ICPC Template

墨染空

2024 年 11 月 8 日

目录

1 Default **3**

2 图论 **4**

2.1 Hall 定理 4
2.2 矩阵树定理 4
2.3 欧拉回路计数 4
2.4 树上路径交 4
2.5 圆方树 5
2.6 有向图 Tarjan 6
2.7 欧拉回路 6
2.8 最大流 7
2.9 Prufer 9
2.10 长链剖分 9
2.11 最小费用最大流 10
2.12 KM 12
2.13 有负圈/上下界费用流 13
2.14 虚树 15
2.15 重链剖分 + LCA 16
2.16 匈牙利 17
2.17 上下界网络流 18
2.18 $O(1)$ LCA 20
2.19 点分治 21

3 Poly 多项式 **22**

3.1 1e18 多项式乘法 22

3.2 正常多项式 + 线性递推 24

4 字符串 **27**

4.1 AC 自动机 27

4.2 KMP 27

4.3 Manacher 28

4.4 SA 29

4.5 哈希 Hash 30

4.6 最小表示法 30

4.7 Z 函数 30

4.8 SAM 31

4.9 广义 SAM 32

4.10 回文自动机 33

5 数学 **34**

5.1 Min-Max 容斥 34

5.2 单位根反演 34

5.3 积分表 34

5.4 扩域 46

5.5 原根 47

5.6 $O(n)$ 预处理逆元 49

5.7 Exgcd 扩展欧里几得 49

5.8 扩展中国剩余定理 exCRT 50

5.9 BSGS 51

5.10 杜教筛 52

5.11 Min25 52

5.12 FMT / FWT 55

5.13 子集卷积 56

6 数据结构 **56**

6.1 线段树合并与分裂 56

6.2 ST 表 58

6.3 Fhq Treap 59

6.4 线段树 60
6.5 主席树 61
6.6 树状数组：区间加区间求和 62
6.7 LCT 63
6.8 左偏树 65
6.9 李超树 65
6.10 回滚莫队 66
6.11 动态凸包 68
6.12 珂朵莉树 69
6.13 HashMap 70
6.14 全局平衡二叉树 71

7 计算几何 74

7.1 Basic 74
7.2 点到线段距离 75
7.3 线段交 75
7.4 凸包 76
7.5 半平面交 76
7.6 最小圆覆盖 77
7.7 自适应辛普森积分 78
7.8 极角排序 78
7.9 Int 下凸包 + 闵可夫斯基和 79

1 Default

```
// Skyqwq
#include <bits/stdc++.h>

#define pb push_back
#define fi first
#define se second
#define mp make_pair

using namespace std;

```

```
typedef pair<int, int> PII;
typedef long long LL;

template <typename T> bool chkMax(T &x, T y) { return (y > x) ? x = y, 1 : 0; }
template <typename T> bool chkMin(T &x, T y) { return (y < x) ? x = y, 1 : 0; }

template <typename T> void inline read(T &x) {
    int f = 1; x = 0; char s = getchar();
    while (s < '0' || s > '9') { if (s == '-') f = -1; s = getchar(); }
    while (s <= '9' && s >= '0') x = x * 10 + (s ^ 48), s = getchar();
}

int main() {

    return 0;
}
```

2 图论

2.1 Hall 定理

完美匹配：集合 $\leq$ 邻域的并

最大匹配：总数 - max(集合 -邻域并)

2.2 矩阵树定理

度数 - 边

2.3 欧拉回路计数

Best 定理

内向生成树个数 $\prod(out_i - 1)$

2.4 树上路径交

从 $lca(u, x), lca(u, y), lca(v, x), lca(v, y)$ 四个点找深度最大的两个点，记为 p_1, p_2。

* 若 $p_1 = p_2$ 且 $dep_{p1} < \max(dep_{lca(x,y)}, dep_{lca(u,v)})$ 那么无相交路径 * 否则相交路径就是 p_1 到 p_2

```cpp

// 树上路径交

PII inline query(int u, int v, int x, int y) {
    int p[4] = { lca(u, x), lca(u, y), lca(v, x), lca(v, y)};
    int w = lca(u, v), z = lca(x, y);
    int p1 = 0, p2 = 0;
    for (int i = 0; i < 4; i++)
        if (dep[p[i]] > dep[p1]) p2 = p1, p1 = p[i];
        else if (dep[p[i]] > dep[p2]) p2 = p[i];
    if (p1 == p2 && (dep[p1] < dep[w] || dep[p1] < dep[z])) return mp(-1, -1);
    // p1 - p2 是子路径
    return mp(p1, p2);
}
```

2.5 圆方树

```cpp
// 圆方树
int dfn[N], low[N], dfncnt, cnt;

int s[N], top;

void inline Add(int x, int y) {
    g[x].pb(y), g[y].pb(x);
}

void tarjan(int u, int fa) {
    dfn[u] = low[u] = ++dfncnt;
    s[++top] = u;
    for (int v: e[u]) {
        if (v == fa) continue;
        if (!dfn[v]) {
            tarjan(v, u);
            chkMin(low[u], low[v]);
            if (low[v] >= dfn[u]) {
                int y; ++cnt;
                do {
                    y = s[top--], Add(y, cnt);
                } while (y != v);
```

```
                Add(cnt, u);
            }
        } else {
            chkMin(low[u], dfn[v]);
        }
    }
}
```

2.6 有向图 Tarjan

```


// 有向图 tarjan
void tarjan(int u) {
    dfn[u] = low[u] = ++dfncnt;
    s[++top] = u, ins[u] = true;
    for (int i = head[u]; i; i = e[i].next) {
        int v = e[i].v;
        if (!dfn[v]) {
            tarjan(v), low[u] = min(low[u], low[v]);
        } else if (ins[v]) low[u] = min(low[u], dfn[v]);
    }
    if (low[u] == dfn[u]) {
        int v; ++cnt;
        do {
            v = s[top--], ins[v] = false, col[v] = cnt;
        } while (v != u);
    }
}
```

2.7 欧拉回路

```
// 欧拉回路
void dfs(int u) {
    for (int &i = head[u]; i; ) {
        int v = e[i].v;
        if(vis[i]) {
            i = e[i].next;
```

```
            continue;
        }

        vis[i] = true;
        if(t == 1) vis[i ^ 1] = true;

        i = e[i].next;
        dfs(v);
    }
}
```

2.8 最大流

```
// 最大流
namespace MF{
    int n, m, s, t, pre[N], cur[N], q[N];
    LL res, maxflow, d[N];
    int head[N], numE = 1;
    struct E{
        int next, v, w;
    } e[M << 1];

    void inline add(int u, int v, int w) {
        e[++numE] = (E) { head[u], v, w };
        head[u] = numE;
    }
    void inline init(int v, int a, int b) {
        for (int i = 1; i <= n; i++) head[i] = 0;
        numE = 1;
        n = v, s = a, t = b;
    }

    bool inline bfs() {
        int hh = 0, tt = -1;
        for (int i = 1; i <= n; i++) d[i] = 0;
        q[++tt] = s, d[s] = 1, cur[s] = head[s];
        while (hh <= tt) {
            int u = q[hh++];
            for (int i = head[u]; i; i = e[i].next) {
```

```
            int v = e[i].v;
            if (!d[v] && e[i].w) {
                cur[v] = head[v];
                q[++tt]= v, d[v] = d[u] + 1;
                if (v == t) return 1;
            }
        }
    }
    return 0;
}
LL dinic(int u, LL flow) {
    if (u == t) return flow;
    LL rest = flow;
    for (int i = cur[u]; i && rest; i = e[i].next) {
        cur[u] = i;
        int v = e[i].v;
        if (e[i].w&& d[v] == d[u] + 1) {
            int k = dinic(v, min((LL)e[i].w, rest));
            if (!k) d[v] = 0;
            rest -= k, e[i].w -= k,e[i ^ 1].w += k;
        }
    }
    return flow - rest;
}
void inline addE(int u, int v, int w) {
    add(u, v, w), add(v, u, 0);
}
LL inline work() {
    maxflow = 0;
    while (bfs())
        while (res = dinic(s, INF)) maxflow += res;
    return maxflow;
}
// Find min-cut
bool vis[N];

void dfs(int u) {
    //cerr << u << " dfs\n";
    vis[u] = 1;
```

```
        for (int i = head[u]; i; i = e[i].next) {
            int v = e[i].v;
            if (!vis[v] && e[i].w) dfs(v);
        }
    }

    void minCut() {
        for (int i = 1; i <= n; i++) vis[i] = 0;
        dfs(s);
    }
}
```

2.9 Prufer

```
void inline fToP() {
        for (int i = 1; i < n; i++) d[f[i]]++;
        for (int i = 1, j = 1; i <= n - 2; j++) {
                while (d[j]) j++;
                p[i++] = f[j];
                while (i <= n - 2 && --d[p[i - 1]] == 0 && p[i - 1] < j) p[i++] = f[p[i - 1]];
        }
}

void inline pToF() {
    for (int i = 1; i <= n - 2; i++) d[p[i]]++;
    p[n - 1] = n;
    for (int i = 1, j = 1; i < n; i++, j++) {
        while (d[j]) j++;
        f[j] = p[i];
        while (i < n - 1 && --d[p[i]] == 0 && p[i] < j) f[p[i]] = p[i + 1], ++i;
    }
}
```

2.10 长链剖分

```
int d[N], dep[N];
int g[N], son[N], fa[N][L], top[N];
LL res;
```

```cpp
vector<int> U[N], D[N];
void dfs1(int u) {
    dep[u] = d[u] = d[fa[u][0]] + 1;
    for (int i = 1; fa[u][i - 1]; i++) fa[u][i] = fa[fa[u][i - 1]][i - 1];
    for (int i = head[u]; i; i = e[i].next) {
        int v = e[i].v;
        dfs1(v);
        if (dep[v] > dep[u]) dep[u] = dep[v], son[u] = v;
    }
}

void dfs2(int u, int tp) {
    top[u] = tp;
    if (u == tp) {
        for (int x = u, i = 0; i <= dep[u] - d[u]; i++)
            U[u].push_back(x), x = fa[x][0];
        for (int x = u, i = 0; i <= dep[u] - d[u]; i++)
            D[u].push_back(x), x = son[x];
    }
    if (son[u]) dfs2(son[u], tp);
    for (int i = head[u]; i; i = e[i].next) {
        int v = e[i].v;
        if (v != son[u]) dfs2(v, v);
    }
}

int inline query(int x, int k) {
    if (!k) return x;
    x = fa[x][g[k]], k -= (1 << g[k]) + d[x] - d[top[x]], x = top[x];
    return k < 0 ? D[x][-k] : U[x][k];
}
```

2.11 最小费用最大流

```cpp
const int N = ?, M = ?;
const int INF = 0x3f3f3f3f;
int n, m, s, t, maxflow, cost, d[N], incf[N], pre[N];
int q[N];
int head[N], numE = 1;
```

```
bool vis[N];

struct E{
    int next, v, w, c;
} e[M];

void inline add(int u, int v, int w, int c) {
    e[++numE] = (E) { head[u], v, w, c };
    head[u] = numE;
}

// Spfa ||
bool spfa() {
    memset(vis, false, sizeof vis);
    memset(d, 0x3f, sizeof d);
    int hh = 0, tt = 1;
    q[0] = s; d[s] = 0; incf[s] = 2e9;
    while (hh != tt) {
        int u = q[hh++]; vis[u] = false;
        if (hh == N) hh = 0;
        for (int i = head[u]; i; i = e[i].next) {
            int v = e[i].v;
            if (e[i].w && d[u] + e[i].c < d[v]) {
                d[v] = d[u] + e[i].c;
                pre[v] = i;
                incf[v] = min(incf[u], e[i].w);
                if (!vis[v]) {
                    q[tt++] = v;
                    vis[v] = true;
                    if (tt == N) tt = 0;
                }
            }
        }
    }
    return d[t] != INF;
}

void update() {
```

```
    int x = t;
    while (x != s) {
        int i = pre[x];
        e[i].w -= incf[t], e[i ^ 1].w += incf[t];
        x = e[i ^ 1].v;
    }
    maxflow += incf[t];
    cost += d[t] * incf[t];
}
```

