

Standard Code Library

Shanghai Jiao Tong University

October, 2015

Contents

1	数论算法	2
1.1	快速数论变换	2
1.2	多项式求逆	2
1.3	中国剩余定理	2
1.4	Miller Rabin	2
1.5	Pollard Rho	2
1.6	坚固的逆元	2
1.7	离散对数	3
1.8	直线下整点个数	3
1.9	原根相关	3
2	数值算法	3
2.1	快速傅立叶变换	3
2.2	单纯形法求解线性规划	3
2.3	自适应辛普森	3
3	数据结构	4
3.1	堆	4
3.2	左偏树	4
3.3	Splay 普通操作版	4
3.4	Splay 区间操作版	4
3.5	坚固的 Treap	6
3.6	k-d 树	6
3.7	树链剖分	7
3.7.1	点操作版本	7
3.7.2	链操作版本	8
3.8	Link-Cut-Tree	8
4	图论	9
4.1	点双连通分量	9
4.1.1	坚固的点双连通分量	9
4.1.2	朴素的点双连通分量	9
4.2	2-SAT 问题	9
4.3	二分图最大匹配	10
4.3.1	Hopcroft Karp 算法	10
4.4	二分图最大权匹配	10
4.5	最大流	11
4.5.1	ISAP	11
4.5.2	SAP	11
4.6	上下界网络流	11
4.6.1	无源汇的上下界可行流	12
4.6.2	有源汇的上下界可行流	12
4.6.3	有源汇的上下界最大流	12
4.6.4	有源汇的上下界最小流	12
4.7	最小费用最大流	12
4.7.1	稀疏图	12
4.7.2	稠密图	12
4.8	一般图最大匹配	12
4.9	无向图全局最小割	13
4.10	最小树形图	13
4.11	有根树的同构	14
4.12	度限制生成树	14
4.13	弦图相关	15
4.13.1	弦图的判定	15
4.13.2	弦图的团数	15
4.14	最大团点数	15
4.15	最大团计数	15
4.16	哈密尔顿回路 (ORE 性质的图)	16

5	字符串	16
5.1	模式串匹配	16
5.2	坚固的模式串匹配	17
5.3	AC 自动机	17
5.4	后缀数组	17
5.5	广义后缀自动机	17
5.6	Manacher 算法	18
5.7	回文树	18
5.8	循环串最小表示	18
6	计算几何	18
6.1	二维基础	18
6.1.1	凸包	18
6.1.2	半平面交	18
6.1.3	最近点对	18
6.1.4	三角形的心	19
6.2	三维基础	19
6.2.1	凸包	19
6.2.2	绕轴旋转	20
6.3	多边形	20
6.3.1	判断点在多边形内部	20
6.3.2	多边形内整点计数	20
6.4	圆	20
6.4.1	最小覆盖圆	20
6.4.2	最小覆盖球	20
6.4.3	多边形与圆的交面积	21
7	其它	22
7.1	STL 使用方法	22
7.1.1	nth_element	22
7.1.2	next_permutation	22
7.2	博弈论相关	22
7.2.1	巴什博弈	22
7.2.2	威佐夫博弈	22
7.2.3	阶梯博弈	22
7.2.4	图上删边游戏	22
7.3	Java Reference	22
7.4	Bug List	22
8	数学公式	23
8.1	常用数学公式	23
8.1.1	求和公式	23
8.1.2	斐波那契数列	23
8.1.3	错排公式	23
8.1.4	莫比乌斯函数	23
8.1.5	Burnside 引理	23
8.1.6	五边形数定理	23
8.1.7	树的计数	23
8.1.8	欧拉公式	23
8.1.9	皮克定理	23
8.1.10	牛顿恒等式	23
8.2	平面几何公式	23
8.2.1	三角形	23
8.2.2	四边形	23
8.2.3	正 n 边形	24
8.2.4	圆	24
8.2.5	棱柱	24
8.2.6	棱锥	24
8.2.7	棱台	24
8.2.8	圆柱	24
8.2.9	圆锥	24
8.2.10	圆台	24
8.2.11	球	24
8.2.12	球台	24
8.2.13	球扇形	24
8.3	立体几何公式	24
8.3.1	球面三角公式	24

