

PHarr XCPC ex Templates

PHarr

Southwest Minzu Universtiy

November 27, 2024

Contents

编程技巧和基础算法	2
Linux 下运行脚本	2
mt19937	2
chrono	2
数据结构	2
图论	2
差分约束系统	2
数学知识	4
计算几何	4
博弈论	4
jly 平面几何	5
jly 立体几何	10
jly 静态凸包	11
jly 素数测试与因式分解 (Miller-Rabin & Pollard-Rho)	12
高斯消元	13
自适应 Simpson 积分	15
字符串	16
jly AC 自动机	16
jly 最长公共前缀 LCP	18
动态规划	19

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`

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

如果用文件输入输出可以使用

```
1 #!/bin/bash
2 g++ $1.cpp -o $1 -g -O2 -std=c++20 \
3 -Wall -fsanitize=undefined -fsanitize=address \
4 && 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

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

mt19937

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

chrono

```
1 #include <iostream>
2 #include <chrono>
3
4 using namespace std;
5
6
7 int main() {
8     auto start = chrono::high_resolution_clock::now();
9
10    int n = 1e8;
11    while(n --);
12
13    auto end = chrono::high_resolution_clock::now();
14
15    auto duration = chrono::duration_cast<chrono::milliseconds>(end - start);
16    cerr << duration.count() << " milliseconds" << endl;
17    return 0;
18 }
```

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, \dots, 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 $x_1 = a_1, x_2 = a_2, \dots, x_n = a_n$ ，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 $x_i - x_j \leq c_k$ 都可以变形成 $x_i \leq x_j + c_k$ ，这与单源最短路中的三角形不等式 $dist[y] \leq dist[x] + z$ 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 $x_i - x_j \leq c_k$ ，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。

```

1 // luogu P1993
2 // 有三种约束条件
3 //  $x[a] \geq x[b] + c \rightarrow x[b] - x[a] \leq -c \rightarrow add(a, b, -c)$ 
4 //  $x[a] \leq x[b] + c \rightarrow x[a] - x[b] \leq c \rightarrow add(b, a, c)$ 
5 //  $x[a] == x[b] \rightarrow x[a] - x[b] \leq 0 \text{ and } x[b] - x[a] \leq 0 \rightarrow add(a, b, 0), add(b, a, 0)$ 
6 #include <bits/stdc++.h>
7
8 using namespace std;
9
10 const int inf = INT_MAX / 2;
11
12 using vi = vector<int>;
13 using pii = pair<int, int>;
14
15 int main() {
16     ios::sync_with_stdio(false), cin.tie(nullptr);
17     int n, m;
18     cin >> n >> m;
19
20     vector<vector<pii>> e(n + 1);
21     for (int op, a, b, c; m; m--) {
22         cin >> op;
23         if (op == 1) {
24             cin >> a >> b >> c;
25             e[a].emplace_back(b, -c);
26         } else if (op == 2) {
27             cin >> a >> b >> c;
28             e[b].emplace_back(a, c);
29         } else {
30             cin >> a >> b;
31             e[a].emplace_back(b, 0);
32             e[b].emplace_back(a, 0);
33         }
34     }
35
36     for (int i = 1; i <= n; i++)
37         e[0].emplace_back(i, 0);
38
39     vector<int> dis(n + 1, inf), vis(n + 1), tot(n + 1);
40     dis[0] = 0, vis[0] = 1, tot[0]++;
41     bool ok = true;
42     queue<int> q;
43     q.push(0);
44     while (not q.empty() and ok) {
45         int x = q.front();
46         q.pop();
47         vis[x] = 0;
48         for (auto [y, w]: e[x]) {
49             if (dis[y] <= dis[x] + w) continue;
50             dis[y] = dis[x] + w;
51             if (vis[y] != 0) continue;
52             vis[y] = 1;

```

```

53         q.push(y);
54         tot[y]++;
55         if (tot[y] > n) {
56             ok = false;
57             break;
58         }
59     }
60 }
61 }
62
63 if (ok) cout << "Yes\n";
64 else cout << "No\n";
65 return 0;
66 }

```

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

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

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

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

然后我们打表找规律可以得到

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

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

