].id, 0)));
                        ss.erase(ss.find(P(event[i].id, 1)));
                }
        }
        for (int i = 1; i <= n; i ++) {
                printf("%d: ", i);
                for (int j = head[i]; ~ j; j = nxt[j])
                        printf("%d ", key[j]);
                printf("\n");
        }
        printf("%s\n", sg(0) ? "Alice" : "Bob");
}
int main()
{
        freopen("L.in", "r", stdin);
        int T;
        scanf("%d", &T);
        while(T --) {
                init();
                solve();
        }
        return 0;
}
```

4.2 GeoBasic

```
#include <cstdio>
#include <deque>
#include <iostream>
#include <cmath>
#include <vector>
#include <cstring>
#include <algorithm>
using namespace std;
const double eps = 1e-10;
int dcmp(double x) {
        return x < -eps ? -1 : x > eps;
}
const double pi = acos(-1.0);
inline double sqr(double x) {
        return x * x;
}
struct point {
        double x, y;
        point() : x(0), y (0) {}
        point(double a, double b) : x(a), y(b) {}
        void input() {
                scanf("%lf%lf", &x, &y);
        }
        void print() {
            printf("%lf %lf\n", x, y);
    }
        friend point operator +(const point &a, const point &b) {
                return point(a.x + b.x, a.y + b.y);
        }
        friend point operator -(const point &a, const point &b) {
                return point(a.x - b.x, a.y - b.y);
        }
        friend bool operator ==(const point &a, const point &b) {
                return dcmp(a.x - b.x) == 0 && dcmp(a.y - b.y) == 0;
        }
        friend point operator *(const double &a, const point &b) {
```

```
                return point(a * b.x, a * b.y);
        }
        friend point operator *(const point &b, const double &a) {
                return point(a * b.x, a * b.y);
        }
        friend point operator /(const point &a, const double &b) {
                return point(a.x / b, a.y / b);
        }
        double norm() const {
                return sqrt(sqr(x) + sqr(y));
        }
};
double det(const point &a, const point &b) {
        return a.x * b.y - a.y * b.x;
}
double dot(const point &a, const point &b) {
        return a.x * b.x + a.y * b.y;
}
double dis(const point &a, const point &b) {
        return (a - b).norm();
}
point rotate(const point &p, double A) {
        double tx = p.x, ty = p.y;
        return point(tx * cos(A) - ty * sin(A), tx * sin(A) + ty * cos
            (A));
}

struct line {
        point a, b;
        double ang;
        line() {}
        line(point x, point y) : a(x), b(y) {
                ang = atan2(b.y - a.y, b.x - a.x);
        }
        void input() {
                a.input();
                b.input();
        }
};

//line and seg are different

double dis(const point p, const point s, const point t) {
        if (dcmp(dot(p - s, t - s)) == -1) return (p - s).norm();
        if (dcmp(dot(p - t, s - t)) == -1) return (p - t).norm();
        return fabs(det(s - p, t - p) / dis(s, t));
}
void proj(const point p, const point s, const point t, point &cp) {
        double r = dot((t - s), (p - s)) / dot(t - s, t - s);
        cp = s + r * (t - s);
}
bool onseg(point p, point s, point t) {
        return dcmp(det(p - s, t - s)) == 0 && dcmp(dot(p - s, p - t))
            <= 0;
}
bool parallel(line a, line b) {
        return dcmp(det(a.a - a.b, b.a - b.b)) == 0;
}
bool inter(line a, line b) {
        double c1 = det(b.a - a.a, a.b - a.a), c2 = det(b.b - a.a, a.b
            - a.a);
        double c3 = det(a.a - b.a, b.b - b.a), c4 = det(a.b - b.a, b.b
             - b.a);
        return dcmp(c1) * dcmp(c2) < 0 && dcmp(c3) * dcmp(c4) < 0;
}
point interpoint(line a, line b) {
        //if (inter(a, b) == false) return false;
        point u = a.a - b.a;
        point v = a.b - a.a;
        point w = b.b - b.a;
        double t = det(w, u) / det(v, w);
        return a.a + v * t;
}
line move_d(line a, const double &len) {
        point d = a.b - a.a;
        d = d / d.norm();
        d = rotate(d, pi / 2);
        return line(a.a + d * len, a.b + d * len);
}
bool cmpxy(const point &a, const point &b) {
        if (dcmp(a.x - b.x) == 0)
                return a.y < b.y;
        return a.x < b.x;
}
#define points vector<point>
#define lines vector<line>
#define next(x) ((x) + 1) % n
double area(points &p) {
        double z = 0;
        for (int i = 0; i < (int)p.size() - 1; i ++)
                z += det(p[i] - p[0], p[i + 1] - p[0]);
        return fabs(z) / 2;
}
void convex_hull(points &a, points &res) {
        res.resize(2 * a.size() + 10);
        sort(a.begin(), a.end(), cmpxy);
        a.erase(unique(a.begin(), a.end()), a.end());
        int m = 0;
        for (int i = 0; i < (int)a.size(); i ++) {
                while(m > 1 && dcmp(det(res[m - 1] - res[m - 2], a[i]
                    - res[m - 2])) <= 0) --m;
                res[m++] = a[i];
        }
        int k = m;
        for (int i = (int)a.size() - 2;  i >= 0; i --) {
                while(m > k && dcmp(det(res[m - 1] - res[m - 2], a[i]
                    - res[m - 2])) <= 0) --m;
                res[m++] = a[i];
        }
        res.resize(m);
        //if (a.size() > 1) res.resize(m - 1);
}
points a, qs;
void max_dis() {
        convex_hull(a, qs);
        //printf("%d\n", qs.size());
        int n = qs.size();
        if (n == 2) {
                printf("%.0f\n", dis(qs[0], qs[1]));
                return ;
        }
        int i = 0, j = 0;
```

