PHarr XCPC ex Templates

PHarr

Southwest Minzu Universtiy

November 27, 2024

Contents

编程技巧和基础算法

Linux 下运行脚本

将以下脚本保存为 `run.sh`，如果要编译则`./run.sh A.cpp`。

如果遇到了权限不足的情况可以 `chmod +x ./run.sh` 或者使用 `sudo`

还有一种使用方法是 `bash ./run.sh A`

```
#!/bin/bash
g++ $1.cpp -o $1 -g -O2 -std=c++20 \
-Wall -fsanitize=undefined -fsanitize=address \
&& echo compile_successfully >&2 && ./$1
```

如果用文件输入输出可以使用

```
#!/bin/bash
g++ $1.cpp -o $1 -g -O2 -std=c++20 \
-Wall -fsanitize=undefined -fsanitize=address \
&& echo compile_successfully >&2 && ./$1 < in.txt > out.txt
```

以下是一个可以在 MAC OS 上使用的版本

```
#!/bin/zsh
g++-11 $1.cpp -o $1 -g -O2 -std=c++20 \
-Wall -fsanitize=undefined -fsanitize=address \
&& echo compile_successfully >&2 && ./$1
```

mt19937

```
std::mt19937 rd(std::random_device{}());
```

chrono

```
#include <iostream>
#include <chrono>

using namespace std;


int main() {
    auto start = chrono::high_resolution_clock::now();

    int n = 1e8;
    while(n --);

    auto end = chrono::high_resolution_clock::now();

    auto duration = chrono::duration_cast<chrono::milliseconds>(end - start);
    cerr << duration.count() << " milliseconds" << endl;
    return 0;
}
```

数据结构

图论

差分约束系统

定义

差分约束系统是一种特殊的 n 元一次不等式组，它包含 n 个变量 $x_1, x_2, \ldots, x_n$ 以及 m 个约束条件，每个约束条件是由两个其中的变量做差构成的，形如 $x_i - x_j \leq c_k$，其中 $1 \leq i, j \leq n, i \neq j, 1 \leq k \leq m$ 并且 c_k 是常数（可以是非负数，也可以是负数）。我们要解决的问题是：求一组解 $x_1 = a_1, x_2 = a_2, \ldots, x_n = a_n$，使得所有的约束条件得到满足，否则判断出无解。

差分约束系统中的每个约束条件 $x_i - x_j \le c_k$ 都可以变形成 $x_i \le x_j + c_k$，这与单源最短路中的三角形不等式 $dist[y] \le dist[x] + z$ 非常相似。因此，我们可以把每个变量 x_i 看做图中的一个结点，对于每个约束条件 $x_i - x_j \le c_k$，从结点 j 向结点 i 连一条长度为 c_k 的有向边。

注意到，如果 $\{a_1, a_2, \dots, a_n\}$ 是该差分约束系统的一组解，那么对于任意的常数 d，$\{a_1 + d, a_2 + d, \dots, a_n + d\}$ 显然也是该差分约束系统的一组解，因为这样做差后 d 刚好被消掉。

过程

设 $dist[0] = 0$ 并向每一个点连一条权重为 0 边，跑单源最短路，若图中存在负环，则给定的差分约束系统无解，否则，$x_i = dist[i]$ 为该差分约束系统的一组解。

性质

一般使用 Bellman–Ford 或队列优化的 Bellman–Ford（俗称 SPFA，在某些随机图跑得很快）判断图中是否存在负环，最坏时间复杂度为 $O(nm)$。

如果题目给定了一个源点，则不需要建立超级源点。

```cpp
// luogu P1993
// 有三种约束条件
// x[a] >= x[b] + c -> x[b] - x[a] <= -c -> add(a, b, -c)
// x[a] <= x[b] + c -> x[a] - x[b] <= c -> add(b, a, c)
// x[a] == x[b] -> x[a] - x[b] <= 0 and x[b] - x[a] <= 0 -> add(a, b, 0), add(b, a, 0)
#include <bits/stdc++.h>

using namespace std;

const int inf = INT_MAX / 2;

using vi = vector<int>;
using pii = pair<int, int>;

int main() {
    ios::sync_with_stdio(false), cin.tie(nullptr);
    int n, m;
    cin >> n >> m;

    vector<vector<pii>> e(n + 1);
    for (int op, a, b, c; m; m--) {
        cin >> op;
        if (op == 1) {
            cin >> a >> b >> c;
            e[a].emplace_back(b, -c);
        } else if (op == 2) {
            cin >> a >> b >> c;
            e[b].emplace_back(a, c);
        } else {
            cin >> a >> b;
            e[a].emplace_back(b, 0);
            e[b].emplace_back(a, 0);
        }
    }

    for (int i = 1; i <= n; i++)
        e[0].emplace_back(i, 0);

    vector<int> dis(n + 1, inf), vis(n + 1), tot(n + 1);
    dis[0] = 0, vis[0] = 1, tot[0]++;
    bool ok = true;
    queue<int> q;
    q.push(0);
    while (not q.empty() and ok) {
        int x = q.front();
        q.pop();
        vis[x] = 0;
        for (auto [y, w]: e[x]) {
            if (dis[y] <= dis[x] + w) continue;
            dis[y] = dis[x] + w;
            if (vis[y] != 0) continue;
            vis[y] = 1;
```

```
            q.push(y);
            tot[y]++;
            if (tot[y] > n) {
                ok = false;
                break;
            }

        }
    }

    if (ok) cout << "Yes\n";
    else cout << "No\n";
    return 0;
}
```

