Platelet

Team Reference Material

(unlimited version)

凌皓煜

陈　彤

顾　逸

Shanghai Jiao Tong University, 2018

Contents

1 Graph Theory **4**

1.1 2-SAT (ct) . . . 4
1.2 双连通分量 . . . 5
1.2.1 点双连通分量 . . . 5
1.2.2 边双连通分量 . . . 5
1.3 K 短路 (lhy) . . . 5
1.4 最大团 . . . 9
1.5 一般图最大匹配 . . . 9
1.6 带花树 . . . 9
1.7 KM 算法 . . . 9
1.8 支配树 . . . 9
1.8.1 DAG (ct) . . . 9
1.8.2 一般图 . . . 10
1.9 虚树 (ct) . . . 10
1.10 树上点分治 (ct) . . . 12
1.11 树上倍增 (ct) . . . 13
1.12 Prufer 编码 . . . 13
1.13 Link-Cut Tree (ct) . . . 13
1.14 仙人掌 . . . 15
1.15 弦图 . . . 15
1.16 最小割 . . . 15
1.17 最大流 (ct) . . . 15
1.18 费用流 (ct) . . . 16
1.19 有上下界的网络流 (Durandal) . . . 17
1.20 zkw 费用流 . . . 17
1.21 差分约束 . . . 17

2 Math **18**

2.1 int64 相乘取模 (Durandal) . . . 18
2.2 扩展欧几里得 (gy) . . . 18
2.3 中国剩余定理 (Durandal) . . . 19
2.4 线性同余不等式 (Durandal) . . . 19
2.5 组合数 . . . 19
2.6 高斯消元 . . . 19
2.7 Miller Rabin & Pollard Rho (gy) . . . 19
2.8 $O(m^2 \log n)$ 线性递推 . . . 21
2.9 Polynomial . . . 21
2.9.1 FFT . . . 21
2.9.2 NTT & 多项式求逆 . . . 21
2.10 拉格朗日插值 . . . 21
2.11 杜教筛 . . . 21
2.12 BSGS (ct,gy) . . . 21
2.12.1 BSGS . . . 21
2.12.2 扩展 BSGS . . . 22

2.13 直线下整点个数 (gy) . . . 23
2.14 单纯形 . . . 23
2.15 辛普森积分 . . . 23

3 Geometry **24**
3.1 点、直线、圆 (gy) . . . 24
3.2 点到凸包切线 . . . 28
3.3 直线凸包交点 . . . 28
3.4 凸包游戏 . . . 28
3.5 半平面交 . . . 28
3.6 旋转卡壳 . . . 28
3.7 判断圆是否有交 . . . 28
3.8 最小圆覆盖 . . . 28
3.9 最小球覆盖 . . . 28
3.10 $O(n^2 \log n)$ 圆交面积和重心 . . . 28
3.11 圆与多边形交 . . . 28
3.12 $O(n \log n)$ 凸多边形内的最大圆 . . . 28
3.13 三维凸包 . . . 28
3.14 三维绕轴旋转 . . . 28

4 String **29**
4.1 KMP . . . 29
4.2 AC 自动机 . . . 29
4.3 后缀数组 . . . 29
4.4 后缀自动机 . . . 29
4.5 Manacher . . . 29
4.6 回文自动机 . . . 29
4.7 最小表示法 . . . 29

5 Data Structure **30**
5.1 莫队 (ct) . . . 30
5.2 ST 表 (ct) . . . 30
5.3 可并堆 (ct) . . . 31
5.4 zkw 线段树 (ct) . . . 31
5.5 Splay (ct) . . . 32
5.6 Treap (ct) . . . 36
5.7 可持久化平衡树 (ct) . . . 37
5.8 CDQ 分治 (ct) . . . 40
5.9 Bitset (ct) . . . 41

6 Others **43**
6.1 vimrc (gy) . . . 43
6.2 STL 释放内存 (Durandal) . . . 43
6.3 开栈 (Durandal) . . . 43
6.4 O3 (gy) . . . 44
6.5 Java Template (gy) . . . 44
6.6 Big Fraction (gy) . . . 46
6.7 模拟退火 (ct) . . . 46
6.8 三分 (ct) . . . 47
6.9 Zeller Congruence (gy) . . . 47
6.10 博弈论模型 (gy) . . . 47
6.11 积分表 . . . 48
6.12 公式、数列、定理 . . . 48

Chapter 1

Graph Theory

1.1 2-SAT (ct)

```
struct Edge {
    Edge *next;
    int to;
} *last[maxn << 1], e[maxn << 2], *ecnt = e;
inline void link(int a, int b)
{
    *++ecnt = (Edge) {last[a], b}; last[a] = ecnt;
}
int dfn[maxn], low[maxn], timer, st[maxn], top, id[maxn], colcnt, n;
bool fail, used[maxn];
void tarjan(int x, int fa)
{
    dfn[x] = low[x] = ++timer; st[++top] = x;
    for (R Edge *iter = last[x]; iter; iter = iter -> next)
        if (iter -> to != fa)
        {
            if (!dfn[iter -> to])
            {
                tarjan(iter -> to, x);
                cmin(low[x], low[iter -> to]);
            }
            else if (!id[iter -> to]) cmin(low[x], dfn[iter -> to]);
        }
    if (dfn[x] == low[x])
    {
        ++colcnt; bool flag = 1;
        for (; ;)
        {
            int now = st[top--];
            id[now] = colcnt;
            if (now <= 2 * n)
            {
                flag &= !used[id[now <= n ? now + n : now - n]];
                now <= n ? fail |= (id[now + n] == id[now]) : fail |= (id[now - n] == id[now]);
            }
            if (now == x) break;
        }
        used[colcnt] = flag;
    }
}
int ans[maxn], tot;
int main()
```

```
{
    /*
        build your graph here.
    */
    for (R int i = 1; !fail && i <= n; ++i) if (!dfn[i]) tarjan(i, 0);
    if (fail)
    {
        puts("Impossible");
        return 0;
    }
    for (R int i = 1; i <= n; ++i) if (used[id[i]]) ans[++tot] = i;
    printf("%d\n", tot);
    std::sort(ans + 1, ans + tot + 1);
    for (R int i = 1; i <= tot; ++i) printf("%d ", ans[i]);
    return 0;
}
```

1.2 双连通分量

1.2.1 点双连通分量

1.2.2 边双连通分量

1.3 K 短路 (lhy)

```
const int MAXNODE = MAXN + MAXM * 2;

bool used[MAXN];
int n, m, cnt, S, T, Kth, N, TT;
int rt[MAXN], seq[MAXN], adj[MAXN], from[MAXN], dep[MAXN];
LL dist[MAXN], w[MAXM], ans[MAXK];

struct GivenEdge{
    int u, v, w;
    GivenEdge() {};
    GivenEdge(int _u, int _v, int _w) : u(_u), v(_v), w(_w){};
}edge[MAXM];

struct Edge{
    int v, nxt, w;
    Edge() {};
    Edge(int _v, int _nxt, int _w) : v(_v), nxt(_nxt), w(_w) {};
}e[MAXM];

inline void addedge(int u, int v, int w)
{
    e[++cnt] = Edge(v, adj[u], w); adj[u] = cnt;
}

void dij(int S)
{
    for(int i = 1; i <= N; i++)
    {
        dist[i] = INF;
        dep[i] = 0x3f3f3f3f;
        used[i] = false;
        from[i] = 0;
    }
    static priority_queue<pair<LL, int>, vector<pair<LL, int> >, greater<pair<LL, int> > > hp;
```

```
    while(!hp.empty())hp.pop();
    hp.push(make_pair(dist[S] = 0, S));
    dep[S] = 1;
    while(!hp.empty())
    {
        pair<LL, int> now = hp.top();
        hp.pop();
        int u = now.second;
        if(used[u])continue;
        else used[u] = true;
        for(int p = adj[u]; p; p = e[p].nxt)
        {
            int v = e[p].v;
            if(dist[u] + e[p].w < dist[v])
            {
                dist[v] = dist[u] + e[p].w;
                dep[v] = dep[u] + 1;
                from[v] = p;
                hp.push(make_pair(dist[v], v));
            }
        }
    }
    for(int i = 1; i <= m; i++)    w[i] = 0;
    for(int i = 1; i <= N; i++)
        if(from[i])w[from[i]] = -1;
    for(int i = 1; i <= m; i++)
    {
        if(~w[i] && dist[edge[i].u] < INF && dist[edge[i].v] < INF)
        {
            w[i] = -dist[edge[i].u] + (dist[edge[i].v] + edge[i].w);
        }
        else
        {
            w[i] = -1;
        }
    }
}

inline bool cmp_dep(int p, int q)
{
    return dep[p] < dep[q];
}

struct Heap{
    LL key;
    int id, lc, rc, dist;
    Heap() {};
    Heap(LL k, int i, int l, int r, int d) : key(k), id(i), lc(l), rc(r), dist(d) {};
    inline void clear()
    {
        key = 0;
        id = lc = rc = dist = 0;
    }
}hp[MAXNODE];

inline int merge_simple(int u, int v)
{
    if(!u)return v;
    if(!v)return u;
    if(hp[u].key > hp[v].key)
    {
```

```
        swap(u, v);
    }
    hp[u].rc = merge_simple(hp[u].rc, v);
    if(hp[hp[u].lc].dist < hp[hp[u].rc].dist)
    {
        swap(hp[u].lc, hp[u].rc);
    }
    hp[u].dist = hp[hp[u].rc].dist + 1;
    return u;
}

inline int merge_full(int u, int v)
{
    if(!u)return v;
    if(!v)return u;
    if(hp[u].key > hp[v].key)
    {
        swap(u, v);
    }
    int nownode = ++cnt;
    hp[nownode] = hp[u];
    hp[nownode].rc = merge_full(hp[nownode].rc, v);
    if(hp[hp[nownode].lc].dist < hp[hp[nownode].rc].dist)
    {
        swap(hp[nownode].lc, hp[nownode].rc);
    }
    hp[nownode].dist = hp[hp[nownode].rc].dist + 1;
    return nownode;
}

priority_queue<pair<LL, int>, vector<pair<LL, int> >, greater<pair<LL, int> > > Q;

int main()
{
    while(scanf("%d%d", &n, &m) != EOF)
    {
        scanf("%d%d%d%d", &S, &T, &Kth, &TT);
        for(int i = 1; i <= m; i++)
        {
            int u, v, w;
            scanf("%d%d%d", &u, &v, &w);
            edge[i] = {u, v, w};
        }
        N = n;
        memset(adj, 0, sizeof(*adj) * (N + 1));
        cnt = 0;
        for(int i = 1; i <= m; i++)
            addedge(edge[i].v, edge[i].u, edge[i].w);
        dij(T);
        if(dist[S] > TT)
        {
            puts("Whitesnake!");
            continue;
        }
        for(int i = 1; i <= N; i++)
            seq[i] = i;
        sort(seq + 1, seq + N + 1, cmp_dep);

        cnt = 0;
        memset(adj, 0, sizeof(*adj) * (N + 1));
        memset(rt, 0, sizeof(*rt) * (N + 1));
```

```
        for(int i = 1; i <= m; i++)
            addedge(edge[i].u, edge[i].v, edge[i].w);
        rt[T] = cnt = 0;
        hp[0].dist = -1;
        for(int i = 1; i <= N; i++)
        {
            int u = seq[i], v = edge[from[u]].v;
            rt[u] = 0;
            for(int p = adj[u]; p; p = e[p].nxt)
            {
                if(~w[p])
                {
                    hp[++cnt] = Heap(w[p], p, 0, 0, 0);
                    rt[u] = merge_simple(rt[u], cnt);
                }
            }
            if(i == 1)continue;
            rt[u] = merge_full(rt[u], rt[v]);
        }
        while(!Q.empty())Q.pop();
        Q.push(make_pair(dist[S], 0));
        edge[0].v = S;
        for(int kth = 1; kth <= Kth; kth++)
        {
            if(Q.empty())
            {
                ans[kth] = -1;
                continue;
            }
            pair<LL, int> now = Q.top(); Q.pop();
            ans[kth] = now.first;
            int p = now.second;
            if(hp[p].lc)
            {
                Q.push(make_pair(+hp[hp[p].lc].key + now.first - hp[p].key, hp[p].lc));
            }
            if(hp[p].rc)
            {
                Q.push(make_pair(+hp[hp[p].rc].key + now.first - hp[p].key, hp[p].rc));
            }
            if(rt[edge[hp[p].id].v])
            {
                Q.push(make_pair(hp[rt[edge[hp[p].id].v]].key + now.first, rt[edge[hp[p].id].v]));
            }
        }
        if(ans[Kth] == -1 || ans[Kth] > TT)
        {
            puts("Whitesnake!");
        }
        else
        {
            puts("yareyaredawa");
        }
    }
}
```