```
        for (int k = 0; k < n; k ++) {
                if (!cmpxy(qs[i], qs[k])) i = k;
                if (cmpxy(qs[j], qs[k])) j = k;
        }
        double res = 0;
        int si = i, sj = j;
        while(i != sj || j != si) {
                res = max(res, dis(qs[i], qs[j]));
                if (det(qs[(i + 1) % n] - qs[i], qs[(j + 1) % n] - qs[
                    j]) < 0) {
                        i = (i + 1) % n;
                } else {
                        j = (j + 1) % n;
                }
        }
        printf("%.0f\n", res);
}
void cut(points &p, point b, point a, points &res) {
        res.clear();
        int n = p.size();
        for (int i = 0; i < n; i ++) {
                point c = p[i];
                point d = p[next(i)];
                if (dcmp(det(b - a, c - a)) >= 0) res.push_back(c);
                if (dcmp(det(b - a, c - d)) != 0) {
                        point cp = interpoint(line(a, b), line(c, d));
                        if (onseg(cp, c, d)) res.push_back(cp);
                }
        }
}

bool onleft(point a, line p) {
        return dcmp(det(a - p.a, p.b - p.a)) < 0;
}
bool cmpang(const line &a, const line &b) {
        if (dcmp(a.ang - b.ang) == 0)
                return onleft(a.a, b);
        return a.ang < b.ang;
}
int halfplane(lines &v, points &res) {
        sort(v.begin(), v.end(), cmpang);
        deque<line> q;
        deque<point> ans;
        q.push_back(v[0]);
        for (int i = 1; i < int(v.size()); i ++) {
                if (dcmp(v[i].ang - v[i - 1].ang) == 0) continue;
                while(ans.size() && !onleft(ans.back(), v[i])) {
                        ans.pop_back();
                        q.pop_back();
                }
                while(ans.size() && !onleft(ans.front(), v[i])) {
                        ans.pop_front();
                        q.pop_front();
                }
                ans.push_back(interpoint(q.back(), v[i]));
                q.push_back(v[i]);
        }
        while(ans.size() && !onleft(ans.back(), q.front())) {
                ans.pop_back();
                q.pop_back();
        }
        while(ans.size() && !onleft(ans.front(), q.back())) {
                ans.pop_front();
                q.pop_front();
        }
        ans.push_back(interpoint(q.back(), q.front()));
        res = points(ans.begin(), ans.end());
        return ans.size(); //you must use the size to assure an empty
            set, area dont has the accuracy we need
}
const int N = 50010;
point p[N];
int n;
double r;
void init() {
}
double mysqrt(double x) {
    return sqrt(max(0.0, x));
}
void circle_inter_line(point a, point b, point o, double r, point ret
    [], int &num) {
    point p = b - a;
    point q = a - o;
    double A = dot(p, p);
    double B = 2 * dot(p, q);
    double C = dot(q, q) - sqr(r);
    double delta = B * B - 4 * A * C;
    num = 0;
    if (dcmp(delta) >= 0) {
        double t1 = (-B - mysqrt(delta)) / (2 * A);
        double t2 = (-B + mysqrt(delta)) / (2 * A);
        if (t1 <= 1 && t1 >= 0) {
            ret[num++] = a + t1 * p;
        }
        if (t2 <= 1 && t2 >= 0) {
            ret[num++] = a + t2 * p;
        }
    }
}

double sector_area(const point &a, const point &b) {
    double theta = atan2(a.y, a.x) - atan2(b.y, b.x);
    while(theta <= 0) theta += 2 * pi;
    while(theta > 2 * pi) theta -= 2 * pi;
    theta = min(theta, 2 * pi - theta);
    return r * r * theta / 2;
}
double calc(const point &a, const point &b) {
    point p[2];
    int num = 0;
    int ina = dcmp(a.norm()- r) < 0;
    int inb = dcmp(b.norm() - r) < 0;
    if (ina) {
        if (inb) {
            return fabs(det(a, b)) / 2;
        } else {
            circle_inter_line(a, b, point(0, 0), r, p, num);
            return sector_area(b, p[0]) + fabs(det(a, p[0])) / 2;
        }
    } else {
        if (inb) {
            circle_inter_line(a, b, point(0, 0), r, p, num);
```

```
            return sector_area(p[0], a) + fabs(det(p[0], b)) / 2;
        } else {
            circle_inter_line(a, b, point(0, 0), r, p, num);
            if (num == 2) {
                return sector_area(a, p[0]) + sector_area(p[1], b) +
                    fabs(det(p[0], p[1])) / 2;
            } else {
                return sector_area(a, b);
            }
        }
    }
}
double area() {
    double ret = 0;
    for (int i = 0; i < n; i ++) {
        int sgn = dcmp(det(p[i], p[i + 1]));
        if (sgn) {
            ret += sgn * calc(p[i], p[i + 1]);
        }
    }
    return ret;
}
void solve() {
    scanf("%d", &n);
        for (int i = 0; i < n; i ++)
                p[i].input();
        p[n] = p[0];
        printf("%.2lf\n", fabs(area()) + eps);

}
int main() {
        while(scanf("%lf", &r) != EOF) {
            init();
                solve();
        }
        return 0;
}
```

4.3 segscan

```
#include <bits/stdc++.h>
using namespace std;
#define next nxt
#define rep(i, s, t) for (int i = s; i <= t; i++)
#define dwn(i, s, t) for (int i = s; i >= t; i--)
#define edg(i, x) for (int i = head[x]; ~ i; i = next[i])
#define ctn(i, x) for (i = x.begin(); i != x.end(); i++)
#define clr(x) memset ((x), 0, sizeof (x))
typedef long long LL;
int read()
{
    int 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;
}
void print(LL x) {
    static int a[24];int n = 0;
    while(x > 0) {
        a[n++] = x % 10;
        x /= 10;
    }
    if (n == 0) a[n++] = 0;
    while(n--) putchar('0' + a[n]);
    putchar('\n');
}
void from(const char *s) {
        freopen(s, "r", stdin);
}
const double eps = 1e-8;
const int INF = 0x3f3f3f3f;
double nowx;
int n, C, m;
const int N = 3001000;
int ans[N];
double cmp(double x) {
        return x < -eps ? -1 : x > eps;
}
struct Q {
        double x1, x2, y1, y2;
        bool vert, high;
        Q() {}
        Q(double x1, double y1, double x2, double y2) :
                x1(x1), y1(y1), x2(x2), y2(y2) {
                if (cmp(y1 - y2) == 0) vert = true;
                else vert = false;
                if (y2 > y1) high = true;
                else high = false;
        }
        double getY() {
                //if (cmp(x1 - x2) == 0) return y1;
                return y1 + (y2 - y1) / (x2 - x1) * (nowx - x1);
                //return y1;
        }
}a[N];

struct E {
        double x, y;
        int id;
        bool pos;
        E() {}
        E(double x, double y, int id, bool pos) :
                x(x), y(y), id(id), pos(pos) {}
        bool operator<(const E& a) const {
                if (cmp(x - a.x) == 0)
                        if (pos == a.pos)
                                return y < a.y;
                        else return pos < a.pos;
                else return x < a.x;
        }
}event[N];