数学知识

计算几何

已知正方形对角线两点坐标，求另外两点坐标

按照顺时针方向，正方形上四点 A, B, C, D，两对角线交点 O。现在已知 $A(ax, ay), C(cx, cy)$ 求另外两点坐标。

令 $\vec{v} = \frac{C-A}{2} = (\frac{cx-ax}{2}, \frac{cy-ay}{2})$，则有 $O = A + \vec{v} = C - \vec{v} = (ax + vx, ay + vy) = (cx - vx, cy - vy)$

根据正方形的对称性可知

$$B = (ox - vy, oy + vx) = (ax + vx - vy, cy + vx - vy)D = (ox + vy, oy - vx) = (cx - vx + vy, ay - vx + vy)$$

令 $vp = vx - vy = \frac{cx-ax-cy+ay}{2}$ 分别代入上式子可得

$$B = (ax + vp, cy + vp)C = (cx - vp, ay - vp)$$

计算三角形面积

对于一个三角形记 $\vec{A} = \vec{ca}, \vec{B} = \vec{cb}$, 三角形的面积就是 $\frac{1}{2}|\vec{A} \times \vec{B}|$

多边形的面积

假设 n 个点的多边形，n 个点按照逆时针顺序标记为 $p_0, p_1, p_2, \dots, p_{n-1}$, 任取一个辅助点记为 O，记向量 $\vec{v_i} = p_i - O$。那么这个多边形的面积可以表示为

$$S = \frac{1}{2} \sum \vec{v}_i \times \vec{v}_{(i+1) \mod n}$$

博弈论

Lasker’s Nim Game

n 堆石子，每次玩家可以从一堆石子中取走若干个石子，或者把一堆石子分成两个非空的堆

考虑暴力的求解每一堆石子的 SG 函数

$$SG(x) = \begin{cases} 0 & x = 0 \\ \text{mex}\{SG(0)\} = 1 & x = 1 \\ \text{mex}\{SG(x-1), SG(x-2), \dots, \text{mex}\{SG(y) \oplus SG(z) | (y > 0 \wedge z > 0 \wedge y + z = 0)\}\} & x \geq 2 \end{cases}$$

然后我们打表找规律可以得到

$$SG(x) = \begin{cases} 0 & x = 0 \\ x & x = 4k + 1 \vee x = 4k + 2 \\ x \oplus 1 & x = 4k + 3 \vee x = 4x + 4 \end{cases}$$

移动棋子

有一个 $1 \times n$ 的棋盘，其中每个格子可以有多个棋子，每次可以选择一个棋子，将其移动到更左边的任意一个格子，两个人轮流移动，不能移动则输。

考虑每一个棋子的 SG 函数。一个棋子如果在从左向右第 $i(i \geq 0)$ 个格子，则 $SG(i) = i$。

[HNOI2007] 分裂游戏

有 n 堆石子，每堆有 a_i 个石子，保证 $0 \leq n \leq 21, 0 \leq a_i \leq 10^4$。两个玩家轮流操作，每次可以从第 i 堆拿出一个石子，并在 $j, k(i < j \leq k \leq n)$ 堆中各放入一个石子。不能操作的人输。

求出每一个石子的 SG 函数，一个在位置 i 的石子 $SG(i) = \text{mex}\{SG(l) \oplus SG(r) | i < j \leq k \leq n\}$。可以 $O(N^3)$ 预处理每一个石子的 SG 函数。

Green Hackenbush Game on Tree(树上删边游戏)