```
1  /** 平面几何 (Point)
2   *    2023-09-22: https://qoj.ac/submission/185408
3   **/
4  template<class T>
5  struct Point {
6      T x;
7      T y;
8      Point(const T &x_ = 0, const T &y_ = 0) : x(x_), y(y_) {}
9
10     template<class U>
11     operator Point<U>() {
12         return Point<U>(U(x), U(y));
13     }
14     Point &operator+=(const Point &p) & {
15         x += p.x;
16         y += p.y;
17         return *this;
18     }
19     Point &operator-=(const Point &p) & {
20         x -= p.x;
21         y -= p.y;
22         return *this;
23     }
24     Point &operator*=(const T &v) & {
25         x *= v;
26         y *= v;
27         return *this;
28     }
29     Point &operator/=(const T &v) & {
30         x /= v;
31         y /= v;
32         return *this;
33     }
34     Point operator-() const {
35         return Point(-x, -y);
36     }
37     friend Point operator+(Point a, const Point &b) {
38         return a + b;
39     }
40     friend Point operator-(Point a, const Point &b) {
41         return a - b;
42     }
43     friend Point operator*(Point a, const T &b) {
44         return a * b;
45     }
46     friend Point operator/(Point a, const T &b) {
47         return a / b;
48     }
49     friend Point operator*(const T &a, Point b) {
50         return b * a;
```

```

51     }
52     friend bool operator==(const Point &a, const Point &b) {
53         return a.x == b.x && a.y == b.y;
54     }
55     friend std::istream &operator>>(std::istream &is, Point &p) {
56         return is >> p.x >> p.y;
57     }
58     friend std::ostream &operator<<(std::ostream &os, const Point &p) {
59         return os << "(" << p.x << ", " << p.y << ")";
60     }
61 };
62
63 template<class T>
64 struct Line {
65     Point<T> a;
66     Point<T> b;
67     Line(const Point<T> &a_ = Point<T>(), const Point<T> &b_ = Point<T>()) : a(a_), b(b_) {}
68 };
69
70 template<class T>
71 T dot(const Point<T> &a, const Point<T> &b) {
72     return a.x * b.x + a.y * b.y;
73 }
74
75 template<class T>
76 T cross(const Point<T> &a, const Point<T> &b) {
77     return a.x * b.y - a.y * b.x;
78 }
79
80 template<class T>
81 T square(const Point<T> &p) {
82     return dot(p, p);
83 }
84
85 template<class T>
86 double length(const Point<T> &p) {
87     return std::sqrt(square(p));
88 }
89
90 template<class T>
91 double length(const Line<T> &l) {
92     return length(l.a - l.b);
93 }
94
95 template<class T>
96 Point<T> normalize(const Point<T> &p) {
97     return p / length(p);
98 }
99
100 template<class T>
101 bool parallel(const Line<T> &l1, const Line<T> &l2) {
102     return cross(l1.b - l1.a, l2.b - l2.a) == 0;
103 }
104
105 template<class T>
106 double distance(const Point<T> &a, const Point<T> &b) {
107     return length(a - b);
108 }
109
110 template<class T>
111 double distancePL(const Point<T> &p, const Line<T> &l) {
112     return std::abs(cross(l.a - l.b, l.a - p)) / length(l);
113 }
114
115 template<class T>
116 double distancePS(const Point<T> &p, const Line<T> &l) {
117     if (dot(p - l.a, l.b - l.a) < 0) {
118         return distance(p, l.a);
119     }
120     if (dot(p - l.b, l.a - l.b) < 0) {
121         return distance(p, l.b);

```