1.4 最大团

1.5 一般图最大匹配

1.6 带花树

1.7 KM 算法

1.8 支配树

1.8.1 DAG (ct)

```
struct Edge {
    Edge *next;
    int to;
} ;
Edge *last[maxn], e[maxm], *ecnt = e; // original graph
Edge *rlast[maxn], re[maxm], *recnt = re; // reversed-edge graph
Edge *tlast[maxn], te[maxn << 1], *tecnt = te; // dominate tree graph
int deg[maxn], q[maxn], fa[maxn][20], all_fa[maxn], fa_cnt, size[maxn], dep[maxn];
inline void link(int a, int b)
{
    *++ecnt = (Edge) {last[a], b}; last[a] = ecnt; ++deg[b];
}
inline void link_rev(R int a, R int b)
{
    *++recnt = (Edge) {rlast[a], b}; rlast[a] = recnt;
}
inline void link_tree(R int a, R int b)
{
    *++tecnt = (Edge) {tlast[a], b}; tlast[a] = tecnt;
}
inline int getlca(R int a, R int b)
{
    if (dep[a] < dep[b]) std::swap(a, b);
    R int temp = dep[a] - dep[b];
    for (R int i; temp; temp -= 1 << i)
        a = fa[a][i = __builtin_ctz(temp)];
    for (R int i = 16; ~i; --i)
        if (fa[a][i] != fa[b][i])
            a = fa[a][i], b = fa[b][i];
    if (a == b) return a;
    return fa[a][0];
}
void dfs(R int x)
{
    size[x] = 1;
    for (R Edge *iter = tlast[x]; iter; iter = iter -> next)
        dfs(iter -> to), size[x] += size[iter -> to];
}
int main()
{
    q[1] = 0;
    R int head = 0, tail = 1;
    while (head < tail)
    {
        R int now = q[++head];
        fa_cnt = 0;
```

```
        for (R Edge *iter = rlast[now]; iter; iter = iter -> next)
            all_fa[++fa_cnt] = iter -> to;
        for (; fa_cnt > 1; --fa_cnt)
            all_fa[fa_cnt - 1] = getlca(all_fa[fa_cnt], all_fa[fa_cnt - 1]);
        fa[now][0] = all_fa[fa_cnt];
        dep[now] = dep[all_fa[fa_cnt]] + 1;
        if (now) link_tree(fa[now][0], now);

        for (R int i = 1; i <= 16; ++i)
            fa[now][i] = fa[fa[now][i - 1]][i - 1];
        for (R Edge *iter = last[now]; iter; iter = iter -> next)
            if (--deg[iter -> to] == 0) q[++tail] = iter -> to;
    }
    dfs(0);
    for (R int i = 1; i <= n; ++i) printf("%d\n", size[i] - 1 );
    return 0;
}
```

1.8.2 一般图

1.9 虚树 (ct)

```
struct Edge {
    Edge *next;
    int to;
} *last[maxn], e[maxn << 1], *ecnt = e;
inline void link(int a, int b)
{
    *++ecnt = (Edge) {last[a], b}; last[a] = ecnt;
    *++ecnt = (Edge) {last[b], a}; last[b] = ecnt;
}
int a[maxn], n, dfn[maxn], pos[maxn], timer, inv[maxn], st[maxn];
int fa[maxn], size[maxn], dep[maxn], son[maxn], top[maxn];
bool vis[maxn];
void dfs1(int x)
{
    vis[x] = 1; size[x] = 1; dep[x] = dep[fa[x]] + 1;
    for (R Edge *iter = last[x]; iter; iter = iter -> next)
        if (!vis[iter -> to])
        {
            fa[iter -> to] = x;
            dfs1(iter -> to);
            size[x] += size[iter -> to];
            size[son[x]] < size[iter -> to] ? son[x] = iter -> to : 0;
        }
}
void dfs2(int x)
{
    vis[x] = 0; top[x] = x == son[fa[x]] ? top[fa[x]] : x;
    dfn[x] = ++timer; pos[timer] = x;
    if (son[x]) dfs2(son[x]);
    for (R Edge *iter = last[x]; iter; iter = iter -> next)
        if (vis[iter -> to]) dfs2(iter -> to);
    inv[x] = timer;
}
inline int getlca(int a, int b)
{
    while (top[a] != top[b])
        dep[top[a]] < dep[top[b]] ? b = fa[top[b]] : a = fa[top[a]];
```

```
    return dep[a] < dep[b] ? a : b;
}
inline bool cmp(int a, int b)
{
    return dfn[a] < dfn[b];
}
inline bool isson(int a, int b)
{
    return dfn[a] <= dfn[b] && dfn[b] <= inv[a];
}
typedef long long ll;
bool imp[maxn];
struct sEdge {
    sEdge *next;
    int to, w;
} *slast[maxn], se[maxn << 1], *secnt = se;
inline void slink(int a, int b, int w)
{
    *++secnt = (sEdge) {slast[a], b, w}; slast[a] = secnt;
}
int main()
{
    scanf("%d", &n);
    for (int i = 1; i < n; ++i)
    {
        int a, b; scanf("%d%d", &a, &b);
        link(a, b);
    }
    int m; scanf("%d", &m);
    dfs1(1); dfs2(1);
    memset(size, 0, (n + 1) << 2);
    for (; m; --m)
    {
        int top = 0; scanf("%d", &k);
        for (int i = 1; i <= k; ++i) scanf("%d", &a[i]), vis[a[i]] = imp[a[i]] = 1;
        std::sort(a + 1, a + k + 1, cmp);
        int p = k;
        for (int i = 1; i < k; ++i)
        {
            int lca = getlca(a[i], a[i + 1]);
            if (!vis[lca]) vis[a[++p] = lca] = 1;
        }
        std::sort(a + 1, a + p + 1, cmp);
        st[++top] = a[1];
        for (int i = 2; i <= p; ++i)
        {
            while (!isson(st[top], a[i])) --top;
            slink(st[top], a[i], dep[a[i]] - dep[st[top]]);
            st[++top] = a[i];
        }
        /*
            write your code here.
        */
        for (int i = 1; i <= p; ++i) vis[a[i]] = imp[a[i]] = 0, slast[a[i]] = 0;
        secnt = se;
    }
    return 0;
}
```

1.10 树上点分治 (ct)

```
int root, son[maxn], size[maxn], sum;
bool vis[maxn];
void dfs_root(int x, int fa)
{
    size[x] = 1; son[x] = 0;
    for (R Edge *iter = last[x]; iter; iter = iter -> next)
    {
        if (iter -> to == fa || vis[iter -> to]) continue;
        dfs_root(iter -> to, x);
        size[x] += size[iter -> to];
        cmax(son[x], size[iter -> to]);
    }
    cmax(son[x], sum - size[x]);
    if (!root || son[x] < son[root]) root = x;
}
void dfs_chain(int x, int fa, int st1, int st2)
{
    /*
        write your code here.
    */
    for (Edge *iter = last[x]; iter; iter = iter -> next)
    {
        if (vis[iter -> to] || iter -> to == fa) continue;
        dfs_chain(iter -> to, x);
    }
}
void calc(int x)
{
    for (Edge *iter = last[x]; iter; iter = iter -> next)
    {
        if (vis[iter -> to]) continue;
        dfs_chain(iter -> to, x);
        /*
            write your code here.
        */
    }
}
void work(int x)
{
    vis[x] = 1;
    calc(x);
    for (R Edge *iter = last[x]; iter; iter = iter -> next)
    {
        if (vis[iter -> to]) continue;
        root = 0;
        sum = size[iter -> to];
        dfs_root(iter -> to, 0);
        work(root);
    }
}
int main()
{
    root = 0; sum = n;
    dfs_root(1, 0);
    work(root);
    return 0;
}
```

1.11 树上倍增 (ct)

```
int fa[maxn][17], mn[maxn][17], dep[maxn];
bool vis[maxn];
void dfs(int x)
{
    vis[x] = 1;
    for (int i = 1; i <= 16; ++i)
    {
        if (dep[x] < (1 << i)) break;
        fa[x][i] = fa[fa[x][i - 1]][i - 1];
        mn[x][i] = dmin(mn[x][i - 1], mn[fa[x][i - 1]][i - 1]);
    }
    for (Edge *iter = last[x]; iter; iter = iter -> next)
        if (!vis[iter -> to])
        {
            fa[iter -> to][0] = x;
            mn[iter -> to][0] = iter -> w;
            dep[iter -> to] = dep[x] + 1;
            dfs(iter -> to);
        }
}
inline int getlca(int x, int y)
{
    if (dep[x] < dep[y]) std::swap(x, y);
    int t = dep[x] - dep[y];
    for (int i = 0; i <= 16 && t; ++i)
        if ((1 << i) & t)
            x = fa[x][i], t ^= 1 << i;
    for (int i = 16; i >= 0; --i)
        if (fa[x][i] != fa[y][i])
        {
            x = fa[x][i];
            y = fa[y][i];
        }
    if (x == y) return x;
    return fa[x][0];
}
inline int getans(int x, int f)
{
    int ans = inf, t = dep[x] - dep[f];
    for (int i = 0; i <= 16 && t; ++i)
        if (t & (1 << i))
        {
            cmin(ans, mn[x][i]);
            x = fa[x][i];
            t ^= 1 << i;
        }
    return ans;
}
```