给一个有根树（森林）, 每次可以删掉一个子树。

叶子点的 SG 值为 0，非叶子点的 $SG(u) = \oplus[SG(v) + 1]$，$v$ 是 u 的子节点。

jly 平面几何

```cpp
/**   平面几何（Point）
 *    2023-09-22: https://qoj.ac/submission/185408
**/
template<class T>
struct Point {
    T x;
    T y;
    Point(const T &x_ = 0, const T &y_ = 0) : x(x_), y(y_) {}

    template<class U>
    operator Point<U>() {
        return Point<U>(U(x), U(y));
    }
    Point &operator+=(const Point &p) & {
        x += p.x;
        y += p.y;
        return *this;
    }
    Point &operator-=(const Point &p) & {
        x -= p.x;
        y -= p.y;
        return *this;
    }
    Point &operator*=(const T &v) & {
        x *= v;
        y *= v;
        return *this;
    }
    Point &operator/=(const T &v) & {
        x /= v;
        y /= v;
        return *this;
    }
    Point operator-() const {
        return Point(-x, -y);
    }
    friend Point operator+(Point a, const Point &b) {
        return a += b;
    }
    friend Point operator-(Point a, const Point &b) {
        return a -= b;
    }
    friend Point operator*(Point a, const T &b) {
        return a *= b;
    }
    friend Point operator/(Point a, const T &b) {
        return a /= b;
    }
    friend Point operator*(const T &a, Point b) {
        return b *= a;
```

```
    }
    friend bool operator==(const Point &a, const Point &b) {
        return a.x == b.x && a.y == b.y;
    }
    friend std::istream &operator>>(std::istream &is, Point &p) {
        return is >> p.x >> p.y;
    }
    friend std::ostream &operator<<(std::ostream &os, const Point &p) {
        return os << "(" << p.x << ", " << p.y << ")";
    }
};

template<class T>
struct Line {
    Point<T> a;
    Point<T> b;
    Line(const Point<T> &a_ = Point<T>(), const Point<T> &b_ = Point<T>()) : a(a_), b(b_) {}
};

template<class T>
T dot(const Point<T> &a, const Point<T> &b) {
    return a.x * b.x + a.y * b.y;
}

template<class T>
T cross(const Point<T> &a, const Point<T> &b) {
    return a.x * b.y - a.y * b.x;
}

template<class T>
T square(const Point<T> &p) {
    return dot(p, p);
}

template<class T>
double length(const Point<T> &p) {
    return std::sqrt(square(p));
}

template<class T>
double length(const Line<T> &l) {
    return length(l.a - l.b);
}

template<class T>
Point<T> normalize(const Point<T> &p) {
    return p / length(p);
}

template<class T>
bool parallel(const Line<T> &l1, const Line<T> &l2) {
    return cross(l1.b - l1.a, l2.b - l2.a) == 0;
}

template<class T>
double distance(const Point<T> &a, const Point<T> &b) {
    return length(a - b);
}

template<class T>
double distancePL(const Point<T> &p, const Line<T> &l) {
    return std::abs(cross(l.a - l.b, l.a - p)) / length(l);
}

template<class T>
double distancePS(const Point<T> &p, const Line<T> &l) {
    if (dot(p - l.a, l.b - l.a) < 0) {
        return distance(p, l.a);
    }
    if (dot(p - l.b, l.a - l.b) < 0) {
        return distance(p, l.b);
```

```
    }
    return distancePL(p, l);
}

template<class T>
Point<T> rotate(const Point<T> &a) {
    return Point(-a.y, a.x);
}

template<class T>
int sgn(const Point<T> &a) {
    return a.y > 0 || (a.y == 0 && a.x > 0) ? 1 : -1;
}

template<class T>
bool pointOnLineLeft(const Point<T> &p, const Line<T> &l) {
    return cross(l.b - l.a, p - l.a) > 0;
}

template<class T>
Point<T> lineIntersection(const Line<T> &l1, const Line<T> &l2) {
    return l1.a + (l1.b - l1.a) * (cross(l2.b - l2.a, l1.a - l2.a) / cross(l2.b - l2.a, l1.a - l1.b));
}

template<class T>
bool pointOnSegment(const Point<T> &p, const Line<T> &l) {
    return cross(p - l.a, l.b - l.a) == 0 && std::min(l.a.x, l.b.x) <= p.x && p.x <= std::max(l.a.x, l.b.x)
        && std::min(l.a.y, l.b.y) <= p.y && p.y <= std::max(l.a.y, l.b.y);
}

template<class T>
bool pointInPolygon(const Point<T> &a, const std::vector<Point<T>> &p) {
    int n = p.size();
    for (int i = 0; i < n; i++) {
        if (pointOnSegment(a, Line(p[i], p[(i + 1) % n]))) {
            return true;
        }
    }

    int t = 0;
    for (int i = 0; i < n; i++) {
        auto u = p[i];
        auto v = p[(i + 1) % n];
        if (u.x < a.x && v.x >= a.x && pointOnLineLeft(a, Line(v, u))) {
            t ^= 1;
        }
        if (u.x >= a.x && v.x < a.x && pointOnLineLeft(a, Line(u, v))) {
            t ^= 1;
        }
    }

    return t == 1;
}

// 0 : not intersect
// 1 : strictly intersect
// 2 : overlap
// 3 : intersect at endpoint
template<class T>
std::tuple<int, Point<T>, Point<T>> segmentIntersection(const Line<T> &l1, const Line<T> &l2) {
    if (std::max(l1.a.x, l1.b.x) < std::min(l2.a.x, l2.b.x)) {
        return {0, Point<T>(), Point<T>()};
    }
    if (std::min(l1.a.x, l1.b.x) > std::max(l2.a.x, l2.b.x)) {
        return {0, Point<T>(), Point<T>()};
    }
    if (std::max(l1.a.y, l1.b.y) < std::min(l2.a.y, l2.b.y)) {
        return {0, Point<T>(), Point<T>()};
    }
    if (std::min(l1.a.y, l1.b.y) > std::max(l2.a.y, l2.b.y)) {
        return {0, Point<T>(), Point<T>()};
```

```
    }
    if (cross(l1.b - l1.a, l2.b - l2.a) == 0) {
        if (cross(l1.b - l1.a, l2.a - l1.a) != 0) {
            return {0, Point<T>(), Point<T>()};
        } else {
            auto maxx1 = std::max(l1.a.x, l1.b.x);
            auto minx1 = std::min(l1.a.x, l1.b.x);
            auto maxy1 = std::max(l1.a.y, l1.b.y);
            auto miny1 = std::min(l1.a.y, l1.b.y);
            auto maxx2 = std::max(l2.a.x, l2.b.x);
            auto minx2 = std::min(l2.a.x, l2.b.x);