struct P {
        int id;
        P() {}
        P(int id) : id(id) {}
        bool operator<(const P& p) const{
                double y1 = a[id].getY();
                double y2 = a[p.id].getY();

                return y1 < y2;
```

```
        }
        bool operator==(const P& p) const {
                return id == p.id;
        }
};
int fa[N], top[N];
int find(int x) {
        return fa[x] == x ? x : fa[x] = find(fa[x]);
}
set<P> st;
set<P>::iterator it, cit, sit[N];

double rec[N];
void init() {
        clr(top);
        clr(ans);
        clr(rec);
}
void solve() {

        int cnt = 0;
        rep(i, 1, n) {
                double x1, x2, y1, y2;
                scanf("%lf%lf%lf%lf", &x1, &y1, &x2, &y2);
                if (x1 > x2) swap(x1, x2), swap(y1, y2);
                a[i] = Q(x1, y1, x2, y2);
                //printf("%d %d\n", a[i].vert, a[i].high);
                event[++cnt] = E(x1, y1, i, 0);
                event[++cnt] = E(x2, y2, i, 1);
                fa[i] = i;
        }
        sort(event + 1, event + 1 + cnt);
        rep(i, 1, cnt) {
                nowx = event[i].x;
                int id = event[i].id;
                if (a[id].vert) {
                        if (event[i].pos == 0) sit[id] = st.insert(P(
                            id)).first;
                        else st.erase(sit[id]);
                        continue;
                }
                if (event[i].pos == 0) {
                        it = st.insert(P(id)).first;
                        sit[id] = it;
                        if (a[id].high == 0) {
                                if (++it == st.end()) {
                                        top[id] = INF;
                                } else {
                                        cit = it;
                                        if (a[cit->id].vert == false)
                                                fa[id] = find(cit->id)
                                                    ;
                                        else
                                                top[id] = cit->id;
                                }
                        }
                } else {
                        it = sit[id];
                        if (it == st.end()) printf("%d\n", id);
                        if (a[id].high == 1) {
                                if (++it == st.end()) {
                                        top[id] = INF;
                                } else {
                                        cit = it;
                                        if (a[cit->id].vert == false)
                                                fa[id] = find(cit->id)
                                                    ;
                                        else
                                                top[id] = cit->id;

                                }
                        }
                        st.erase(sit[id]);
                }
        }
        rep(i, 1, n) {
                fa[i] = find(fa[i]);
                //printf("%d\n", fa[i]);
        }
        rep(i, 1, m) {
                scanf("%lf", &rec[i]);
                a[n + i] = Q(rec[i], 0, rec[i], 0);
                event[++cnt] = E(rec[i], 0, i + n, 1);
        }
        sort(event + 1, event + 1 + cnt);
        st.clear();
        rep(i, 1, cnt) {
                nowx = event[i].x;
                int id = event[i].id;
                if (id > n) {
                        if (st.size()) {
                                it = st.lower_bound(P(id));
                                ans[id - n] = it->id;
                        } else {
                                ans[id - n] = -1;
                        }
                        continue;
                }
                if (a[id].vert) {
                        if (event[i].pos == 0) sit[id] = st.insert(P(
                            id)).first;
                        else st.erase(sit[id]);
                        continue;
                }
                if (event[i].pos == 0) {
                        sit[id] = st.insert(P(id)).first;
                } else {
                        st.erase(sit[id]);
                }
        }

        rep(i, 1, m) {
                if (ans[i] == -1) {
                        printf("%.0lf\n", rec[i]);
                } else {
                        ans[i] = fa[ans[i]];
                        if (top[ans[i]] == INF) {
                                double tmp;
                                if (a[ans[i]].high == 0)
                                        printf("%.0lf\n", a[ans[i]].x1
                                            );
                                else
```

```
                                printf("%.0lf\n", a[ans[i]].x2
                                    );
                        } else {
                                if (a[ans[i]].vert == true) {
                                        printf("%.0lf %.0lf\n", rec[i
                                            ], a[ans[i]].y1);
                                        continue;
                                }
                                if (a[ans[i]].high == 0)
                                        printf("%.0lf %.0lf\n", a[ans[
                                            i]].x1, a[top[ans[i]]].y1)
                                            ;
                                else
                                        printf("%.0lf %.0lf\n", a[ans[
                                            i]].x2, a[top[ans[i]]].y2)
                                            ;
                        }
                }
        }


}
int main() {
        while(scanf("%d%d", &n, &m) != EOF) {
                init();
                solve();
        }
        return 0;
}
```

5 GraphTheory

5.1 costflow

```
int S, T;
struct Flow {
        typedef LL int;
        const int INF = 0x3f3f3f3f;
        int key[M], next[M], head[N], cnt, f[M];
        int pe[N], pv[N], S, T;
        int  q[N];
        LL dis[N], cost[M];
        bool vis[N];
        void init() {
                cnt = 0;
                memset (head, -1, sizeof head);
        }
        void add(int x, int y, LL 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() {
                rep(i, S, T) dis[i] = INF;
                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;
        }
        LL 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;
        }
        LL work() {
                LL ans = 0;
                while(spfa())
                        ans += z();
                return ans;
        }
}flow;
```

5.2 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;
```

5.3 Hungary

```
#include <cstdio>
#include <cstring>
const int N = 1010;
bool vis[N], map[N][N];
int n, m, t, x, lnk[N], left[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;
}
int k;
int main() {
        scanf("%d%d%d", &n, &m, &k);
        for (int i = 1, a, b; i <= k; i ++)
                scanf("%d%d", &a, &b), map[a][b] = true;
        int ans = hungary();
        printf("%d\n", n + m - hungary());
        return 0;
}
```

5.4 km

```
#include <cstdio>
#include <iostream>
#include <algorithm>
#include <cmath>
#include <cstring>
using namespace std;
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;
}
```