1.12 Prufer 编码

1.13 Link-Cut Tree (ct)

```
struct Node *null;
struct Node {
    Node *ch[2], *fa, *pos;
```

```
    int val, mn, l, len; bool rev;
    // min_val in chain
    inline bool type()
    {
        return fa -> ch[1] == this;
    }
    inline bool check()
    {
        return fa -> ch[type()] == this;
    }
    inline void pushup()
    {
        pos = this; mn = val;
        ch[0] -> mn < mn ? mn = ch[0] -> mn, pos = ch[0] -> pos : 0;
        ch[1] -> mn < mn ? mn = ch[1] -> mn, pos = ch[1] -> pos : 0;
        len = ch[0] -> len + ch[1] -> len + l;
    }
    inline void pushdown()
    {
        if (rev)
        {
            ch[0] -> rev ^= 1;
            ch[1] -> rev ^= 1;
            std::swap(ch[0], ch[1]);
            rev ^= 1;
        }
    }
    inline void pushdownall()
    {
        if (check()) fa -> pushdownall();
        pushdown();
    }
    inline void rotate()
    {
        bool d = type(); Node *f = fa, *gf = f -> fa;
        (fa = gf, f -> check()) ? fa -> ch[f -> type()] = this : 0;
        (f -> ch[d] = ch[!d]) != null ? ch[!d] -> fa = f : 0;
        (ch[!d] = f) -> fa = this;
        f -> pushup();
    }
    inline void splay(bool need = 1)
    {
        if (need) pushdownall();
        for (; check(); rotate())
            if (fa -> check())
                (type() == fa -> type() ? fa : this) -> rotate();
        pushup();
    }
    inline Node *access()
    {
        Node *i = this, *j = null;
        for (; i != null; i = (j = i) -> fa)
        {
            i -> splay();
            i -> ch[1] = j;
            i -> pushup();
        }
        return j;
    }
    inline void make_root()
    {
```

```
        access();
        splay();
        rev ^= 1;
    }
    inline void link(Node *that)
    {
        make_root();
        fa = that;
        splay(0);
    }
    inline void cut(Node *that)
    {
        make_root();
        that -> access();
        that -> splay(0);
        that -> ch[0] = fa = null;
        that -> pushup();
    }
} mem[maxn];
inline Node *query(Node *a, Node *b)
{
    a -> make_root(); b -> access(); b -> splay(0);
    return b -> pos;
}
inline int dist(Node *a, Node *b)
{
    a -> make_root(); b -> access(); b -> splay(0);
    return b -> len;
}
```

1.14 仙人掌

1.15 弦图

1.16 最小割

1.17 最大流 (ct)

```
struct Edge {
    Edge *next, *rev;
    int to, cap;
} *last[maxn], *cur[maxn], e[maxm], *ecnt = e;
inline void link(R int a, R int b, R int w)
{
    *++ecnt = (Edge) {last[a], ecnt + 1, b, w}; last[a] = ecnt;
    *++ecnt = (Edge) {last[b], ecnt - 1, a, 0}; last[b] = ecnt;
}
int ans, s, t, q[maxn], dep[maxn];
inline bool bfs()
{
    memset(dep, -1, (t + 1) << 2);
    dep[q[1] = t] = 0; int head = 0, tail = 1;
    while (head < tail)
    {
        int now = q[++head];
        for (Edge *iter = last[now]; iter; iter = iter -> next)
            if (dep[iter -> to] == -1 && iter -> rev -> cap)
```

```
                dep[q[++tail] = iter -> to] = dep[now] + 1;
    }
    return dep[s] != -1;
}
int dfs(int x, int f)
{
    if (x == t) return f;
    int used = 0;
    for (Edge* &iter = cur[x]; iter; iter = iter -> next)
        if (iter -> cap && dep[iter -> to] + 1 == dep[x])
        {
            int v = dfs(iter -> to, dmin(f - used, iter -> cap));
            iter -> cap -= v;
            iter -> rev -> cap += v;
            used += v;
            if (used == f) return f;
        }
    return used;
}
inline void dinic()
{
    while (bfs())
    {
        memcpy(cur, last, sizeof cur);
        ans += dfs(s, inf);
    }
}
```

1.18 费用流 (ct)

```
struct Edge {
    Edge *next, *rev;
    int from, to, cap, cost;
} *last[maxn], *prev[maxn], e[maxm], *ecnt = e;
inline void link(int a, int b, int w, int c)
{
    *++ecnt = (Edge) {last[a], ecnt + 1, a, b, w, c}; last[a] = ecnt;
    *++ecnt = (Edge) {last[b], ecnt - 1, b, a, 0, -c}; last[b] = ecnt;
}
int s, t, q[maxn << 2], dis[maxn];
ll ans;
bool inq[maxn];
#define inf 0x7fffffff
inline bool spfa()
{
    for (int i = 1; i <= t; ++i) dis[i] = inf;
    int head = 0, tail = 1; dis[q[1] = s] = 0;
    while (head < tail)
    {
        int now = q[++head]; inq[now] = 0;
        for (Edge *iter = last[now]; iter; iter = iter -> next)
            if (iter -> cap && dis[iter -> to] > dis[now] + iter -> cost)
            {
                dis[iter -> to] = dis[now] + iter -> cost;
                prev[iter -> to] = iter;
                !inq[iter -> to] ? inq[q[++tail] = iter -> to] = 1 : 0;
            }
    }
    return dis[t] != inf;
```

```
}
inline void mcmf()
{
    int x = inf;
    for (Edge *iter = prev[t]; iter; iter = prev[iter -> from]) cmin(x, iter -> cap);
    for (Edge *iter = prev[t]; iter; iter = prev[iter -> from])
    {
        iter -> cap -= x;
        iter -> rev -> cap += x;
        ans += 1ll * x * iter -> cost;
    }
}
```

1.19 有上下界的网络流 (Durandal)

$B(u,v)$ 表示边 (u,v) 流量的下界，$C(u,v)$ 表示边 (u,v) 流量的上界，设 $F(u,v)$ 表示边 (u,v) 的实际流量
设 $G(u,v)=F(u,v)-B(u,v)$，则 $0 \le G(u,v) \le C(u,v)-B(u,v)$

- 无源汇的上下界可行流
  建立超级源点 S^* 和超级汇点 T^*，对于原图每一条边 (u,v) 在新网络中连如下三条边：$S^* \to v$，容量为 $B(u,v)$；$u \to T^*$，容量为 $B(u,v)$；$u \to v$，容量为 $C(u,v)-B(u,v)$。最后求新网络的最大流，判断从超级源点 S^* 出发的边是否都满流即可，边 (u,v) 的最终解中的实际流量为 $G(u,v)+B(u,v)$。

- 有源汇的上下界可行流
  从汇点 T 到源点 S 连一条上界为 ∞，下界为 0 的边。按照无源汇的上下界可行流一样做即可，流量即为 $T \to S$ 边上的流量。

- 有源汇的上下界最大流
  - 在有源汇的上下界可行流中，从汇点 T 到源点 S 的边改为连一条上界为 ∞，下界为 x 的边。x 满足二分性质，找到最大的 x 使得新网络存在有源汇的上下界可行流即为原图的最大流。
  - 从汇点 T 到源点 S 连一条上界为 ∞，下界为 0 的边，变成无源汇的网络。按照无源汇的上下界可行流的方法，建立超级源点 S^* 与超级汇点 T^*，求一遍 $S^* \to T^*$ 的最大流，再将从汇点 T 到源点 S 的这条边拆掉，求一次 $S \to T$ 的最大流即可。

- 有源汇的上下界最小流
  - 在有源汇的上下界可行流中，从汇点 T 到源点 S 的边改为连一条上界为 x，下界为 0 的边。x 满足二分性质，找到最小的 x 使得新网络存在有源汇的上下界可行流即为原图的最大流。
  - 按照无源汇的上下界可行流的方法，建立超级源点 S^* 与超级汇点 T^*，求一遍 $S^* \to T^*$ 的最大流，但是注意不加上汇点 T 到源点 S 的这条边，即不使之改为无源汇的网络去求解。求完后，再加上那条汇点 T 到源点 S 的边，上界为 ∞ 的边。因为这条边的下界为 0，所以 S^*, T^* 无影响，再求一次 $S^* \to T^*$ 的最大流。若超级源点 S^* 出发的边全部满流，则 $T \to S$ 边上的流量即为原图的最小流，否则无解。

1.20 zkw 费用流

1.21 差分约束

Chapter 2

Math

2.1 int64 相乘取模 (Durandal)

```
int64_t mul(int64_t x, int64_t y, int64_t p) {
    int64_t t = (x * y - (int64_t) ((long double) x / p * y + 1e-3) * p) % p;
    return t < 0 ? t + p : t;
}
```

2.2 扩展欧几里得 (gy)

```
// return gcd(a, b)
// ax+by=gcd(a,b)
int extend_gcd(int a, int b, int &x, int &y) {
    if (b == 0) {
        x = 1, y = 0;
        return a;
    }
    int res = extend_gcd(b, a % b, x, y);
    int t = y;
    y = x - a / b * y;
    x = t;
    return res;
}

// return minimal positive integer x so that ax+by=c
// or -1 if such x does not exist
int solve_equ(int a, int b, int c) {
    int x, y, d;
    d = extend_gcd(a, b, x, y);
    if (c % d)
        return -1;
    int t = c / d;
    x *= t;
    y *= t;
    int k = b / d;
    x = (x % k + k) % k;
    return x;
}

// return minimal positive integer x so that ax==b(mod p)
// or -1 if such x does not exist
int solve(int a, int b, int p) {
    a = (a % p + p) % p;
    b = (b % p + p) % p;
```

```
    return solve_equ(a, p, b);
}
```

2.3 中国剩余定理 (Durandal)

返回是否可行，余数和模数结果为 r_1, m_1

```
bool CRT(int &r1, int &m1, int r2, int m2) {
    int x, y, g = extend_gcd(m1, m2, x, y);
    if ((r2 - r1) % g != 0) return false;
    x = 1ll * (r2 - r1) * x % m2;
    if (x < 0) x += m2;
    x /= g;
    r1 += m1 * x;
    m1 *= m2 / g;
    return true;
}
```

2.4 线性同余不等式 (Durandal)

必须满足 $0 \leq d < m, 0 \leq l \leq r < m$，返回 $\min\{x \geq 0 \mid l \leq x \cdot d \bmod m \leq r\}$，无解返回 -1

```
int64_t calc(int64_t d, int64_t m, int64_t l, int64_t r) {
    if (l == 0) return 0;
    if (d == 0) return -1;
    if (d * 2 > m) return calc(m - d, m, m - r, m - l);
    if ((l - 1) / d < r / d) return (l - 1) / d + 1;
    int64_t k = calc((-m % d + d) % d, d, l % d, r % d);
    if (k == -1) return -1;
    return (k * m + l - 1) / d + 1;
}
```