            auto maxy2 = std::max(l2.a.y, l2.b.y);
            auto miny2 = std::min(l2.a.y, l2.b.y);
            Point<T> p1(std::max(minx1, minx2), std::max(miny1, miny2));
            Point<T> p2(std::min(maxx1, maxx2), std::min(maxy1, maxy2));
            if (!pointOnSegment(p1, l1)) {
                std::swap(p1.y, p2.y);
            }
            if (p1 == p2) {
                return {3, p1, p2};
            } else {
                return {2, p1, p2};
            }
        }
    }
    auto cp1 = cross(l2.a - l1.a, l2.b - l1.a);
    auto cp2 = cross(l2.a - l1.b, l2.b - l1.b);
    auto cp3 = cross(l1.a - l2.a, l1.b - l2.a);
    auto cp4 = cross(l1.a - l2.b, l1.b - l2.b);

    if ((cp1 > 0 && cp2 > 0) || (cp1 < 0 && cp2 < 0) || (cp3 > 0 && cp4 > 0) || (cp3 < 0 && cp4 < 0)) {
        return {0, Point<T>(), Point<T>()};
    }

    Point p = lineIntersection(l1, l2);
    if (cp1 != 0 && cp2 != 0 && cp3 != 0 && cp4 != 0) {
        return {1, p, p};
    } else {
        return {3, p, p};
    }
}

template<class T>
double distanceSS(const Line<T> &l1, const Line<T> &l2) {
    if (std::get<0>(segmentIntersection(l1, l2)) != 0) {
        return 0.0;
    }
    return std::min({distancePS(l1.a, l2), distancePS(l1.b, l2), distancePS(l2.a, l1), distancePS(l2.b, l1)});
}

template<class T>
bool segmentInPolygon(const Line<T> &l, const std::vector<Point<T>> &p) {
    int n = p.size();
    if (!pointInPolygon(l.a, p)) {
        return false;
    }
    if (!pointInPolygon(l.b, p)) {
        return false;
    }
    for (int i = 0; i < n; i++) {
        auto u = p[i];
        auto v = p[(i + 1) % n];
        auto w = p[(i + 2) % n];
        auto [t, p1, p2] = segmentIntersection(l, Line(u, v));

        if (t == 1) {
            return false;
        }
        if (t == 0) {
            continue;
        }
```

```
        if (t == 2) {
            if (pointOnSegment(v, l) && v != l.a && v != l.b) {
                if (cross(v - u, w - v) > 0) {
                    return false;
                }
            }
        } else {
            if (p1 != u && p1 != v) {
                if (pointOnLineLeft(l.a, Line(v, u))
                    || pointOnLineLeft(l.b, Line(v, u))) {
                    return false;
                }
            } else if (p1 == v) {
                if (l.a == v) {
                    if (pointOnLineLeft(u, l)) {
                        if (pointOnLineLeft(w, l)
                            && pointOnLineLeft(w, Line(u, v))) {
                            return false;
                        }
                    } else {
                        if (pointOnLineLeft(w, l)
                            || pointOnLineLeft(w, Line(u, v))) {
                            return false;
                        }
                    }
                } else if (l.b == v) {
                    if (pointOnLineLeft(u, Line(l.b, l.a))) {
                        if (pointOnLineLeft(w, Line(l.b, l.a))
                            && pointOnLineLeft(w, Line(u, v))) {
                            return false;
                        }
                    } else {
                        if (pointOnLineLeft(w, Line(l.b, l.a))
                            || pointOnLineLeft(w, Line(u, v))) {
                            return false;
                        }
                    }
                } else {
                    if (pointOnLineLeft(u, l)) {
                        if (pointOnLineLeft(w, Line(l.b, l.a))
                            || pointOnLineLeft(w, Line(u, v))) {
                            return false;
                        }
                    } else {
                        if (pointOnLineLeft(w, l)
                            || pointOnLineLeft(w, Line(u, v))) {
                            return false;
                        }
                    }
                }
            }
        }
    }
    return true;
}

template<class T>
std::vector<Point<T>> hp(std::vector<Line<T>> lines) {
    std::sort(lines.begin(), lines.end(), [&](auto l1, auto l2) {
        auto d1 = l1.b - l1.a;
        auto d2 = l2.b - l2.a;

        if (sgn(d1) != sgn(d2)) {
            return sgn(d1) == 1;
        }

        return cross(d1, d2) > 0;
    });

    std::deque<Line<T>> ls;
    std::deque<Point<T>> ps;
```

```
    for (auto l : lines) {
        if (ls.empty()) {
            ls.push_back(l);
            continue;
        }

        while (!ps.empty() && !pointOnLineLeft(ps.back(), l)) {
            ps.pop_back();
            ls.pop_back();
        }

        while (!ps.empty() && !pointOnLineLeft(ps[0], l)) {
            ps.pop_front();
            ls.pop_front();
        }

        if (cross(l.b - l.a, ls.back().b - ls.back().a) == 0) {
            if (dot(l.b - l.a, ls.back().b - ls.back().a) > 0) {

                if (!pointOnLineLeft(ls.back().a, l)) {
                    assert(ls.size() == 1);
                    ls[0] = l;
                }
                continue;
            }
            return {};
        }

        ps.push_back(lineIntersection(ls.back(), l));
        ls.push_back(l);
    }

    while (!ps.empty() && !pointOnLineLeft(ps.back(), ls[0])) {
        ps.pop_back();
        ls.pop_back();
    }
    if (ls.size() <= 2) {
        return {};
    }
    ps.push_back(lineIntersection(ls[0], ls.back()));

    return std::vector(ps.begin(), ps.end());
}

using real = long double;
using P = Point<real>;

constexpr real eps = 0;