2.12 KM

```
namespace KM{
    int n, va[N], vb[N], match[N], last[N];
    LL a[N], b[N], upd[N], w[N][N];
    bool dfs(int u, int fa) {
        va[u] = 1;
        for (int v = 1; v <= n; v++) {
            if (vb[v]) continue;
            if (a[u] + b[v] == w[u][v]) {
                vb[v] = 1, last[v] = fa;
                if (!match[v] || dfs(match[v], v)) {
                    match[v] = u; return true;
                }
            } else if (a[u] + b[v] - w[u][v] < upd[v])
                upd[v] = a[u] + b[v] - w[u][v], last[v] = fa;
        }
        return false;
    }
    void inline calc(int len, LL d[N][N]) {
        n = len;
        for (int i = 1; i <= n; i++)
            for (int j = 1; j <= n; j++) w[i][j] = d[i][j];
        for (int i = 1; i <= n; i++) {
            a[i] = -1e18, b[i] = 0;
            for (int j = 1; j <= n; j++)
                a[i] = max(a[i], w[i][j]);
        }
        for (int i = 1; i <= n; i++) {
```

```
            memset(va, 0, sizeof va);
            memset(vb, 0, sizeof vb);
            memset(upd, 0x3f, sizeof upd);
            int st = 0; match[0] = i;
            while (match[st]) {
                LL delta = 1e18;
                if (dfs(match[st], st)) break;
                for (int j = 1; j <= n; j++) {
                    if (!vb[j] && upd[j] < delta)
                        delta = upd[j], st = j;
                }
                for (int j = 1; j <= n; j++) {
                    if (va[j]) a[j] -= delta;
                    if (vb[j]) b[j] += delta;
                    else upd[j] -= delta;
                }
                vb[st] = true;
            }
            while (st) {
                match[st] = match[last[st]];
                st = last[st];
            }
        }
    }
}
```

2.13 有负圈/上下界费用流

```
// 有负圈 / 上下界
struct MCMF2{
    const int N = 205, M = 10005;
    const int INF = 0x3f3f3f3f;
    int n, m, s, t, maxflow, cost, d[N], incf[N], pre[N];
    int q[N], in, S, T;
    int head[N], a[N], numE = 1, a0, a1;
    bool vis[N];
    struct E{
        int next, v, w, c;
    } e[M << 2];
```

```
void inline add(int u, int v, int w, int c) {
    e[++numE] = (E) { head[u], v, w, c };
    head[u] = numE;
}
void inline addE(int u, int v, int w, int c) {
    add(u, v, w, c), add(v, u, 0, -c);
}
bool spfa() {
    memset(vis, false, sizeof vis);
    memset(d, 0x3f, sizeof d);
    int hh = 0, tt = 1;
    q[0] = S; d[S] = 0; incf[S] = 2e9;
    while (hh != tt) {
        int u = q[hh++]; vis[u] = false;
        if (hh == N) hh = 0;
        for (int i = head[u]; i; i = e[i].next) {
            int v = e[i].v;
            if (e[i].w && d[u] + e[i].c < d[v]) {
                d[v] = d[u] + e[i].c;
                pre[v] = i;
                incf[v] = min(incf[u], e[i].w);
                if (!vis[v]) {
                    q[tt++] = v;
                    vis[v] = true;
                    if (tt == N) tt = 0;
                }
            }
        }
    }
    return d[T] != INF;
}
void update() {
    int x = T;
    while (x != S) {
        int i = pre[x];
        e[i].w -= incf[T], e[i ^ 1].w += incf[T];
        x = e[i ^ 1].v;
    }
    maxflow += incf[T];
```

```
        cost += d[T] * incf[T];
    }

    void inline addEdge(int u, int v, int l, int d, int c) {
        a[v] += l, a[u] -= l;
        addE(u, v, d - l, c);
    }

    void inline work() {
        while (spfa()) update();
    }

    void inline ADD(int u, int v, int w, int c) {
        if (c >= 0) addEdge(u, v, 0, w, c);
        else a[v] += w, a[u] -= w, addEdge(v, u, 0, w, -c), a1 += c * w;
    }

    void inline solve() {
        for (int i = 1; i <= n; i++) {
            if (!a[i]) continue;
            if (a[i] > 0) addEdge(S, i, 0, a[i], 0);
            else addEdge(i, T, 0, -a[i], 0);
        }
        addEdge(T, S, 0, INF, 0);
        work();
        S = s, T = t;
        a1 += cost;
        maxflow = cost = 0;
        e[numE].w = e[numE - 1].w = 0;
        work();
        a0 += maxflow, a1 += cost;
    }
}
```

2.14 虚树

```
void insert(int x) {
    if (!top) { s[++top] = x; return; }
    int p = lca(x, s[top]);
```

```
    while (top > 1 && dep[s[top - 1]] >= dep[p]) e[s[top - 1]].pb(s[top]), top--;
    if (s[top] != p) {
        e[p].pb(s[top]);
        s[top] = p;
    }
    s[++top] = x;
}


bool inline cmp(int x, int y) {
    return dfn[x] < dfn[y];
}
int inline build(vector<int> &A) {
    top = 0;
    sort(A.begin(), A.end(), cmp);
    for (int x: A) {
        insert(x);
    }
    for (int i = 1; i < top; i++)
        e[s[i]].pb(s[i + 1]);
    return s[1];
}
```

2.15 重链剖分 + LCA

```
int sz[SZ], fa[SZ], dep[SZ], top[SZ], hson[SZ];

void dfs1(int u) {
    sz[u] = 1;
    for (int i = head[u]; i; i = e[i].next) {
        int v = e[i].v;
        if (v == fa[u]) continue;
        dep[v] = dep[u] + 1, fa[v] = u;
        dfs1(v);
        sz[u] += sz[v];
        if (sz[v] > sz[hson[u]]) hson[u] = v;
    }
}

```

```cpp
void dfs2(int u, int tp) {
    top[u] = tp;
    if (hson[u]) dfs2(hson[u], tp);
    for (int i = head[u]; i; i = e[i].next) {
        int v = e[i].v;
        if (v == fa[u] || v == hson[u]) continue;
        dfs2(v, v);
    }
}

int lca(int x, int y) {
    while (top[x] != top[y]) {
        if (dep[top[x]] < dep[top[y]]) swap(x, y);
        x = fa[top[x]];
    }
    if (dep[x] < dep[y]) swap(x, y);
    return y;
}
```

2.16 匈牙利

```cpp
int match[N];
bool vis[N];

bool find(int u) {
    for (int i = head[u]; i; i = e[i].next) {
        int v = e[i].v;
        if (vis[v]) continue;
        vis[v] = true;
        if (!match[v] || find(match[v])) {
            match[v] = u; return true;
        }
    }
    return false;
}


for (int i = 1; i <= n; i++) {
    memset(vis, 0, sizeof vis);
```

```
    if (find(i)) ans++;
}
```

2.17 上下界网络流

```
// 上下界网络流

struct NF{
    int n, S, T, head[N], numE = 0, q[N], d[N], a[N], ans;

    struct E{
        int next, v, w;
    } e[M];

    void inline init(int len, int s, int t) {
        n = len, S = n + 1, T = n + 2, ans = 0;
        memset(head, 0, sizeof head);
        memset(a, 0, sizeof a);
        numE = 1;
    }

    void inline addEdge(int u, int v, int w) {
        e[++numE] = (E) { head[u], v, w };
        head[u] = numE;
    }

    void inline add(int u, int v, int c, int d) {
        a[v] += c, a[u] -= c;
        addEdge(u, v, d - c), addEdge(v, u, 0);
    }

    bool inline bfs() {
        memset(d, 0, sizeof d);
        int hh = 0, tt = 0; q[0] = S;
        d[S] = 1;
        while (hh <= tt) {
            int u = q[hh++];
            if (u == T) return true;
            for (int i = head[u]; i; i = e[i].next) {
```

```cpp
                int v = e[i].v;
                if (e[i].w && !d[v]) {
                    d[v] = d[u] + 1;
                    q[++tt] = v;
                }
            }
        }
        return false;
    }

    int dinic(int u, int flow) {
        if (u == T) return flow;
        int rest = flow;
        for (int i = head[u]; i && rest; i = e[i].next) {
            int v = e[i].v;
            if (e[i].w && d[v] == d[u] + 1) {
                int k = dinic(v, min(rest, e[i].w));
                if (!k) d[v] = 0;
                e[i].w -= k, e[i ^ 1].w += k, rest -= k;
            }
        }
        return flow - rest;
    }

    void inline prework() {
        for (int i = 1; i <= n; i++)
            if (a[i] > 0) addEdge(S, i, a[i]), addEdge(i, S, 0), ans += a[i];
            else if (a[i] < 0) addEdge(i, T, -a[i]), addEdge(T, i, 0);
    }

    int inline run() {
        int res;
        addEdge(n, n - 1, INF);
        addEdge(n - 1, n, 0);
        while (bfs())
            while(res = dinic(S, INF)) ans -= res;
        if (ans) return -1;
        ans = e[numE].w;
        e[numE].w = e[numE - 1].w = 0;
```

```
        S = n - 1, T = n;
        while (bfs())
            while(res = dinic(S, INF)) ans += res;
        return ans;
    }

} t;

int S = n + m + 1, T = n + m + 2;
t.init(T, S, T);
t.add(u, v, c, d);
t.prework();
output:: t.run()
```

2.18 $O(1)$ LCA

```
const int N = 5e5 + 5, L = 19;

int n, m, dfncnt, rt, st[L][N], Lg[N], dfn[N], d[N], fa[N];

vector<int> g[N];

void dfs0(int u) {
    st[0][dfncnt] = fa[u];
    dfn[u] = ++dfncnt;
    for (int v: g[u]) {
        if (v == fa[u]) continue;
        d[v] = d[u] + 1;
        fa[v] = u;
        dfs0(v);
    }
}

int inline cmp(int x, int y) {
    return d[x] < d[y] ? x : y;
}

void inline bd() {
    Lg[0] = -1;
```

```cpp
    for (int i = 1; i <= n; i++)
        Lg[i] = Lg[i >> 1] + 1;
    for (int j = 1; j <= Lg[n]; j++)
        for (int i = 1; i + (1 << j) - 1 <= n; i++)
            st[j][i] = cmp(st[j - 1][i], st[j - 1][i + (1 << (j - 1))]);
}

int inline lca(int x, int y) {
    if (x == y) return x;
    x = dfn[x], y = dfn[y];
    if (x > y) swap(x, y); --y;
    int k = Lg[y - x + 1];
    return cmp(st[k][x], st[k][y - (1 << k) + 1]);
}

void prework() {
    dfs0(rt);
    bd();
}

// Use lca(a, b)
```

2.19 点分治

```cpp
int val;

void findRoot(int u, int last, int &rt) {
    sz[u] = 1; int s = 0;
    for (int i = head[u]; i; i = e[i].next) {
        int v = e[i].v;
        if (st[v] || v == last) continue;
        findRoot(v, u, rt);
        sz[u] += sz[v], s = max(s, sz[v]);
    }
    s = max(s, S - sz[u]);
    if (s < val) val = s, rt = u;
}

void solve(int u) {
```

```cpp
    if (st[u]) return;
    val = INF, findRoot(u, 0, u), st[u] = true;
    for (int i = head[u], j = 0; i; i = e[i].next) {
        int v = e[i].v;
        if (st[v]) continue;
        // Do sth
    }
    for (int i = head[u]; i; i = e[i].next) S = sz[e[i].v], solve(e[i].v);
}

S = n, solve(1);
```

3 Poly 多项式

3.1 1e18 多项式乘法

```cpp
// 1e18 多项式乘法》。。。别用fft（mtt也不会写

#define I __int128_t
typedef vector<I> Poly;
const I P = 1945555039024054273ll, G = 5;
// p=1945555039024054273=27\times 2^{56}+1,g=5


I A[N], rev[N];
I lim = 1, len = 0;
LL W[19][N];

I inline power(I a, I b, I Mod = P) {
        I res = 1;
        while (b) {
                if (b & 1) res = res * a % Mod;
                a = a * a % Mod;
                b >>= 1;
        }
        return res;
}