```

122     }
123     return distancePL(p, l);
124 }
125
126 template<class T>
127 Point<T> rotate(const Point<T> &a) {
128     return Point(-a.y, a.x);
129 }
130
131 template<class T>
132 int sgn(const Point<T> &a) {
133     return a.y > 0 || (a.y == 0 && a.x > 0) ? 1 : -1;
134 }
135
136 template<class T>
137 bool pointOnLineLeft(const Point<T> &p, const Line<T> &l) {
138     return cross(l.b - l.a, p - l.a) > 0;
139 }
140
141 template<class T>
142 Point<T> lineIntersection(const Line<T> &l1, const Line<T> &l2) {
143     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));
144 }
145
146 template<class T>
147 bool pointOnSegment(const Point<T> &p, const Line<T> &l) {
148     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)
149         && std::min(l.a.y, l.b.y) <= p.y && p.y <= std::max(l.a.y, l.b.y);
150 }
151
152 template<class T>
153 bool pointInPolygon(const Point<T> &a, const std::vector<Point<T>> &p) {
154     int n = p.size();
155     for (int i = 0; i < n; i++) {
156         if (pointOnSegment(a, Line(p[i], p[(i + 1) % n]))) {
157             return true;
158         }
159     }
160
161     int t = 0;
162     for (int i = 0; i < n; i++) {
163         auto u = p[i];
164         auto v = p[(i + 1) % n];
165         if (u.x < a.x && v.x >= a.x && pointOnLineLeft(a, Line(v, u))) {
166             t ^= 1;
167         }
168         if (u.x >= a.x && v.x < a.x && pointOnLineLeft(a, Line(u, v))) {
169             t ^= 1;
170         }
171     }
172
173     return t == 1;
174 }
175
176 // 0 : not intersect
177 // 1 : strictly intersect
178 // 2 : overlap
179 // 3 : intersect at endpoint
180 template<class T>
181 std::tuple<int, Point<T>, Point<T>> segmentIntersection(const Line<T> &l1, const Line<T> &l2) {
182     if (std::max(l1.a.x, l1.b.x) < std::min(l2.a.x, l2.b.x)) {
183         return {0, Point<T>(), Point<T>()};
184     }
185     if (std::min(l1.a.x, l1.b.x) > std::max(l2.a.x, l2.b.x)) {
186         return {0, Point<T>(), Point<T>()};
187     }
188     if (std::max(l1.a.y, l1.b.y) < std::min(l2.a.y, l2.b.y)) {
189         return {0, Point<T>(), Point<T>()};
190     }
191     if (std::min(l1.a.y, l1.b.y) > std::max(l2.a.y, l2.b.y)) {
192         return {0, Point<T>(), Point<T>()};
193     }

```



```

193     }
194     if (cross(l1.b - l1.a, l2.b - l2.a) == 0) {
195         if (cross(l1.b - l1.a, l2.a - l1.a) != 0) {
196             return {0, Point<T>(), Point<T>()};
197         } else {
198             auto maxx1 = std::max(l1.a.x, l1.b.x);
199             auto minx1 = std::min(l1.a.x, l1.b.x);
200             auto maxy1 = std::max(l1.a.y, l1.b.y);
201             auto miny1 = std::min(l1.a.y, l1.b.y);
202             auto maxx2 = std::max(l2.a.x, l2.b.x);
203             auto minx2 = std::min(l2.a.x, l2.b.x);
204             auto maxy2 = std::max(l2.a.y, l2.b.y);
205             auto miny2 = std::min(l2.a.y, l2.b.y);
206             Point<T> p1(std::max(minx1, minx2), std::max(miny1, miny2));
207             Point<T> p2(std::min(maxx1, maxx2), std::min(maxy1, maxy2));
208             if (!pointOnSegment(p1, l1)) {
209                 std::swap(p1.y, p2.y);
210             }
211             if (p1 == p2) {
212                 return {3, p1, p2};
213             } else {
214                 return {2, p1, p2};
215             }
216         }
217     }
218     auto cp1 = cross(l2.a - l1.a, l2.b - l1.a);
219     auto cp2 = cross(l2.a - l1.b, l2.b - l1.b);
220     auto cp3 = cross(l1.a - l2.a, l1.b - l2.a);
221     auto cp4 = cross(l1.a - l2.b, l1.b - l2.b);
222
223     if ((cp1 > 0 && cp2 > 0) || (cp1 < 0 && cp2 < 0) || (cp3 > 0 && cp4 > 0) || (cp3 < 0 && cp4 < 0)) {
224         return {0, Point<T>(), Point<T>()};
225     }
226
227     Point p = lineIntersection(l1, l2);
228     if (cp1 != 0 && cp2 != 0 && cp3 != 0 && cp4 != 0) {
229         return {1, p, p};
230     } else {
231         return {3, p, p};
232     }
233 }
234
235 template<class T>
236 double distanceSS(const Line<T> &l1, const Line<T> &l2) {
237     if (std::get<0>(segmentIntersection(l1, l2)) != 0) {
238         return 0.0;
239     }
240     return std::min({distancePS(l1.a, l2), distancePS(l1.b, l2), distancePS(l2.a, l1), distancePS(l2.b, l1)});
241 }
242
243 template<class T>
244 bool segmentInPolygon(const Line<T> &l, const std::vector<Point<T>> &p) {
245     int n = p.size();
246     if (!pointInPolygon(l.a, p)) {
247         return false;
248     }
249     if (!pointInPolygon(l.b, p)) {
250         return false;
251     }
252     for (int i = 0; i < n; i++) {
253         auto u = p[i];
254         auto v = p[(i + 1) % n];
255         auto w = p[(i + 2) % n];
256         auto [t, p1, p2] = segmentIntersection(l, Line(u, v));
257
258         if (t == 1) {
259             return false;
260         }
261         if (t == 0) {
262             continue;
263         }

```