```

jly 立体几何

```
/**   立体几何（Point）
 *    2023-09-25 (i64): https://qoj.ac/submission/188519
 *    2023-09-28 (double): https://qoj.ac/submission/190463
**/
using i64 = long long;
using real = double;

struct Point {
    real x = 0;
    real y = 0;
    real z = 0;
};

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

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

```
}

Point operator*(const Point &a, real b) {
    return {a.x * b, a.y * b, a.z * b};
}

Point operator/(const Point &a, real b) {
    return {a.x / b, a.y / b, a.z / b};
}

real length(const Point &a) {
    return std::hypot(a.x, a.y, a.z);
}

Point normalize(const Point &a) {
    real l = length(a);
    return {a.x / l, a.y / l, a.z / l};
}

real getAng(real a, real b, real c) {
    return std::acos((a * a + b * b - c * c) / 2 / a / b);
}

std::ostream &operator<<(std::ostream &os, const Point &a) {
    return os << "(" << a.x << ", " << a.y << ", " << a.z << ")";
}

real dot(const Point &a, const Point &b) {
    return a.x * b.x + a.y * b.y + a.z * b.z;
}

Point cross(const Point &a, const Point &b) {
    return {
        a.y * b.z - a.z * b.y,
        a.z * b.x - a.x * b.z,
        a.x * b.y - a.y * b.x
    };
}
```

jly 静态凸包

```
/**   静态凸包（with. Point，新版）
 *    2024-04-06: https://qoj.ac/submission/379920)
**/
struct Point {
    i64 x;
    i64 y;
    Point() : x{0}, y{0} {}
    Point(i64 x_, i64 y_) : x{x_}, y{y_} {}
};

i64 dot(Point a, Point b) {
    return a.x * b.x + a.y * b.y;
}

i64 cross(Point a, Point b) {
    return a.x * b.y - a.y * b.x;
}

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

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

auto getHull(std::vector<Point> p) {
    std::sort(p.begin(), p.end(),
        [&](auto a, auto b) {
```