```

```
void inline NTT(I c[], int lim, int o) {
	for (int i = 0; i < lim; i++)
		if (i < rev[i]) swap(c[i], c[rev[i]]);
	for (int k = 1, t = 0; k < lim; k <<= 1, t++) {
		for (int i = 0; i < lim; i += (k << 1)) {
			for (int j = 0; j < k; j++) {
				I u = c[i + j], v = (I)c[i + k + j] * W[t][j] % P;
				c[i + j] = u + v >= P ? u + v - P : u + v;
				c[i + j + k] = u - v < 0 ? u - v + P : u - v;
			}
		}
	}
	if (o == -1) {
		reverse(c + 1, c + lim);
		I inv = power(lim, P - 2, P);
		for (int i = 0; i < lim; i++)
			c[i] = c[i] * inv % P;
	}
}

void inline setN(int n) {
	lim = 1, len = 0;
	while (lim < n) lim <<= 1, len++;
	for (int i = 0; i < lim; i++)
		rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (len - 1));
}

Poly inline NTT(Poly a, int o) {
	int n = a.size();
	for (int i = 0; i < n; i++) A[i] = a[i];
	NTT(A, lim, o);
	a.clear();
	for (int i = 0; i < lim; i++) a.push_back(A[i]), A[i] = 0;
	return a;
}

Poly inline mul (Poly a, Poly b, int newn = -1) {
	if (newn == -1) newn = a.size() + b.size() - 1;
```

```
	setN(a.size() + b.size() - 1);
	Poly c = NTT(a, 1), d = NTT(b, 1);
	for (int i = 0; i < lim; i++) c[i] = (I)c[i] * d[i] % P;
	d = NTT(c, -1); d.resize(newn);
	return d;
}

// 用到的最大的 n
void inline init(int n) {
	setN(n);
	for (int k = 1, t = 0; k < lim; k <<= 1, t++) {
		I wn = power(G, (P - 1) / (k << 1));
		W[t][0] = 1;
		for (int j = 1; j < k; j++) W[t][j] = (I)W[t][j - 1] * wn % P;
	}
}

// --
```

3.2 正常多项式 + 线性递推

```
typedef vector<int> Poly;

#define pb push_back

const int N = 8e5 + 5, P = 998244353, G = 3;

int A[N], rev[N], mod, inv[N], fact[N], infact[N];
int lim = 1, len = 0, W[20][N];

int inline power(int a, int b, int Mod = P) {
	int res = 1;
	while (b) {
		if (b & 1) res = (LL)res * a % Mod;
		a = (LL)a * a % Mod;
		b >>= 1;
	}
	return res;
}
```

```
int Gi = power(G, P - 2, P), inv2 = power(2, P - 2, P);

void inline NTT(int c[], int lim, int o) {
	for (int i = 0; i < lim; i++)
		if (i < rev[i]) swap(c[i], c[rev[i]]);
	for (int k = 1, t = 0; k < lim; k <<= 1, t++) {
		for (int i = 0; i < lim; i += (k << 1)) {
			for (int j = 0; j < k; j++) {
				int u = c[i + j], v = (LL)c[i + k + j] * W[t][j] % P;
				c[i + j] = u + v >= P ? u + v - P : u + v;
				c[i + j + k] = u - v < 0 ? u - v + P : u - v;
			}
		}
	}
	if (o == -1) {
		reverse(c + 1, c + lim);
		int inv = power(lim, P - 2, P);
		for (int i = 0; i < lim; i++)
			c[i] = (LL)c[i] * inv % P;
	}
}

void inline setN(int n) {
	lim = 1, len = 0;
	while (lim < n) lim <<= 1, len++;
	for (int i = 0; i < lim; i++)
		rev[i] = (rev[i >> 1] >> 1) | ((i & 1) << (len - 1));
}

Poly inline NTT(Poly a, int o) {
	int n = a.size();
	for (int i = 0; i < n; i++) A[i] = a[i];
	NTT(A, lim, o);
	a.clear();
	for (int i = 0; i < lim; i++) a.push_back(A[i]), A[i] = 0;
	return a;
}

```

```
Poly inline mul (Poly a, Poly b, int newn = -1) {
        if (newn == -1) newn = a.size() + b.size() - 1;
        setN(a.size() + b.size() - 1);
        Poly c = NTT(a, 1), d = NTT(b, 1);
        for (int i = 0; i < lim; i++) c[i] = (LL)c[i] * d[i] % P;
        d = NTT(c, -1); d.resize(newn);
        return d;
}

// 用到的最大的 n
void inline init(int n) {
        setN(2 * n);
        for (int k = 1, t = 0; k < lim; k <<= 1, t++) {
                int wn = power(G, (P - 1) / (k << 1));
                W[t][0] = 1;
                for (int j = 1; j < k; j++) W[t][j] = (LL)W[t][j - 1] * wn % P;
        }
}

// f[0 ... n] 线性递推第 b 项
// g[1 ~ k] 为递推多项式

int inline LRS(int b, Poly f, Poly g) {
        int k = g.size() - 1;
        g[0] = 1;
        for (int i = 1; i <= k; i++) g[i] = (P - g[i]) % P;
        Poly h = mul(f, g, k);
        while (b) {
                Poly g2 = g;
                for (int i = 0; i < g2.size(); i += 2)
                        g2[i] = (P - g2[i]) % P;
                Poly t = mul(g2, g); g.clear();
                for (int i = 0; i < t.size(); i += 2)
                        g.pb(t[i]);
                t = mul(g2, h); h.clear();
                for (int i = (b & 1); i < t.size(); i += 2)
                        h.pb(t[i]);
                b >>= 1;
        }
```

```
        return (LL)h[0] * power(g[0], P - 2) % P;
}
```

4 字符串

4.1 AC 自动机

```
struct ACAutomation{
        int tr[SZ][26], nxt[SZ], idx, q[SZ];
        void inline insert(char s[]) {
                int p = 0;
                for (int j = 0; s[j]; j++) {
                        int ch = s[j] - 'a';
                        if(!tr[p][ch]) tr[p][ch] = ++idx;
                        p = tr[p][ch];
                }
        }
        void build() {
                int hh = 0, tt = -1;
                for (int i = 0; i < 26; i++)
                        if (tr[0][i]) q[++tt] = tr[0][i];
                while (hh <= tt) {
                        int u = q[hh++];
                        for (int i = 0; i < 26; i++) {
                                int v = tr[u][i];
                                if (!v) tr[u][i] = tr[nxt[u]][i];
                                else nxt[v] = tr[nxt[u]][i], q[++tt] = v;
                        }
                }
        }
}
```

4.2 KMP

```
struct KMP{
        int n, nxt[SZ];
        void inline build(char s[]) {
                n = strlen(s + 1);
```

```
        nxt[1] = 0;
        for (int i = 2, j = 0; i <= n; i++) {
            while (j && s[j + 1] != s[i]) j = nxt[j];
            if (s[j + 1] == s[i]) j++;
            nxt[i] = j;
        }
    }
    void inline match(char a[], int m) {
        for (int i = 1, j = 0; i <= m; i++) {
            while (j && s[j + 1] != a[i]) j = nxt[j];
            if (s[j + 1] == a[i]) j++;
            if (j == n) {
                j = nxt[j];
            }
        }
    }
} kmp;
```

4.3 Manacher

```
// 中间添加 #
char s[N], g[N];

void change() {
    n = strlen(s + 1) * 2;
    g[0] = 0;
    for (int i = 1; i <= n; i++) {
        if (i % 2) g[i] = 1;
        else g[i] = s[i >> 1];
    }
    g[++n] = 1, g[n + 1] = 2;
    manacher();
}

void manacher() {
    int r = 0, mid = 0;
    for (int i = 1; i <= n; i++) {
        p[i] = i <= r ? min(r - i + 1, p[2 * mid - i]) : 1;
        while (g[i - p[i]] == g[i + p[i]]) ++p[i];
```

```
            if (i + p[i] - 1 > r) mid = i, r = i + p[i] - 1;
            ans = max(ans, p[i] - 1);
        }
}
```

4.4 SA

```
struct SA{
        int rk[SZ], sa[SZ], cnt[SZ], oldrk[SZ], id[SZ], n, m, p, height[SZ];
        bool inline cmp(int i, int j, int k) {
                return oldrk[i] == oldrk[j] && oldrk[i + k] == oldrk[j + k];
        }
        void inline build(char s[]) {
                n = strlen(s + 1), m = 221;
                for (int i = 1; i <= n; i++) cnt[rk[i] = s[i]]++;
                for (int i = 1; i <= m; i++) cnt[i] += cnt[i - 1];
                for (int i = n; i; i--) sa[cnt[rk[i]]--] = i;
                for (int w = 1; w < n; w <<= 1, m = p) {
                        p = 0;
                        for (int i = n; i > n - w; i--) id[++p] = i;
                        for (int i = 1; i <= n; i++)
                                if (sa[i] > w) id[++p] = sa[i] - w;
                        for (int i = 1; i <= m; i++) cnt[i] = 0;
                        for (int i = 1; i <= n; i++) cnt[rk[i]]++, oldrk[i] = rk[i];
                        for (int i = 1; i <= m; i++) cnt[i] += cnt[i - 1];
                        for (int i = n; i; i--) sa[cnt[rk[id[i]]]--] = id[i];
                        p = 0;
                        for (int i = 1; i <= n; i++) {
                                rk[sa[i]] = cmp(sa[i], sa[i - 1], w) ? p : ++p;
                        }
                        if (p == n) break;
                }
                for (int i = 1; i <= n; i++) {
                        int j = sa[rk[i] - 1], k = max(0, height[rk[i - 1]] - 1);
                        while (s[i + k] == s[j + k]) k++;
                        height[rk[i]] = k;
                }
        }
};
```

4.5 哈希 Hash

```cpp
// 哈希

struct Hash{
        int b, P, p[N], h[N];
        int inline get(int l, int r){
            return (h[r] - (LL)h[l - 1] * p[r - l + 1] % P + P) % P;
        }
        void inline build(int n, int tb, int tp) {
                b = tb, P = tp;
                p[0] = 1;
            for(int i = 1; i <= n; i++){
                p[i] = (LL)p[i - 1] * b % P;
                h[i] = ((LL)h[i - 1] * b + s[i]) % P;
            }
        }
}
```

4.6 最小表示法

```cpp
// 切记复制一倍到后面, 最小表示法, 返回开始下标
int inline minExp(int a[], int n) {
        int i = 1, j = 2;
        while (i <= n && j <= n) {
                int k;
                for (k = 0; k < n && a[i + k] == a[j + k]; k++);
                if (k == n) break;
                if (a[i + k] < a[j + k]) j += k + 1;
                else i += k + 1;
                if (i == j) i++;
        }
        return min(i, j);
}
```

4.7 Z 函数

```cpp
// Z 函数
```

```
z[1] = n;
for (int i = 2, r = 0, j = 0; i <= n; i++) {
        if (i <= r) z[i] = min(r - i + 1, z[i - j + 1]);
        while (i + z[i] <= n && a[i + z[i]] == a[1 + z[i]]) z[i]++;
        if (i + z[i] - 1 > r) r = i + z[i] - 1, j = i;
}

for (int i = 1, r = 0, j = 0; i <= m; i++) {
        if (i <= r) p[i] = min(r - i + 1, z[i - j + 1]);
        while (i + p[i] <= m && b[i + p[i]] == a[1 + p[i]]) p[i]++;
        if (i + p[i] - 1 > r) r = i + p[i] - 1, j = i;
}
```

4.8 SAM

```
struct SAM{
        int idx, last;
        struct SAM_{
                int nxt[26], len, link;
        } t[N];
        void inline init() {
                last = idx = 1;
        }

        void inline extend(int c) {
                int x = ++idx, p = last; sz[x] = 1;
                t[x].len = t[last].len + 1;
                while (p && !t[p].nxt[c])
                        t[p].nxt[c] = x, p = t[p].link;
                if (!p) t[x].link = 1;
                else {
                        int q = t[p].nxt[c];
                        if (t[p].len + 1 == t[q].len) t[x].link = q;
                        else {
                                int y = ++idx;
                                t[y] = t[q], t[y].len = t[p].len + 1;
                                while (p && t[p].nxt[c] == q)
                                        t[p].nxt[c] = y, p = t[p].link;
                                t[q].link = t[x].link = y;
```

```
                }
            }
            last = x;
        }
} t;
```

4.9 广义 SAM

```
struct GSAM{
        int idx, last;
        struct SAM{
                int ch[26], len, link;
        } t[N];
        void inline init() {
                last = idx = 1;
        }
        void inline insert(int c) {
                int p = last;
                if (t[p].ch[c]) {
                        int q = t[p].ch[c];
                        if (t[q].len == t[p].len + 1) last = q;
                        else {
                                int y = ++idx; t[y] = t[q];
                                t[y].len = t[p].len + 1;
                                while (p && t[p].ch[c] == q)
                                        t[p].ch[c] = y, p = t[p].link;
                                t[q].link = y;
                                last = y;
                        }
                        return;
                }
                int x = ++idx; t[x].len = t[p].len + 1;
                while (p && !t[p].ch[c]) t[p].ch[c] = x, p = t[p].link;
                int q, y;
                if (!p) t[x].link = 1;
                else {
                        q = t[p].ch[c];
                        if (t[q].len == t[p].len + 1) t[x].link = q;
                        else {
```

```
                                int y = ++idx; t[y] = t[q];
                                t[y].len = t[p].len + 1;
                                while (p && t[p].ch[c] == q)
                                        t[p].ch[c] = y, p = t[p].link;
                                t[q].link = t[x].link = y;
                                last = y;
                        }
                }
                last = x;
        }
} t;
```

