Standard Code Library

111

zu

August 7, 2022

Contents

一切的开始

宏定义

- 需要 C++11

```cpp
#include <bits/stdc++.h>
using namespace std;
using LL = long long;
#define FOR(i, x, y) for (decay<decltype(y)>::type i = (x), _##i = (y); i < _##i; ++i)
#define FORD(i, x, y) for (decay<decltype(x)>::type i = (x), _##i = (y); i > _##i; --i)
#ifdef zerol
#define dbg(x...) do { cout << "\033[32;1m" << #x << " -> "; err(x); } while (0)
void err() { cout << "\033[39;0m" << endl; }
template<template<typename...> class T, typename t, typename... A>
void err(T<t> a, A... x) { for (auto v: a) cout << v << ' '; err(x...); }
template<typename T, typename... A>
void err(T a, A... x) { cout << a << ' '; err(x...); }
#else
#define dbg(...)
#endif
// ------------------------------------------------------------------------------
```

数据结构

ST 表

- 二维

```cpp
int f[maxn][maxn][10][10];
inline int highbit(int x) { return 31 - __builtin_clz(x); }
inline int calc(int x, int y, int xx, int yy, int p, int q) {
    return max(
        max(f[x][y][p][q], f[xx - (1 << p) + 1][yy - (1 << q) + 1][p][q]),
        max(f[xx - (1 << p) + 1][y][p][q], f[x][yy - (1 << q) + 1][p][q])
    );
}
void init() {
    FOR (x, 0, highbit(n) + 1)
    FOR (y, 0, highbit(m) + 1)
        FOR (i, 0, n - (1 << x) + 1)
        FOR (j, 0, m - (1 << y) + 1) {
            if (!x && !y) { f[i][j][x][y] = a[i][j]; continue; }
            f[i][j][x][y] = calc(
                i, j,
                i + (1 << x) - 1, j + (1 << y) - 1,
                max(x - 1, 0), max(y - 1, 0)
            );
        }
}
inline int get_max(int x, int y, int xx, int yy) {
    return calc(x, y, xx, yy, highbit(xx - x + 1), highbit(yy - y + 1));
}
```

树状数组

```cpp
int t[N], n;

inline int lowbit(int x) { return x & (-x); }

void add(int x, int k) {
    while (x <= n) {
        t[x] += k;
        x += lowbit(x);
    }
}

int qry(int x) {
    int ret = 0;
```

```
    while (x) {
        ret = max(ret,t[x]);
        x -= lowbit(x);
    }
    return ret;
}
```

线段树

```
struct node {
    ll sum;
    ll plz, mlz;
    int l, r;
} tree[N * 4];

void build(int i, int l, int r) {
    tree[i].l = l;
    tree[i].r = r;
    tree[i].plz = 0;
    tree[i].mlz = 1;
    if (l == r) {
        cin >> tree[i].sum;
        tree[i].sum = tree[i].sum % p;
        return;
    }
    int mid = (l + r) >> 1;
    build(i * 2, l, mid);
    build(i * 2 + 1, mid + 1, r);
    tree[i].sum = (tree[i * 2].sum + tree[i * 2 + 1].sum) % p;
}

inline void push_down(ll i) {
    ll k1 = tree[i].mlz, k2 = tree[i].plz;
    tree[i << 1].sum = (tree[i << 1].sum * k1 + k2 * (tree[i << 1].r - tree[i << 1].l + 1)) % p;
    tree[i << 1 | 1].sum = (tree[i << 1 | 1].sum * k1 + k2 * (tree[i << 1 | 1].r - tree[i << 1 | 1].l + 1)) % p;
    tree[i << 1].mlz = (tree[i << 1].mlz * k1) % p;
    tree[i << 1 | 1].mlz = (tree[i << 1 | 1].mlz * k1) % p;
    tree[i << 1].plz = (tree[i << 1].plz * k1 + k2) % p;
    tree[i << 1 | 1].plz = (tree[i << 1 | 1].plz * k1 + k2) % p;
    tree[i].plz = 0;
    tree[i].mlz = 1;
}

inline void add(int i, int l, int r, ll k) {
    if (tree[i].l >= l && tree[i].r <= r) {
        tree[i].sum = (tree[i].sum + k * (tree[i].r - tree[i].l + 1)) % p;
        tree[i].plz += k;
        return;
    }
    push_down(i);
    if (tree[i * 2].r >= l) add(i * 2, l, r, k);
    if (tree[i * 2 + 1].l <= r) add(i * 2 + 1, l, r, k);
    tree[i].sum = tree[i * 2].sum + tree[i * 2 + 1].sum;
}

inline void mul(int i, int l, int r, ll k) {
    if (tree[i].l >= l && tree[i].r <= r) {
        tree[i].mlz = tree[i].mlz * k % p;
        tree[i].plz = tree[i].plz * k % p;
        tree[i].sum = tree[i].sum * k % p;
        return;
    }
    push_down(i);
    if (tree[i * 2].r >= l) mul(i * 2, l, r, k);
    if (tree[i * 2 + 1].l <= r) mul(i * 2 + 1, l, r, k);
    tree[i].sum = tree[i * 2].sum + tree[i * 2 + 1].sum;
}

ll search(int i, int l, int r) {
    if (tree[i].l >= l && tree[i].r <= r) {
        return tree[i].sum % p;
```

```
    }
    if (tree[i].r < l || tree[i].l > r) return 0;
    push_down(i);
    ll s = 0;
    if (tree[i * 2].r >= l) s = (s + search(i * 2, l, r)) % p;
    if (tree[i * 2 + 1].l <= r) s = (s + search(i * 2 + 1, l, r)) % p;
    return s % p;
}
```