```
            return a.x < b.x || (a.x == b.x && a.y < b.y);
        });

    std::vector<Point> hi, lo;
    for (auto p : p) {
        while (hi.size() > 1 && cross(hi.back() - hi[hi.size() - 2], p - hi.back()) >= 0) {
            hi.pop_back();
        }
        while (!hi.empty() && hi.back().x == p.x) {
            hi.pop_back();
        }
        hi.push_back(p);
        while (lo.size() > 1 && cross(lo.back() - lo[lo.size() - 2], p - lo.back()) <= 0) {
            lo.pop_back();
        }
        if (lo.empty() || lo.back().x < p.x) {
            lo.push_back(p);
        }
    }
    return std::make_pair(hi, lo);
}

const double inf = INFINITY;

```

jly 素数测试与因式分解（Miller-Rabin & Pollard-Rho）

```
/**   素数测试与因式分解（Miller-Rabin & Pollard-Rho）
 *    2023-05-16: https://cf.dianhsu.com/gym/104354/submission/206130894
**/
i64 mul(i64 a, i64 b, i64 m) {
    return static_cast<__int128>(a) * b % m;
}
i64 power(i64 a, i64 b, i64 m) {
    i64 res = 1 % m;
    for (; b; b >>= 1, a = mul(a, a, m))
        if (b & 1)
            res = mul(res, a, m);
    return res;
}
bool isprime(i64 n) {
    if (n < 2)
        return false;
    static constexpr int A[] = {2, 3, 5, 7, 11, 13, 17, 19, 23};
    int s = __builtin_ctzll(n - 1);
    i64 d = (n - 1) >> s;
    for (auto a : A) {
        if (a == n)
            return true;
        i64 x = power(a, d, n);
        if (x == 1 || x == n - 1)
            continue;
        bool ok = false;
        for (int i = 0; i < s - 1; ++i) {
            x = mul(x, x, n);
            if (x == n - 1) {
                ok = true;
                break;
            }
        }
        if (!ok)
            return false;
    }
    return true;
}
std::vector<i64> factorize(i64 n) {
    std::vector<i64> p;
    std::function<void(i64)> f = [&](i64 n) {
        if (n <= 10000) {
            for (int i = 2; i * i <= n; ++i)
                for (; n % i == 0; n /= i)
```

```
                p.push_back(i);
            if (n > 1)
                p.push_back(n);
            return;
        }
        if (isprime(n)) {
            p.push_back(n);
            return;
        }
        auto g = [&](i64 x) {
            return (mul(x, x, n) + 1) % n;
        };
        i64 x0 = 2;
        while (true) {
            i64 x = x0;
            i64 y = x0;
            i64 d = 1;
            i64 power = 1, lam = 0;
            i64 v = 1;
            while (d == 1) {
                y = g(y);
                ++lam;
                v = mul(v, std::abs(x - y), n);
                if (lam % 127 == 0) {
                    d = std::gcd(v, n);
                    v = 1;
                }
                if (power == lam) {
                    x = y;
                    power *= 2;
                    lam = 0;
                    d = std::gcd(v, n);
                    v = 1;
                }
            }
            if (d != n) {
                f(d);
                f(n / d);
                return;
            }
            ++x0;
        }
    };
    f(n);
    std::sort(p.begin(), p.end());
    return p;
}

```

高斯消元

整数版

```
// The 15-th BIT Campus Programming Contest - Onsite Round, problem: (J) Teacher Long and Machine Learning
using vi = vector<int>;

vi gauss(int n, vector<vi> a) { // a is n * (n + 1), 1 base
    for(int i = 1; i <= n; i++) {
        int r = i;
        for(int j = i + 1; j <= n; j++)
            if(abs(a[r][i]) < abs(a[j][i]))
                r = j;

        if(abs(a[r][i]) == 0)
            return vi();

        if(i != r ) swap(a[i], a[r]);
        int div = a[i][i];
        for(int j = i; j <= n + 1; j++) {
            if(a[i][j] % div != 0) return vi();
            a[i][j] /= div;
```

```
        }

        for(int j = i + 1; j <= n; j++) {
            div = a[j][i];
            for(int k = i; k <= n + 1; k++)
                a[j][k] -= a[i][k] * div;
        }
    }
    vi ret(n + 1);
    ret[n] = a[n][n + 1];
    for(int i= n - 1 ; i >= 1; i --) {
        ret[i] = a[i][n + 1];
        for(int j=i + 1; j <= n;j++)
            ret[i] -= (a[i][j] * ret[j]);
    }
    return ret;
}
```

浮点数