4.10 回文自动机

```
// 回文自动机
struct PAM{
    int n, ch[N][26], fail[N], len[N], sz[N], idx = -1, last;
    char s[N];
    void clr() {
        n = 0;
        for (int i = 0; i <= idx; i++) {
            sz[i] = len[i] = fail[i] = 0;
            for (int j = 0; j < 26; j++)
                ch[i][j] = 0;
        }
        idx = -1;
        last = 0;
    }

    int newNode(int x) { len[++idx] = x; return idx; }
    int getFail(int x) {
        while (s[n - len[x] - 1] != s[n]) x = fail[x];
        return x;
    }
    int insert(char c) {
        int k = c - 'a';
        s[++n] = c;
        int p = getFail(last), x;
        if (!ch[p][k]) {
```

```
            x = newNode(len[p] + 2);
            fail[x] = ch[getFail(fail[p])][k];
            ch[p][k] = x, sz[x] = 1 + sz[fail[x]];
        } else x = ch[p][k];
        last = x;
        return x;
    }
    void bd() {
        // -1:idx jigen
        newNode(0), newNode(-1);
        s[0] = '$', fail[0] = 1, last = 0;
    }
} pam;
```

5 数学

5.1 Min-Max 容斥

期望也对。

$$\max(S) = \sum(-1)^{|T|-1}\min(T)$$

$$kMax(S) = \sum_{|T|\geq k}(-1)^{|T|-k}\binom{|T|-1}{k-1}\min(T)$$

5.2 单位根反演

$$[n|k] = \frac{1}{n}\sum_{i=1}^{n-1} w_n^{ik}$$

5.3 积分表

Basic Forms

$$\int x^n dx = \frac{1}{n+1}x^{n+1} \tag{1}$$

$$\int \frac{1}{x} dx = \ln |x| \tag{2}$$

$$\int u dv = uv - \int v du \tag{3}$$

$$\int \frac{1}{ax+b} dx = \frac{1}{a} \ln |ax+b| \tag{4}$$

Integrals of Rational Functions

$$\int \frac{1}{(x+a)^2} dx = -\frac{1}{x+a} \tag{5}$$

$$\int (x+a)^n dx = \frac{(x+a)^{n+1}}{n+1}, n \neq -1 \tag{6}$$

$$\int x(x+a)^n dx = \frac{(x+a)^{n+1}((n+1)x-a)}{(n+1)(n+2)} \tag{7}$$

$$\int \frac{1}{1+x^2} dx = \tan^{-1} x \tag{8}$$

$$\int \frac{1}{a^2+x^2} dx = \frac{1}{a} \tan^{-1} \frac{x}{a} \tag{9}$$

$$\int \frac{x}{a^2+x^2} dx = \frac{1}{2} \ln |a^2+x^2| \tag{10}$$

$$\int \frac{x^2}{a^2+x^2} dx = x - a \tan^{-1} \frac{x}{a} \tag{11}$$

$$\int \frac{x^3}{a^2+x^2} dx = \frac{1}{2} x^2 - \frac{1}{2} a^2 \ln |a^2+x^2| \tag{12}$$

$$\int \frac{1}{ax^2+bx+c} dx = \frac{2}{\sqrt{4ac-b^2}} \tan^{-1} \frac{2ax+b}{\sqrt{4ac-b^2}} \tag{13}$$

$$\int \frac{1}{(x+a)(x+b)} dx = \frac{1}{b-a} \ln \frac{a+x}{b+x},\ a \neq b \tag{14}$$

$$\int \frac{x}{(x+a)^2} dx = \frac{a}{a+x} + \ln |a+x| \tag{15}$$

$$\int \frac{x}{ax^2+bx+c}dx = \frac{1}{2a}\ln|ax^2+bx+c| - \frac{b}{a\sqrt{4ac-b^2}}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}} \tag{16}$$

Integrals with Roots

$$\int \sqrt{x-a}dx = \frac{2}{3}(x-a)^{3/2} \tag{17}$$

$$\int \frac{1}{\sqrt{x\pm a}}dx = 2\sqrt{x\pm a} \tag{18}$$

$$\int \frac{1}{\sqrt{a-x}}dx = -2\sqrt{a-x} \tag{19}$$

$$\int x\sqrt{x-a}dx = \frac{2}{3}a(x-a)^{3/2} + \frac{2}{5}(x-a)^{5/2} \tag{20}$$

$$\int \sqrt{ax+b}dx = \left(\frac{2b}{3a}+\frac{2x}{3}\right)\sqrt{ax+b} \tag{21}$$

$$\int (ax+b)^{3/2}dx = \frac{2}{5a}(ax+b)^{5/2} \tag{22}$$

$$\int \frac{x}{\sqrt{x\pm a}}dx = \frac{2}{3}(x\mp 2a)\sqrt{x\pm a} \tag{23}$$

$$\int \sqrt{\frac{x}{a-x}}dx = -\sqrt{x(a-x)} - a\tan^{-1}\frac{\sqrt{x(a-x)}}{x-a} \tag{24}$$

$$\int \sqrt{\frac{x}{a+x}}dx = \sqrt{x(a+x)} - a\ln\left[\sqrt{x}+\sqrt{x+a}\right] \tag{25}$$

$$\int x\sqrt{ax+b}dx = \frac{2}{15a^2}(-2b^2+abx+3a^2x^2)\sqrt{ax+b} \tag{26}$$

$$\int \sqrt{x(ax+b)}dx = \frac{1}{4a^{3/2}}\left[(2ax+b)\sqrt{ax(ax+b)} \right.$$
$$\left. -b^2 \ln\left|a\sqrt{x}+\sqrt{a(ax+b)}\right|\right] \tag{27}$$

$$\int \sqrt{x^3(ax+b)}dx = \left[\frac{b}{12a}-\frac{b^2}{8a^2x}+\frac{x}{3}\right]\sqrt{x^3(ax+b)}$$
$$+\frac{b^3}{8a^{5/2}}\ln\left|a\sqrt{x}+\sqrt{a(ax+b)}\right| \tag{28}$$

$$\int \sqrt{x^2 \pm a^2}dx = \frac{1}{2}x\sqrt{x^2\pm a^2} \pm \frac{1}{2}a^2 \ln\left|x+\sqrt{x^2\pm a^2}\right| \tag{29}$$

$$\int \sqrt{a^2 - x^2}dx = \frac{1}{2}x\sqrt{a^2-x^2} + \frac{1}{2}a^2 \tan^{-1}\frac{x}{\sqrt{a^2-x^2}} \tag{30}$$

$$\int x\sqrt{x^2 \pm a^2}dx = \frac{1}{3}\left(x^2\pm a^2\right)^{3/2} \tag{31}$$

$$\int \frac{1}{\sqrt{x^2\pm a^2}}dx = \ln\left|x+\sqrt{x^2\pm a^2}\right| \tag{32}$$

$$\int \frac{1}{\sqrt{a^2-x^2}}dx = \sin^{-1}\frac{x}{a} \tag{33}$$

$$\int \frac{x}{\sqrt{x^2\pm a^2}}dx = \sqrt{x^2\pm a^2} \tag{34}$$

$$\int \frac{x}{\sqrt{a^2-x^2}}dx = -\sqrt{a^2-x^2} \tag{35}$$

$$\int \frac{x^2}{\sqrt{x^2\pm a^2}}dx = \frac{1}{2}x\sqrt{x^2\pm a^2} \mp \frac{1}{2}a^2 \ln\left|x+\sqrt{x^2\pm a^2}\right| \tag{36}$$

$$\int \sqrt{ax^2+bx+c}dx = \frac{b+2ax}{4a}\sqrt{ax^2+bx+c}$$
$$+\frac{4ac-b^2}{8a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx^+c)}\right| \tag{37}$$

$$\int x\sqrt{ax^2+bx+c} = \frac{1}{48a^{5/2}}\left(2\sqrt{a}\sqrt{ax^2+bx+c}\right.$$
$$\times\left(-3b^2+2abx+8a(c+ax^2)\right)$$
$$\left.+3(b^3-4abc)\ln\left|b+2ax+2\sqrt{a}\sqrt{ax^2+bx+c}\right|\right) \tag{38}$$

$$\int\frac{1}{\sqrt{ax^2+bx+c}}dx = \frac{1}{\sqrt{a}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right| \tag{39}$$

$$\int\frac{x}{\sqrt{ax^2+bx+c}}dx = \frac{1}{a}\sqrt{ax^2+bx+c}$$
$$-\frac{b}{2a^{3/2}}\ln\left|2ax+b+2\sqrt{a(ax^2+bx+c)}\right| \tag{40}$$

$$\int\frac{dx}{(a^2+x^2)^{3/2}} = \frac{x}{a^2\sqrt{a^2+x^2}} \tag{41}$$

Integrals with Logarithms

$$\int \ln ax dx = x\ln ax - x \tag{42}$$

$$\int\frac{\ln ax}{x}dx = \frac{1}{2}\left(\ln ax\right)^2 \tag{43}$$

$$\int \ln(ax+b)dx = \left(x+\frac{b}{a}\right)\ln(ax+b)-x, a\neq 0 \tag{44}$$

$$\int \ln(x^2+a^2)\,\mathrm{dx} = x\ln(x^2+a^2)+2a\tan^{-1}\frac{x}{a}-2x \tag{45}$$

$$\int \ln(x^2-a^2)\,\mathrm{dx} = x\ln(x^2-a^2)+a\ln\frac{x+a}{x-a}-2x \tag{46}$$

$$\int \ln\left(ax^2+bx+c\right)dx = \frac{1}{a}\sqrt{4ac-b^2}\tan^{-1}\frac{2ax+b}{\sqrt{4ac-b^2}}$$
$$-2x+\left(\frac{b}{2a}+x\right)\ln\left(ax^2+bx+c\right) \tag{47}$$

$$\int x\ln(ax+b)dx = \frac{bx}{2a} - \frac{1}{4}x^2 + \frac{1}{2}\left(x^2 - \frac{b^2}{a^2}\right)\ln(ax+b) \tag{48}$$

$$\int x\ln\left(a^2 - b^2x^2\right)dx = -\frac{1}{2}x^2 + \frac{1}{2}\left(x^2 - \frac{a^2}{b^2}\right)\ln\left(a^2 - b^2x^2\right) \tag{49}$$

Integrals with Exponentials

$$\int e^{ax}dx = \frac{1}{a}e^{ax} \tag{50}$$

$$\int \sqrt{x}e^{ax}dx = \frac{1}{a}\sqrt{x}e^{ax} + \frac{i\sqrt{\pi}}{2a^{3/2}}\operatorname{erf}\left(i\sqrt{ax}\right), \text{ where } \operatorname{erf}(x) = \frac{2}{\sqrt{\pi}}\int_0^x e^{-t^2}dt \tag{51}$$

$$\int xe^{x}dx = (x-1)e^{x} \tag{52}$$

$$\int xe^{ax}dx = \left(\frac{x}{a} - \frac{1}{a^2}\right)e^{ax} \tag{53}$$

$$\int x^2e^{x}dx = \left(x^2 - 2x + 2\right)e^{x} \tag{54}$$

$$\int x^2e^{ax}dx = \left(\frac{x^2}{a} - \frac{2x}{a^2} + \frac{2}{a^3}\right)e^{ax} \tag{55}$$

$$\int x^3e^{x}dx = \left(x^3 - 3x^2 + 6x - 6\right)e^{x} \tag{56}$$

$$\int x^ne^{ax}\,\mathrm{d}x = \frac{x^ne^{ax}}{a} - \frac{n}{a}\int x^{n-1}e^{ax}\,\mathrm{d}x \tag{57}$$

$$\int x^n e^{ax}\,\mathrm{d}x = \frac{(-1)^n}{a^{n+1}}\Gamma[1+n,-ax], \quad \text{where } \Gamma(a,x) = \int_x^\infty t^{a-1}e^{-t}\,\mathrm{d}t \tag{58}$$

$$\int e^{ax^2}\,\mathrm{d}x = -\frac{i\sqrt{\pi}}{2\sqrt{a}}\mathrm{erf}\left(ix\sqrt{a}\right) \tag{59}$$

$$\int e^{-ax^2}\,\mathrm{d}x = \frac{\sqrt{\pi}}{2\sqrt{a}}\mathrm{erf}\left(x\sqrt{a}\right) \tag{60}$$

$$\int xe^{-ax^2}\ \mathrm{dx} = -\frac{1}{2a}e^{-ax^2} \tag{61}$$

$$\int x^2e^{-ax^2}\ \mathrm{dx} = \frac{1}{4}\sqrt{\frac{\pi}{a^3}}\mathrm{erf}(x\sqrt{a}) - \frac{x}{2a}e^{-ax^2} \tag{62}$$