```

264     if (t == 2) {
265         if (pointOnSegment(v, l) && v != l.a && v != l.b) {
266             if (cross(v - u, w - v) > 0) {
267                 return false;
268             }
269         }
270     } else {
271         if (p1 != u && p1 != v) {
272             if (pointOnLineLeft(l.a, Line(v, u))
273                 || pointOnLineLeft(l.b, Line(v, u))) {
274                 return false;
275             }
276         } else if (p1 == v) {
277             if (l.a == v) {
278                 if (pointOnLineLeft(u, l)) {
279                     if (pointOnLineLeft(w, l)
280                         && pointOnLineLeft(w, Line(u, v))) {
281                         return false;
282                     }
283                 } else {
284                     if (pointOnLineLeft(w, l)
285                         || pointOnLineLeft(w, Line(u, v))) {
286                         return false;
287                     }
288                 }
289             } else if (l.b == v) {
290                 if (pointOnLineLeft(u, Line(l.b, l.a))) {
291                     if (pointOnLineLeft(w, Line(l.b, l.a))
292                         && pointOnLineLeft(w, Line(u, v))) {
293                         return false;
294                     }
295                 } else {
296                     if (pointOnLineLeft(w, Line(l.b, l.a))
297                         || pointOnLineLeft(w, Line(u, v))) {
298                         return false;
299                     }
300                 }
301             } else {
302                 if (pointOnLineLeft(u, l)) {
303                     if (pointOnLineLeft(w, Line(l.b, l.a))
304                         || pointOnLineLeft(w, Line(u, v))) {
305                         return false;
306                     }
307                 } else {
308                     if (pointOnLineLeft(w, l)
309                         || pointOnLineLeft(w, Line(u, v))) {
310                         return false;
311                     }
312                 }
313             }
314         }
315     }
316 }
317 return true;
318 }
319
320 template<class T>
321 std::vector<Point<T>> hp(std::vector<Line<T>> lines) {
322     std::sort(lines.begin(), lines.end(), [&](auto l1, auto l2) {
323         auto d1 = l1.b - l1.a;
324         auto d2 = l2.b - l2.a;
325
326         if (sgn(d1) != sgn(d2)) {
327             return sgn(d1) == 1;
328         }
329
330         return cross(d1, d2) > 0;
331     });
332
333     std::deque<Line<T>> ls;
334     std::deque<Point<T>> ps;

```

```

335     for (auto l : lines) {
336         if (ls.empty()) {
337             ls.push_back(l);
338             continue;
339         }
340
341         while (!ps.empty() && !pointOnLineLeft(ps.back(), l)) {
342             ps.pop_back();
343             ls.pop_back();
344         }
345
346         while (!ps.empty() && !pointOnLineLeft(ps[0], l)) {
347             ps.pop_front();
348             ls.pop_front();
349         }
350
351         if (cross(l.b - l.a, ls.back().b - ls.back().a) == 0) {
352             if (dot(l.b - l.a, ls.back().b - ls.back().a) > 0) {
353
354                 if (!pointOnLineLeft(ls.back().a, l)) {
355                     assert(ls.size() == 1);
356                     ls[0] = l;
357                 }
358                 continue;
359             }
360             return {};
361         }
362
363         ps.push_back(lineIntersection(ls.back(), l));
364         ls.push_back(l);
365     }
366
367     while (!ps.empty() && !pointOnLineLeft(ps.back(), ls[0])) {
368         ps.pop_back();
369         ls.pop_back();
370     }
371     if (ls.size() <= 2) {
372         return {};
373     }
374     ps.push_back(lineIntersection(ls[0], ls.back()));
375
376     return std::vector(ps.begin(), ps.end());
377 }
378
379 using real = long double;
380 using P = Point<real>;
381
382 constexpr real eps = 0;
383

```

jly 立体几何

```

1  /** 立体几何 (Point)
2   *   2023-09-25 (i64): https://qoj.ac/submission/188519
3   *   2023-09-28 (double): https://qoj.ac/submission/190463
4   **/
5  using i64 = long long;
6  using real = double;
7
8  struct Point {
9      real x = 0;
10     real y = 0;
11     real z = 0;
12 };
13
14 Point operator+(const Point &a, const Point &b) {
15     return {a.x + b.x, a.y + b.y, a.z + b.z};
16 }
17
18 Point operator-(const Point &a, const Point &b) {
19     return {a.x - b.x, a.y - b.y, a.z - b.z};

```