```
const double eps = 1e-6;

vector<double> gauss(int n, vector<vector<double>> a) { // a is n * (n + 1), 1 base
    for (int i = 1; i <= n; i++) {
        int r = i;
        for (int j = i + 1; j <= n; j++)
            if (abs(a[r][i]) < abs(a[j][i]))
                r = j;
        if (abs(a[r][i]) < eps) return vector<double>();
        if (i != r) swap(a[i], a[r]);
        double div = a[i][i];
        for (int j = i; j <= n + 1; j++)
            a[i][j] /= div;
        for (int j = i + 1; j <= n; j++) {
            div = a[j][i];
            for (int k = i; k <= n + 1; k++)
                a[j][k] -= a[i][k] * div;
        }
    }
    vector<double> ret(n + 1);
    ret[n] = a[n][n + 1];
    for (int i = n - 1; i >= 1; i--) {
        ret[i] = a[i][n + 1];
        for (int j = i + 1; j <= n; j++) {
            ret[i] -= a[i][j] * ret[j];
        }
    }
    return ret;
}
```

求行列式，浮点数

```
double gauss(int n, vector<vector<double>> a) { // a is n * n, 0 base
    double det = 1;
    for (int i = 0; i < n; ++i) {
        int k = i;
        for (int j = i + 1; j < n; ++j)
            if (abs(a[j][i]) > abs(a[k][i])) k = j;
        if (abs(a[k][i]) < EPS) return 0;
        swap(a[i], a[k]);
        if (i != k) det = -det;
        det *= a[i][i];
        for (int j = i + 1; j < n; ++j) a[i][j] /= a[i][i];
        for (int j = 0; j < n; ++j)
            if (j != i && abs(a[j][i]) > EPS)
                for (int k = i + 1; k < n; ++k) a[j][k] -= a[i][k] * a[j][i];
    }
    return det;
}
```

求行列式，整数，对 p 取模，不需要保证 p 为质数

```
// luogo P7112, 1 <= n <= 600
```

```cpp
using i64 = long long;
using vi = vector<i64>;

i64 gauss(i64 n, vector<vi> a, i64 p) { // a is n * n, 1 base
    i64 det = 1, w = 1;
    for (int i = 1; i <= n; i++) {
        for (int j = i + 1; j <= n; j++) {
            while (a[i][i]) {
                int div = a[j][i] / a[i][i];
                for (int k = i; k <= n; k++)
                    a[j][k] = (a[j][k] - div * a[i][k] % p + p) % p;
                swap(a[i], a[j]), w = -w;
            }
            swap(a[i], a[j]), w = -w;
        }
    }
    for (int i = 1; i <= n; i++) det = det * a[i][i] % p;
    det = ((det * w) % p + p) % p;
    return det;
}
```

自适应 Simpson 积分

对于一些比较难求导的函数求解近似的积分。下面的函数可以任意替换，模板题要求精度 10^{-6}。

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

using namespace std;

using i32 = int32_t;

double a, b, c, d;

double f(double x) {
    return (c * x + d) / (a * x + b);
}

double simpson(double l, double r) {
    double mid = (l + r) / 2;
    return (r - l) * (f(l) + 4 * f(mid) + f(r)) / 6;  // 辛普森公式
}

double asr(double l, double r, double eps, double ans, int step) {
    double mid = (l + r) / 2;
    double fl = simpson(l, mid), fr = simpson(mid, r);
    if (abs(fl + fr - ans) <= 15 * eps && step < 0)
        return fl + fr + (fl + fr - ans) / 15;  // 足够相似的话就直接返回
    return asr(l, mid, eps / 2, fl, step - 1) +
           asr(mid, r, eps / 2, fr, step - 1);  // 否则分割成两段递归求解
}

double calc(double l, double r, double eps) {
    return asr(l, r, eps, simpson(l, r), 12);
}

i32 main() {
    ios::sync_with_stdio(false), cin.tie(nullptr);
    cin >> a >> b >> c >> d;

    double l, r;
    cin >> l >> r;
    cout << fixed << setprecision(6) << calc(l, r, 1e-6);
    return 0;
}
```