Integrals with Trigonometric Functions

$$\int \sin axdx = -\frac{1}{a}\cos ax \tag{63}$$

$$\int \sin^2 axdx = \frac{x}{2} - \frac{\sin 2ax}{4a} \tag{64}$$

$$\int \sin^n axdx = -\frac{1}{a}\cos ax\ \ {}_2F_1\left[\frac{1}{2},\frac{1-n}{2},\frac{3}{2},\cos^2 ax\right] \tag{65}$$

$$\int \sin^3 axdx = -\frac{3\cos ax}{4a} + \frac{\cos 3ax}{12a} \tag{66}$$

$$\int \cos axdx = \frac{1}{a}\sin ax \tag{67}$$

$$\int \cos^2 axdx = \frac{x}{2} + \frac{\sin 2ax}{4a} \tag{68}$$

$$\int \cos^p ax dx = -\frac{1}{a(1+p)} \cos^{1+p} ax \times \\ {}_2F_1\left[\frac{1+p}{2}, \frac{1}{2}, \frac{3+p}{2}, \cos^2 ax\right] \tag{69}$$

$$\int \cos^3 ax dx = \frac{3 \sin ax}{4a} + \frac{\sin 3ax}{12a} \tag{70}$$

$$\int \cos ax \sin bx dx = \frac{\cos[(a-b)x]}{2(a-b)} - \frac{\cos[(a+b)x]}{2(a+b)}, a \neq b \tag{71}$$

$$\int \sin^2 ax \cos bx dx = -\frac{\sin[(2a-b)x]}{4(2a-b)} \\ + \frac{\sin bx}{2b} - \frac{\sin[(2a+b)x]}{4(2a+b)} \tag{72}$$

$$\int \sin^2 x \cos x dx = \frac{1}{3} \sin^3 x \tag{73}$$

$$\int \cos^2 ax \sin bx dx = \frac{\cos[(2a-b)x]}{4(2a-b)} - \frac{\cos bx}{2b} \\ - \frac{\cos[(2a+b)x]}{4(2a+b)} \tag{74}$$

$$\int \cos^2 ax \sin ax dx = -\frac{1}{3a} \cos^3 ax \tag{75}$$

$$\int \sin^2 ax \cos^2 bx dx = \frac{x}{4} - \frac{\sin 2ax}{8a} - \frac{\sin[2(a-b)x]}{16(a-b)} \\ + \frac{\sin 2bx}{8b} - \frac{\sin[2(a+b)x]}{16(a+b)} \tag{76}$$

$$\int \sin^2 ax \cos^2 ax dx = \frac{x}{8} - \frac{\sin 4ax}{32a} \tag{77}$$

$$\int \tan ax dx = -\frac{1}{a} \ln \cos ax \tag{78}$$

$$\int \tan^2 ax dx = -x + \frac{1}{a} \tan ax \tag{79}$$

$$\int \tan^n ax dx = \frac{\tan^{n+1} ax}{a(1+n)} \times$$
$$ {}_2F_1\left(\frac{n+1}{2}, 1, \frac{n+3}{2}, -\tan^2 ax\right) \tag{80}$$

$$\int \tan^3 ax dx = \frac{1}{a} \ln \cos ax + \frac{1}{2a} \sec^2 ax \tag{81}$$

$$\int \sec x dx = \ln|\sec x + \tan x| = 2 \tanh^{-1}\left(\tan \frac{x}{2}\right) \tag{82}$$

$$\int \sec^2 ax dx = \frac{1}{a} \tan ax \tag{83}$$

$$\int \sec^3 x \, \mathrm{dx} = \frac{1}{2} \sec x \tan x + \frac{1}{2} \ln|\sec x + \tan x| \tag{84}$$

$$\int \sec x \tan x dx = \sec x \tag{85}$$

$$\int \sec^2 x \tan x dx = \frac{1}{2} \sec^2 x \tag{86}$$

$$\int \sec^n x \tan x dx = \frac{1}{n} \sec^n x, n \neq 0 \tag{87}$$

$$\int \csc x dx = \ln\left|\tan \frac{x}{2}\right| = \ln|\csc x - \cot x| + C \tag{88}$$

$$\int \csc^2 ax dx = -\frac{1}{a} \cot ax \tag{89}$$

$$\int \csc^3 x dx = -\frac{1}{2} \cot x \csc x + \frac{1}{2} \ln|\csc x - \cot x| \tag{90}$$

$$\int \csc^n x \cot x dx = -\frac{1}{n} \csc^n x, n \neq 0 \tag{91}$$

$$\int \sec x \csc x dx = \ln|\tan x| \tag{92}$$

Products of Trigonometric Functions and Monomials

$$\int x \cos x dx = \cos x + x \sin x \tag{93}$$

$$\int x \cos ax dx = \frac{1}{a^2} \cos ax + \frac{x}{a} \sin ax \tag{94}$$

$$\int x^2 \cos x dx = 2x \cos x + \left(x^2 - 2\right) \sin x \tag{95}$$

$$\int x^2 \cos ax dx = \frac{2x \cos ax}{a^2} + \frac{a^2x^2 - 2}{a^3} \sin ax \tag{96}$$

$$\begin{aligned}\int x^n cosxdx = &-\frac{1}{2}(i)^{n+1} \left[\Gamma(n+1, -ix)\right. \\ &\left.+(-1)^n \Gamma(n+1, ix)\right]\end{aligned} \tag{97}$$

$$\begin{aligned}\int x^n cosaxdx = &\frac{1}{2}(ia)^{1-n} \left[(-1)^n \Gamma(n+1, -iax)\right. \\ &\left.-\Gamma(n+1, ixa)\right]\end{aligned} \tag{98}$$

$$\int x \sin x dx = -x \cos x + \sin x \tag{99}$$

$$\int x \sin ax dx = -\frac{x \cos ax}{a} + \frac{\sin ax}{a^2} \tag{100}$$

$$\int x^2 \sin x dx = \left(2 - x^2\right) \cos x + 2x \sin x \tag{101}$$

$$\int x^2 \sin ax dx = \frac{2 - a^2x^2}{a^3} \cos ax + \frac{2x \sin ax}{a^2} \tag{102}$$

$$\int x^n \sin x dx = -\frac{1}{2}(i)^n \left[\Gamma(n+1, -ix) - (-1)^n \Gamma(n+1, -ix)\right] \tag{103}$$

Products of Trigonometric Functions and Exponentials

$$\int e^x \sin x dx = \frac{1}{2}e^x(\sin x - \cos x) \tag{104}$$

$$\int e^{bx} \sin ax dx = \frac{1}{a^2+b^2}e^{bx}(b\sin ax - a\cos ax) \tag{105}$$

$$\int e^x \cos x dx = \frac{1}{2}e^x(\sin x + \cos x) \tag{106}$$

$$\int e^{bx} \cos ax dx = \frac{1}{a^2+b^2}e^{bx}(a\sin ax + b\cos ax) \tag{107}$$

$$\int xe^x \sin x dx = \frac{1}{2}e^x(\cos x - x\cos x + x\sin x) \tag{108}$$

$$\int xe^x \cos x dx = \frac{1}{2}e^x(x\cos x - \sin x + x\sin x) \tag{109}$$

Integrals of Hyperbolic Functions

$$\int \cosh ax dx = \frac{1}{a}\sinh ax \tag{110}$$

$$\int e^{ax} \cosh bx dx = \begin{cases} \dfrac{e^{ax}}{a^2-b^2}[a\cosh bx - b\sinh bx] & a \neq b \\ \dfrac{e^{2ax}}{4a} + \dfrac{x}{2} & a = b \end{cases} \tag{111}$$

$$\int \sinh ax dx = \frac{1}{a}\cosh ax \tag{112}$$

$$\int e^{ax} \sinh bx dx = \begin{cases} \dfrac{e^{ax}}{a^2-b^2}[-b\cosh bx + a\sinh bx] & a \neq b \\ \dfrac{e^{2ax}}{4a} - \dfrac{x}{2} & a = b \end{cases} \tag{113}$$

$$\int e^{ax}\tanh bx dx = \begin{cases} \dfrac{e^{(a+2b)x}}{(a+2b)}{}_2F_1\left[1+\dfrac{a}{2b},1,2+\dfrac{a}{2b},-e^{2bx}\right] \\ \qquad -\dfrac{1}{a}e^{ax}{}_2F_1\left[\dfrac{a}{2b},1,1E,-e^{2bx}\right] & a\neq b \\ \dfrac{e^{ax}-2\tan^{-1}[e^{ax}]}{a} & a=b \end{cases} \tag{114}$$

$$\int \tanh ax\, dx = \frac{1}{a}\ln\cosh ax \tag{115}$$

$$\int \cos ax\cosh bx dx = \frac{1}{a^2+b^2}\left[a\sin ax\cosh bx + b\cos ax\sinh bx\right] \tag{116}$$

$$\int \cos ax\sinh bx dx = \frac{1}{a^2+b^2}\left[b\cos ax\cosh bx + a\sin ax\sinh bx\right] \tag{117}$$

$$\int \sin ax\cosh bx dx = \frac{1}{a^2+b^2}\left[-a\cos ax\cosh bx + b\sin ax\sinh bx\right] \tag{118}$$

$$\int \sin ax\sinh bx dx = \frac{1}{a^2+b^2}\left[b\cosh bx\sin ax - a\cos ax\sinh bx\right] \tag{119}$$

$$\int \sinh ax\cosh ax dx = \frac{1}{4a}\left[-2ax+\sinh 2ax\right] \tag{120}$$

$$\int \sinh ax\cosh bx dx = \frac{1}{b^2-a^2}\left[b\cosh bx\sinh ax - a\cosh ax\sinh bx\right] \tag{121}$$

5.4 扩域

```cpp
// 扩域
struct C{
    int x, y;
    // x + y * sqrt(o);
};

int o = 2;

// fn = Aa^n + Bb^n

int inline power(int a, int b) {
    int ret = 1;
    while (b) {
        if (b & 1) ret = 1ll * ret * a % P;
        a = 1ll * a * a % P;
        b >>= 1;
    }
    return ret;
}




int mod(int x) {
    return x >= P ? x - P : x;
}

C operator + (const C &a, const C &b) {
    return (C) { mod(a.x + b.x), mod(a.y + b.y) };
};

C operator * (const C &a, const C &b) {
    C c;
    c.x = (1ll * a.x * b.x + 1ll * a.y * b.y % P * o) % P;
    c.y = (1ll * a.x * b.y + 1ll * a.y * b.x) % P;
    return c;
};
```

```
C operator * (const C &a, const int &b) {
    C c;
    c.x = 1ll * a.x * b % P;
    c.y = 1ll * a.y * b % P;

    return c;
};


C inline power(C a, int b) {
    C ret = (C) { 1, 0 } ;
    while (b) {
        if (b & 1) ret = ret * a;
        a = a * a;
        b >>= 1;
    }
    return ret;
}

C operator / (const C &a, const C &b) {
    C c, d;
    c = a;
    d = b;
    d.y = mod(P - d.y);
    c = c * d;
    int I = (((LL)b.x * b.x - (LL)b.y * b.y * o) % P + P) % P;
    I = power(I, P - 2);
    c = c * I;
    return c;
};
```

5.5 原根

```
// 原根 / 封装不太好


int n, D, phi[N], primes[N], tot, d[N], len;
int ans[N], cnt;
```

```
bool st[N], pr[N];

void inline init() {
	phi[1] = 1, pr[2] = pr[4] = true;
	for (int i = 2; i < N; i++) {
		if (!st[i]) primes[tot++] = i, phi[i] = i - 1;
		for (int j = 0; i * primes[j] < N; j++) {
			st[i * primes[j]] = true;
			if (i % primes[j] == 0) {
				phi[i * primes[j]] = phi[i] * primes[j];
				break;
			}
			phi[i * primes[j]] = phi[i] * (primes[j] - 1);
		}
	}
	for (int i = 1; i < tot; i++) {
		for (LL j = primes[i]; j < N; j *= primes[i]) pr[j] = true;
		for (LL j = 2 * primes[i]; j < N; j *= primes[i]) pr[j] = true;
	}
}


void inline factor(int m) {
	len = 0;
	for (int i = 0; i < tot && primes[i] * primes[i] <= m; i++) {
		int j = primes[i];
		if (m % j == 0) {
			d[len++] = j;
			while (m % j == 0) m /= j;
		}
	}
	if (m > 1) d[len++] = m;
}

int inline power(int a, int b, int P) {
	int res = 1;
	while (b) {
		if (b & 1) res = (LL)res * a % P;
```

```
            a = (LL)a * a % P;
            b >>= 1;
        }
        return res;
}

bool inline check(int x, int P) {
        if (power(x, phi[P], P) != 1) return false;
        for (int i = 0; i < len; i++)
                if(power(x, phi[P] / d[i], P) == 1) return false;
        return true;
}

// 输入 P, 返回最小原根

int inline get(int P) {
        for (int i = 1; i < P; i++)
                if (check(i, P)) return i;
        return 0;
}
//-
```