```

20 }
21
22 Point operator*(const Point &a, real b) {
23     return {a.x * b, a.y * b, a.z * b};
24 }
25
26 Point operator/(const Point &a, real b) {
27     return {a.x / b, a.y / b, a.z / b};
28 }
29
30 real length(const Point &a) {
31     return std::hypot(a.x, a.y, a.z);
32 }
33
34 Point normalize(const Point &a) {
35     real l = length(a);
36     return {a.x / l, a.y / l, a.z / l};
37 }
38
39 real getAng(real a, real b, real c) {
40     return std::acos((a * a + b * b - c * c) / 2 / a / b);
41 }
42
43 std::ostream &operator<<(std::ostream &os, const Point &a) {
44     return os << "(" << a.x << ", " << a.y << ", " << a.z << ")";
45 }
46
47 real dot(const Point &a, const Point &b) {
48     return a.x * b.x + a.y * b.y + a.z * b.z;
49 }
50
51 Point cross(const Point &a, const Point &b) {
52     return {
53         a.y * b.z - a.z * b.y,
54         a.z * b.x - a.x * b.z,
55         a.x * b.y - a.y * b.x
56     };
57 }
58

```

jly 静态凸包

```

1  /** 静态凸包 (with. Point, 新版)
2   *   2024-04-06: https://qoj.ac/submission/379920)
3   **/
4   struct Point {
5       i64 x;
6       i64 y;
7       Point() : x{0}, y{0} {}
8       Point(i64 x_, i64 y_) : x{x_}, y{y_} {}
9   };
10
11   i64 dot(Point a, Point b) {
12       return a.x * b.x + a.y * b.y;
13   }
14
15   i64 cross(Point a, Point b) {
16       return a.x * b.y - a.y * b.x;
17   }
18
19   Point operator+(Point a, Point b) {
20       return Point(a.x + b.x, a.y + b.y);
21   }
22
23   Point operator-(Point a, Point b) {
24       return Point(a.x - b.x, a.y - b.y);
25   }
26
27   auto getHull(std::vector<Point> p) {
28       std::sort(p.begin(), p.end(),
29               [&](auto a, auto b) {

```

```

30         return a.x < b.x || (a.x == b.x && a.y < b.y);
31     });
32
33     std::vector<Point> hi, lo;
34     for (auto p : p) {
35         while (hi.size() > 1 && cross(hi.back() - hi[hi.size() - 2], p - hi.back()) >= 0) {
36             hi.pop_back();
37         }
38         while (!hi.empty() && hi.back().x == p.x) {
39             hi.pop_back();
40         }
41         hi.push_back(p);
42         while (lo.size() > 1 && cross(lo.back() - lo[lo.size() - 2], p - lo.back()) <= 0) {
43             lo.pop_back();
44         }
45         if (lo.empty() || lo.back().x < p.x) {
46             lo.push_back(p);
47         }
48     }
49     return std::make_pair(hi, lo);
50 }
51
52 const double inf = INFINITY;
53

```

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

```

1  /** 素数测试与因式分解 (Miller-Rabin & Pollard-Rho)
2   *   2023-05-16: https://cf.dianhsu.com/gym/104354/submission/206130894
3   **/
4  i64 mul(i64 a, i64 b, i64 m) {
5      return static_cast<__int128>(a) * b % m;
6  }
7  i64 power(i64 a, i64 b, i64 m) {
8      i64 res = 1 % m;
9      for (; b >= 1, a = mul(a, a, m))
10         if (b & 1)
11             res = mul(res, a, m);
12     return res;
13 }
14 bool isprime(i64 n) {
15     if (n < 2)
16         return false;
17     static constexpr int A[] = {2, 3, 5, 7, 11, 13, 17, 19, 23};
18     int s = __builtin_ctzll(n - 1);
19     i64 d = (n - 1) >> s;
20     for (auto a : A) {
21         if (a == n)
22             return true;
23         i64 x = power(a, d, n);
24         if (x == 1 || x == n - 1)
25             continue;
26         bool ok = false;
27         for (int i = 0; i < s - 1; ++i) {
28             x = mul(x, x, n);
29             if (x == n - 1) {
30                 ok = true;
31                 break;
32             }
33         }
34         if (!ok)
35             return false;
36     }
37     return true;
38 }
39 std::vector<i64> factorize(i64 n) {
40     std::vector<i64> p;
41     std::function<void(i64)> f = [&](i64 n) {
42         if (n <= 10000) {
43             for (int i = 2; i * i <= n; ++i)
44                 for (; n % i == 0; n /= i)

```