2.5 组合数

2.6 高斯消元

2.7 Miller Rabin & Pollard Rho (gy)

```
/*
 * In Java, use BigInteger.isProbablePrime(int certainty) to replace miller_rabin(BigInteger
 ↪ number)
 * Test Set / First Wrong Answer
 * 2 / 2,047
 * 2, 3 / 1,373,653
 * 31, 73 / 9,080,191
 * 2, 3, 5 / 25,326,001
 * 2, 3, 5, 7 / 3,215,031,751 (> Int.MAX_VALUE)
 * 2, 7, 61 / 4,759,123,141
 * 2, 13, 23, 1662803 / 1,122,004,669,633
 * 2, 3, 5, 7, 11 / 2,152,302,898,747
 * 2, 3, 5, 7, 11, 13 / 3,474,749,660,383
 * 2, 3, 5, 7, 11, 13, 17 / 341,550,071,728,321
 * 2, 3, 5, 7, 11, 13, 17, 19, 23 / 3,825,123,056,546,413,051
 * 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37 / 318,665,857,834,031,151,167,461 (> Long.MAX_VALUE)
 * 2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41 / 3,317,044,064,679,887,385,961,981
```

```
 */
const int test_case_size = 12;
const int test_cases[test_case_size] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};

int64_t multiply_mod(int64_t x, int64_t y, int64_t p) {
    int64_t t = (x * y - (int64_t) ((long double) x / p * y + 1e-3) * p) % p;
    return t < 0 ? t + p : t;
}

int64_t add_mod(int64_t x, int64_t y, int64_t p) {
    return (0ull + x + y) % p;
}

int64_t power_mod(int64_t x, int64_t exp, int64_t p) {
    int64_t ans = 1;
    while (exp) {
        if (exp & 1)
            ans = multiply_mod(ans, x, p);
        x = multiply_mod(x, x, p);
        exp >>= 1;
    }
    return ans;
}

bool miller_rabin_check(int64_t prime, int64_t base) {
    int64_t number = prime - 1;
    for (; ~number & 1; number >>= 1)
        continue;
    int64_t result = power_mod(base, number, prime);
    for (; number != prime - 1 && result != 1 && result != prime - 1; number <<= 1)
        result = multiply_mod(result, result, prime);
    return result == prime - 1 || (number & 1) == 1;
}

bool miller_rabin(int64_t number) {
    if (number < 2)
        return false;
    if (number < 4)
        return true;
    if (~number & 1)
        return false;
    for (int i = 0; i < test_case_size && test_cases[i] < number; i++)
        if (!miller_rabin_check(number, test_cases[i]))
            return false;
    return true;
}

int64_t gcd(int64_t x, int64_t y) {
    return y == 0 ? x : gcd(y, x % y);
}

int64_t pollard_rho_test(int64_t number, int64_t seed) {
    int64_t x = rand() % (number - 1) + 1, y = x;
    int head = 1, tail = 2;
    while (true) {
        x = multiply_mod(x, x, number);
        x = add_mod(x, seed, number);
        if (x == y)
            return number;
        int64_t answer = gcd(std::abs(x - y), number);
        if (answer > 1 && answer < number)
```

```
            return answer;
        if (++head == tail) {
            y = x;
            tail <<= 1;
        }
    }
}

void factorize(int64_t number, std::vector<int64_t> &divisor) {
    if (number > 1) {
        if (miller_rabin(number)) {
            divisor.push_back(number);
        } else {
            int64_t factor = number;
            while (factor >= number)
                factor = pollard_rho_test(number, rand() % (number - 1) + 1);
            factorize(number / factor, divisor);
            factorize(factor, divisor);
        }
    }
}
```

2.8 $O(m^2 \log n)$ 线性递推

2.9 Polynomial

2.9.1 FFT

2.9.2 NTT & 多项式求逆

2.10 拉格朗日插值

2.11 杜教筛

2.12 BSGS (ct,gy)

2.12.1 BSGS

p 是素数，返回 $\min\{x \geq 0 \mid y^x \equiv z \mod p\}$

```
const int mod = 19260817;
struct Hash
{
    Hash *next;
    int key, val;
} *last[mod], mem[100000], *tot = mem;
inline void insert(R int x, R int v)
{
    *++tot = (Hash) {last[x % mod], x, v}; last[x % mod] = tot;
}
inline int query(R int x)
{
    for (R Hash *iter = last[x % mod]; iter; iter = iter -> next)
        if (iter -> key == x) return iter -> val;
    return -1;
}
inline void del(R int x)
```

```
{
    last[x % mod] = 0;
}
int main()
{
    for (; T; --T)
    {
        R int y, z, p; scanf("%d%d%d", &y, &z, &p);
        R int m = (int) sqrt(p * 1.0);
        y %= p; z %= p;
        if (!y && !z) {puts("0"); continue;}
        if (!y) {puts("Orz, I cannot find x!"); continue;}
        R int pw = 1;
        for (R int i = 0; i < m; ++i, pw = 1ll * pw * y % p) insert(1ll * z * pw % p, i);
        R int ans = -1;
        for (R int i = 1, t, pw2 = pw; i <= p / m + 1; ++i, pw2 = 1ll * pw2 * pw % p)
            if ((t = query(pw2)) != -1) {ans = i * m - t; break;}
        if (ans == -1) puts("Orz, I cannot find x!");
        else printf("%d\n", ans );
        tot = mem; pw = 1;
        for (R int i = 0; i < m; ++i, pw = 1ll * pw * y % p) del(1ll * z * pw % p);
    }
    return 0;
}
```

2.12.2 扩展 BSGS

必须满足 $0 \le a < p, 0 \le b < p$，返回 $\min\{x \ge 0 \mid a^x \equiv b \mod p\}$

```
int64_t ex_bsgs(int64_t a, int64_t b, int64_t p) {
    if (b == 1)
        return 0;
    int64_t t, d = 1, k = 0;
    while ((t = std::__gcd(a, p)) != 1) {
        if (b % t) return -1;
        k++, b /= t, p /= t, d = d * (a / t) % p;
        if (b == d) return k;
    }
    map.clear();
    int64_t m = std::ceil(std::sqrt((long double) p));
    int64_t a_m = pow_mod(a, m, p);
    int64_t mul = b;
    for (int j = 1; j <= m; j++) {
        (mul *= a) %= p;
        map[mul] = j;
    }
    for (int i = 1; i <= m; i++) {
        (d *= a_m) %= p;
        if (map.count(d))
            return i * m - map[d] + k;
    }
    return -1;
}

int main() {
    int64_t a, b, p;
    while (scanf("%lld%lld%lld", &a, &b, &p) != EOF)
        printf("%lld\n", ex_bsgs(a, b, p));
    return 0;
}
```

2.13 直线下整点个数 (gy)

必须满足 $a \geq 0, b \geq 0, m > 0$，返回 $\sum_{i=0}^{n-1} \frac{a+bi}{m}$

```
int64_t count(int64_t n, int64_t a, int64_t b, int64_t m) {
    if (b == 0)
        return n * (a / m);
    if (a >= m)
        return n * (a / m) + count(n, a % m, b, m);
    if (b >= m)
        return (n - 1) * n / 2 * (b / m) + count(n, a, b % m, m);
    return count((a + b * n) / m, (a + b * n) % m, m, b);
}
```

2.14 单纯形

2.15 辛普森积分

Chapter 3

Geometry

3.1 点、直线、圆 (gy)

```
using number = long double;
const number eps = 1e-8;

number _sqrt(number x) {
    return std::sqrt(std::max(x, (number) 0));
}
number _asin(number x) {
    x = std::min(x, (number) 1), x = std::max(x, (number) -1);
    return std::asin(x);
}
number _acos(number x) {
    x = std::min(x, (number) 1), x = std::max(x, (number) -1);
    return std::acos(x);
}

int sgn(number x) {
    return (x > eps) - (x < -eps);
}
int cmp(number x, number y) {
    return sgn(x - y);
}

struct point {
    number x, y;
    point() {}
    point(number x, number y) : x(x), y(y) {}

    number len2() const {
        return x * x + y * y;
    }
    number len() const {
        return _sqrt(len2());
    }
    point unit() const {
        return point(x / len(), y / len());
    }
    point rotate90() const {
        return point(-y, x);
    }

    friend point operator+(const point &a, const point &b) {
        return point(a.x + b.x, a.y + b.y);
```

```cpp
    }
    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, number b) {
        return point(a.x * b, a.y * b);
    }
    friend point operator/(const point &a, number b) {
        return point(a.x / b, a.y / b);
    }
    friend number dot(const point &a, const point &b) {
        return a.x * b.x + a.y * b.y;
    }
    friend number det(const point &a, const point &b) {
        return a.x * b.y - a.y * b.x;
    }
    friend number operator==(const point &a, const point &b) {
        return cmp(a.x, b.x) == 0 && cmp(a.y, b.y) == 0;
    }
};

number dis2(const point &a, const point &b) {
    return (a - b).len2();
}
number dis(const point &a, const point &b) {
    return (a - b).len();
}

struct line {
    point a, b;
    line() {}
    line(point a, point b) : a(a), b(b) {}
    point value() const {
        return b - a;
    }
};

bool point_on_line(const point &p, const line &l) {
    return sgn(det(p - l.a, p - l.b)) == 0;
}
// including endpoint
bool point_on_ray(const point &p, const line &l) {
    return sgn(det(p - l.a, p - l.b)) == 0 &&
        sgn(dot(p - l.a, l.b - l.a)) >= 0;
}
// including endpoints
bool point_on_seg(const point &p, const line &l) {
    return sgn(det(p - l.a, p - l.b)) == 0 &&
        sgn(dot(p - l.a, l.b - l.a)) >= 0 &&
        sgn(dot(p - l.b, l.a - l.b)) >= 0;
}
bool seg_has_intersection(const line &a, const line &b) {
    if (point_on_seg(a.a, b) || point_on_seg(a.b, b) ||
            point_on_seg(b.a, a) || point_on_seg(b.b, a))
        return /* including endpoints */ true;
    return sgn(det(a.a - b.a, b.b - b.a)) * sgn(det(a.b - b.a, b.b - b.a)) < 0
        && sgn(det(b.a - a.a, a.b - a.a)) * sgn(det(b.b - a.a, a.b - a.a)) < 0;
}
point intersect(const line &a, const line &b) {
    number s1 = det(a.b - a.a, b.a - a.a);
    number s2 = det(a.b - a.a, b.b - a.a);
```

```
    return (b.a * s2 - b.b * s1) / (s2 - s1);
}
point projection(const point &p, const line &l) {
    return l.a + (l.b - l.a) * dot(p - l.a, l.b - l.a) / (l.b - l.a).len2();
}
number dis(const point &p, const line &l) {
    return std::abs(dot(p - l.a, l.b - l.a)) / (l.b - l.a).len();
}
point symmetry_point(const point &a, const point &o) {
    return o + o - a;
}
point reflection(const point &p, const line &l) {
    return symmetry_point(p, projection(p, l));
}

struct circle {
    point o;
    number r;
    circle() {}
    circle(point o, number r) : o(o), r(r) {}
};

bool intersect(const line &l, const circle &a, point &p1, point &p2) {
    number x = dot(l.a - a.o, l.b - l.a);
    number y = (l.b - l.a).len2();
    number d = x * x - y * ((l.a - a.o).len2() - a.r * a.r);
    if (sgn(d) < 0) return false;
    point p = l.a - (l.b - l.a) * (x / y), delta = (l.b - l.a) * (_sqrt(d) / y);
    p1 = p + delta, p2 = p - delta;
    return true;
}
bool intersect(const circle &a, const circle &b, point &p1, point &p2) {
    if (a.o == b.o && cmp(a.r, b.r) == 0)
        return /* value for coincident circles */ false;
    number s1 = (b.o - a.o).len();
    if (cmp(s1, a.r + b.r) > 0 || cmp(s1, std::abs(a.r - b.r)) < 0)
        return false;
    number s2 = (a.r * a.r - b.r * b.r) / s1;
    number aa = (s1 + s2) / 2, bb = (s1 - s2) / 2;
    point p = (b.o - a.o) * (aa / (aa + bb)) + a.o;
    point delta = (b.o - a.o).unit().rotate90() * _sqrt(a.r * a.r - aa * aa);
    p1 = p + delta, p2 = p - delta;
    return true;
}
bool tangent(const point &p0, const circle &c, point &p1, point &p2) {
    number x = (p0 - c.o).len2();
    number d = x - c.r * c.r;
    if (sgn(d) < 0) return false;
    if (sgn(d) == 0)
        return /* value for point_on_line */ false;
    point p = (p0 - c.o) * (c.r * c.r / x);
    point delta = ((p0 - c.o) * (-c.r * _sqrt(d) / x)).rotate90();
    p1 = c.o + p + delta;
    p2 = c.o + p - delta;
    return true;
}
bool ex_tangent(const circle &a, const circle &b, line &l1, line &l2) {
    if (cmp(std::abs(a.r - b.r), (b.o - a.o).len()) == 0) {
        point p1, p2;
        intersect(a, b, p1, p2);
        l1 = l2 = line(p1, p1 + (a.o - p1).rotate90());
```

```
        return true;
    } else if (cmp(a.r, b.r) == 0) {
        point dir = b.o - a.o;
        dir = (dir * (a.r / dir.len())).rotate90();
        l1 = line(a.o + dir, b.o + dir);
        l2 = line(a.o - dir, b.o - dir);
        return true;
    } else {
        point p = (b.o * a.r - a.o * b.r) / (a.r - b.r);
        point p1, p2, q1, q2;
        if (tangent(p, a, p1, p2) && tangent(p, b, q1, q2)) {
            l1 = line(p1, q1);
            l2 = line(p2, q2);
            return true;
        } else {
            return false;
        }
    }
}
bool in_tangent(const circle &a, const circle &b, line &l1, line &l2) {
    if (cmp(a.r + b.r, (b.o - a.o).len()) == 0) {
        point p1, p2;
        intersect(a, b, p1, p2);
        l1 = l2 = line(p1, p1 + (a.o - p1).rotate90());
        return true;
    } else {
        point p = (b.o * a.r + a.o * b.r) / (a.r + b.r);
        point p1, p2, q1, q2;
        if (tangent(p, a, p1, p2) && tangent(p, b, q1, q2)) {
            l1 = line(p1, q1);
            l2 = line(p2, q2);
            return true;
        } else {
            return false;
        }
    }
}
```