5.5 kruscal

```
const int N = 201000, M = 2001000;
int n, m, T;
LL ans;
struct Q {
        int x, y, w;
        void scan() {
                scanf("%d%d%d", &x, &y, &w);
        }
        bool operator < (const Q& a) const {
                return w < a.w;
        }
}e[M];
void init() {
        ans = 0;
}
int find(int x) {
        return fa[x] == x ? x : fa[x] = find(fa[x]);
}
void solve() {
        scanf("%d%d", &n, &m);
        rep(i, 1, m) e[i].scan();
        rep(i, 1, n) fa[i] = i;
        sort(e + 1, e + 1 + m);
        rep(i, 1, m)
                if (find(e[i].x) != find(e[i].y)) {
                        fa[fa[e[i].x]] = fa[e[i].y];
                        ans += e[i].w;
                }
        cout << ans;
}
int main() {
        scanf("%d", &T);
        rep(i, 1, T) {
                init();
                solve();
        }
        return 0;
}
```

5.6 scc

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i, s, t) for (int i = s; i <= t; i++)
#define dwn(i, s, t) for (int i = s; i >= t; i--)
#define edg(i, x) for (int i = head[x]; ~ i; i = next[i])
#define ctn(i, x) for (i = x.begin(); i != x.end(); i++)
#define clr(x) memset ((x), 0, sizeof (x))
#define SZ(x) (int)x.size()
#define next nxt
typedef long long LL;
int read()
{
    int 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;
}
void print(LL x) {
    static int a[24];int n = 0;
    while(x > 0) {
        a[n++] = x % 10;
        x /= 10;
    }
    if (n == 0) a[n++] = 0;
    while(n--) putchar('0' + a[n]);
    putchar('\n');
}
void from(const char *s) {
    freopen(s, "r", stdin);
}
void to(const char *s) {
    freopen(s, "w", stdout);
}
const int N = 101000;
int key[N], next[N], head[N], cnt;
int dfn[N], stk[N], top, low[N], tmct, scc_c, scc[N];
bool instk[N];
bool in[N], out[N];
int n, m, no_in, no_out;
void init() {
        clr(dfn);
        clr(low);
        clr(scc);
        clr(instk);
        clr(stk);
        scc_c = top = tmct = cnt = 0;
        memset (head, -1, sizeof head);
        no_in = no_out = 0;
        clr(in);
        clr(out);
}
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);
        }
}
void add(int x, int y) {
        key[cnt] = y;
        next[cnt] = head[x];
        head[x] = cnt++;
}

int solve() {
        n = read(), m = read();
        rep(i, 1, m) {
                int x = read(), y = read();
                add(x, y);
        }
        rep(i, 1, n)
                if (dfn[i] == 0)
                        DFS(i);
        rep(u, 1, n) edg(j, u) {
                int v = key[j];
                if (scc[u] == scc[v]) continue;
                in[scc[v]] = true;
                out[scc[u]] = true;
        }
        rep(i, 1, scc_c) if (in[i] == false) no_in++;
        rep(i, 1, scc_c) if (out[i] == false) no_out++;
        if (scc_c == 1) return 0;
        return max(no_in, no_out);
}

int main() {
        int T = read();
        rep(i, 1, T) {
                init();
                printf("%d\n", solve());
        }
        return 0;
}
```

6 NumberTheory

6.1 binominal

```
const int N = 1001000, P = 1e9 + 7;
LL inv[N], fac[N], faci[N];
LL C(int n, int m) {
        if (n < 0 || m < 0 || m > n) return 0;
        return fac[n] * faci[n - m] % P * faci[m] % P;
}
void pre() {
        const int P = 1e9 + 7, N = 1000000;
        inv[1] = 1;
        rep(i, 2, N) inv[i] = (P - P / i) * inv[P % i] % P;
        fac[0] = 1;
        rep(i, 1, N) fac[i] = fac[i - 1] * i % P;
        faci[0] = 1;
        rep(i, 1, N) faci[i] = faci[i - 1] * inv[i] % P;
}
```

6.2 broot

```
#include <cstdio>
#include <algorithm>
#include <cstring>
#include <iostream>
#include <map>
#include <cmath>
using namespace std;
typedef long long LL;
const int N = 1001000;
LL pw(LL a, LL k) {
        LL z(1);
        for (; k; k >>= 1) {
                if (k & 1) z = z * a;
                a = a * a;
        }
        return z;
}
int gcd(int a, int b) {
        return b ? gcd(b, a % b) : a;
}
LL pw(LL x, LL k, LL p) {
        LL z = 1;
        for (; k; k >>= 1) {
                if (k & 1) z = z * x % p;
                x = x * x % p;
        }
        return z;
}
bool vis[N];
int pr[N];
void getpr() {
        int N = 1000000;
        for (int i = 1; i <= N; i ++)
                vis[i] = false;
        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;
                }
        }
}
struct ROOT {
        LL n;
        int a_c;
        int a[N];
        void divide(LL 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(LL _n) {
                a_c = 0;
                n = _n;
                divide(n - 1);
                int i;
                for (i = 2; ; i ++) {
                        if (ck(i))
                                return i;
                }
        }
}root;

const int HN = 30000, M = 1000000, HEAD = 29997;
struct HASH {
        int cnt, head[HN], next[M], len[M];
        LL key[M];
        HASH() {
                clear();
        }
        inline void clear() {
                memset (head, -1, sizeof head);
                cnt = 0;
        }
        inline void ADD(int x, LL y, int w) {
                key[cnt] = y;
                next[cnt] = head[x];
                len[cnt] = w;
                head[x] = cnt ++;
        }
        inline int GETHEAD(LL idx) {
                return idx % HEAD;
        }
        inline void add(LL idx, int val) {
                int h = GETHEAD(idx);
                ADD(h, idx, val);
        }
        bool find(LL idx) {
                int h = GETHEAD(idx);
                for (int i = head[h]; ~ i; i = next[i])
                        if (key[i] == idx)
                                return true;
                return false;
        }
        int get(LL idx) {
                int h = GETHEAD(idx);
                for (int i = head[h]; ~ i; i = next[i])
                        if (key[i] == idx)
                                return len[i];
        }
}_hash;
int BSGS(LL a, LL b, LL p) {
        a %= p, b %= p;
        if (b == 1) return 0;
        int cnt = 0;
        LL t = 1;
        for (LL 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;
}
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 A, B, p;
LL a[N];
int a_c;
int main() {
        freopen("a.in", "r", stdin);
        getpr();
        int Case = 0;
        while(scanf("%lld%lld%lld", &A, &p, &B) != EOF) {
                a_c = 0;
                int g = root.get(B);
                LL I = BSGS(g, B, p);
                LL x, y, d;
                exgcd(A, p - 1, d, x, y);
```