5.6 $O(n)$ 预处理逆元

```
void inline preInv(int n) {
    inv[1] = 1;
    for (int i = 2; i <= n; i++)
        inv[i] = ((LL)P - P / i) * inv[P % i] % P;
}
```

5.7 Exgcd 扩展欧里几得

```
LL inline exgcd(LL a, LL b, LL &x, LL &y) {
        if (b == 0) {
                x = 1, y = 0;
                return a;
        }
        LL d = exgcd(b, a % b, y, x);
```

```
        y -= a / b * x;
        return d;
}
```

5.8 扩展中国剩余定理 exCRT

```
// 扩展中国剩余定理 exCRT
typedef pair<LL, LL> PLL;

LL gcd(LL a, LL b) {
        return b ? gcd(b, a % b) : a;
}

LL exgcd(LL a, LL b, LL &x, LL &y) {
        if (!b) {
                x = 1, y = 0;
                return a;
        }
        LL d = exgcd(b, a % b, y, x);
        y -= a / b * x;
        return d;
}

LL mul(LL x, LL y, LL P) {
        return (__int128)x * y % P;
//      return x * y % P;
}


// x mod m = a (m1, a1) (m2, a2) return x


PLL inline merge(PLL A, PLL B) {
        LL a1 = A.fi, b1 = A.se;
        LL a2 = B.fi, b2 = B.se;
        LL a = a1 / gcd(a1, a2) * a2;
        LL x, y;
        LL d = exgcd(a1, a2, x, y);
        assert((b2 - b1) % d == 0);
```

```
        x = mul(x, (b2 - b1) / d, a);
        if (x < 0) x += a;
        LL o = mul(x, a1, a) + b1;
        if (o >= a) o -= a;
        PLL c = mp(a, o);
        return c;
}
```

5.9 BSGS

```
// BSGS

unordered_map<int, int> mp;

int BSGS(int a, int b, int P) {
        int t = sqrt(P) + 1; mp.clear(); b %= P;
        for (int j = 0, s = b; j < t; j++)
                mp[s] = j, s = (LL)s * a % P;
        a = power(a, t, P);
        for (int i = 1, s = 1; i <= t; i++) {
                s = (LL)s * a % P;
                if (mp.count(s) && i * t - mp[s] >= 0)
                        return i * t - mp[s];
        }
        return -1;
}

int exBSGS(int a, int b, int P) {
        int x, y, d, A = 1, k = 0;
        while ((d = gcd(a, P)) > 1) {
                if (b % d) return -1;
                b /= d, P /= d, k++, A = (LL)A * (a / d) % P;
                if (A == b) return k;
        }
        exgcd(A, P, x, y); x = (x % P + P) % P;
        int res = BSGS(a, (LL)b * x % P, P);
        return res == -1 ? -1 : res + k;
}
```

5.10 杜教筛

```
const int N = 5000005, S = 3000;
const LL INF = 9e18;

LL p1[N], p2[S], m1[N], m2[S];

int n, primes[N], tot;

bool vis[N];

// 杜教筛 phi
LL s1(int x) {
	if (x < N) return p1[x];
	else if (p2[n / x] != INF) return p2[n / x];
	LL res = x * (x + 1ll) / 2;
	for (LL l = 2, r; l <= x; l = r + 1) {
		r = x / (x / l);
		res -= (r - l + 1) * s1(x / l);
	}
	return p2[n / x] = res;
}

// 杜教筛 mu

LL s2(int x) {
	if (x < N) return m1[x];
	else if (m2[n / x] != INF) return m2[n / x];
	LL res = 1;
	for (LL l = 2, r; l <= x; l = r + 1) {
		r = x / (x / l);
		res -= (r - l + 1) * s2(x / l);
	}
	return m2[n / x] = res;
}
```

5.11 Min25

```
// Min25
```

```cpp

int inv2 = power(2, P - 2), inv6 = power(6, P - 2);

// 求 g_k 函数: <= x 的和
int inline getS(LL x, int k) {
	if (k == 1) return (x % P * (x % P + 1ll) % P * inv2 + P - 1ll)  % P;
	if (k == 2) return (P - 1ll + x % P * (x % P + 1ll) % P * (2ll * x % P + 1) % P * inv6) %
		P;
}

int inline getV(LL x, int k) {
	if (k == 1) return x % P;
	if (k == 2) return (LL)x % P * x % P;
}

bool vis[M];

int primes[M], tot;

void inline linear(int n) {
	for (int i = 2; i <= n; i++) {
		if (!vis[i]) primes[++tot] = i;
		for (int j = 1; primes[j] <= n / i; j++) {
			vis[i * primes[j]] = true;
			if (i % primes[j] == 0) break;
		}
	}
}

// 预处理 g_k 处所有 n / i 形式的质数前缀和
struct MP1{
	int m, g[M], pos1[M], pos2[M], len, id;
	LL n, d[M];
	int inline getPos(LL x) {
		return x <= m ? pos1[x] : pos2[n / x];
	}
	void inline add(LL v) {
		d[++len] = v;
		g[len] = getS(v, id);
```

```
			if (v <= m) pos1[v] = len;
			else pos2[n / v] = len;
	}
	void build(LL sum, int t) {
		m = sqrt(n = sum); id = t;
		for (LL i = 1, j; i <= n; i = j + 1) {
			LL v = n / i; j = n / v;
			if (v <= m) break;
			add(v);
		}
		for (int i = m; i; i--) add(i);
		for (int i = 1; i <= tot && (LL)primes[i] * primes[i] <= n; i++) {
			LL pr = primes[i];
			for (int j = 1; j <= len && pr * pr <= d[j]; j++) {
				int k = getPos(d[j] / pr);
				g[j] = (g[j] - (LL)getV(pr, id) * (g[k] - g[getPos(primes[i - 1])
					] + P) % P + P) % P;
			}
		}
	}
	int inline s(LL x) { return g[getPos(x)]; }
} t1, t2;

int inline get(LL x) {
	return (t2.s(x) - t1.s(x) + P) % P;
}

int inline calc(LL x) {
	return x % P * (x % P - 1ll + P) % P;
}

void inline add(int &x, int y) {
	(x += y) %= P;
}

int inline s(LL n, int t) {
	if (primes[t] >= n) return 0;
	int ans = (get(n) - get(primes[t]) + P) % P;
	for (int i = t + 1; i <= tot && (LL)primes[i] * primes[i] <= n; i++) {
```

```
                int pr = primes[i];
                LL v = pr;
                for (int j = 1; v <= n; v = v * pr, j++) {
                        add(ans, (LL)calc(v) * ((j != 1) + s(n / v, i)) % P);
                }
        }
        return ans;
}
```

5.12 FMT / FWT

```
// FMT / FWT

void inline OR(int n, int a[], int o) {
        for (int w = 1; w < n; w <<= 1)
                for (int i = 0; i < n; i += (w << 1))
                        for (int j = 0; j < w; j++)
                                add(a[i + j + w], o * a[i + j]);
}

void inline AND(int n, int a[], int o) {
        for (int w = 1; w < n; w <<= 1)
                for (int i = 0; i < n; i += (w << 1))
                        for (int j = 0; j < w; j++)
                                add(a[i + j], o * a[i + j + w]);
}


// 反向传 1/2
void inline XOR(int n, int a[], int o) {
        for (int w = 1; w < n; w <<= 1)
                for (int i = 0; i < n; i += (w << 1))
                        for (int j = 0; j < w; j++) {
                                int u = a[i + j], v = a[i + j + w];
                                a[i + j] = ((LL)u + v + P) * o % P;
                                a[i + j + w] = ((LL)u - v + P) * o % P;
                        }
}
```

5.13 子集卷积

```cpp
// 子集卷积

void inline SubConv(int n, int a[], int b[], int c[]) {
	for (int i = 0; i < (1 << n); i++) {
		f[get(i)][i] = a[i];
		g[get(i)][i] = b[i];
	}
	for (int i = 0; i <= n; i++)
		OR(1 << n, f[i], 1), OR(1 << n, g[i], 1);
	for (int i = 0; i <= n; i++)
		for (int j = 0; j <= i; j++)
			for (int k = 0; k < (1 << n); k++)
				add(h[i][k], (LL)f[j][k] * g[i - j][k] % P);
	for (int i = 0; i <= n; i++) OR(1 << n, h[i], -1);
	for (int i = 0; i < (1 << n); i++) c[i] = h[get(i)][i];
}
```

6 数据结构

6.1 线段树合并与分裂

```cpp

// 线段树合并与分裂

struct Seg{
	int idx;

	struct T{
		int l, r;
		LL v;
	} t[N * 22];

	void inline pushup(int p) {
		t[p].v = t[t[p].l].v + t[t[p].r].v;
	}

```

```
	void build(int &p, int l, int r) {
		if (!p) p = ++idx;
		if (l == r) {
			t[p].v = a[l];
			return;
		}
		int mid = (l + r) >> 1;
		build(t[p].l, l, mid);
		build(t[p].r, mid + 1, r);
		pushup(p);
	}

	void change(int &p, int &q, int l, int r, int x, int y) {
		if (x <= l && r <= y) {
			q = p; p = 0;
			return;
		}
		if (!q) q = ++idx;
		int mid = (l + r) >> 1;
		if (x <= mid) change(t[p].l, t[q].l, l, mid, x, y);
		if (mid < y) change(t[p].r, t[q].r, mid + 1, r, x, y);
		pushup(p); pushup(q);
	}

	void merge(int &p, int &q, int l, int r) {
		if (!p) return;
		if (!q) { q = p; return; }
		if (l == r) { t[q].v += t[p].v; return; }
		int mid = (l + r) >> 1;
		merge(t[p].l, t[q].l, l, mid);
		merge(t[p].r, t[q].r, mid + 1, r);
		pushup(q);
	}

	void insert(int &p, int l, int r, int x, int k) {
		if (!p) p = ++idx;
		if (l == r) { t[p].v += k; return ; }
		int mid = (l + r) >> 1;
		if (x <= mid) insert(t[p].l, l, mid, x, k);
```

```
			else insert(t[p].r, mid + 1, r, x, k);
			pushup(p);
	}

	LL query(int p, int l, int r, int x, int y) {
			if (!p) return 0;
			if (x <= l && r <= y) return t[p].v;
			int mid = (l + r) >> 1; LL res = 0;
			if (x <= mid) res += query(t[p].l, l, mid, x, y);
			if (mid < y) res += query(t[p].r, mid + 1, r, x, y);
			return res;
	}

	int kth(int p, int l, int r, int k) {
			if (l == r) return l;
			int mid = (l + r) >> 1;
			if (k <= t[t[p].l].v) return kth(t[p].l, l, mid, k);
			else return kth(t[p].r, mid + 1, r, k - t[t[p].l].v);
	}
}
```

6.2 ST 表

```
// ST 表
struct ST{
	void inline STPrework(int n) {
			g[0] = -1;
			for (int i = 1; i <= n; i++)
					f[i][0] = a[i], g[i] = g[i >> 1] + 1;
			for (int j = 1; j <= g[n]; j++)
					for (int i = 1; i + (1 << j) - 1 <= n; i++)
							f[i][j] = max(f[i][j - 1], f[i + (1 << (j - 1))][j - 1]);
	}

	int inline query(int l, int r) {
			int k = g[r - l + 1];
			return max(f[l][k], f[r - (1 << k) + 1][k]);
	}
}
```