3.2 点到凸包切线

3.3 直线凸包交点

3.4 凸包游戏

3.5 半平面交

3.6 旋转卡壳

3.7 判断圆是否有交

3.8 最小圆覆盖

3.9 最小球覆盖

3.10 $O(n^2 \log n)$ 圆交面积和重心

3.11 圆与多边形交

3.12 $O(n \log n)$ 凸多边形内的最大圆

3.13 三维凸包

3.14 三维绕轴旋转

Chapter 4

String

4.1 KMP

4.2 AC 自动机

4.3 后缀数组

4.4 后缀自动机

4.5 Manacher

4.6 回文自动机

4.7 最小表示法

Chapter 5

Data Structure

5.1 莫队 (ct)

```
int size;
struct Query {
    int l, r, id;
    inline bool operator < (const Queuy &that) const {return l / size != that.l / size ? l < that.l
      ↪ : ((l / size) & 1 ? r < that.r : r > that.r);}
} q[maxn];
int main()
{
    size = (int) sqrt(n * 1.0);
    std::sort(q + 1, q + m + 1);
    int l = 1, r = 0;
    for (int i = 1; i <= m; ++i)
    {
        for (; r < q[i].r; ) add(++r);
        for (; r > q[i].r; ) del(r--);
        for (; l < q[i].l; ) del(l++);
        for (; l > q[i].l; ) add(--l);
        /*
            write your code here.
        */
    }
    return 0;
}
```

5.2 ST 表 (ct)

```
int a[maxn], f[20][maxn], n;
int Log[maxn];

void build()
{
    for (int i = 1; i <= n; ++i) f[0][i] = a[i];

    int lim = Log[n];
    for (int j = 1; j <= lim; ++j)
    {
        int *fj = f[j], *fj1 = f[j - 1];
        for (int i = 1; i <= n - (1 << j) + 1; ++i)
            fj[i] = dmax(fj1[i], fj1[i + (1 << (j - 1))]);
    }
}
```

```
int Query(int l, int r)
{
    int k = Log[r - l + 1];
    return dmax(f[k][l], f[k][r - (1 << k) + 1]);
}
int main()
{
    scanf("%d", &n);
    Log[0] = -1;
    for (int i = 1; i <= n; ++i)
    {
        scanf("%d", &a[i]);
        Log[i] = Log[i >> 1] + 1;
    }
    build();
    int q;
    scanf("%d", &q);
    for (; q; --q)
    {
        int l, r; scanf("%d%d", &l, &r);
        printf("%d\n", Query(l, r) );
    }
}
```

5.3 可并堆 (ct)

```
struct Node {
    Node *ch[2];
    ll val; int size;
    inline void update()
    {
        size = ch[0] -> size + ch[1] -> size + 1;
    }
} mem[maxn], *rt[maxn];
Node *merge(Node *a, Node *b)
{
    if (a == mem) return b;
    if (b == mem) return a;
    if (a -> val < b -> val) std::swap(a, b);
    // a -> pushdown();
    std::swap(a -> ch[0], a -> ch[1]);
    a -> ch[1] = merge(a -> ch[1], b);
    a -> update();
    return a;
}
```

5.4 zkw 线段树 (ct)

```
// must be 0-based !
inline void build()
{
    for (int i = M - 1; i; --i) tr[i] = dmax(tr[i << 1], tr[i << 1 | 1]);
}
inline void Change(int x, int v)
{
    x += M; tr[x] = v; x >>= 1;
    for (; x; x >>= 1) tr[x] = dmax(tr[x << 1], tr[x << 1 | 1]);
}
```

```
inline int Query(int s, int t)
{
    int ret = -0x7fffffff;
    for (s = s + M - 1, t = t + M + 1; s ^ t ^ 1; s >>= 1, t >>= 1)
    {
        if (~s & 1) cmax(ret, tr[s ^ 1]);
        if (t & 1) cmax(ret, tr[t ^ 1]);
    }
    return ret;
}
int main()
{
    int n; scanf("%d", &n);
    for (M = 1; M < n; M <<= 1) ;
    for (int i = 0; i < n; ++i)
        scanf("%d", &tr[i + M]);
    for (int i = n; i < M; ++i) tr[i + M] = -0x7fffffff;
    build();
    int q; scanf("%d", &q);
    for (; q; --q)
    {
        int l, r; scanf("%d%d", &l, &r); --l, --r;
        printf("%d\n", Query(l, r));
    }
    return 0;
}
```

5.5 Splay (ct)

指针版

```
struct Node *null;
struct Node {
    Node *ch[2], *fa;
    int val; bool rev;
    inline bool type()
    {
        return fa -> ch[1] == this;
    }
    inline void pushup()
    {
    }
    inline void pushdown()
    {
        if (rev)
        {
            ch[0] -> rev ^= 1;
            ch[1] -> rev ^= 1;
            std::swap(ch[0], ch[1]);
            rev ^= 1;
        }
    }
    inline void rotate()
    {
        bool d = type(); Node *f = fa, *gf = f -> fa;
        (fa = gf, f -> fa != null) ? fa -> ch[f -> type()] = this : 0;
        (f -> ch[d] = ch[!d]) != null ? ch[!d] -> fa = f : 0;
        (ch[!d] = f) -> fa = this;
        f -> pushup();
    }
```

```
    inline void splay()
    {
        for (; fa != null; rotate())
            if (fa -> fa != null)
                (type() == fa -> type() ? fa : this) -> rotate();
        pushup();
    }
} mem[maxn];
```

数组版

```
// BZOJ - 1500 维修数列
int fa[maxn], ch[maxn][2], a[maxn], size[maxn], cnt;
int sum[maxn], lmx[maxn], rmx[maxn], mx[maxn], v[maxn], id[maxn], root;
bool rev[maxn], tag[maxn];
inline void update(R int x)
{
    R int ls = ch[x][0], rs = ch[x][1];
    size[x] = size[ls] + size[rs] + 1;
    sum[x] = sum[ls] + sum[rs] + v[x];
    mx[x] = gmax(mx[ls], mx[rs]);
    cmax(mx[x], lmx[rs] + rmx[ls] + v[x]);
    lmx[x] = gmax(lmx[ls], sum[ls] + v[x] + lmx[rs]);
    rmx[x] = gmax(rmx[rs], sum[rs] + v[x] + rmx[ls]);
}
inline void pushdown(R int x)
{
    R int ls = ch[x][0], rs = ch[x][1];
    if (tag[x])
    {
        rev[x] = tag[x] = 0;
        if (ls) tag[ls] = 1, v[ls] = v[x], sum[ls] = size[ls] * v[x];
        if (rs) tag[rs] = 1, v[rs] = v[x], sum[rs] = size[rs] * v[x];
        if (v[x]>= 0)
        {
            if (ls) lmx[ls] = rmx[ls] = mx[ls] = sum[ls];
            if (rs) lmx[rs] = rmx[rs] = mx[rs] = sum[rs];
        }
        else
        {
            if (ls) lmx[ls] = rmx[ls] = 0, mx[ls] = v[x];
            if (rs) lmx[rs] = rmx[rs] = 0, mx[rs] = v[x];
        }
    }
    if (rev[x])
    {
        rev[x] ^= 1; rev[ls] ^= 1; rev[rs] ^= 1;
        swap(lmx[ls], rmx[ls]);swap(lmx[rs], rmx[rs]);
        swap(ch[ls][0], ch[ls][1]); swap(ch[rs][0], ch[rs][1]);
    }
}
inline void rotate(R int x)
{
    R int f = fa[x], gf = fa[f], d = ch[f][1] == x;
    if (f == root) root = x;
    (ch[f][d] = ch[x][d ^ 1]) > 0 ? fa[ch[f][d]] = f : 0;
    (fa[x] = gf) > 0 ? ch[gf][ch[gf][1] == f] = x : 0;
    fa[ch[x][d ^ 1] = f] = x;
    update(f);
}
inline void splay(R int x, R int rt)
```

```
{
    while (fa[x] != rt)
    {
        R int f = fa[x], gf = fa[f];
        if (gf != rt) rotate((ch[gf][1] == f) ^ (ch[f][1] == x) ? x : f);
        rotate(x);
    }
    update(x);
}
void build(R int l, R int r, R int rt)
{
    if (l > r) return ;
    R int mid = l + r >> 1, now = id[mid], last = id[rt];
    if (l == r)
    {
        sum[now] = a[l];
        size[now] = 1;
        tag[now] = rev[now] = 0;
        if (a[l] >= 0) lmx[now] = rmx[now] = mx[now] = a[l];
        else lmx[now] = rmx[now] = 0, mx[now] = a[l];
    }
    else
    {
        build(l, mid - 1, mid);
        build(mid + 1, r, mid);
    }
    v[now] = a[mid];
    fa[now] = last;
    update(now);
    ch[last][mid >= rt] = now;
}
int find(R int x, R int rank)
{
    if (tag[x] || rev[x]) pushdown(x);
    R int ls = ch[x][0], rs = ch[x][1], lsize = size[ls];
    if (lsize + 1 == rank) return x;
    if (lsize >= rank)
        return find(ls, rank);
    else
        return find(rs, rank - lsize - 1);
}
inline int prepare(R int l, R int tot)
{
    R int x = find(root, l - 1), y = find(root, l + tot);
    splay(x, 0);
    splay(y, x);
    return ch[y][0];
}
std::queue <int> q;
inline void Insert(R int left, R int tot)
{
    for (R int i = 1; i <= tot; ++i ) a[i] = FastIn();
    for (R int i = 1; i <= tot; ++i )
        if (!q.empty()) id[i] = q.front(), q.pop();
        else id[i] = ++cnt;
    build(1, tot, 0);
    R int z = id[(1 + tot) >> 1];
    R int x = find(root, left), y = find(root, left + 1);
    splay(x, 0);
    splay(y, x);
    fa[z] = y;
```

```
    ch[y][0] = z;
    update(y);
    update(x);
}
void rec(R int x)
{
    if (!x) return ;
    R int ls = ch[x][0], rs = ch[x][1];
    rec(ls); rec(rs); q.push(x);
    fa[x] = ch[x][0] = ch[x][1] = 0;
    tag[x] = rev[x] = 0;
}
inline void Delete(R int l, R int tot)
{
    R int x = prepare(l, tot), f = fa[x];
    rec(x); ch[f][0] = 0;
    update(f); update(fa[f]);
}
inline void Makesame(R int l, R int tot, R int val)
{
    R int x = prepare(l, tot), y = fa[x];
    v[x] = val; tag[x] = 1; sum[x] = size[x] * val;
    if (val >= 0) lmx[x] = rmx[x] = mx[x] = sum[x];
    else lmx[x] = rmx[x] = 0, mx[x] = val;
    update(y); update(fa[y]);
}
inline void Reverse(R int l, R int tot)
{
    R int x = prepare(l, tot), y = fa[x];
    if (!tag[x])
    {
        rev[x] ^= 1;
        swap(ch[x][0], ch[x][1]);
        swap(lmx[x], rmx[x]);
        update(y); update(fa[y]);
    }
}
inline void Query(R int l, R int tot)
{
    R int x = prepare(l, tot);
    printf("%d\n",sum[x] );
}
#define inf ((1 << 30))
int main()
{
    R int n = FastIn(),  m = FastIn(), l, tot, val;
    R char op, op2;
    mx[0] = a[1] = a[n + 2] = -inf;
    for (R int i = 2; i <= n + 1; i++ )
    {
        a[i] = FastIn();
    }
    for (R int i = 1; i <= n + 2; ++i) id[i] = i;
    n += 2; cnt = n; root = (n + 1) >> 1;
    build(1, n, 0);
    for (R int i = 1; i <= m; i++ )
    {
        op = getc();
        while (op < 'A' || op > 'Z') op = getc();
        getc(); op2 = getc();getc();getc();getc();getc();
        if (op == 'M' && op2 == 'X')
```

```
        {
            printf("%d\n",mx[root] );
        }
        else
        {
            l = FastIn() + 1; tot = FastIn();
            if (op == 'I') Insert(l, tot);
            if (op == 'D') Delete(l, tot);
            if (op == 'M') val = FastIn(), Makesame(l, tot, val);
            if (op == 'R')
                Reverse(l, tot);
            if (op == 'G')
                Query(l, tot);
        }
    }
    return 0;
}
```