8.3.2 四面体体积公式	24
8.4 积分表	24

第1章 数论算法

1.1 快速数论变换

使用条件及注意事项: mod 必须要是一个形如 $a2^b + 1$ 的数, prt 表示 mod 的原根。

```

1 const int mod = 998244353;
2 const int prt = 3;
3 int prepare(int n) {
4     int len = 1;
5     for (; len <= 2 * n; len <= 1);
6     for (int i = 0; i <= len; i++) {
7         e[0][i] = fpm(prt, (mod - 1) / len * i, mod);
8         e[1][i] = fpm(prt, (mod - 1) / len * (len - i), mod);
9     }
10    return len;
11 }
12 void DFT(int *a, int n, int f) {
13     for (int i = 0, j = 0; i < n; i++) {
14         if (i > j) std::swap(a[i], a[j]);
15         for (int t = n >> 1; (j ^= t) < t; t >>= 1);
16     }
17     for (int i = 2; i <= n; i <= 1)
18         for (int j = 0; j < n; j += i)
19             for (int k = 0; k < (i >> 1); k++) {
20                 int A = a[j + k];
21                 int B = (long long)a[j + k + (i >> 1)] * e[f][n / i * k] % mod;
22                 a[j + k] = (A + B) % mod;
23                 a[j + k + (i >> 1)] = (A - B + mod) % mod;
24             }
25     if (f == 1) {
26         long long rev = fpm(n, mod - 2, mod);
27         for (int i = 0; i < n; i++) {
28             a[i] = (long long)a[i] * rev % mod;
29         }
30     }
31 }
```

1.2 多项式求逆

使用条件及注意事项: 求一个多项式在模意义下的逆元。

```

1 void getInv(int *a, int *b, int n) {
2     static int tmp[1000];
3     std::fill(b, b + n, 0);
4     b[0] = fpm(a[0], mod - 2, mod);
5     for (int c = 1; c <= n; c <= 1) {
6         for (int i = 0; i < c; i++) tmp[i] = a[i];
7         std::fill(b + c, b + (c << 1), 0);
8         std::fill(tmp + c, tmp + (c << 1), 0);
9         DFT(tmp, c << 1, 0);
10        DFT(b, c << 1, 0);
11        for (int i = 0; i < (c << 1); i++) {
12            b[i] = (long long)(2 - (long long)tmp[i] * b[i] % mod + mod) * b[i] % mod;
13        }
14        DFT(b, c << 1, 1);
15        std::fill(b + c, b + (c << 1), 0);
16    }
17 }
```

1.3 中国剩余定理

使用条件及注意事项: 模数可以不互质。

```

1 bool solve(int n, std::pair<long long, long long> input[],
2             std::pair<long long, long long> &output) {
3     output = std::make_pair(1, 1);
4     for (int i = 0; i < n; ++i) {
5         long long number, useless;
6         euclid(output.second, input[i].second, number, useless);
7         long long divisor = std::__gcd(output.second, input[i].second);
8         if ((input[i].first - output.first) % divisor) {
9             return false;
10        }
11        number *= (input[i].first - output.first) / divisor;
12        fix(number, input[i].second);
13    }
```

```

13     output.first += output.second * number;
14     output.second *= input[i].second / divisor;
15     fix(output.first, output.second);
16 }
17 return true;
18 }
```

1.4 Miller Rabin

```

1 const int BASE[12] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37};
2 bool check(const long long &prime, const long long &base) {
3     long long number = prime - 1;
4     for (; ~number & 1; number >>= 1);
5     long long result = power_mod(base, number, prime);
6     for (; number != prime - 1 && result != 1 && result != prime - 1; number <= 1) {
7         result = multiply_mod(result, result, prime);
8     }
9     return result == prime - 1 || (number & 1) == 1;
10 }
11 bool miller_rabin(const long long &number) {
12     if (number < 2) return false;
13     if (number < 4) return true;
14     if (~number & 1) return false;
15     for (int i = 0; i < 12 && BASE[i] < number; ++i)
16         if (!check(number, BASE[i])) return false;
17     return true;
18 }
```

1.5 Pollard Rho

```

1 long long pollard_rho(const long long &number, const long long &seed) {
2     long long x = rand() % (number - 1) + 1, y = x;
3     for (int head = 1, tail = 2; ; ) {
4         x = multiply_mod(x, x, number);
5         x = add_mod(x, seed, number);
6         if (x == y) {
7             return number;
8         }
9         long long answer = std::__gcd(abs(x - y), number);
10        if (answer > 1 && answer < number) {
11            return answer;
12        }
13        if (++head == tail) {
14            y = x;
15            tail <= 1;
16        }
17    }
18 }
19 void factorize(const long long &number, std::vector<long long> &divisor) {
20     if (number > 1) {
21         if (miller_rabin(number)) {
22             divisor.push_back(number);
23         } else {
24             long long factor = number;
25             for (; factor >= number;
26                 factor = pollard_rho(number, rand() % (number - 1) + 1));
27             factorize(number / factor, divisor);
28             factorize(factor, divisor);
29         }
30     }
31 }
```

1.6 坚固的逆元

```

1 long long inverse(const long long &x, const long long &mod) {
2     if (x == 1) {
3         return 1;
4     } else {
5         return (mod - mod / x) * inverse(mod % x, mod) % mod;
6     }
7 }
```