```

45         p.push_back(i);
46         if (n > 1)
47             p.push_back(n);
48         return;
49     }
50     if (isprime(n)) {
51         p.push_back(n);
52         return;
53     }
54     auto g = [&](i64 x) {
55         return (mul(x, x, n) + 1) % n;
56     };
57     i64 x0 = 2;
58     while (true) {
59         i64 x = x0;
60         i64 y = x0;
61         i64 d = 1;
62         i64 power = 1, lam = 0;
63         i64 v = 1;
64         while (d == 1) {
65             y = g(y);
66             ++lam;
67             v = mul(v, std::abs(x - y), n);
68             if (lam % 127 == 0) {
69                 d = std::gcd(v, n);
70                 v = 1;
71             }
72             if (power == lam) {
73                 x = y;
74                 power *= 2;
75                 lam = 0;
76                 d = std::gcd(v, n);
77                 v = 1;
78             }
79         }
80         if (d != n) {
81             f(d);
82             f(n / d);
83             return;
84         }
85         ++x0;
86     }
87 };
88 f(n);
89 std::sort(p.begin(), p.end());
90 return p;
91 }
92

```

高斯消元

整数版

```

1 // The 15-th BIT Campus Programming Contest - Onsite Round, problem: (J) Teacher Long and Machine Learning
2 using vi = vector<int>;
3
4 vi gauss(int n, vector<vi> a) { // a is n * (n + 1), 1 base
5     for(int i = 1; i <= n; i++) {
6         int r = i;
7         for(int j = i + 1; j <= n; j++)
8             if(abs(a[r][i]) < abs(a[j][i]))
9                 r = j;
10
11         if(abs(a[r][i]) == 0)
12             return vi();
13
14         if(i != r) swap(a[i], a[r]);
15         int div = a[i][i];
16         for(int j = i; j <= n + 1; j++) {
17             if(a[i][j] % div != 0) return vi();
18             a[i][j] /= div;
19

```

```

19     }
20
21     for(int j = i + 1; j <= n; j++) {
22         div = a[j][i];
23         for(int k = i; k <= n + 1; k++)
24             a[j][k] -= a[i][k] * div;
25     }
26 }
27 vi ret(n + 1);
28 ret[n] = a[n][n + 1];
29 for(int i = n - 1; i >= 1; i--) {
30     ret[i] = a[i][n + 1];
31     for(int j = i + 1; j <= n; j++)
32         ret[i] -= (a[i][j] * ret[j]);
33 }
34 return ret;
35 }

```

浮点数

```

1  const double eps = 1e-6;
2
3  vector<double> gauss(int n, vector<vector<double>> a) { // a is n * (n + 1), 1 base
4      for (int i = 1; i <= n; i++) {
5          int r = i;
6          for (int j = i + 1; j <= n; j++)
7              if (abs(a[r][i]) < abs(a[j][i]))
8                  r = j;
9          if (abs(a[r][i]) < eps) return vector<double>();
10         if (i != r) swap(a[i], a[r]);
11         double div = a[i][i];
12         for (int j = i; j <= n + 1; j++)
13             a[i][j] /= div;
14         for (int j = i + 1; j <= n; j++) {
15             div = a[j][i];
16             for (int k = i; k <= n + 1; k++)
17                 a[j][k] -= a[i][k] * div;
18         }
19     }
20     vector<double> ret(n + 1);
21     ret[n] = a[n][n + 1];
22     for (int i = n - 1; i >= 1; i--) {
23         ret[i] = a[i][n + 1];
24         for (int j = i + 1; j <= n; j++) {
25             ret[i] -= a[i][j] * ret[j];
26         }
27     }
28     return ret;
29 }

```

求行列式，浮点数