5.6 Treap (ct)

```
struct Treap {
    Treap *ls, *rs;
    int size;
    bool rev;
    inline void update()
    {
        size = ls -> size + rs -> size + 1;
    }
    inline void set_rev()
    {
        rev ^= 1;
        std::swap(ls, rs);
    }
    inline void pushdown()
    {
        if (rev)
        {
            ls -> set_rev();
            rs -> set_rev();
            rev = 0;
        }
    }
} mem[maxn], *root, *null = mem;
struct Pair {
    Treap *fir, *sec;
};
Treap *build(R int l, R int r)
{
    if (l > r) return null;
    R int mid = l + r >> 1;
    R Treap *now = mem + mid;
    now -> rev = 0;
    now -> ls = build(l, mid - 1);
    now -> rs = build(mid + 1, r);
    now -> update();

    return now;
}
inline Treap *Find_kth(R Treap *now, R int k)
```

```
{
    if (!k) return mem;
    if (now -> ls -> size >= k) return Find_kth(now -> ls, k);
    else if (now -> ls -> size + 1 == k) return now;
    else return Find_kth(now -> rs, k - now -> ls -> size - 1);
}
Treap *merge(R Treap *a, R Treap *b)
{
    if (a == null) return b;
    if (b == null) return a;
    if (rand() % (a -> size + b -> size) < a -> size)
    {
        a -> pushdown();
        a -> rs = merge(a -> rs, b);
        a -> update();
        return a;
    }
    else
    {
        b -> pushdown();
        b -> ls = merge(a, b -> ls);
        b -> update();
        return b;
    }
}
Pair split(R Treap *now, R int k)
{
    if (now == null) return (Pair) {null, null};
    R Pair t = (Pair) {null, null};
    now -> pushdown();
    if (k <= now -> ls -> size)
    {
        t = split(now -> ls, k);
        now -> ls = t.sec;
        now -> update();
        t.sec = now;
    }
    else
    {
        t = split(now -> rs, k - now -> ls -> size - 1);
        now -> rs = t.fir;
        now -> update();
        t.fir = now;
    }
    return t;
}
inline void set_rev(int l, int r)
{
    R Pair x = split(root, l - 1);
    R Pair y = split(x.sec, r - l + 1);
    y.fir -> set_rev();
    root = merge(x.fir, merge(y.fir, y.sec));
}
```

5.7 可持久化平衡树 (ct)

```
char str[maxn];
struct Treap
{
```

```
    Treap *ls, *rs;
    char data; int size;
    inline void update()
    {
        size = ls -> size + rs -> size + 1;
    }
} *root[maxn], mem[maxcnt], *tot = mem, *last = mem, *null = mem;
inline Treap* new_node(char ch)
{
    *++tot = (Treap) {null, null, ch, 1};
    return tot;
}
struct Pair
{
    Treap *fir, *sec;
};
inline Treap *copy(Treap *x)
{
    if (x == null) return null;
    if(x > last) return x;
    *++tot = *x;
    return tot;
}
Pair Split(Treap *x, int k)
{
    if (x == null) return (Pair) {null, null};
    Pair y;
    Treap *nw = copy(x);
    if (nw -> ls -> size >= k)
    {
        y = Split(nw -> ls, k);
        nw -> ls = y.sec;
        nw -> update();
        y.sec = nw;
    }
    else
    {
        y = Split(nw -> rs, k - nw -> ls -> size - 1);
        nw -> rs = y.fir;
        nw -> update();
        y.fir = nw;
    }
    return y;
}
Treap *Merge(Treap *a, Treap *b)
{
    if (a == null) return b;
    if (b == null) return a;
    Treap *nw;
    if (rand() % (a -> size + b -> size) < a -> size)
    {
        nw = copy(a);
        nw -> rs = Merge(nw -> rs, b);
    }
    else
    {
        nw = copy(b);
        nw -> ls = Merge(a, nw -> ls);
    }
    nw -> update();
    return nw;
```

```
}
Treap *Build(int l, int r)
{
    if (l > r) return null;
    R int mid = l + r >> 1;
    Treap *nw = new_node(str[mid]);
    nw -> ls = Build(l, mid - 1);
    nw -> rs = Build(mid + 1, r);
    nw -> update();
    return nw;
}
int now;
inline void Insert(int k, char ch)
{
    Pair x = Split(root[now], k);
    Treap *nw = new_node(ch);
    root[++now] = Merge(Merge(x.fir, nw), x.sec);
}
inline void Del(int l, int r)
{
    Pair x = Split(root[now], l - 1);
    Pair y = Split(x.sec, r - l + 1);
    root[++now] = Merge(x.fir, y.sec);
}
inline void Copy(int l, int r, int ll)
{
    Pair x = Split(root[now], l - 1);
    Pair y = Split(x.sec, r - l + 1);
    Pair z = Split(root[now], ll);
    Treap *ans = y.fir;
    root[++now] = Merge(Merge(z.fir, ans), z.sec);
}
void Print(Treap *x, int l, int r)
{
    if (!x) return ;
    if (l > r) return;
    R int mid = x -> ls -> size + 1;
    if (r < mid)
    {
        Print(x -> ls, l, r);
        return ;
    }
    if (l > mid)
    {
        Print(x -> rs, l - mid, r - mid);
        return ;
    }
    Print(x -> ls, l, mid - 1);
    printf("%c", x -> data );
    Print(x -> rs, 1, r - mid);
}
void Printtree(Treap *x)
{
    if (!x) return;
    Printtree(x -> ls);
    printf("%c", x -> data );
    Printtree(x -> rs);
}
int main()
{
    srand(time(0) + clock());
```

```
    null -> ls = null -> rs = null; null -> size = 0; null -> data = 0;
    int n = F();
    gets(str + 1);
    int len = strlen(str + 1);
    root[0] = Build(1, len);
    while (1)
    {
        last = tot;
        R char opt = getc();
        while (opt < 'A' || opt > 'Z')
        {
            if (opt == EOF) return 0;
            opt = getc();
        }
        if (opt == 'I')
        {
            R int x = F();
            R char ch = getc();
            Insert(x, ch);
        }
        else if (opt == 'D')
        {
            R int l = F(), r = F();
            Del(l, r);
        }
        else if (opt == 'C')
        {
            R int x = F(), y = F(), z = F();
            Copy(x, y, z);
        }
        else if (opt == 'P')
        {
            R int x = F(), y = F(), z = F();
            Print(root[now - x], y, z);
            puts("");
        }
    }
    return 0;
}
```

5.8 CDQ 分治 (ct)

```
struct event
{
    int x, y, id, opt, ans;
} t[maxn], q[maxn];
void cdq(int left, int right)
{
    if (left == right) return ;
    R int mid = left + right >> 1;
    cdq(left, mid);
    cdq(mid + 1, right);
    //分成若干个子问题
    ++now;
    for (int i = left, j = mid + 1; j <= right; ++j)
    {
        for (; i <= mid && q[i].x <= q[j].x; ++i)
            if (!q[i].opt)
                add(q[i].y, q[i].ans);
```

```
        //考虑前面的修改操作对后面的询问的影响
        if (q[j].opt)
            q[j].ans += query(q[j].y);
    }
    R int i, j, k = 0;
    //以下相当于归并排序
    for (i = left, j = mid + 1; i <= mid && j <= right; )
    {
        if (q[i].x <= q[j].x)
            t[k++] = q[i++];
        else
            t[k++] = q[j++];
    }
    for (; i <= mid; )
        t[k++] = q[i++];
    for (; j <= right; )
        t[k++] = q[j++];
    for (int i = 0; i < k; ++i)
        q[left + i] = t[i];
}
```

5.9 Bitset (ct)

```
namespace Game {
#define maxn 300010
#define maxs 30010
uint b1[32][maxs], b2[32][maxs];
int popcnt[256];
inline void set(R uint *s, R int pos)
{
    s[pos >> 5] |= 1u << (pos & 31);
}
inline int popcount(R uint x)
{
    return popcnt[x >> 24 & 255]
         + popcnt[x >> 16 & 255]
         + popcnt[x >> 8  & 255]
         + popcnt[x       & 255];
}
void main() {
    int n, q;
    scanf("%d%d", &n, &q);

    char *s1 = new char[n + 1];
    char *s2 = new char[n + 1];
    scanf("%s%s", s1, s2);

    uint *anss = new uint[q];

    for (R int i = 1; i < 256; ++i) popcnt[i] = popcnt[i >> 1] + (i & 1);

    #define modify(x, _p)\
    {\
        for (R int j = 0; j < 32 && j <= _p; ++j)\
            set(b##x[j], _p - j);\
    }
    for (R int i = 0; i < n; ++i)
        if (s1[i] == '0') modify(1, 3 * i)
        else if (s1[i] == '1') modify(1, 3 * i + 1)
```

```
        else modify(1, 3 * i + 2)

    for (R int i = 0; i < n; ++i)
        if (s2[i] == '1') modify(2, 3 * i)
        else if (s2[i] == '2') modify(2, 3 * i + 1)
        else modify(2, 3 * i + 2)

    for (int Q = 0; Q < q; ++Q) {
        R int x, y, l;
        scanf("%d%d%d", &x, &y, &l); x *= 3; y *= 3; l *= 3;
        uint *f1 = b1[x & 31], *f2 = b2[y & 31], ans = 0;
        R int i = x >> 5, j = y >> 5, p, lim;
        for (p = 0, lim = l >> 5; p + 8 < lim; p += 8, i += 8, j += 8)
        {
            ans += popcount(f1[i + 0] & f2[j + 0]);
            ans += popcount(f1[i + 1] & f2[j + 1]);
            ans += popcount(f1[i + 2] & f2[j + 2]);
            ans += popcount(f1[i + 3] & f2[j + 3]);
            ans += popcount(f1[i + 4] & f2[j + 4]);
            ans += popcount(f1[i + 5] & f2[j + 5]);
            ans += popcount(f1[i + 6] & f2[j + 6]);
            ans += popcount(f1[i + 7] & f2[j + 7]);
        }
        for (; p < lim; ++p, ++i, ++j) ans += popcount(f1[i] & f2[j]);
        R uint S = (1u << (l & 31)) - 1;
        ans += popcount(f1[i] & f2[j] & S);
        anss[Q] = ans;
    }

    output_arr(anss, q * sizeof(uint));
}
}
```