主席树

```
#include <bits/stdc++.h>
using namespace std;
typedef long long ll;
const int N = 2e5 + 7;
struct node{
    int sum,ls,rs;
}tree[N << 5];
int tot = 0;
int a[N];
int v[N];
int rt[N];
int getid(int x,int n) {
    return lower_bound(v + 1,v + n + 1,x) - v;
}
int build(int l,int r) {
    int root = ++tot;
    tree[root].sum = 0;
    if (l >= r) return root;
    int mid = l + r >> 1;
    tree[root].ls = build(l, mid);
    tree[root].rs = build(mid + 1, r);
    return root;
}
int update(int x,int l, int r,int root) {
    int d = ++tot;
    tree[d].ls = tree[root].ls;
    tree[d].rs = tree[root].rs;
    tree[d].sum = tree[root].sum + 1;
    if (l >= r) return d;
    int mid = l + r >> 1;
    if (x <= mid) tree[d].ls = update(x, l, mid, tree[d].ls);
    else tree[d].rs = update(x, mid + 1, r, tree[d].rs);
    return d;
}
int query(int u, int v, int l, int r, int k) {
    if (l >= r) return l;
    int mid = l + r >> 1;
    int x = tree[tree[v].ls].sum - tree[tree[u].ls].sum;
    if (x >= k) return query(tree[u].ls,tree[v].ls,l,mid,k);
    else query(tree[u].rs, tree[v].rs, mid + 1, r, k - x);
}

void solve() {
    tot = 0;
    int n, m;
    cin >> n >> m;
    for (int i = 1; i <= n; ++i) {
        cin >> a[i];
        v[i] = a[i];
    }
    sort(v + 1,v + n + 1);
    int len = unique(v + 1, v + n + 1) - v - 1;
    rt[0] = build(1,len);
    for (int i = 1; i <= n; ++i) {
        rt[i] = update(getid(a[i],len),1,len,rt[i - 1]);
    }
    for (int i = 1; i <= m; ++i) {
        int l, r, k;
        cin >> l >> r >> k;
        cout << v[query(rt[l - 1],rt[r],1,len,k)] << '\n';
```

```
    }
}

# 数学

## 中国剩余定理

```c++
ll exgcd(ll a, ll b, LL &x, LL &y) {
    if (!b) {
        x = 1;
        y = 0;
        return a;
    }
    ll d = exgcd(b, a % b, x, y);
    ll c = x;
    x = y;
    y = c - (a / b) * y;
    return d;
}
ll a[N],b[N];
ll crt(int n) {
    LL sum = 1;
    LL res = 0;
    for (int i = 1; i <= n; ++i) sum *= b[i];
    for (int i = 1; i <= n; ++i) {
        LL m = sum / b[i];
        LL x,y;
        exgcd(m,b[i],x,y);
        a[i] = (a[i] + sum) % sum;
        res = (res + a[i] * m * x % sum) % sum;
    }
    return (res % sum + sum) % sum;
}
```

FFT

```
const int N = 2e6 + 7, Log = 20;
const double Pi = acos(-1);
int n, m;

struct CP {
    CP(double xx = 0, double yy = 0) { x = xx, y = yy; }

    double x, y;

    CP operator+(CP const &B) const { return CP(x + B.x, y + B.y); }

    CP operator-(CP const &B) const { return CP(x - B.x, y - B.y); }

    CP operator*(CP const &B) const { return CP(x * B.x - y * B.y, x * B.y + y * B.x); }
} f[N << 1];//只用了一个复数数组
int tr[N << 1];

void fft(CP *f, bool flag) {
    for (int i = 0; i < n; i++)
        if (i < tr[i])swap(f[i], f[tr[i]]);
    for (int p = 2; p <= n; p <<= 1) {
        int len = p >> 1;
        CP tG(cos(2 * Pi / p), sin(2 * Pi / p));
        if (!flag)tG.y *= -1;
        for (int k = 0; k < n; k += p) {
            CP buf(1, 0);
            for (int l = k; l < k + len; l++) {
                CP tt = buf * f[len + l];
                f[len + l] = f[l] - tt;
                f[l] = f[l] + tt;
                buf = buf * tG;
            }
        }
    }
```
```

```
}

void solve() {
    string a, b;
    cin >> a >> b;
    n = a.size();
    m = b.size();
    for (int i = 0; i < n; ++i) f[i].x = a[n - i - 1] - '0';
    for (int i = 0; i < m; ++i) f[i].y = b[m - i - 1] - '0';
    for (m += n, n = 1; n <= m; n <<= 1);
    for (int i = 0; i < n; ++i) tr[i] = (tr[i >> 1] >> 1) | ((i & 1) ? n >> 1 : 0);
    fft(f, 1);
    for (int i = 0; i < n; ++i) f[i] = f[i] * f[i];
    fft(f, 0);
    for (int i = 0; i <= m; ++i) f[i].y = int(f[i].y / n / 2 + 0.49);
}
```