```

1  double gauss(int n, vector<vector<double>> a) { // a is n * n, 0 base
2      double det = 1;
3      for (int i = 0; i < n; ++i) {
4          int k = i;
5          for (int j = i + 1; j < n; ++j)
6              if (abs(a[j][i]) > abs(a[k][i])) k = j;
7          if (abs(a[k][i]) < EPS) return 0;
8          swap(a[i], a[k]);
9          if (i != k) det = -det;
10         det *= a[i][i];
11         for (int j = i + 1; j < n; ++j) a[i][j] /= a[i][i];
12         for (int j = 0; j < n; ++j)
13             if (j != i && abs(a[j][i]) > EPS)
14                 for (int k = i + 1; k < n; ++k) a[j][k] -= a[i][k] * a[j][i];
15     }
16     return det;
17 }

```

求行列式，整数，对 p 取模，不需要保证 p 为质数

```

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

```

```

2  using i64 = long long;
3  using vi = vector<i64>;
4
5  i64 gauss(i64 n, vector<vi> a, i64 p) { // a is n * n, 1 base
6      i64 det = 1, w = 1;
7      for (int i = 1; i <= n; i++) {
8          for (int j = i + 1; j <= n; j++) {
9              while (a[i][i]) {
10                 int div = a[j][i] / a[i][i];
11                 for (int k = i; k <= n; k++)
12                     a[j][k] = (a[j][k] - div * a[i][k] % p + p) % p;
13                 swap(a[i], a[j]), w = -w;
14             }
15             swap(a[i], a[j]), w = -w;
16         }
17     }
18     for (int i = 1; i <= n; i++) det = det * a[i][i] % p;
19     det = ((det * w) % p + p) % p;
20     return det;
21 }

```

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} 。

```

1  #include <bits/stdc++.h>
2
3  using namespace std;
4
5  using i32 = int32_t;
6
7  double a, b, c, d;
8
9  double f(double x) {
10     return (c * x + d) / (a * x + b);
11 }
12
13 double simpson(double l, double r) {
14     double mid = (l + r) / 2;
15     return (r - l) * (f(l) + 4 * f(mid) + f(r)) / 6; // 辛普森公式
16 }
17
18 double asr(double l, double r, double eps, double ans, int step) {
19     double mid = (l + r) / 2;
20     double fl = simpson(l, mid), fr = simpson(mid, r);
21     if (abs(fl + fr - ans) <= 15 * eps && step < 0)
22         return fl + fr + (fl + fr - ans) / 15; // 足够相似的话就直接返回
23     return asr(l, mid, eps / 2, fl, step - 1) +
24            asr(mid, r, eps / 2, fr, step - 1); // 否则分割成两段递归求解
25 }
26
27 double calc(double l, double r, double eps) {
28     return asr(l, r, eps, simpson(l, r), 12);
29 }
30
31 i32 main() {
32     ios::sync_with_stdio(false), cin.tie(nullptr);
33     cin >> a >> b >> c >> d;
34
35     double l, r;
36     cin >> l >> r;
37     cout << fixed << setprecision(6) << calc(l, r, 1e-6);
38     return 0;
39 }

```


字符串

jly AC 自动机

```
1  /** AC 自动机 (AhoCorasick, with string 新版)
2  *    2024-04-09: https://www.luogu.com.cn/record/155114676 【模板】
3  **/
4  struct AhoCorasick {
5      static constexpr int ALPHABET = 26;
6      struct Node {
7          int len;
8          int link;
9          std::array<int, ALPHABET> next;
10         Node() : len{0}, link{0}, next{} {}
11     };
12
13     std::vector<Node> t;
14
15     AhoCorasick() {
16         init();
17     }
18
19     void init() {
20         t.assign(2, Node());
21         t[0].next.fill(1);
22         t[0].len = -1;
23     }
24
25     int newNode() {
26         t.emplace_back();
27         return t.size() - 1;
28     }
29
30     int add(const std::string &a) {
31         int p = 1;
32         for (auto c : a) {
33             int x = c - 'a';
34             if (t[p].next[x] == 0) {
35                 t[p].next[x] = newNode();
36                 t[t[p].next[x]].len = t[p].len + 1;
37             }
38             p = t[p].next[x];
39         }
40         return p;
41     }
42
43     void work() {
44         std::queue<int> q;
45         q.push(1);
46
47         while (!q.empty()) {
48             int x = q.front();
49             q.pop();
50
51             for (int i = 0; i < ALPHABET; i++) {
52                 if (t[x].next[i] == 0) {
53                     t[x].next[i] = t[t[x].link].next[i];
54                 } else {
55                     t[t[x].next[i]].link = t[t[x].link].next[i];
56                     q.push(t[x].next[i]);
57                 }
58             }
59         }
60     }
61
62     int next(int p, int x) {
63         return t[p].next[x];
64     }
65
66     int link(int p) {
67         return t[p].link;
```