Chapter 6

Others

6.1 vimrc (gy)

```
se et ts=4 sw=4 sts=4 nu sc sm lbr is hls mouse=a
sy on
ino <tab> <c-n>
ino <s-tab> <tab>
au bufwinenter * winc L

nm <f6> ggVG"+y
nm <f7> :w<cr>:make<cr>
nm <f8> :!@<cr>
nm <f9> :!@ < in<cr>
nm <s-f9> :!(time @ < in &> out) &>> out<cr>:sp out<cr>

au filetype cpp cm @ ./a.out | se cin fdm=syntax mp=g++\ %\ -std=c++11\ -Wall\ -Wextra\ -Wconversion\ -O2

map <c-p> :ha<cr>
se pheader=%N@%F popt=number:y

au filetype java cm @ java %< | se cin fdm=syntax mp=javac\ %
au filetype python cm @ python % | se si fdm=indent
au bufenter *.kt setf kotlin
au filetype kotlin cm @ kotlin _%<Kt | se si mp=kotlinc\ %
```

6.2 STL 释放内存 (Durandal)

```
template <typename T>
__inline void clear(T &container) {
    container.clear();
    T(container).swap(container);
}
```

6.3 开栈 (Durandal)

```
register char *_sp __asm__("rsp");
int main() {
    const int size = 400 << 20; // 400 MB
    static char *sys, *mine(new char[size] + size - 4096);
    sys = _sp; _sp = mine;
    _main(); // main method
```

```
    _sp = sys;
    return 0;
}
```

6.4 O3 (gy)

```
__attribute__((optimize("-O3"))) int main() { return 0; }
```

6.5 Java Template (gy)

```java
import java.io.*;
import java.math.*;
import java.util.*;

public class Template {
    // Input
    private static BufferedReader reader;
    private static StringTokenizer tokenizer;
    private static String next() {
        try {
            while (tokenizer == null || !tokenizer.hasMoreTokens())
                tokenizer = new StringTokenizer(reader.readLine());
        } catch (IOException e) {
            // do nothing
        }
        return tokenizer.nextToken();
    }
    private static int nextInt() {
        return Integer.parseInt(next());
    }
    private static double nextDouble() {
        return Double.parseDouble(next());
    }
    private static BigInteger nextBigInteger() {
        return new BigInteger(next());
    }

    public static void main(String[] args) {
        reader = new BufferedReader(new InputStreamReader(System.in));
        Scanner scanner = new Scanner(System.in);
        while (scanner.hasNext())
            scanner.next();
    }

    // BigInteger & BigDecimal
    private static void bigDecimal() {
        BigDecimal a = BigDecimal.valueOf(1.0);
        BigDecimal b = a.setScale(50, RoundingMode.HALF_EVEN);
        BigDecimal c = b.abs();
        // if scale omitted, b.scale is used
        BigDecimal d = c.divide(b, 50, RoundingMode.HALF_EVEN);
        // since Java 9
        BigDecimal e = d.sqrt(new MathContext(50, RoundingMode.HALF_EVEN));
        BigDecimal x = new BigDecimal(BigInteger.ZERO);
        BigInteger y = BigDecimal.ZERO.toBigInteger(); // RoundingMode.DOWN
        y = BigDecimal.ZERO.setScale(0, RoundingMode.HALF_EVEN).unscaledValue();
    }
    // sqrt for Java 8
```

```java
    // can solve scale=100 for 10000 times in about 1 second
    private static BigDecimal sqrt(BigDecimal a, int scale) {
        if (a.compareTo(BigDecimal.ZERO) < 0)
            return BigDecimal.ZERO.setScale(scale, RoundingMode.HALF_EVEN);
        int length = a.precision() - a.scale();
        BigDecimal ret = new BigDecimal(BigInteger.ONE, -length / 2);
        for (int i = 1; i <= Integer.highestOneBit(scale) + 10; i++)
            ret = ret.add(a.divide(ret, scale,
              ↪ RoundingMode.HALF_EVEN)).divide(BigDecimal.valueOf(2), scale,
              ↪ RoundingMode.HALF_EVEN);
        return ret;
    }
    // can solve a=2^10000 for 100000 times in about 1 second
    private static BigInteger sqrt(BigInteger a) {
        int length = a.bitLength() - 1;
        BigInteger l = BigInteger.ZERO.setBit(length / 2), r = BigInteger.ZERO.setBit(length / 2);
        while (!l.equals(r)) {
            BigInteger m = l.add(r).shiftRight(1);
            if (m.multiply(m).compareTo(a) < 0)
                l = m.add(BigInteger.ONE);
            else
                r = m;
        }
        return l;
    }

    // Collections
    private static void arrayList() {
        List<Integer> list = new ArrayList<>();
        // Generic array is banned
        List[] lists = new List[100];
        lists[0] = new ArrayList<Integer>();
        // for List<Integer>, remove(Integer) stands for element, while remove(int) stands for
          ↪ index
        list.remove(list.get(1));
        list.remove(list.size() - 1);
        list.clear();
        Queue<Integer> queue = new LinkedList<>();
        // return the value without popping
        queue.peek();
        // pop and return the value
        queue.poll();
        Queue<Integer> priorityQueue = new PriorityQueue<>();
        Deque<Integer> deque = new ArrayDeque<>();
        deque.peekFirst();
        deque.peekLast();
        deque.pollFirst();
        TreeSet<Integer> set = new TreeSet<>();
        TreeSet<Integer> anotherSet = new TreeSet<>(Comparator.reverseOrder());
        set.ceiling(1);
        set.floor(1);
        set.lower(1);
        set.higher(1);
        set.contains(1);
        HashSet<Integer> hashSet = new HashSet<>();
        HashMap<String, Integer> map = new HashMap<>();
        map.put("", 1);
        map.get("");
        map.forEach((string, integer) -> System.out.println(string + integer));
        TreeMap<String, Integer> treeMap = new TreeMap<>();
        Arrays.sort(new int[10]);
```

```
        Arrays.sort(new Integer[10], (a, b) -> {
            if (a.equals(b)) return 0;
            if (a > b) return -1;
            return 1;
        });
        Arrays.sort(new Integer[10], Comparator.comparingInt((a) -> (int) a).reversed());
        long a = 1_000_000_000_000_000_000L;
        int b = Integer.MAX_VALUE;
        int c = 'a';
    }
}
```

6.6 Big Fraction (gy)

```
fun gcd(a: Long, b: Long): Long = if (b == 0L) a else gcd(b, a % b)

class Fraction(val a: BigInteger, val b: BigInteger) {
    constructor(a: Long, b: Long) : this(BigInteger.valueOf(a / gcd(a, b)), BigInteger.valueOf(b /
      ↪ gcd(a, b)))

    operator fun plus(o: Fraction): Fraction {
        var gcd = b.gcd(o.b)
        val tempProduct = (b / gcd) * (o.b / gcd)
        var ansA = a * (o.b / gcd) + o.a * (b / gcd)
        val gcd2 = ansA.gcd(gcd)
        ansA /= gcd2
        gcd /= gcd2
        return Fraction(ansA, gcd * tempProduct)
    }

    operator fun minus(o: Fraction): Fraction {
        var gcd = b.gcd(o.b)
        val tempProduct = (b / gcd) * (o.b / gcd)
        var ansA = a * (o.b / gcd) - o.a * (b / gcd)
        val gcd2 = ansA.gcd(gcd)
        ansA /= gcd2
        gcd /= gcd2
        return Fraction(ansA, gcd * tempProduct)
    }

    operator fun times(o: Fraction): Fraction {
        val gcd1 = a.gcd(o.b)
        val gcd2 = b.gcd(o.a)
        return Fraction((a / gcd1) * (o.a / gcd2), (b / gcd2) * (o.b / gcd1))
    }
}
```

6.7 模拟退火 (ct)

```
db ans_x, fans;
inline double rand01() {return rand() / 2147483647.0;}
inline double randp() {return (rand() & 1 ? 1 : -1) * rand01();}
inline double f(double x)
{
    /*
        write your function here.
    */
    if (maxx < fans) {fans = maxx; ans_x = x;}
```

```
    return maxx;
}
int main()
{
    srand(time(NULL) + clock());
    db x = 0, fnow = f(x);
    fans = 1e30;
    for (db T = 1e4; T > 1e-4; T *= 0.997)
    {
        db nx = x + randp() * T, fnext = f(nx);
        db delta = fnext - fnow;
        if (delta < 1e-9 || exp(-delta / T) > rand01())
        {
            x = nx;
            fnow = fnext;
        }
    }
    return 0;
}
```

6.8 三分 (ct)

```
inline db cubic_search()
{
    double l = -1e4, r = 1e4;
    for (int i = 1; i <= 100; ++i)
    {
        double ll = (l + r) * 0.5;
        double rr = (ll + r) * 0.5;
        if (check(ll) < check(rr)) r = rr;
        else l = ll;
    }
    return (l + r) * 0.5;
}
```

6.9 Zeller Congruence (gy)

```
int day_in_week(int year, int month, int day) {
    if (month == 1 || month == 2)
        month += 12, year--;
    int c = year / 100, y = year % 100, m = month, d = day;
    int ret = (y + y / 4 + c / 4 + 5 * c + 13 * (m + 1) / 5 + d + 6) % 7;
    return ret >= 0 ? ret : ret + 7;
}
```

6.10 博弈论模型 (gy)

- Wythoff's game
  给定两堆石子，每次可以从任意一堆中取至少一个石子，或从两堆中取相同的至少一个石子，取走最后石子的胜
  先手胜当且仅当石子数满足：
  $\lfloor (b-a) \times \phi \rfloor = a, (a \le b, \phi = \frac{\sqrt{5}+1}{2})$
  先手胜对应的石子数构成两个序列：
  Lower Wythoff sequence: $a_n = \lfloor n \times \phi \rfloor$
  Upper Wythoff sequence: $b_n = \lfloor n \times \phi^2 \rfloor$

- Fibonacci nim
  给定一堆石子，第一次可以取至少一个、少于石子总数数量的石子，之后每次可以取至少一个、不超过上次取石子数量两倍的石子，取走最后石子的胜
  先手胜当且仅当石子数为斐波那契数