图论

LCA

```
vector<int> e[max_n];
int fa[max_n][25];
int d[max_n];
int vis[max_n];
int n;

void dfs(int x) {
    vis[x]++;
    for (int i : e[x]) {
        if (!vis[i]) {
            d[i] = d[x] + 1;
            fa[i][0] = x;
            dfs(i);
        }
    }
}

void bz() {
    for (int j = 1; j <= 20; ++j) {
        for (int i = 1; i <= n; ++i) {
            fa[i][j] = fa[fa[i][j - 1]][j - 1];
        }
    }
}

int LCA(int u, int v) {
    if (d[u] < d[v]) swap(u, v);
    int de = d[u] - d[v];
    for (int i = 0; i <= 20; ++i) {
        if ((1<<i) & de) {
            u = fa[u][i];
        }
    }
    if (u == v) return u;
    for (int i = 20; i >= 0; --i) {
        if (fa[u][i] != fa[v][i]) {
            u = fa[u][i];
            v = fa[v][i];
        }
    }
    return fa[u][0];
}
```

最短路

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

using namespace std;
```

```cpp
typedef long long ll;
typedef pair<int, int> pii;
const ll INF = 0x3f3f3f3f3f3f3f3f;
const int inf = 0x3f3f3f3f;
const int N = 1e5 + 7;

vector<pii> e[N];
ll dis[N];
priority_queue<pii, vector<pii>, greater<>> q;
vector<int> v1;
int vis[N];

void init() {
    memset(dis, inf, sizeof dis);
    memset(vis, 0, sizeof vis);
}

void dij(int s, int n) {
    v1.push_back(s);
    int cnt = 0;
    dis[s] = 0;
    vis[s]++;
    while (cnt < n - 1) {
        for (auto &i : e[v1[cnt]]) {
            if (vis[i.second])continue;
            q.push({i.first + dis[v1[cnt]], i.second});
        }
        while (vis[q.top().second]) q.pop();
        v1.push_back(q.top().second);
        vis[q.top().second]++;
        dis[q.top().second] = q.top().first;
        q.pop();
        cnt++;
    }
}
```

最大流

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

using namespace std;
typedef long long ll;
typedef pair<int, int> pii;
const ll INF = 0x3f3f3f3f3f3f3f3f;
const int inf = 0x3f3f3f3f;
const int N = 2e2+ 7;
typedef __int128 LL;
struct node {
    int to, next;
    ll val;
} e[20011];
int cur[2001];
int head[2001];
int cnt = 1;

void add(int u, int v, ll w) {
    e[++cnt].val = w;
    e[cnt].to = v;
    e[cnt].next = head[u];
    head[u] = cnt;
}

int s, t;
int d[2001];

int bfs() {
    queue<int> q;
    q.push(s);
    memset(d, 0, sizeof d);
    memcpy(cur, head, sizeof head);
    d[s] = 1;
```

```
    while (!q.empty()) {
        int u = q.front();
        q.pop();
        for (int i = head[u]; i; i = e[i].next) {
            int v = e[i].to;
            if (e[i].val && !d[v]) q.push(v), d[v] = d[u] + 1;
        }
    }
    return d[t];
}

ll dfs(ll u, ll mn) {
    ll a;
    if (u == t) return mn;
    ll tmp = 0;
    for (int &i = cur[u]; i && mn; i = e[i].next) {
        int v = e[i].to;
        if (e[i].val && d[v] == d[u] + 1) {
            a = dfs(v, min(e[i].val, mn));
            e[i].val -= a;
            e[i ^ 1].val += a;
            mn -= a;
            tmp += a;
        }
    }
    if (!mn) d[u] = -1;
    return tmp;
}
```

费用流 (dij)

```
struct MFMC {
    struct edge {
        int v, next;
        ll f, c;
    } e[1000005];

    struct node {
        int v, e;
    } p[100005];

    struct mypair {
        ll dis;
        int id;

        bool operator<(const mypair &a) const { return dis > a.dis; }

        mypair(ll d, int x) { dis = d, id = x; }
    };

    int head[50005], vis[50005];
    ll dis[50005], h[50005];
    int n, m, s, t, cnt = 1;
    ll maxf, minc;

    void addedge(int u, int v, ll f, ll c) {
        e[++cnt].v = v;
        e[cnt].f = f;
        e[cnt].c = c;
        e[cnt].next = head[u];
        head[u] = cnt;
    }

    void add(int u, int v, ll f, ll c) {
        addedge(u, v, f, c);
        addedge(v, u, 0, -c);
    }

    bool dijkstra() {
        priority_queue<mypair> q;
        for (int i = 1; i <= n; i++) dis[i] = inf;
```

```
        memset(vis, 0, sizeof(vis));
        dis[s] = 0;
        q.push(mypair(0, s));
        while (!q.empty()) {
            int u = q.top().id;
            q.pop();
            if (vis[u]) continue;
            vis[u] = 1;
            for (int i = head[u]; i; i = e[i].next) {
                int v = e[i].v;
                ll nc = e[i].c + h[u] - h[v];
                if (e[i].f && dis[v] > dis[u] + nc) {
                    dis[v] = dis[u] + nc;
                    p[v].v = u;
                    p[v].e = i;
                    if (!vis[v]) q.push(mypair(dis[v], v));
                }
            }
        }
        return dis[t] != inf;
    }

    void spfa() {
        queue<int> q;
        memset(h, 63, sizeof(h));
        h[s] = 0, vis[s] = 1;
        q.push(s);
        while (!q.empty()) {
            int u = q.front();
            q.pop();
            vis[u] = 0;
            for (int i = head[u]; i; i = e[i].next) {
                int v = e[i].v;
                if (e[i].f && h[v] > h[u] + e[i].c) {
                    h[v] = h[u] + e[i].c;
                    if (!vis[v]) {
                        vis[v] = 1;
                        q.push(v);
                    }
                }
            }
        }
    }

    void mfmc() {
        maxf = 0;
        minc = 0;
        spfa();
        while (dijkstra()) {
            ll minf = inf;
            for (int i = 1; i <= n; i++) h[i] += dis[i];
            for (int i = t; i != s; i = p[i].v) minf = min(minf, e[p[i].e].f);
            for (int i = t; i != s; i = p[i].v) {
                e[p[i].e].f -= minf;
                e[p[i].e ^ 1].f += minf;
            }
            maxf += minf;
            minc += minf * h[t];
        }
    }

    void init() {
        cnt = 1;
        memset(head, 0, sizeof head);
    }
} mfmc;
```