6.3 Fhq Treap

```
// 用来动态开点的池
struct T{
	int l, r, val, rnd, sz;
} t[SZ];
int idx;

struct Fhq{
	int rt;
	void pushup(int p) {

	}
	// value(A) < value(B)
	int merge(int A, int B) {
		if (!A || !B) return A + B;
		else if(t[A].rnd > t[B].rnd) {
			t[A].r = merge(t[A].r, B);
			pushup(A);
			return A;
		} else {
			t[B].l = merge(A, t[B].l);
			pushup(B);
			return B;
		}
	}

	// 按值分裂
	void split(int p, int k, int &x, int &y) {
		if (!p) x = y = 0;
		else {
			if (t[p].val <= k)
			x = p, split(t[p].r, k, t[p].r, y);
			else y = p, split(t[p].l, k, x, t[p].l);
			pushup(p);
		}
	}
	int getNode(int val) {
		t[++idx] = (T) { 0, 0, val, rand(), 1 };
		return idx;
```

```
        }

        void insert(int val) {
                int x, y;
                split(rt, val, x, y);
                rt = merge(merge(x, getNode(val)), y);
        }

        int get(int l, int r) {
                int x, y, z;
                split(rt, l - 1, x, y);
                split(y, r, y, z);
                int res = t[y].N;
                rt = merge(x, merge(y, z));
                return res;
        }

        void del(int val) {
                int x, y, z;
                split(rt, val - 1, x, y);
                split(y, val, y, z);
                y = merge(t[y].l, t[y].r);
                rt = merge(x, merge(y, z));
        }
}
```

6.4 线段树

```
// 普通线段树

struct Seg{
        #define ls (p << 1)
        #define rs (p << 1 | 1)
        void inline pu(int p) {

        }

        void inline pd(int p) {

```

```
	}

	void bd(int p, int l, int r) {
		if(l == r) {
			return;
		}
		int mid = (l + r) >> 1;
	    bd(ls, l, mid);
	    bd(rs, mid + 1, r);
	    pu(p);
	}
	void chg(int p, int l, int r, int x, int y, int k, int c) {
	    if(x <= l && r <= y) {
	        return ;
	    }
	    int mid = (l + r) >> 1;
	    pd(p);
	    if(x <= mid) chg(ls, l, mid, x, y, k, c);
	    if(mid + 1 <= y) chg(rs, mid + 1, r, x, y, k, c);
	    pu(p);
	}

	int qry(int p, int l, int r, int x, int y) {
	    if(x <= l && r <= y) return ?;
	    int mid = (l + r) >> 1, s = 0;
	    pd(p);
	    if(x <= mid) s += qry(ls, l, mid, x, y);
	    if(mid + 1 <= y) s += qry(rs, mid + 1, r, x, y);
	    return s % P;
	}
}
```

6.5 主席树

```
// 主席树
struct PersisSeg{
	struct T{
		int l, r;
		LL v;
```

```
	} t[SZ];

	int rt[SZ], idx;

	void inline update(int &p, int q, int l, int r, int x, int k) {
		t[p = ++idx] = t[q];
		t[p].v += k;
		if (l == r) return;
		int mid = (l + r) >> 1;
		if (x <= mid) update(t[p].l, t[q].l, l, mid, x, k);
		else update(t[p].r, t[q].r, mid + 1, r, x, k);
	}

	LL inline query(int p, int l, int r, int x, int y) {
		if (!p || x > y) return 0;
		if (x <= l && r <= y) return t[p].v;
		int mid = (l + r) >> 1; LL res = 0;
		if (x <= mid) res += query(t[p].l, l, mid, x, y);
		if (mid < y) res += query(t[p].r, mid + 1, r, x, y);
		return res;
	}
}
```

6.6 树状数组：区间加区间求和

```
// 区间加 区间查的树状数组
struct exBIT{
	BIT t1, t2;
	int n;
	void inline init(int len, int a[]) {
		n = len;
		for (int i = 1; i <= n; i++)
			b[i] = a[i] - a[i - 1];
		t1.init(n, b);
		for (int i = 1; i <= n; i++) b[i] *= i;
		t2.init(n, b);
	}
	void inline add(int l, int r, LL c) {
		t1.add(l, c), t1.add(r + 1, -c);
```

```
            t2.add(l, c * l), t2.add(r + 1, -c * (r + 1));
        }
        LL inline ask(int x) {
            return (x + 1) * t1.ask(x) - t2.ask(x);
        }
        LL inline ask(int x, int y) { return ask(y) - ask(x - 1); }
};
```

6.7 LCT

```
struct LCT{
        #define get(x) (ch[fa[x]][1] == x)
        #define isRoot(x) (ch[fa[x]][0] != x && ch[fa[x]][1] != x)
        #define ls ch[p][0]
        #define rs ch[p][1]

        int ch[N][2], fa[N], mx[N], w[N], rev[N];

        void inline pushup(int p) {

        }

        void inline pushdown(int p) {
                if (rev[p]) { swap(ls, rs), rev[ls] ^= 1, rev[rs] ^= 1, rev[p] = 0; }
        }

        void inline rotate(int x) {
                int y = fa[x], z = fa[y], k = get(x);
                if (!isRoot(y)) ch[z][get(y)] = x;
                ch[y][k] = ch[x][!k], fa[ch[y][k]] = y;
                ch[x][!k] = y, fa[y] = x, fa[x] = z;
                pushup(y); pushup(x);
        }

        void inline update(int p) {
                if (!isRoot(p)) update(fa[p]);
                pushdown(p);
        }

```

```
        void inline splay(int p) {
                update(p);
                for (int f = fa[p]; !isRoot(p); rotate(p), f = fa[p])
                        if (!isRoot(f)) rotate(get(p) == get(f) ? f : p);
        }

        void inline access(int x) {
                for (int p = 0; x; p = x, x = fa[x]) {
                        splay(x), ch[x][1] = p, pushup(x);
                }
        }

        int inline find(int p) {
                access(p), splay(p);
                while (ls) pushdown(p), p = ls;
                splay(p);
                return p;
        }

        void inline makeRoot(int x) {
                access(x), splay(x), rev[x] ^= 1;
        }

        void inline split(int x, int y) {
                makeRoot(x), access(y), splay(y);
        }

        void inline link(int x, int y) {
                makeRoot(x), fa[x] = y;
        }

        void inline cut(int x, int y) {
                split(x, y);
                ch[y][0] = 0, fa[x] = 0;
                pushup(y);
        }

}
```

6.8 左偏树

```
// 左偏树
struct LeftistTree{
        struct T{
            int l, r, v, d, f;
            // l, r 表示左右儿子, v 表示值
            // d 表示从当前节点到最近叶子节点的距离, f 表示当前节点的父亲
        } t[SZ];

        int find(int x) {
            return t[x].f == x ? x : t[x].f = find(t[x].f);
        }

        int merge(int x, int y) { // 递归合并函数
            if (!x || !y) return x + y;
            if (t[x].v > t[y].v || (t[x].v == t[y].v && x > y)) swap(x, y);
            rs = merge(rs, y);
            if (t[ls].d < t[rs].d) swap(ls, rs);
            t[x].d = t[rs].d + 1;
            return x;
        }

        int work(int x, int y) { // 合并 x, y 两个堆。
            if (x == y) return 0;
                if (!x || !y) return t[x + y].f = x + y;
            if (t[x].v > t[y].v || (t[x].v == t[y].v && x > y)) swap(x, y);
            t[x].f = t[y].f = x;
            merge(x, y); return x;
        }

        void del(int x) {
            t[x].f = work(ls, rs), t[x].v = -1;
        }
}
```

6.9 李超树

```
// 李超树
```

```cpp

struct LC{
	struct Tree{
		int l, r;
		Line v;
	} t[N << 2];
	LL inline calc(Line e, LL x) {
		return e.k * x + e.b;
	}
	int idx, rt;
	void inline clr() {
		idx = 0; rt = 0;
	}
	// 这里写法非常简洁的原因是，让计算机人工帮你判断了单调 / 需要 upd 的位置，事实上只会走一边。
	void inline ins(int &p, int l, int r, Line e) {
		if (!p) {
			t[p = ++idx] = (Tree) { 0, 0, e };
			return;
		}
		int mid = (l + r) >> 1;
		if (calc(t[p].v, mid) > calc(e, mid)) swap(e, t[p].v);
		if (calc(e, l) < calc(t[p].v, l)) ins(t[p].l, l, mid, e);
		if (calc(e, r) < calc(t[p].v, r)) ins(t[p].r, mid + 1, r, e);
	}
	LL ask(int p, int l, int r, int x) {
		if (!p) return INF;
		if (l == r) return calc(t[p].v, x);
		int mid = (l + r) >> 1; LL ret = calc(t[p].v, x);
		if (x <= mid) chkMin(ret, ask(t[p].l, l, mid, x));
		else chkMin(ret, ask(t[p].r, mid + 1, r, x));
		return ret;
	}

} ;
```

6.10 回滚莫队

```cpp
// 莫队
```

```cpp

int pos[N], L[N], R[N], t;

struct Q {
	int l, r, id;
	bool operator < (const Q &b) const {
		if (pos[l] != pos[b.l]) return pos[l] < pos[b.l];
		return r < b.r;
	}
} q[N];

t = sqrt(n);
for (int i = 1; i <= n; i++) {
	pos[i] = (i - 1) / t + 1;
	if (!L[pos[i]]) L[pos[i]] = i;
	R[pos[i]] = i;
}

sort(q + 1, q + 1 + m);

// 回滚

int l = 1, r = 0, last = -1;
for (int i = 1; i <= m; i++) {
	if (pos[q[i].l] == pos[q[i].r]) {
		// 块内暴力
		continue;
	}
	if (pos[q[i].l] != last) {
		// 新的左块
		res = 0, top = 0, r = R[pos[q[i].l]], l = r + 1;
		last = pos[q[i].l];
	}
	while (r < q[i].r) {
		++r;
		// insert r
	}
	int bl = l, tp = res; // 记录
	while (l > q[i].l) {
```

```
            --l;
            // insert l
        }
        // 恢复
        ans[q[i].id] = res; res = tp;
}
```

6.11 动态凸包

```
typedef pair<LL, LL> PII;
typedef set<PII>::iterator SIT;
typedef set<PII> SI;

PII operator - (const PII &a, const PII &b) {
        return mp(a.x - b.x, a.y - b.y);
}

LL inline cross(PII a, PII b) {
        return a.x * b.y - a.y * b.x;
}

LL inline cross(PII a, PII b, PII c) {
        PII u = b - a, v = c - a;
        return cross(u, v);
}

// 动态凸包

struct Hull {
        SI su, sd;
        bool inline query(SI &s, PII u, int o) {
                SIT l = s.upper_bound(u), r = s.lower_bound(u);
                if (r == s.end() || l == s.begin()) return false;
                l--;
                return cross(*l, u, *r) * o <= 0;
        }
        void inline insert(SI &s, PII u, int o) {
                if (query(s, u, o)) return;
                SIT it = s.insert(u).first;
```

```
                while (1) {
                        SIT mid = it;
                        if (mid == s.begin()) break; --mid;
                        SIT l = mid;
                        if (l == s.begin()) break; --l;
                        if (cross(*l, *mid, u) * o >= 0) break;
                        s.erase(mid);
                }
                while (1) {
                        SIT mid = it; ++mid;
                        if (mid == s.end()) break;
                        SIT r = mid; ++r;
                        if (r == s.end()) break;
                        if (cross(u, *mid, *r) * o >= 0) break;
                        s.erase(mid);
                }
        }
        void inline ins(PII u) {
                insert(su, u, 1), insert(sd, u, -1);
        }
        int inline chk(PII u) {
                return query(su, u, 1) && query(sd, u, -1);
        }
} t;
```

6.12 珂朵莉树

```
// 珂朵莉树？？

struct E{
    int l, r, v;
    bool operator < (const E &b) const {
        return r < b.r;
    }
};

set<E> s;

typedef set<E>::iterator SIT;
```

```cpp

void split(int i) {
    SIT u = s.lower_bound((E){ 0, i + 1, 0 });
    if (u == s.end()) return;
    if (u -> r > i && u -> l <= i) {
        E t = *u;
        s.erase(u);
        s.insert((E){ t.l, i, t.v });
        s.insert((E){ i + 1, t.r, t.v });
    }
}

void inline ins(int l, int r, int v) {
        split(l - 1), split(r);
    while (1) {
        SIT u = s.lower_bound((E){ 0, l, 0, 0 });
        if (u == s.end()) break;
        if (u -> r > r) break;

        s.erase(u);
    }
    s.insert((E){ l, r, v });
}
```

6.13 HashMap

```cpp
// Hashmap


struct E{
        int next, v, w;
};

const int MOD = 999997;

struct Hash{
        E e[MOD];
        int numE, head[MOD];
        void inline clear() {
```

```
            for (int i = 1; i <= numE; i++)
                head[e[i].v % MOD] = 0;
            numE = 0;
        }
        int &operator[] (int x) {
            int t = x % MOD;
            for (int i = head[t]; i; i = e[i].next) {
                if (e[i].v == x) {
                    return e[i].w;
                }
            }
            e[++numE] = (E) { head[t], x, 0 };
            head[t] = numE;
            return e[numE].w;
        }
} t
```