```
                ++Case;printf("case%d:\n", Case);
                if (I % d) {
                        printf("-1\n");
                        continue;
                } else {
                        x = (I / d) * x % (p - 1);
                        for (int i = 0; i < d; i ++) {
                                LL tx = (x + i * (p - 1) / d) % (p -
                                    1);
                                a[++a_c] = pw(g, tx, p);
                        }
                }
                sort(a + 1, a + 1 + a_c);
                for (int i = 1; i <= a_c; i ++)
                        printf("%lld\n", a[i]);
        }
        return 0;
}
```

6.3 god

```
#include <cstdio>
#include <cstring>
#include <iostream>
#include <map>
#include <cmath>
using namespace std;
typedef long long LL;
const int N = 101000;
LL pw(LL a, int k) {
        LL z(1);
        for (; k; k >>= 1) {
                if (k & 1) z = z * a;
                a = a * a;
        }
        return z;
}
int gcd(int a, int b) {
        return b ? gcd(b, a % b) : a;
}
LL pw(LL x, int k, LL p) {
        LL z = 1;
        for (; k; k >>= 1) {
                if (k & 1) z = z * x % p;
                x = x * x % p;
        }
        return z;
}
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;
                }
        }
}
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;

const int HN = 40000, M = 100000, HEAD = 39997;
struct HASH {
        int cnt, head[HN], 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];
        }
}_hash;
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;
}
int a[N], c[N], a_c;
void Divide(int n) {
        a_c = 0;
        memset (c, 0, sizeof c);
        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], c[a_c]++;
        }
        if (n != 1) {
                a[++ a_c] = n;
                c[a_c] = 1;
        }
}
int A, B, p;
int main() {
        freopen("a.in", "r", stdin);
        getpr();
        int T;
        scanf("%d", &T);
        while(T --) {
                scanf("%d%d%d", &A, &B, &p);
                a_c = 0;
                p = 2 * p + 1;
                Divide(p);
                bool flag = false;
                LL ans = 1;
                for (int i = 1; i <= a_c; i ++) {
                        int t = pw(a[i], c[i]);
                        int b = B % t;
                        if (!b) {
                                ans = ans * pw(a[i], c[i] - (c[i] - 1)
                                    / A - 1);
                        }
                        else {
                                int g = gcd(b, t);
                                b /= g, t /= g;
                                int cnt = 0;
                                while(g % a[i] == 0) g /= a[i], cnt
                                    ++;
                                if (cnt % A) {
                                        ans = 0;
                                        break;
                                }
                                int rt = root.get(a[i]);
                                int I = BSGS(rt, b, t);
                                int phit = t - pw(a[i], c[i] - 1);
                                int D = gcd(A, phit);
                                if (I % D) {
                                        ans = 0;
                                        break;
                                }
                                if (cnt)
                                        ans = ans * D * pw(a[i], cnt -
                                            cnt / A);
                                else
                                        ans = ans * D;
                        }
                }
                printf("%d\n", ans);
        }
        return 0;
}
```

6.4 log

```
#include <cstdio>
#include <cstring>
#include <iostream>
#include <map>
#include <cmath>
using namespace std;
typedef long long LL;
int gcd(int a, int b) {
        return b ? gcd(b, a % b) : a;
}
int pw(LL x, int k, LL p) {
        LL z = 1;
        for (; k; k >>= 1) {
                if (k & 1) z = z * x % p;
                x = x * x % p;
        }
        return z;
}
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];
        }
};
struct HASH hash;
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;
}
int main() {
        freopen("a.in", "r", stdin);
        int a, b, p;
        while(scanf("%d%d%d", &p, &a, &b) != EOF) {
                int tmp = BSGS(a, b, p);
                if (tmp == -1) printf("no solution\n");
                else printf("%d\n", tmp);
        }
        return 0;
}
```

6.5 primitive

```
#include <cstdio>
#include <cstring>
typedef long long LL;
const int N = 4001000;
int P, n, a_c;
int a[N];
LL pw(LL a, int k) {
        LL z(1);
        for (; k; k >>= 1) {
                if (k & 1) z = z * a % P;
                a = a * a % P;
        }
        return z;
}
bool vis[N];
int pr[N];
void getpr() {
        const int N = 4000000;
        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;
                }
        }
}
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]) == 1)
                        return false;
        return true;
}
```

```
int main() {
        scanf("%d", &n);
        getpr();
        divide(n - 1);
        P = n;
        int i;
        for (i = 2; ; i ++) {
                if (ck(i))
                        break;
        }
        printf("%d\n", i);
        return 0;
}
```

6.6 quadratic

```
#include <cstdio>
#include <iostream>
using namespace std;
#include <ctime>
#include <cstdlib>
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;
}
```

6.7 sieve

```
bool vis[N];
int pr[N];
void getpr() {
        int N = 1000000;
        for (int i = 1; i <= N; i ++)
                vis[i] = false;
        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;
                }
        }
}
```

7 NumericalMethod

7.1 SPNM

```
#include <cstdio>
#include <iostream>
using namespace std;
class Poly {
```

```
        typedef vector<double> vd;
        vd v;
        N() {}
        //a[0] * x ^ 0 + a[1] * x ^ 1 + ....
        N(int n, double a[]) {
                v.resize(n);
                for (int i = 0; i < n; i ++) v[i] = a[i];
        }
        double valAt(double x) {
                double z = 1;
                for (int i = v.size() - 1; i >= 0; i --)
                        z = z * x + a[i];
                return z;
        }
        double DerivAt(double x) {

                for (int i = v.size() - 1; i >= 0; i --) {
                        z = z * x + a[i];
                        d = d * x +
}

int main() {
        return 0;
}
```

8 Others

8.1 BigInteger

```
#include <cstdio>
#include <cstring>
#include <iostream>
#include <algorithm>
using namespace std;
typedef long long LL;
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 get() {
                LL x;
                cin >> x;
                while(x) {
                        s[++len] = x % base;
                        x /= base;
                }
        }
        void print() {
                printf("%d", s[len]);
                for (int i = len - 1; i >= 1; i --)
                        printf("%04d", s[i]);
                printf("\n");
        }
};
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 = max(a.len, b.len);
        for(int i = 1; i <= c.len; i++) {
                c[i] += a[i] - b[i];
                if(c[i] < 0) {
                        c[i + 1]--;
                        c[i] += base;
                }
        }
        while(c[c.len] == 0 && 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++;
        while(c[c.len] == 0 && 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;
}
bool operator== (const Num& a, const Num& b) {
        if (a.len != b.len)
                return false;
        for (int i = a.len; i >= 0; i --)
                if (a[i] != b[i])
                        return false;
        return true;
}
struct Q {
        Num x, y;
```