费用流 (spfa)

```
const int N = 5e3 + 5, M = 1e5 + 5;
const int INF = 0x3f3f3f3f;
```

```cpp
int n, m, tot = 1, lnk[N], cur[N], ter[M], nxt[M], cap[M], cost[M], dis[N], ret;
bool vis[N];

void add(int u, int v, int w, int c) {
  ter[++tot] = v, nxt[tot] = lnk[u], lnk[u] = tot, cap[tot] = w, cost[tot] = c;
}

void addedge(int u, int v, int w, int c) { add(u, v, w, c), add(v, u, 0, -c); }

bool spfa(int s, int t) {
  memset(dis, 0x3f, sizeof(dis));
  memcpy(cur, lnk, sizeof(lnk));
  std::queue<int> q;
  q.push(s), dis[s] = 0, vis[s] = 1;
  while (!q.empty()) {
    int u = q.front();
    q.pop(), vis[u] = 0;
    for (int i = lnk[u]; i; i = nxt[i]) {
      int v = ter[i];
      if (cap[i] && dis[v] > dis[u] + cost[i]) {
        dis[v] = dis[u] + cost[i];
        if (!vis[v]) q.push(v), vis[v] = 1;
      }
    }
  }
  return dis[t] != INF;
}

int dfs(int u, int t, int flow) {
  if (u == t) return flow;
  vis[u] = 1;
  int ans = 0;
  for (int &i = cur[u]; i && ans < flow; i = nxt[i]) {
    int v = ter[i];
    if (!vis[v] && cap[i] && dis[v] == dis[u] + cost[i]) {
      int x = dfs(v, t, std::min(cap[i], flow - ans));
      if (x) ret += x * cost[i], cap[i] -= x, cap[i ^ 1] += x, ans += x;
    }
  }
  vis[u] = 0;
  return ans;
}

int mcmf(int s, int t) {
  int ans = 0;
  while (spfa(s, t)) {
    int x;
    while ((x = dfs(s, t, INF))) ans += x;
  }
  return ans;
}
```

差分约束

```cpp
xa - xb >= c --> add(a, b, -c);
xa - xb <= c --> add(b, a, c);
xa == xb --> add(b, a, 0), add(a, b, 0);
```

Kruskal 重构树

```cpp
struct kruskal {
    struct edge {
        int u, v, w;
    } ed[N];

    vector<int> e[N];
    int w[N];
    int n, m;
    int fa[N];
    int root;
```

```
    int find(int x) {
        return fa[x] == x ? x : fa[x] = find(fa[x]);
    }

    void init() {
        for (int i = 1; i <= 2 * n; ++i) {
            fa[i] = i;
            e[i].clear();
            w[i] = 0;
        }
        root = build();
    }

    int build() {
        int cnt = n;
        for (int i = 1; i <= m; ++i) {
            int u = find(ed[i].u);
            int v = find(ed[i].v);
            if (u != v) {
                w[++cnt] = ed[i].w;
                fa[u] = fa[v] = cnt;
                e[cnt].push_back(u);
                e[cnt].push_back(v);
            }
        }
        return cnt;
    }

} k;
```

计算几何

二维几何：点与向量

```
#define y1 yy1
#define nxt(i) ((i + 1) % s.size())
typedef double LD;
const LD PI = 3.14159265358979323846;
const LD eps = 1E-10;
int sgn(LD x) { return fabs(x) < eps ? 0 : (x > 0 ? 1 : -1); }
struct L;
struct P;
typedef P V;
struct P {
    LD x, y;
    explicit P(LD x = 0, LD y = 0): x(x), y(y) {}
    explicit P(const L& l);
};
struct L {
    P s, t;
    L() {}
    L(P s, P t): s(s), t(t) {}
};

P operator + (const P& a, const P& b) { return P(a.x + b.x, a.y + b.y); }
P operator - (const P& a, const P& b) { return P(a.x - b.x, a.y - b.y); }
P operator * (const P& a, LD k) { return P(a.x * k, a.y * k); }
P operator / (const P& a, LD k) { return P(a.x / k, a.y / k); }
inline bool operator < (const P& a, const P& b) {
    return sgn(a.x - b.x) < 0 || (sgn(a.x - b.x) == 0 && sgn(a.y - b.y) < 0);
}
bool operator == (const P& a, const P& b) { return !sgn(a.x - b.x) && !sgn(a.y - b.y); }
P::P(const L& l) { *this = l.t - l.s; }
ostream &operator << (ostream &os, const P &p) {
    return (os << "(" << p.x << "," << p.y << ")");
}
istream &operator >> (istream &is, P &p) {
    return (is >> p.x >> p.y);
}

```

```
LD dist(const P& p) { return sqrt(p.x * p.x + p.y * p.y); }
LD dot(const V& a, const V& b) { return a.x * b.x + a.y * b.y; }
LD det(const V& a, const V& b) { return a.x * b.y - a.y * b.x; }
LD cross(const P& s, const P& t, const P& o = P()) { return det(s - o, t - o); }
// ------------------------------------------
```