6.11 积分表

- $\sin x \to -\cos x$
- $\cos x \to \sin x$
- $\tan x \to -\ln\cos x$
- $\sec x \to \ln\left|\sin\frac{x}{2}+\cos\frac{x}{2}\right| - \ln\left|-\sin\frac{x}{2}+\cos\frac{x}{2}\right|$
- $\csc x \to \ln\left|\sin\frac{x}{2}\right| - \ln\left|\cos\frac{x}{2}\right|$
- $\sin^2 x \to \frac{x}{2} - \frac{1}{2}\sin x\cos x$
- $\cos^2 x \to \frac{x}{2} + \frac{1}{2}\sin x\cos x$
- $\tan^2 x \to \tan x - x$
- $\sec^2 x \to \tan x$
- $\csc^2 x \to -\tan x$
- $\arcsin x \to \frac{1}{\sqrt{1-x^2}}$
- $\arccos x \to -\frac{1}{\sqrt{1-x^2}}$
- $\arctan x \to \frac{1}{1+x^2}$
- $a^x \to \frac{a^x}{\ln a}$
- $\frac{1}{x^2+a^2} \to \frac{1}{|a|}\arctan\frac{x}{|a|}$
- $\frac{1}{x^2-a^2} \to \frac{1}{2}\ln|x-a| - \frac{1}{2}\ln|x+a|$
- $\frac{x}{ax+b} \to \frac{x}{a} - \frac{b}{a^2}\ln|ax+b|$
- $\frac{x}{ax^2+c} \to \frac{1}{2a}\ln\left|ax^2+c\right|$
- $\sqrt{c+x^2} \to \frac{x}{2}\sqrt{c+x^2} + \frac{c}{2}\ln\left|x+\sqrt{c+x^2}\right|$
- $\sqrt{c-x^2} \to \frac{x}{2}\sqrt{c-x^2} + \frac{c}{2}\arctan\frac{x}{\sqrt{c-x^2}}$
- $\frac{1}{\sqrt{c+x^2}} \to \ln\left|x+\sqrt{c+x^2}\right|$
- $\frac{1}{\sqrt{c-x^2}} \to \arctan\frac{x}{\sqrt{c-x^2}}$

6.12 公式、数列、定理

- **求和公式**
  - $\sum\limits_{k=1}^{n}(2k-1)^2 = \frac{1}{3}n(4n^2-1)$
  - $\sum\limits_{k=1}^{n}k^3 = \frac{1}{4}n^2(n+1)^2$

- $\sum\limits_{k=1}^{n}(2k-1)^3 = n^2(2n^2-1)$
- $\sum\limits_{k=1}^{n}k^4 = \frac{1}{30}n(n+1)(2n+1)(3n^2+3m-1)$
- $\sum\limits_{k=1}^{n}k^5 = \frac{1}{12}n^2(n+1)^2(2n^2+2n-1)$
- $\sum\limits_{k=1}^{n}k(k+1) = \frac{1}{3}n(n+1)(n+2)$
- $\sum\limits_{k=1}^{n}k(k+1)(k+2) = \frac{1}{4}n(n+1)(n+2)(n+3)$
- $\sum\limits_{k=1}^{n}k(k+1)(k+2)(k+3) = \frac{1}{5}n(n+1)(n+2)(n+3)(n+4)$

- **错排公式**
  D_n 表示 n 个元素错位排列的方案数
  $D_1 = 0, D_2 = 1$
  $D_n = (n-1)(D_{n-2}+D_{n-1}), n \geq 3$
  $D_n = n! \cdot (1 - \frac{1}{1!} + \frac{1}{2!} - \cdots + (-1)^n \frac{1}{n!})$

- **Fibonacci sequence**
  $F_0 = 0, F_1 = 1$
  $F_n = F_{n-1} + F_{n-2}$
  $F_{n+1} \cdot F_{n-1} - F_n^2 = (-1)^n$
  $F_{-n} = (-1)^n F_n$
  $F_{n+k} = F_k \cdot F_{n+1} + F_{k-1} \cdot F_n$
  $\gcd(F_m, F_n) = F_{\gcd(m,n)}$
  $F_m \mid F_n^2 \Leftrightarrow nF_n \mid m$
  $F_n = \frac{\varphi^n - \Psi^n}{\sqrt{5}}, \varphi = \frac{1+\sqrt{5}}{2}, \Psi = \frac{1-\sqrt{5}}{2}$
  $F_n = \lfloor \frac{\varphi^n}{\sqrt{5}} + \frac{1}{2} \rfloor, n \geq 0$
  $n(F) = \lfloor \log_{\varphi}(F \cdot \sqrt{5} + \frac{1}{2}) \rfloor$

- **第一类 Stirling number**
  用 $s(n,k) = (-1)^{n-k}\left[{n \atop k}\right]$ 表示第一类 Stirling number
  $\left[{n+1 \atop k}\right] = n\left[{n \atop k}\right] + \left[{n \atop k-1}\right], k > 0$
  $\left[{0 \atop 0}\right] = 1, \left[{n \atop 0}\right] = \left[{0 \atop n}\right] = 0, n > 0$
  $\left[{n \atop k}\right]$ 为将 n 个元素分成 k 个环的方案数

- **第二类 Stirling number**
  用 $S(n,k) = \left\{{n \atop k}\right\}$ 表示第二类 Stirling number
  $\left\{{n+1 \atop k}\right\} = k\left\{{n \atop k}\right\} + \left\{{n \atop k-1}\right\}, k > 0$
  $\left\{{0 \atop 0}\right\} = 1, \left\{{n \atop 0}\right\} = \left\{{0 \atop n}\right\} = 0, n > 0$
  $\left\{{n \atop k}\right\} = \frac{1}{k!}\sum\limits_{j=0}^{k}(-1)^{k-j}\binom{k}{j}j^n$
  $\left\{{n \atop k}\right\}$ 为将 n 个元素划分成 k 个非空集合的方案数

- **Catalan number**
  c_n 表示长度为 $2n$ 的合法括号序的数量
  $c_1 = 1, c_{n+1} = \sum\limits_{i=1}^{n} c_i \times c_{n+1-i}$
  $c_n = \frac{\binom{2n}{n}}{n+1}$

- **Bell number**
  B_n 表示基数为 n 的集合的划分方案数

$$B_i = \begin{cases} 1 & i = 0 \\ \sum\limits_{k=0}^{n} \binom{n}{k} B_k & i > 0 \end{cases}$$
$$B_n = \sum_{k=0}^{n} \left\{ {n \atop k} \right\}$$

- **五边形数定理**
  $p(n)$ 表示将 n 划分为若干个正整数之和的方案数
  $p(n) = \sum\limits_{k \in \mathbb{N}^*} (-1)^{k-1} p(n - \frac{k(3k-1)}{2})$

- **Bernoulli number**
  $\sum\limits_{j=0}^{m} \binom{m+1}{j} B_j = 0, m > 0$
  $$B_i = \begin{cases} 1 & i = 0 \\ -\frac{\sum\limits_{j=0}^{i-1} \binom{i+1}{j}}{i+1} & i > 0 \end{cases}$$
  $\sum\limits_{k=1}^{n} k^m = \frac{1}{m+1} \sum\limits_{k=0}^{m} \binom{m+1}{k} B_k n^{m+1-k}$

- **Möbius function**
  $$\mu(n) = \begin{cases} 1 & n \text{ is a square-free positive integer with an even number of prime factors} \\ -1 & n \text{ is a square-free positive integer with an odd number of prime factors} \\ 0 & n \text{ has a squared prime factor} \end{cases}$$
  $$\sum_{d|n} \mu(d) = \begin{cases} 1 & n = 1 \\ 0 & n > 1 \end{cases}$$
  $g(n) = \sum\limits_{d|n} f(d) \Leftrightarrow f(n) = \sum\limits_{d|n} \mu(d) g(\frac{n}{d})$

- **Lagrange polynomial**
  给定次数为 n 的多项式函数 $L(x)$ 上的 $n+1$ 个点 $(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$
  则 $L(x) = \sum\limits_{j=0}^{n} y_j \prod\limits_{0 \le m \le n, m \ne j} \frac{x - x_m}{x_j - x_m}$

- **树的计数**
  - **有根树计数**
    $a_1 = 1$
    $a_{n+1} = \frac{\sum\limits_{j=1}^{n} j \cdot a_j \cdot S_{n,j}}{n}$
    $S_{n,j} = \sum\limits_{i=1}^{n/j} a_{n+1-ij} = S_{n-j,j} + a_{n+1-j}$
  - **无根树计数**
    $$\begin{cases} a_n - \sum\limits_{i=1}^{n/2} a_i a_{n-i} & n \text{ is odd} \\ a_n - \sum\limits_{i=1}^{n/2} a_i a_{n-i} + \frac{1}{2} a_{\frac{n}{2}} (a_{\frac{n}{2}} + 1) & n \text{ is even} \end{cases}$$
  - **完全图生成树计数**
    n^{n-2}
  - **矩阵-树定理**
    设 $\mathbf{A}[G]$ 为图 G 的邻接矩阵、$\mathbf{D}[G]$ 为图 G 的度数矩阵，则图 G 的不同生成树的个数为 $\mathbf{C}[G] = \mathbf{D}[G] - \mathbf{A}[G]$ 的任意一个 $n-1$ 阶主子式的行列式值。

- **Euler characteristic**
  平面图的顶点个数 V，边数 E，平面被划分的区域数 F，组成图形的连通部分的数目 C 满足:
  $V - E + F = C + 1$

- **Pick theorem**
  顶点为整点的简单多边形，其面积 A，内部格点数 i，边上格点数 b 满足：
  $A = i + \frac{b}{2} - 1$

- **平面几何公式**

  - **三角形**
    半周长 $p = \frac{a+b+c}{2}$
    面积 $S = \frac{1}{2}aH_a = \frac{1}{2}ab \cdot \sin C = \sqrt{p(p-a)(p-b)(p-c)} = pr = \frac{abc}{4R}$
    中线长 $M_a = \frac{1}{2}\sqrt{2(b^2+c^2)-a^2} = \frac{1}{2}\sqrt{b^2+c^2+2bc\cdot\cos A}$
    角平分线长 $T_a = \frac{\sqrt{bc((b+c)^2-a^2)}}{b+c} = \frac{2bc}{b+c}\cos\frac{A}{2}$
    高 $H_a = b\sin C = \sqrt{b^2 - (\frac{a^2+b^2-c^2}{2a})^2}$
    内切圆半径 $r = \frac{S}{p} = 4R\sin\frac{A}{2}\sin\frac{B}{2}\sin\frac{C}{2} = \sqrt{\frac{(p-a)(p-b)(p-c)}{p}} = p\tan\frac{A}{2}\tan\frac{B}{2}\tan\frac{C}{2}$
    外接圆半径 $R = \frac{abc}{4S} = \frac{a}{2\sin A}$
    旁切圆半径 $r_A = \frac{2S}{-a+b+c}$
    重心 $\left(\frac{x_1+x_2+x_3}{3}, \frac{y_1+y_2+y_3}{3}\right)$
    外心 $\left(\frac{\begin{vmatrix} x_1^2+y_1^2 & y_1 & 1 \\ x_2^2+y_2^2 & y_2 & 1 \\ x_3^2+y_3^2 & y_3 & 1 \end{vmatrix}}{2\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}, \frac{\begin{vmatrix} x_1 & x_1^2+y_1^2 & 1 \\ x_2 & x_2^2+y_2^2 & 1 \\ x_3 & x_3^2+y_3^2 & 1 \end{vmatrix}}{2\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}\right)$
    内心 $\left(\frac{ax_1+bx_2+cx_3}{a+b+c}, \frac{ay_1+by_2+cy_3}{a+b+c}\right)$
    垂心 $\left(\frac{\begin{vmatrix} x_2x_3+y_2y_3 & 1 & y_1 \\ x_3x_1+y_3y_1 & 1 & y_2 \\ x_1x_2+y_1y_2 & 1 & y_3 \end{vmatrix}}{2\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}, \frac{\begin{vmatrix} x_2x_3+y_2y_3 & x_1 & 1 \\ x_3x_1+y_3y_1 & x_2 & 1 \\ x_1x_2+y_1y_2 & x_3 & 1 \end{vmatrix}}{2\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}}\right)$
    旁心 $\left(\frac{-ax_1+bx_2+cx_3}{-a+b+c}, \frac{-ay_1+by_2+cy_3}{-a+b+c}\right)$