6.14 全局平衡二叉树

```
// 全局平衡二叉树


vector<int> g[N];

int lim[N];

bool vis[N];

int fa[N], sz[N], son[N], d[N];

struct Mat{
        // 定义矩阵的地方
        Mat operator * (const Mat &b) const {

        }
};

void dfs1(int u) {
        sz[u] = 1;
```

```
        for (int v: g[u]) {
                if (v == fa[u]) continue;
                fa[v] = u;
                d[v] = d[u] + 1;
                dfs1(v);
                sz[u] += sz[v];
                if (sz[v] > sz[son[u]]) son[u] = v;
        }
}

int len, b[N], val[N], rt[N], ps[N];

struct T{
        int l, r, f;
        Mat v, s;
} t[N];

int inline getM(int x, int y) {
        int mn = 2e9, p = -1;
        for (int i = x; i <= y; i++)
                if (chkMin(mn, max(val[i - 1] - val[x - 1], val[y] - val[i]))) p = i;
        return p;
}

#define ls t[p].l
#define rs t[p].r

void pu(int p) {
        if (ls && rs) t[p].s = t[rs].s * t[p].v * t[ls].s;
        else if (ls) t[p].s = t[p].v * t[ls].s;
        else if (rs) t[p].s = t[rs].s * t[p].v;
        else t[p].s = t[p].v;
}

void inline bd(int &p, int l, int r, int F) {
        if (l > r) return;
        int mid = getM(l, r);
        p = b[mid];
        t[p].f = F;
```

```cpp
        bd(ls, l, mid - 1, p), bd(rs, mid + 1, r, p);
        pu(p);
}

void inline remake(int u) {
        // 更新 u 的子树了, 更新矩阵
}

void inline updF(int v) {
        // u 的轻儿子 v 变了, 更新轻儿子对自己的影响
}

void inline bd(int tp) {
        int x = tp; vector<int> z;
        while (x) z.pb(x), x = son[x];
        for (int u: z) {
                for (int v: g[u])
                        if (v != fa[u] && v != son[u]) bd(v), updF(v);
                remake(u);
        }
        len = 0;
        for (int v: z) b[++len] = v, val[len] = sz[v] - sz[son[v]];
        for (int i = 1; i <= len; i++) val[i] += val[i - 1];
        bd(rt[tp], 1, len, 0);
        ps[rt[tp]] = tp;
}

void inline sop(int x) {
        while (x) {
                remake(x); int p = x, y = 0;
                while (p) y = ps[p], pu(p), p = t[p].f;
                if (!fa[y]) break;
                updF(y), x = fa[y];
        }
}
```

7 计算几何

7.1 Basic

```
const double eps = 1e-4;
typedef pair<double, double> PDD;
struct Line{
    PDD s, t;
};

int inline cmp(double x, double y) {
    if (fabs(x - y) < eps) return 0;
    return x < y ? -1 : 1;
}

double inline cross(PDD a, PDD b) { return a.fi * b.se - a.se * b.fi; }
PDD operator - (const PDD &a, const PDD &b) { return make_pair(a.fi - b.fi, a.se - b.se); }
PDD operator + (const PDD &a, const PDD &b) { return make_pair(a.fi+ b.fi, a.se+ b.se); }
PDD operator / (const PDD &a, double b) { return make_pair(a.fi / b, a.se / b); }
PDD operator * (const PDD &a, double b) { return make_pair(a.fi * b, a.se * b); }
double inline area(PDD a, PDD b, PDD c) { return cross(b - a, c - a); }
double inline dot(PDD a, PDD b) { return a.fi * b.fi + a.se * b.se; }
double inline len(PDD a) { return sqrt(dot(a, a)); }
double inline project(PDD a, PDD b, PDD c) { return dot(b - a, c - a) / len(b - a); }
double inline dist(PDD a, PDD b) { return sqrt((a.fi - b.fi) * (a.fi - b.fi) + (a.se - b.se) * (a.se - b.se)); }
// 顺时针转 x
PDD inline rotate(PDD a, double x) { return make_pair ( cos(x) * a.fi + sin(x) * a.se, -sin(x) * a.fi + cos(x) * a.se ); }
PDD inline norm(PDD a) { return a / len(a); }
double angle(PDD a, PDD b) {
    return acos(dot(a, b) / len(a) / len(b));
}
int sign(double fi) {
    if (fabs(fi) < eps) return 0;
    if (fi < 0) return -1;
    return 1;
}
```

7.2 点到线段距离

```
LD getD(PDD a, PDD u, PDD v) {
        LD w = min(dis(a, u), dis(a, v));
        LD c = dot(a - u, v - u);
        LD t = dis(u, v);
        c /= t;
        if (cmp(c, 0) >= 0 && cmp(c, t) <= 0) {
                LD z = norm(u - a);
                LD val = sqrt(z - c * c);
                w = val;
        }
        return w;
}
```

7.3 线段交

```
bool segInter(PDD a1, PDD a2, PDD b1, PDD b2) {
    double c1 = cross(a2 - a1, b1 - a1), c2 = cross(a2 - a1, b2 - a1);
    double c3 = cross(b2 - b1, a2 - b1), c4 = cross(b2 - b1, a1 - b1);
    return sign(c1) * sign(c2) <= 0 && sign(c3) * sign(c4) <= 0;
}

bool cmp2 (const Line &a, const Line &b) {
    double A = getAngle(a), B = getAngle(b);
    if (A != B) return A < B;
    else return area(a.s, a.t, b.t) < 0;
}

PDD getInter(PDD p, PDD v, PDD q, PDD w) {
    PDD u = p - q;
    double t = cross(w, u) / cross(v, w);
    return make_pair(p.fi + t * v.fi, p.se + t * v.se);
}

PDD getInter(Line a, Line b) { return getInter(a.s, a.t - a.s, b.s, b.t - b.s); }

bool inline Right(Line a, Line b, Line c) {
    PDD u = getInter(b, c);
```

```
    return area(a.s, a.t, u) <= 0;
}
```

7.4 凸包

```
void inline andrew() {
    sort(p + 1, p + 1 + n);
    for (int i = 1; i <= n; i++) {
        while (top > 1 && area(p[s[top - 1]], p[s[top]], p[i]) < 0) {
            if (area(p[s[top - 1]], p[s[top]], p[i]) <= 0) st[s[top--]] = false;
            else top--;
        }
        st[i] = true, s[++top] = i;
    }
    st[1] = false;
    for (int i = n; i; i--) {
        if (!st[i]) {
            while (top > 1 && area(p[s[top - 1]], p[s[top]], p[i]) <= 0)
                st[s[top--]] = false;
            st[i] = true, s[++top] = i;
        }
    }
    for (int i = 0; i < top; i++) s[i] = s[i + 1];
    top--;
}
```

7.5 半平面交

```
struct Line{
    PDD s, t;
    int id;
} e[N];

// 半平面交
double HPI() {
    sort(e + 1, e + 1 + n, cmp2);
    int hh = 0, tt = -1;
    for (int i = 1; i <= n; i++) {
```

```cpp
        if (i && getAngle(e[i]) == getAngle(e[i - 1])) continue;
        while (hh < tt && Right(e[i], e[q[tt - 1]], e[q[tt]])) tt--;
        while (hh < tt && Right(e[i], e[q[hh]], e[q[hh + 1]])) hh++;
        q[++tt] = i;
    }
    while (hh < tt && Right(e[q[hh]], e[q[tt - 1]], e[q[tt]])) tt--;
    while (hh < tt && Right(e[q[tt]], e[q[hh]], e[q[hh + 1]])) hh++;
    q[++tt] = q[hh];
    tot = 0;
    for (int i = hh; i < tt; i++)
        p[++tot] = getInter(e[q[i]], e[q[i + 1]]);
    double res = 0;
    for (int i = 1; i < tot; i++)
        res += area(p[1], p[i], p[i + 1]);
    return res / 2;
}
```

7.6 最小圆覆盖

```cpp
Point inline getCircle(Point a, Point b, Point c) {
    return Inter((a + b) / 2, rotate(b - a, PI / 2), (a + c) / 2, rotate(c - a, PI / 2));
}

// 最小圆覆盖

void inline minCircle(PDD a[]) {
    random_shuffle(a + 1, a + 1 + n);
    double r = 0; Point u = a[1];
    for (int i = 2; i <= n; i++) {
        if (cmp(r, len(u - a[i])) == -1) {
            r = 0, u = a[i];
            for (int j = 1; j < i; j++) {
                if (cmp(r, len(u - a[j])) == -1) {
                    r = len(a[i] - a[j]) / 2, u = (a[i] + a[j]) / 2;
                    for (int k = 1; k < j; k++) {
                        if (cmp(r, len(u - a[k])) == -1) {
                            u = getCircle(a[i], a[j], a[k]), r = len(a[i] - u);
                        }
                    }
```

```
                }
            }
        }
    }
}
```

7.7 自适应辛普森积分

```
// 自适应辛普森积分
double inline f(double fi) {
    return ?;
}
double inline s(double l, double r) {
    double mid = (l + r) / 2;
    return (r - l) * (f(l) + 4 * f(mid) + f(r)) / 6;
}

double inline asr(double l, double r) {
    double mid = (l + r) / 2, v = s(l, r);
    double a = s(l, mid), b = s(mid, r);
    if (fabs(a + b - v) < eps) return v;
    else return asr(l, mid) + asr(mid, r);
}
```

7.8 极角排序

```
// https://codeforces.com/contest/1284/problem/E 的怨念 不丢精度的极角排序

LL inline cross(PII x, PII y) {
    return 1ll * x.fi * y.se - 1ll * x.se * y.fi;
}

int inline quad(PII x) {
    if (x.fi >= 0 && x.se >= 0) return 1;
    if (x.fi <= 0 && x.se >= 0) return 2;
    if (x.fi <= 0 && x.se <= 0) return 3;
    if (x.fi >= 0 && x.se <= 0) return 4;
    return 0;
```

```
}
```

7.9 Int 下凸包 + 闵可夫斯基和

凸包面积 $*2$ = 环相邻两个向量乘起来加起来 $(\sum p_i \times p_{(i+1)\%n})$

```
// PII andrew + mincowf


LL operator * (PII a, PII b) {
    return (LL)a.fi * b.se - (LL)a.se * b.fi;
}

PII operator + (PII a, PII b) {
    return mp(a.fi + b.fi, a.se + b.se);
}

PII operator - (PII a, PII b) {
    return mp(a.fi - b.fi, a.se - b.se);
}

LL dot (PII a, PII b) {
    return (LL)a.fi * a.se + (LL)b.fi * b.se;
}

vector<PII> inline andrew(vector<PII> a) {
    int n = a.size();
    top = 0;
    sort(a.begin(), a.end());
    for (int i = 0; i < n; i++) {
        while (top > 1 && (a[i] - a[s[top - 1]]) * (a[s[top]] - a[s[top - 1]]) > 0) {
            vis[s[top--]] = 0;
        }
        vis[i] = 1, s[++top] = i;
    }
    vis[0] = 0;
    for (int i = n - 1; i >= 0; i--) {
        if (!vis[i]) {
            while (top > 1 && (a[i] - a[s[top - 1]]) * (a[s[top]] - a[s[top - 1]]) > 0)
                vis[s[top--]] = 0;
```

```
            vis[i] = 1, s[++top] = i;
        }
    }
    --top;
    vector<PII> ret;
    for (int i = 1; i <= top; i++) ret.pb(a[s[i]]);
    for (int i = 0; i < n; i++) vis[i] = 0;
    return ret;
}

// 有

vector<PII> calc(vector<PII> a, vector<PII> b) {
    vector<PII> c;
    c.pb(a[0] + b[0]);
    vector<PII> dx, dy;
    for (int i = 1; i < a.size(); i++) dx.pb(a[i] - a[i - 1]);
    dx.pb(a[0] - a.back());
    for (int i = 1; i < b.size(); i++) dy.pb(b[i] - b[i - 1]);
    dy.pb(b[0] - b.back());
    int i = 0, j = 0;
    while (i < dx.size() && j < dy.size()) {
        if (dx[i] * dy[j] > 0)
            c.pb(c.back() + dx[i++]);
        else if (dx[i] * dy[j] == 0 && c.size() > 1) {
            // 共线放一起不然是错的！！！！
            if (dot(c.back() - c[c.size() - 2], dx[i]) > 0)
                c.pb(c.back() + dx[i++]);
            else c.pb(c.back() + dy[j++]);
        } else {
            c.pb(c.back() + dy[j++]);
        }
    }
    while (i < dx.size()) c.pb(c.back() + dx[i++]);
    while (j < dy.size()) c.pb(c.back() + dy[j++]);
    assert(c.back() == c[0]);
    c.pop_back();
    return c;
}
```