点

```
const double eps = 1e-8;
const double pi = acos(-1.0);

int sgn(double x) {
    if (fabs(x) < eps) return 0;
    if (x < 0) return -1;
    return 1;
}

struct Point {
    double x, y;

    Point() {}

    Point(double _x, double _y) {
        x = _x;
        y = _y;
    }

    bool operator==(Point b) const {
        return sgn(x - b.x) == 0 && sgn(y - b.y) == 0;
    }

    bool operator<(Point b) const {
        return sgn(x - b.x) == 0 ? sgn(y - b.y) < 0 : x < b.x;
    }

    Point operator-(const Point &b) const {
        return Point(x - b.x, y - b.y);
    }

    Point operator+(const Point &b) const {
        return Point(x + b.x, y + b.y);
    }

    Point operator*(const double &k) const {
        return Point(x * k, y * k);
    }

    Point operator/(const double &k) const {
        return Point(x / k, y / k);
    }

    double operator^(const Point &b) const {
        return x * b.y - y * b.x;
    }

    double operator*(const Point &b) const {
        return x * b.x + y * b.y;
    }


    double len() {
        return hypot(x, y);
    }

    double len2() {
        return x * x + y * y;
    }

    double distance(Point p) {
        return hypot(x - p.x, y - p.y);
    }
```

```
    double rad(Point a, Point b) {
        Point p = *this;
        return fabs(atan2(fabs((a - p) ^ (b - p)), (a - p) * (b - p)));
    }

    Point trunc(double r) {
        double l = len();
        if (!sgn(l)) return *this;
        r /= l;
        return Point(x * r, y * r);
    }

    Point rotleft() {
        return Point(-y, x);
    }

    Point rotright() {
        return Point(y, -x);
    }

    Point rotate(Point p, double angle) {
        Point v = (*this) - p;
        double c = cos(angle), s = sin(angle);
        return Point(p.x + v.x * c - v.y * s, p.y + v.x * s + v.y * c);
    }
};
```

圆

```
struct circle {
    Point p;
    double r;

    circle() {}

    circle(Point _p, double _r) {
        p = _p;
        r = _r;
    }

    circle(double _x, double _y, double _r) {
        p = Point(_x, _y);
        r = _r;
    }

    bool operator==(circle v) const {
        return (p == v.p) && sgn(r - v.r) == 0;
    }

    bool operator<(circle v) const {
        return ((p < v.p) || ((p == v.p) && sgn(r - v.r) < 0));
    }

    double area() {
        return pi * r * r;
    }

    double circumference() {
        return 2 * pi * r;
    }

    int relation(Point b) {
        double dis = b.distance(p);
        if (sgn(dis - r) < 0) return 2; //圆内
        if (sgn(dis - r) == 0) return 1; //圆上
        return 0; //圆外
    }

    int relation(circle v) {
        double d = p.distance(v.p);
```

```
        if (sgn(d - r - v.r) > 0) return 5;
        if (sgn(d - r - v.r) == 0) return 4;
        double l = fabs(r - v.r);
        if (sgn(d - l) > 0) return 3;
        if (sgn(d - l) == 0) return 2;
        return 1;
    }

    int pointcrosscircle(circle v, Point &p1, Point &p2) {
        int rel = relation(v);
        if (rel == 5) return 0;
        if (rel == 1) {
            if (sgn(r - v.r) < 0) return 0;
            return 3;
        }
        double d = p.distance(v.p);
        double l = (d * d + r * r - v.r * v.r) / (2 * d);
        double h = sqrt(r * r - l * l);
        Point tmp = p + (v.p - p).trunc(l);
        p1 = tmp + ((v.p - p).rotleft().trunc(h));
        p2 = tmp + ((v.p - p).rotright().trunc(h));
        if (rel == 2 || rel == 4) return 1;
        return 2;
    }
}
```

字符串

AC 自动机

```
#include <bits/stdc++.h>
using namespace std;
const int N = 1e6 + 7;
int z[N][26];
int fail[N];
int res[N];
int cnt = 0;
int re = 0;i
nt mp[N];
void insert(string s) {
    int r = 0;
    re++;
    for (int i = 0; i < s.size(); ++i) {
        if (!z[r][s[i] - 'a']) {
            z[r][s[i] - 'a'] = ++cnt;
        }
        r = z[r][s[i] - 'a'];
    }
    mp[re] = r;
}
void bfs() {
    queue <int> q;
    for (int i = 0; i < 26; ++i) {
        if (z[0][i]) {
            fail[z[0][i]] = 0;
            q.push(z[0][i]);
        }
    }
    while (!q.empty()) {
        int now = q.front();
        q.pop();
        for (int i = 0; i < 26; ++i) {
            if (z[now][i]) {
                fail[z[now][i]] = z[fail[now]][i];
                q.push(z[now][i]);
            }
            else z[now][i] = z[fail[now]][i];
        }
    }
}
```

```
void quert(string s) {
    int now = 0;
    int ans = 0;
    for (int i = 0; i < s.size(); ++i) {
        now = z[now][s[i]-'a'];
        for (int j = now;j ; j = fail[j]) {
            res[j]++;;
        }
    }
}int n;
string ss[155];
void solve() {
    memset(z,0,sizeof z);
    memset(res,0,sizeof res);
    memset(fail,0,sizeof fail);
    cnt = 0;
    re = 0;
    for (int i = 1; i <= n; ++i) {
        cin >> ss[i];
        insert(ss[i]);
    }
    bfs();
    string s;
    cin >> s;
    quert(s);
    int tmp = 0;
    int ans;
    for (int i = 1; i <= n; ++i) {
        if (res[mp[i]] > tmp) {
            tmp = res[mp[i]];
            ans = i;
        }
    }
    cout << res[mp[ans]] << '\n';
    for (int i = 1; i <= n; ++i) {
        if (res[mp[i]] == tmp) {
            cout << ss[i] << '\n';
        }
    }

}
int main() {
    ios::sync_with_stdio(0);
    while (cin >> n && n) {
        solve();
    }
}
```