```
}a, b, A, B;
Q operator* (const Q& a, const Q& b) {
        Q c;
        c.x = a.x * b.x - a.y * b.y;
        c.y = a.x * b.y + a.y * b.x;
        return c;
}
int n, m;
int main() {
        freopen("a.in", "r", stdin);
        A.x.get();A.y.get();
        cin >> n;
        B.x.get();B.y.get();
        cin >> m;
        //(a + bi)(c + di) = (ac - bd) + (ad + bc)
        for (int i = 1; i <= m; i ++)
                A = A * a;
        for (int i = 1; i <= n; i ++)
                B = B * b;
        if (A.x == B.x && A.y == B.y)
                printf("%d\n", __gcd(n, m));
        else
                printf("0\n");
        return 0;
}
```

8.2 bit-op

```
bool test(int s, int i) {
        return (s >> i) & 1;
}
void set(int& s, int i) {
        s |= (1 << i);
}

void flip(int &s, int i) {
        s ^= (1 << i);
}
void clear(int& s, int i) {
        if (test(s, i))
                flip(s, i);
}
int count(int s) {
        int z = 0;
        for (int i = 0; i < 8 * sizeof(s); i ++)
                if (test(s, i))
                        z ++;
        return z;
}
```

8.3 CountingColors

```
#include <cstdio>
#include <iostream>
#include <algorithm>
#define rep(i, s, t) for (int i = s; i <= t; i ++)
using namespace std;
const int N = 1001000;
int n, m, a[N], s[N], ans[N], last[N];
struct Q {
        int l, r, id;
        bool operator<(const Q& a) const {
                return r < a.r;
        }
}b[N];
void update(int x, int v) {
        for (; x <= n; x += x & -x) s[x] += v;
}
int query(int x) {
        int z = 0;
        for (; x; x -= x & -x) z += s[x];
        return z;
}
int main() {
        scanf("%d", &n);
        rep(i, 1, n) scanf("%d", &a[i]);
        scanf("%d", &m);
        rep(i, 1, m) scanf("%d%d", &b[i].l, &b[i].r), b[i].id = i;
        sort(b + 1, b + 1 + m);
        int k = 1;
        rep(i, 1, n) {
                update(i, 1);
                if (last[a[i]]) update(last[a[i]], -1);
                last[a[i]] = i;
                while(b[k].r == i) {
                        ans[b[k].id] = query(b[k].r) - query(b[k].l -
                            1);
                        k++;
                }
        }
        rep(i, 1, m) printf("%d\n", ans[i]);
        return 0;
}
```

8.4 CountingSort

```
for (int i = 1; i <= n; i ++)
        a[i] = (1LL * a[i - 1] * A + B) % p, s[a[i]]++;
for (int i = 1; i <= p; i ++)
        s[i] += s[i - 1];
for (int i = n; i >= 1; i --)
        id[s[a[i]]--] = i;
```

8.5 dicretize

```
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 ++)
    a[i] = lower_bound (b + 1, b + 1 + b_c, a[i]) - b;
```

8.6 fraction&powersum

```
#include <cstdio>
#include <cstdlib>
#include <iostream>
#include <algorithm>
using namespace std;
const int N = 35;
typedef long long LL;
struct Q {
        LL a, b;
        Q () { a = 0; b = 1; }
        Q (LL x) { a = x; b = 1; }
        Q (LL x, LL y) {
                a = x, b = y;
                uni();
        }
        void uni() {
                LL t = __gcd(a, b);
                a /= t;
                b /= t;
                if (b < 0) {
                        a = -a;
                        b = -b;
                }
        }
        Q operator + (const Q& x) const {
                Q c;
                c.a = a * x.b + x.a * b;
                c.b = b * x.b;
                c.uni();
                return c;
        }
        Q operator - (const Q& x) const {
                Q c;
                c.a = a * x.b - x.a * b;
                c.b = b * x.b;
                c.uni();
                return c;
        }
        Q operator * (const Q& x) const {
                Q c;
                c.a = a * x.a;
                c.b = b * x.b;
                c.uni();
                return c;
        }
        Q operator / (const Q& x) const {
                Q c;
                c.a = a * x.b;
                c.b = b * x.a;
                c.uni();
                return c;
        }
        bool operator==(int x) {
                uni();
                return a == 0;
        }
        bool operator!=(int x) {
                uni();
                return a != 0;
        }
        void print() {
                printf("X = %lld/%lld\n", a, b);
        }
}C[N][N], B[N];
int main() {
        for (int i = 1; i <= 31; i ++) {
                C[i][0] = C[i][i] = Q(1, 1);
                for (int j = 1; j < i; j ++)
                        C[i][j] = C[i - 1][j - 1] + C[i - 1][j];//, C[
                            i][j].print();
        }
        B[0] = Q(1, 1);
        for (int i = 1; i <= 30; i ++) {
                B[i] = Q(0, 1);
                for (int j = 0; j < i; j ++)
                        B[i] = B[i] + C[i + 1][j] * B[j];
                B[i] = B[i] * Q(-1, i + 1);
                //printf("%d ", i);B[i].print();

        }
        int m;
        scanf("%d", &m);
        for (int k = 0; k <= m; k ++) {
                Q t(1, m + 1);
                t = t * C[m + 1][k];
                t = t * B[k];
                if (k == 1) t = t + Q(1, 1);
                t.print();
                //C[m + 1][k] * B[k].print();
        }
        Q t(0, 1);
        t.print();
        return 0;
}
```

8.7 liner_bound+fast_read

```
#include <cstdio>
#include <iostream>
#include <algorithm>
using namespace std;
typedef long long LL;
const int N = 2001000;
int a[N], l[N], r[N], c[N];
int read()
{
    int 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;
}
int main() {
        freopen("a.in", "r", stdin);
        int n;
        scanf("%d", &n);
        for (int i = 1; i <= n; i ++)
                a[i] = read();
        rotate(a + 1, max_element(a + 1, a + 1+n), a + 1 + n);
        for(int i = n; i >= 1; --i) {
            r[i] = i + 1;
            while (r[i] <= n && a[i] > a[r[i]]) r[i] = r[r[i]];
```