字符串

jly AC 自动机

```cpp
/**   AC 自动机（AhoCorasick, with string 新版）
 *    2024-04-09: https://www.luogu.com.cn/record/155114676 【模板】
**/
struct AhoCorasick {
    static constexpr int ALPHABET = 26;
    struct Node {
        int len;
        int link;
        std::array<int, ALPHABET> next;
        Node() : len{0}, link{0}, next{} {}
    };

    std::vector<Node> t;

    AhoCorasick() {
        init();
    }

    void init() {
        t.assign(2, Node());
        t[0].next.fill(1);
        t[0].len = -1;
    }

    int newNode() {
        t.emplace_back();
        return t.size() - 1;
    }

    int add(const std::string &a) {
        int p = 1;
        for (auto c : a) {
            int x = c - 'a';
            if (t[p].next[x] == 0) {
                t[p].next[x] = newNode();
                t[t[p].next[x]].len = t[p].len + 1;
            }
            p = t[p].next[x];
        }
        return p;
    }

    void work() {
        std::queue<int> q;
        q.push(1);

        while (!q.empty()) {
            int x = q.front();
            q.pop();

            for (int i = 0; i < ALPHABET; i++) {
                if (t[x].next[i] == 0) {
                    t[x].next[i] = t[t[x].link].next[i];
                } else {
                    t[t[x].next[i]].link = t[t[x].link].next[i];
                    q.push(t[x].next[i]);
                }
            }
        }
    }

    int next(int p, int x) {
        return t[p].next[x];
    }

    int link(int p) {
        return t[p].link;
```

```
    }

    int len(int p) {
        return t[p].len;
    }

    int size() {
        return t.size();
    }
};

/**   AC 自动机（AhoCorasick, with vector 新版）
 *    2023-04-07: https://codeforces.com/contest/1801/submission/201155712
**/
struct AhoCorasick {
    static constexpr int ALPHABET = 26;
    struct Node {
        int len;
        int link;
        std::array<int, ALPHABET> next;
        Node() : link{}, next{} {}
    };

    std::vector<Node> t;

    AhoCorasick() {
        init();
    }

    void init() {
        t.assign(2, Node());
        t[0].next.fill(1);
        t[0].len = -1;
    }

    int newNode() {
        t.emplace_back();
        return t.size() - 1;
    }

    int add(const std::vector<int> &a) {
        int p = 1;
        for (auto x : a) {
            if (t[p].next[x] == 0) {
                t[p].next[x] = newNode();
                t[t[p].next[x]].len = t[p].len + 1;
            }
            p = t[p].next[x];
        }
        return p;
    }

    int add(const std::string &a, char offset = 'a') {
        std::vector<int> b(a.size());
        for (int i = 0; i < a.size(); i++) {
            b[i] = a[i] - offset;
        }
        return add(b);
    }

    void work() {
        std::queue<int> q;
        q.push(1);

        while (!q.empty()) {
            int x = q.front();
            q.pop();

            for (int i = 0; i < ALPHABET; i++) {
                if (t[x].next[i] == 0) {
                    t[x].next[i] = t[t[x].link].next[i];
```

```
                } else {
                    t[t[x].next[i]].link = t[t[x].link].next[i];
                    q.push(t[x].next[i]);
                }
            }
        }
    }

    int next(int p, int x) {
        return t[p].next[x];
    }

    int next(int p, char c, char offset = 'a') {
        return next(p, c - 'a');
    }

    int link(int p) {
        return t[p].link;
    }

    int len(int p) {
        return t[p].len;
    }

    int size() {
        return t.size();
    }
};

```

jly 最长公共前缀 LCP

```
/**   最长公共前缀 LCP（例题）
 *    2024-03-02: https://qoj.ac/submission/343378
**/
constexpr int L = 2E6 + 10;

int len[L];
int lnk[L];
int nxt[L][26];

int f[L];
int tot = 1;

std::vector<int> adj[L];

int extend(int p, int c) {
    if (nxt[p][c]) {
        int q = nxt[p][c];
        if (len[q] == len[p] + 1) {
            return q;
        }
        int r = ++tot;
        len[r] = len[p] + 1;
        lnk[r] = lnk[q];
        std::copy(nxt[q], nxt[q] + 26, nxt[r]);
        lnk[q] = r;
        while (nxt[p][c] == q) {
            nxt[p][c] = r;
            p = lnk[p];
        }
        return r;
    }
    int cur = ++tot;
    len[cur] = len[p] + 1;
    while (!nxt[p][c]) {
        nxt[p][c] = cur;
        p = lnk[p];
    }
    lnk[cur] = extend(p, c);
    return cur;
```

```cpp
}

int main() {
    std::ios::sync_with_stdio(false);
    std::cin.tie(nullptr);

    std::fill(nxt[0], nxt[0] + 26, 1);
    len[0] = -1;

    int N;
    std::cin >> N;

    std::vector<std::string> S(N);
    for (int i = 0; i < N; i++) {
        std::cin >> S[i];
        int p = 1;
        for (auto c : S[i]) {
            p = extend(p, c - 'a');
            if (f[p] != -1) {
                if (f[p] == 0) {
                    f[p] = i + 1;
                } else if (f[p] != i + 1) {
                    f[p] = -1;
                }
            }
        }
    }

    for (int i = 1; i <= tot; i++) {
        adj[lnk[i]].push_back(i);
    }
}

```

动态规划