KMP

```
void get(string s) {
    int j = 0, k = -1;
    next[0] = -1;
    while (j < s.length()) {
        if (k == -1 || s[j] == s[k]) {
            j++, k++;
            if (s[j] != s[k]) {
                next[j] = k;
            }
            else next[j] = next[k];
        }
        else k = next[k];
    }
}
```

SA（dc3）

```
//大小开 3 倍
/*
    suffix[i]: 以 i 为起始位置的后缀
```

```
    sa[i]: 排名第 i 的后缀的起始位置
    rk[i]: 表示 suffix[i] 的排名
    height[i]: suffix(sa[i-1]) 和 suffix(sa[i]) 的最长公共前缀
        · h[i] = height[rak[i]], h[i] >= h[i-1]-1
        · suffix[i] 和 suffix[j] 之间的最长公共前缀 = min(height[rak[i]+1]...height[rak[j]])
*/

#define F(x) ((x) / 3 + ((x) % 3 == 1 ? 0 : tb))
#define G(x) ((x) < tb ? (x) * 3 + 1 : ((x) - tb) * 3 + 2)

int wa[N], wb[N], wss[N], wv[N], sa[N * 3];
int rk[N], height[N], r[N], lcp[N][30];


int c0(int *r, int a, int b) {
    return r[a] == r[b] && r[a + 1] == r[b + 1] && r[a + 2] == r[b + 2];
}

int c12(int k, int *r, int a, int b) {
    if (k == 2)
        return r[a] < r[b] || r[a] == r[b] && c12(1, r, a + 1, b + 1);
    return r[a] < r[b] || r[a] == r[b] && wv[a + 1] < wv[b + 1];
}

void Rsort(int *r, int *a, int *b, int n, int m) {
    for (int i = 0; i < n; i++) wv[i] = r[a[i]];
    for (int i = 0; i < m; i++) wss[i] = 0;
    for (int i = 0; i < n; i++) wss[wv[i]]++;
    for (int i = 1; i < m; i++) wss[i] += wss[i - 1];
    for (int i = n - 1; i >= 0; i--) b[--wss[wv[i]]] = a[i];
}

void dc3(int *r, int *sa, int n, int m) {
    int i, j, *rn = r + n, *san = sa + n, ta = 0, tb = (n + 1) / 3, tbc = 0, p;
    r[n] = r[n + 1] = 0;
    for (i = 0; i < n; i++) if (i % 3 != 0) wa[tbc++] = i;
    Rsort(r + 2, wa, wb, tbc, m);
    Rsort(r + 1, wb, wa, tbc, m);
    Rsort(r, wa, wb, tbc, m);
    for (p = 1, rn[F(wb[0])] = 0, i = 1; i < tbc; i++)
        rn[F(wb[i])] = c0(r, wb[i - 1], wb[i]) ? p - 1 : p++;
    if (p < tbc) dc3(rn, san, tbc, p);
    else for (i = 0; i < tbc; i++) san[rn[i]] = i;
    for (i = 0; i < tbc; i++) if (san[i] < tb) wb[ta++] = san[i] * 3;
    if (n % 3 == 1) wb[ta++] = n - 1;
    Rsort(r, wb, wa, ta, m);
    for (i = 0; i < tbc; i++) wv[wb[i] = G(san[i])] = i;
    for (i = 0, j = 0, p = 0; i < ta && j < tbc; p++)
        sa[p] = c12(wb[j] % 3, r, wa[i], wb[j]) ? wa[i++] : wb[j++];
    for (; i < ta; p++) sa[p] = wa[i++];
    for (; j < tbc; p++) sa[p] = wb[j++];
}

void calHeight(int *r, int *sa, int n) {
    int i, j, k = 0;
    for (i = 1; i <= n; i++) rk[sa[i]] = i;
    for (i = 0; i < n; height[rk[i++]] = k)
        for (k ? k-- : 0, j = sa[rk[i] - 1]; r[i + k] == r[j + k]; k++);

    // for (int i = 1; i <= n; ++i) {
    //     dbg(i, height[i]);
    // }
    // 用 getLcp 要去掉下面的注释
    // for (int i = 1; i <= n; ++i) lcp[i][0] = height[i];
    // for (int l = 1; (1 << l) <= n; l++) {
    //     for (int i = 1; i + (1 << l) - 1 <= n; ++i) {
    //         lcp[i][l] = min(lcp[i][l - 1], lcp[i + (1 << (l - 1))][l - 1]);
    //     }
    // }
}