```
            if (r[i] <= n && a[i] == a[r[i]]) {
                    c[i] = c[r[i]] + 1;
                    r[i] = r[r[i]];
                }
        }
        for(int i = 1; i <= n; i ++) {
            l[i] = i - 1;
            while (l[i] >= 1 && a[i] > a[l[i]]) l[i] = l[l[i]];
            if (l[i] >= 1 && a[i] == a[l[i]]) {
                    l[i] = l[l[i]];
                }
        }
        LL ans = 0;
        for (int i = 1; i <= n; i ++) {
                ans += c[i];
                if (a[i] == a[1]) continue;
                ans += 2;
                if (l[i] == 1 && r[i] == n + 1) ans--;
        }
        printf("%I64d\n", ans);
        return 0;
}
```

8.8 Matrix&QuickPower

```
int mod(int x) { return x % P; }
LL mod(LL x) { return x % P; }
struct Q {
        int s[N][N];
        Q () {
                memset (s, 0, sizeof s);
        }
        Q operator * (const Q& a) {
                Q c;
                for (int i = 0; i < N; i ++)
                        for (int j = 0; j < N; j ++)
                                for (int k = 0; k < N; k ++)
                                        c.s[i][j] = mod(c.s[i][j] +
                                            mod(s[i][k] * a.s[k][j]));
                return c;
        }
};
Q qk(Q& A, LL k) {
        Q z;z.s[0][0] = z.s[1][1] = 1;
        for (; k; k >>= 1) {
                if (k & 1)
                        z = z * A;
                A = A * A;
        }
        return z;
}
```

8.9 Mode

```
struct Mode {
        int mx, c[N], cnt[N];
        void init() {
                memset(c, 0, sizeof c);
                memset(cnt, 0, sizeof cnt);
                mx = 0;
                cnt[0] = 0x3f3f3f3f;
        }
        void inc(int x) {
                cnt[c[x]]--;
                c[x]++;
                cnt[c[x]]++;
                mx = max(mx, c[x]);
        }
        void dec(int x) {
                cnt[c[x]]--;
                c[x]--;
                cnt[c[x]]++;
                while(cnt[mx] == 0) mx--;
        }
        int get() {
                return mx;
        }
}mode;
```

9 String

9.1 HASH

```
const int N = 400000, M = 5000000, HEAD = 399997;
const int P = 1e9 + 7;
struct HASH {
        int cnt, head[N], next[M], len[M];
        LL key[M];
        HASH() {
                clear();
        }
        inline void clear() {
                memset (head, -1, sizeof head);
                cnt = 0;
        }
        inline void ADD(int x, LL y, int w) {
                key[cnt] = y;
                next[cnt] = head[x];
                len[cnt] = w;
                head[x] = cnt ++;
        }
        inline int GETHEAD(LL idx) {
                return idx % HEAD;
        }
        inline void add(LL idx, int w) {
                int h = GETHEAD(idx);
                ADD(h, idx, w);
        }
        bool find(LL idx, int w) {
                int h = GETHEAD(idx);
                for (int i = head[h]; ~ i; i = next[i])
                        if (key[i] == idx && len[i] == w)
                                return true;
                return false;
        }
}mp;
```

9.2 PAM-with-sfail

```
#include <cstdio>
#include <iostream>
#include <cstdlib>
#include <cstring>
using namespace std;
const int N = 501000, M = 30;
int ans[N][2];
const int MAX = 0x3f3f3f3f;
struct PAM {
        int next[N][M], fail[N], cnt[N], len[N], S[N], num[N], diff[N
            ], sfail[N], dp[N][2];
        int last, n, p;
        int newnode(int l) {
                for (int i = 0; i < M; i ++) next[p][i] = 0;
                cnt[p] = 0;
                num[p] = 0;
                len[p] = l;
                return p++;
        }
        void init() {
                p = 0;
                newnode(0);
                newnode(-1);
                last = 0;
                n = 0;
                S[n] = -1;
                fail[0] = 1;
                ans[0][0] = 0;
                ans[0][1] = MAX;
        }
        int get_fail(int x) {
                while(S[n - len[x] - 1] != S[n]) x = fail[x];
                return x;
        }
        int getmin(int x, int i) {
                dp[x][i] = ans[n - len[sfail[x]] - diff[x]][i];
                if (diff[x] == diff[fail[x]]) {
                        dp[x][i] = min(dp[x][i], dp[fail[x]][i]);
                }
                return dp[x][i] + 1;
        }
        void add(int c) {
                c -= 'a';
                S[++n] = c;
                int cur = get_fail(last);
                if (!next[cur][c]) {
                        int now = newnode(len[cur] + 2);
                        fail[now] = next[get_fail(fail[cur])][c];
                        next[cur][c] = now;
                        num[now] = num[fail[now]] + 1;
                }
                last = next[cur][c];
                diff[last] = len[last] - len[fail[last]];
                sfail[last] = diff[last] != diff[fail[last]] ? fail[
                    last] : sfail[fail[last]];
                cnt[last]++;
                //==========================================
                ans[n][1] = ans[n][0] = MAX;
                for (int x = last; x; x = sfail[x]) {
                        ans[n][0] = min(ans[n][0], getmin(x, 1));
                        ans[n][1] = min(ans[n][1], getmin(x, 0));
                }
        }
        void count() {
                for (int i = p - 1; i >= 0; i --)
                        cnt[fail[i]] += cnt[i];
                for (int i = 1; i <= n; i ++)
                        printf("%d %d\n",
                                ans[i][1] == MAX ? -1 : ans[i][1],
                                ans[i][0] == MAX ? -2 : ans[i][0]);
        }
}pa, pb;
char s[N];
int main() {
        freopen("a.in", "r", stdin);
        int T;

        scanf("%s", s);
        int len = strlen(s);
        pa.init();
        for (int i = 0; i < len; i ++)
                pa.add(s[i]);
        pa.count();

        return 0;
}
```

9.3 PAM

```
#include <cstdio>
#include <iostream>
#include <cstdlib>
#include <cstring>
using namespace std;
const int N = 501000, M = 30;
/*
1.len[i]          ł i
2.next[i][c]
              ł i                                  c

3.fail[i]
             i            ł                 i                                  A C

4.cnt[i]
             i                                                          c o u n t
    ()
5.num[i]
               i                               ł
6. l a s t
7.S[i]       i                 ł     S [0] = -1
                             S
8. p
9. n
*/
struct PAM {
        int next[N][M], fail[N], cnt[N], len[N], S[N], num[N];
        int last, n, p;
```