```

68     }
69
70     int len(int p) {
71         return t[p].len;
72     }
73
74     int size() {
75         return t.size();
76     }
77 };
78
79
80 /** AC 自动机 (AhoCorasick, with vector 新版)
81  * 2023-04-07: https://codeforces.com/contest/1801/submission/201155712
82  */
83 struct AhoCorasick {
84     static constexpr int ALPHABET = 26;
85     struct Node {
86         int len;
87         int link;
88         std::array<int, ALPHABET> next;
89         Node() : link{}, next{} {}
90     };
91
92     std::vector<Node> t;
93
94     AhoCorasick() {
95         init();
96     }
97
98     void init() {
99         t.assign(2, Node());
100         t[0].next.fill(1);
101         t[0].len = -1;
102     }
103
104     int newNode() {
105         t.emplace_back();
106         return t.size() - 1;
107     }
108
109     int add(const std::vector<int> &a) {
110         int p = 1;
111         for (auto x : a) {
112             if (t[p].next[x] == 0) {
113                 t[p].next[x] = newNode();
114                 t[t[p].next[x]].len = t[p].len + 1;
115             }
116             p = t[p].next[x];
117         }
118         return p;
119     }
120
121     int add(const std::string &a, char offset = 'a') {
122         std::vector<int> b(a.size());
123         for (int i = 0; i < a.size(); i++) {
124             b[i] = a[i] - offset;
125         }
126         return add(b);
127     }
128
129     void work() {
130         std::queue<int> q;
131         q.push(1);
132
133         while (!q.empty()) {
134             int x = q.front();
135             q.pop();
136
137             for (int i = 0; i < ALPHABET; i++) {
138                 if (t[x].next[i] == 0) {
139                     t[x].next[i] = t[t[x].link].next[i];
140                 }
141             }
142         }
143     }
144 };

```

```

61         } else {
62             t[t[x].next[i]].link = t[t[x].link].next[i];
63             q.push(t[x].next[i]);
64         }
65     }
66 }
67 }
68
69 int next(int p, int x) {
70     return t[p].next[x];
71 }
72
73 int next(int p, char c, char offset = 'a') {
74     return next(p, c - 'a');
75 }
76
77 int link(int p) {
78     return t[p].link;
79 }
80
81 int len(int p) {
82     return t[p].len;
83 }
84
85 int size() {
86     return t.size();
87 }
88 };
89

```

jly 最长公共前缀 LCP

```

1  /** 最长公共前缀 LCP (例题)
2   *   2024-03-02: https://oj.ac/submission/343378
3   **/
4  constexpr int L = 2E6 + 10;
5
6  int len[L];
7  int lnk[L];
8  int nxt[L][26];
9
10 int f[L];
11 int tot = 1;
12
13 std::vector<int> adj[L];
14
15 int extend(int p, int c) {
16     if (nxt[p][c]) {
17         int q = nxt[p][c];
18         if (len[q] == len[p] + 1) {
19             return q;
20         }
21         int r = ++tot;
22         len[r] = len[p] + 1;
23         lnk[r] = lnk[q];
24         std::copy(nxt[q], nxt[q] + 26, nxt[r]);
25         lnk[q] = r;
26         while (nxt[p][c] == q) {
27             nxt[p][c] = r;
28             p = lnk[p];
29         }
30         return r;
31     }
32     int cur = ++tot;
33     len[cur] = len[p] + 1;
34     while (!nxt[p][c]) {
35         nxt[p][c] = cur;
36         p = lnk[p];
37     }
38     lnk[cur] = extend(p, c);
39     return cur;

```

```

40 }
41
42 int main() {
43     std::ios::sync_with_stdio(false);
44     std::cin.tie(nullptr);
45
46     std::fill(nxt[0], nxt[0] + 26, 1);
47     len[0] = -1;
48
49     int N;
50     std::cin >> N;
51
52     std::vector<std::string> S(N);
53     for (int i = 0; i < N; i++) {
54         std::cin >> S[i];
55         int p = 1;
56         for (auto c : S[i]) {
57             p = extend(p, c - 'a');
58             if (f[p] != -1) {
59                 if (f[p] == 0) {
60                     f[p] = i + 1;
61                 } else if (f[p] != i + 1) {
62                     f[p] = -1;
63                 }
64             }
65         }
66     }
67
68     for (int i = 1; i <= tot; i++) {
69         adj[lnk[i]].push_back(i);
70     }
71 }
72

```

动态规划