```

```cpp
int getLcp(int i, int j, int n) {
    if (i == j) return n - i;
    int l = rk[i], r = rk[j];
    if (l > r) swap(l, r);
    l++;
    int k = __lg(r - l + 1);
    return min(lcp[l][k], lcp[r - (1 << k) + 1][k]);
}

char s[N];

void solve() {
    int n = 0;
    cin >> s;
    for (int i = 0; s[i]; ++i) {
        r[n++] = s[i];
    }
    r[n] = 0;
    dc3(r, sa, n + 1, 256);
    calHeight(r, sa, n);
    for (int i = 1; i <= n; ++i) {
        cout << sa[i] + 1 << " \n" [i == n];
    }
    for (int i = 1; i <= n; ++i) {
        cout << height[i] << " \n"[i == n];
    }
    // 不同子串个数
    /*
    ll ans = 1ll * n * (n + 1) / 2;
    for (int i = 1; i <= n; ++i) {
        ans -= height[i];
    }
    // 两个串的最长公共子串
    int n = 0;
    scanf("%s",s);
    scanf("%s",t);
    int l = strlen(s);
    s[l] = '!';
    int tag = l;
    for (int i = 0; t[i] ; ++i) {
        s[++l] += t[i];
    }
    for (int i = 0; s[i]; ++i) {
        r[n++] = s[i];
    }
    r[n] = 0;
    dc3(r, sa, n + 1, 256);
    calHeight(r, sa, n);
    int ans = 0;
    for (int i = 1; i <= n; ++i) {
        int x1 = sa[i - 1],x2 = sa[i];
        if ((x1 < tag && x2 > tag) || (x1 > tag && x2 < tag)) {
            ans = max(ans,height[i]);
        }
    }
    printf("%lld",ans);
    //不同公共子串的个数
    ll ans = 0;
    int tmp = 0;
    for (int i = 1; i <= n; ++i) {
        int x1 = sa[i - 1], x2 = sa[i];
        if ((x1 < tag && x2 > tag) || (x1 > tag && x2 < tag)) {
            ans += height[i];
            if (tmp > 0) ans -= min(getLcp(sa[i], sa[tmp], n), getLcp(sa[i - 1], sa[tmp - 1], n));
            tmp = i;
        }
    }
    printf("%lld", ans);
    */
}
```

SAM

```cpp
struct SAM {
    struct state {
        int len, link;
        map<char, int> next;
    };

    state st[N * 2];
    int sz, last;
    int cnt[N * 2];
    int siz[N * 2];
    int a[N * 2];

    void init() {
        st[0].len = 0;
        st[0].link = -1;
        sz = 1;
        last = 0;
    }

    void extend(char c) {
        int cur = sz++;
        st[cur].len = st[last].len + 1;
        int p = last;
        while (p != -1 && !st[p].next.count(c)) {
            st[p].next[c] = cur;
            p = st[p].link;
        }
        if (p == -1) {
            st[cur].link = 0;
        } else {
            int q = st[p].next[c];
            if (st[p].len + 1 == st[q].len) {
                st[cur].link = q;
            } else {
                int clone = sz++;
                st[clone].len = st[p].len + 1;
                st[clone].next = st[q].next;
                st[clone].link = st[q].link;
                while (p != -1 && st[p].next[c] == q) {
                    st[p].next[c] = clone;
                    p = st[p].link;
                }
                st[q].link = st[cur].link = clone;
            }
        }
        last = cur;
        siz[cur]++;
    }

    void run() { // 求子串出现次数
        ll ans = 0;
        for (int i = 1; i <= sz; ++i) cnt[st[i].len]++;
        for (int i = 1; i <= sz; ++i) cnt[i] += cnt[i - 1];
        for (int i = 1; i <= sz; ++i) a[cnt[st[i].len]--] = i;
        for (int i = sz; i; --i) {
            int p = a[i];
            siz[st[p].link] += siz[p];
            if (siz[p] > 1) ans = max(ans, 1LL * siz[p] * st[p].len);
        }
        cout << ans;
    }
    /* 不同子串个数
    void run1() {
        for (int i = 1; i <= sz; ++i) cnt[st[i].len]++;
        for (int i = 1; i <= sz; ++i) cnt[i] += cnt[i - 1];
        for (int i = 1; i <= sz; ++i) a[cnt[st[i].len]--] = i;
        for (int i = sz; i >= 0; --i) {
            int p = a[i];
            siz[p] = 1;
```

```
            for (auto j : st[p].next) {
                siz[p] += siz[j.second];
            }
        }
        cout << siz[0] - 1;
    }
    void run2(int x) {
        dp[x] = 1;
        for (auto i : st[x].next) {
            if (!dp[i.second]) run2(i.second);
            dp[x] += dp[i.second];
        }
    }
    dp[0] - 1;
    void run3() {
        ll ans = 0;
        for (int i = 1; i <= sz; ++i) {
            ans += st[i].len - st[st[i].link].len;
        }
        cout << ans;
    }
    */

    string lcs(string t) { //最长公共子串
        int v = 0, l = 0, mx = 0, mx_end = 0;
        for (int i = 1; i <= t.size(); ++i) {
            while (v && !st[v].next.count(t[i - 1])) {
                v = st[v].link;
                l = st[v].len;
            }
            if (st[v].next.count(t[i - 1])) {
                v = st[v].next[t[i - 1]];
                l++;
            }
            if (l > mx) {
                mx = l;
                mx_end = i;
            }
        }
        return t.substr(mx_end - mx + 1, mx);
    }
} sam;


void solve() {
    string s;
    cin >> s;
    sam.init();
    for (char i: s) {
        sam.extend(i);
    }
}
```