```
        int newnode(int l) {
                for (int i = 0; i < M; i ++) next[p][i] = 0;
                cnt[p] = 0;
                num[p] = 0;
                len[p] = l;
                return p++;
        }
        void init() {
                p = 0;
                newnode(0);
                newnode(-1);
                last = 0;
                n = 0;
                S[n] = -1;
                fail[0] = 1;
        }
        int get_fail(int x) {
                while(S[n - len[x] - 1] != S[n]) x = fail[x];
                return x;
        }
        void add(int c) {
                c -= 'a';
                S[++n] = c;
                int cur = get_fail(last);
                if (!next[cur][c]) {
                        int now = newnode(len[cur] + 2);
                        fail[now] = next[get_fail(fail[cur])][c];
                        next[cur][c] = now;
                        num[now] = num[fail[now]] + 1;
                }
                last = next[cur][c];
                cnt[last]++;
        }
        void count() {
                for (int i = p - 1; i >= 0; i --)
                        cnt[fail[i]] += cnt[i];
        }

}pa, pb;
int main() {
        int T;
        scanf("%d", &T);
                scanf("%s", sa);
                int lb = strlen(sb);
                pa.init();
                for (int i = 0; i < la; i ++)
                        pa.add(sa[i]);
                pa.count();
        }
        return 0;
}
```

9.4 SAM

```
#include <cstdio>
#include <cstring>
#include <algorithm>
#include <iostream>
#include <climits>
#include <numeric>
#include <vector>
using namespace std;

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;
}
```

10 TreeTheory

10.1 Divide&Conquer

```
#include <bits/stdc++.h>
using namespace std;
#define rep(i, s, t) for (int i = s; i <= t; i ++)
#define dwn(i, s, t) for (int i = s; i >= t; i --)
#define edg(i, x) for (int i = head[x]; ~ i; i = next[i])
#define ctn(i, x) for (i = x.begin(); i != x.end(); i++)
#define clr(x) memset ((x), 0, sizeof (x))
typedef long long LL;
typedef pair<int, int> pi;
int read()
{
    int 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;
}
void print(LL x) {
        if (x / 10) print(x / 10);
        putchar(x % 10 + '0');
}
void from(const char *s) {
        freopen(s, "r", stdin);
}
const int N = 501000, INF = 0x3f3f3f3f;
int key[N], next[N], len[N], head[N], cnt, f[N], sz[N], flag[N], root[
    N], tot, K, n, t, ans;
vector<pi> son[N];
void add(int x, int y, int w) {
        key[cnt] = y;
        next[cnt] = head[x];
        len[cnt] = w;
        head[x] = cnt++;
}
void find(int u, int fa) {
        f[u] = 0, sz[u] = 1;
        edg(i, u) {
                int v = key[i];
                if (flag[v] || v == fa) continue;
                find(v, u);
                sz[u] += sz[v];
                f[u] = max(f[u], sz[v]);
        }
        f[u] = max(f[u], tot - sz[u]);
        if (f[u] < f[root[t]])
                root[t] = u;
}
bool cmp(pi x, pi y) {
        return sz[x.first] < sz[y.first];
}
void solve(int u) {
        flag[u] = t;
        if (tot == 1) return ;
        edg(i, u) {
                int v = key[i];
                if (flag[v]) continue;
                if (sz[v] > sz[u]) sz[v] = tot - sz[u];
                son[u].push_back(pi(v, len[i]));
        }
        sort(son[u].begin(), son[u].end(), cmp);
        edg(i, u) {
                int v = key[i];
                if (flag[v]) continue;
                ++t;
                tot = sz[v];
                find(v, 0);
                solve(root[t]);
        }
}
map<int, int> res, path;
map<int, int>::iterator it;
int dep1, dep2, clk;
void update(map<int, int>& mp, int key, int val) {
        if (mp.find(key) != mp.end()) mp[key] = min(mp[key], val);
        else mp[key] = val;
}
void DFS(int u, int fa, int dep, int dis) {
        update(path, dis, dep);
        edg(i, u) {
                int v = key[i];
                if (flag[v] <= clk || v == fa) continue;
                DFS(v, u, dep + 1, dis + len[i]);
        }
}
bool conquer() {
        for (clk = 1; clk <= n; clk++) {
                int u = root[clk];
                res[0] = 0;
                rep(i, 0, (int)son[u].size() - 1) {
                        int v = son[u][i].first;
                        int len = son[u][i].second;
                        DFS(v, u, 1, len);
                        ctn(it, path)
                                if (res.find(K - it->first) != res.end
                                    ())
                                        ans = min(ans, it->second +
                                            res[K - it->first]);
                        ctn(it, path)
                                update(res, it->first, it->second);
                        path.clear();
                }
                res.clear();
        }
        return 0;
}
int main() {
        memset (head, -1, sizeof head);
        n = read(), K = read();
        rep(i, 1, n - 1) {
                int a, b, c;
                a = read(), b = read(), c = read();
                ++a,++b;
```

```
                add(a, b, c);
                add(b, a, c);
        }
        ++t;
        tot = n;f[0] = INF;
        find(1, 0);
        solve(root[1]);
        ans = 0x3f3f3f3f;
        conquer();
        printf("%d\n", ans != 0x3f3f3f3f ? ans : -1);
        return 0;
}
```

10.2 LCA

```
const int N = 501000, M = 601000;
int key[M], nxt[M], head[N], cnt;
void add(int x, int y) {
        key[cnt] = y;
        nxt[cnt] = head[x];
        head[x] = cnt++;
}
struct LCA {
        int f[N][25], d[N], q[N];
        void BFS(int S) {
                int h = 1, t = 2, u;
                q[1] = S;
                while(h < t) {
                        u = q[h++];
                        for (int i = head[u]; ~ i; i = nxt[i]) {
                                int v = key[i];
                                if (d[v] != 0) continue;
                                f[v][0] = u, d[v] = d[u] + 1;
                                q[t++] = v;
                        }
                }
        }
        void init(int n) {
                d[1] = 1;
                BFS(1);
                for (int j = 1; j <= 20; j ++)
                        for (int i = 1; i <= n; i ++) {
                                f[i][j] = f[f[i][j - 1]][j - 1];
                        }
        }
        int get(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;                       //return
                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];                 //return
        }
        int dis(int x, int y) {
                int t = get(x, y);
                return d[x] - d[t];
        }
}lca;
```