PAM

```
struct PAM {
    struct PAM_Trie {
        int ch[26];
        int fail, len, num, pre; //pre 回文串串个数，num 以 i 结尾回文串个数
    } b[N];
    int last, cnt, s[N];

    void init() {
        s[0] = 26;
        b[0].len = 0;
        b[1].len = -1;
        b[0].fail = 1;
        b[1].fail = 0;
        last = 0;
        cnt = 1;
    }
```

```
    int get_fail(int x, int n) {
        while (s[n - b[x].len - 1] != s[n]) x = b[x].fail;
        return x;
    }

    void insert(int n) {
        int p = get_fail(last, n);
        if (!b[p].ch[s[n]]) {
            b[++cnt].len = b[p].len + 2;
            int tmp = get_fail(b[p].fail, n);
            b[cnt].fail = b[tmp].ch[s[n]];
            b[cnt].num = b[b[cnt].fail].num + 1;
            b[p].ch[s[n]] = cnt;
        }
        last = b[p].ch[s[n]];
        b[last].pre++;
    }

    ll run() {
        ll ans = 0;
        for (int i = cnt; i ; --i) {
            b[b[i].fail].pre += b[i].pre;
            ans = max(ans,1ll * b[i].pre * b[i].len);
        }
        return ans;
    }

} pam;

void solve() {
    string s;
    cin >> s;
    s = ' ' + s;
    int ans = 0;
    pam.init();
    for (int i = 1; i < s.size(); ++i) {
        s[i] = (s[i] - 97 + ans) % 26 + 97;
        pam.s[i] = s[i] - 'a';
        pam.insert(i);
        ans = pam.b[pam.last].num;
        cout << ans << ' ';
    }
}
```

杂项

STL

- copy

```
template <class InputIterator, class OutputIterator>
  OutputIterator copy (InputIterator first, InputIterator last, OutputIterator result);
```

int_128

```
typedef __int128 ll;
inline __int128 read() {
    __int128 x = 0, f = 1;
    char ch = getchar();
    while (ch < '0' || ch > '9') {
        if (ch == '-') f = -1;
        ch = getchar();
    }
    while (ch >= '0' && ch <= '9') {
        x = x * 10 + ch - '0';
        ch = getchar();
    }
    return x * f;
```

```
}

inline void print(__int128 x) {
    if (x < 0) {
        putchar(-1);
        x = -x;
    }
    if (x > 9) print(x / 10);
    putchar(x % 10 + '0');
}
```

背包

```
int val[105];
int num[105];
int nva;
int dp[100010];

void ZeroOnePack(int cost, int weight) {
    for (int i = nva; i >= cost; i--)
        dp[i] = max(dp[i], dp[i - cost] + weight);
}

void CompletePack(int cost, int weight) {
    for (int i = cost; i <= nva; ++i)
        dp[i] = max(dp[i], dp[i - cost] + weight);
}

void MultiplePack(int cost, int weight, int amount) {
    if (cost * amount >= nva) {
        CompletePack(cost, weight);
        return;
    }
    for (int i = 1; i < amount; i <<= 1) {
        ZeroOnePack(cost * i, weight * i);
        amount -= i;
    }
    if (amount > 0) ZeroOnePack(cost * amount, weight * amount);
}

int main() {
    int n, m;
    while (cin >> n >> m && n && m) {
        for (int i = 0; i < n; ++i) {
            cin >> val[i];
        }
        for (int i = 0; i < n; ++i) {
            cin >> num[i];
        }
        nva = m;
        memset(dp, 0, sizeof dp);
        for (int i = 0; i < n; ++i) {
            MultiplePack(val[i], val[i], num[i]);
        }
        int ans = 0;
        for (int i = 1; i <= m; ++i) {
            if (dp[i] == i) ans++;
        }
        cout << ans << endl;
    }
}
```

高精度

```
string a[100][100];
int t[100];
int w[100];
string ans;
string pl(string x,string y) {
    reverse(x.begin(),x.end());
```

```
    reverse(y.begin(),y.end());
    int z[1000];
    memset(z,0,sizeof z);
    for (int i = 0; i < min(x.size(),y.size()); ++i) {
        z[i] = x[i] -'0' + y[i] - '0';
    }
    for (int i = min(x.size(),y.size()); i < max(x.size(),y.size()); ++i) {
        if (x.size() < y.size()) z[i] = y[i] - '0';
        else z[i] = x[i] - '0';
    }
    int cnt = max(x.size(),y.size());
    for (int i = 0; i < cnt; ++i) {
        If (z[i] >= 10) {
            z[i + 1] += (z[i]) / 10;
            z[i] = (z[i]) % 10;
            if (i == cnt - 1) cnt++;
        }
    }
    string p;
    for (int i = 0; i < cnt; ++i) {
        p += z[i] + '0';
    }
    reverse(p.begin(),p.end());
    return p;
}
string me(string x, string y) {
    reverse(x.begin(),x.end());
    reverse(y.begin(),y.end());
    int z[1000];
    memset(z,0,sizeof z);
    for (int i = 0; i < y.size(); ++i) {
        for (int j = 0; j < x.size(); ++j) {
            z[i + j] += (x[j] - '0') * (y[i] - '0');
        }
    }
    int cnt = max(x.size(),y.size());
    for (int i = 0; i < cnt; ++i) {
        if (z[i] >= 10) {
            z[i + 1] += (z[i]) / 10;
            z[i] = (z[i]) % 10;
            if (i == cnt - 1) cnt++;
        }
    }
    string p;
    for (int i = 0; i < cnt; ++i) {
        p += z[i] + '0';
    }
    reverse(p.begin(),p.end());
    return p;
}
```