1.7 离散对数

使用条件及注意事项： a 必须有关于 MOD 的逆元。

```
1 int log_mod(int a, int b) {
2     int m, v, e = 1, i;
3     m = (int)sqrt(MOD);
4     v = inv(pow_mod(a, m));
5     std::map<int, int> x;
6     x[1] = 0;
7     for(i = 1; i < m; i++) { e = mul_mod(e, a); if (!x.count(e)) x[e] = i; }
8     for(i = 0; i < m; i++) {
9         if (x.count(b)) return i * m + x[b];
10        b = mul_mod(b, v);
11    }
12    return -1;
13 }
```

1.8 直线下整点个数

```
1 long long solve(const long long &n, const long long &a,
2                 const long long &b, const long long &m) {
3     if (b == 0) return n * (a / m);
4     if (a >= m) return n * (a / m) + solve(n, a % m, b, m);
5     if (b >= m) return (n - 1) * n / 2 * (b / m) + solve(n, a, b % m, m);
6     return solve((a + b * n) / m, (a + b * n) % m, m, b);
7 }
```

1.9 原根相关

- 定义：设 $m > 1$, $(a, m) = 1$, 使得 $a^r \equiv 1 \pmod{m}$ 成立的最小的 r , 称为 a 对模 m 的阶, 记作 $\delta_m(a)$ 。
- 定义：设 m 是正整数, a 是整数, 若 a 模 m 的阶等于 $\Phi(m)$, 则称 a 为模 m 的一个原根。
- 定理：如果模 m 有原根, 那么它一共有 $\Phi(\Phi(m))$ 个原根。
- 定理：如果 $m > 1$, $(a, m) = 1$, $a^n \equiv 1 \pmod{m}$, 则 $\delta_m(a) | n$ 。
- 定理：模 m 有原根的充要条件是 $m = 2, 4, p^a, 2p^a$ 。
- 求模素数 p 原根的方法：对 $p-1$ 素因子分解, 即 $p-1 = p_1^{k_1} p_2^{k_2} \cdots p_n^{k_n}$, 若 $g^{\frac{p-1}{p_i}}$ 恒成立, 那么 g 为 p 的一个原根。(对于合数求原根, 只需要将 $p-1$ 换成 $\Phi(p)$ 即可)

第2章 数值算法

2.1 快速傅立叶变换

```
1 int prepare(int n) {
2     int len = 1;
3     for (; len <= 2 * n; len <= 1);
4     for (int i = 0; i < len; i++) {
5         e[0][i] = Complex(cos(2 * pi * i / len), sin(2 * pi * i / len));
6         e[1][i] = Complex(cos(2 * pi * i / len), -sin(2 * pi * i / len));
7     }
8     return len;
9 }
10 void DFT(Complex *a, int n, int f) {
11     for (int i = 0, j = 0; i < n; i++) {
12         if (i > j) std::swap(a[i], a[j]);
13         for (int t = n >> 1; (j ^= t) < t; t >>= 1);
14     }
15     for (int i = 2; i <= n; i <= 1)
16         for (int j = 0; j < n; j += i)
17             for (int k = 0; k < (i >> 1); k++) {
18                 Complex A = a[j + k];
19                 Complex B = e[f][n / i * k] * a[j + k + (i >> 1)];
20                 a[j + k] = A + B;
21                 a[j + k + (i >> 1)] = A - B;
22             }
23     if (f == 1) {
24         for (int i = 0; i < n; i++)
25             a[i].a /= n;
26     }
27 }
```

2.2 单纯形法求解线性规划

使用条件及注意事项：返回结果为 $max\{c_1 \times m \cdot x_{m \times 1} \mid x_{m \times 1} \geq 0_{m \times 1}, a_{n \times m} \cdot x_{m \times 1} \leq b_{n \times 1}\}$

```
1 std::vector<double> solve(const std::vector<std::vector<double>> &a,
2                           const std::vector<double> &b, const std::vector<double> &c) {
3     int n = (int)a.size(), m = (int)a[0].size() + 1;
4     std::vector<std::vector<double>> > value(n + 2, std::vector<double>(m + 1));
5     std::vector<int> index(n + m);
6     int r = n, s = m - 1;
7     for (int i = 0; i < n + m; ++i) index[i] = i;
8     for (int i = 0; i < n; ++i) {
9         for (int j = 0; j < m - 1; ++j)
10            value[i][j] = -a[i][j];
11        value[i][m - 1] = 1;
12        value[i][m] = b[i];
13        if (value[r][m] > value[i][m]) r = i;
14    }
15    for (int j = 0; j < m - 1; ++j) value[n][j] = c[j];
16    value[n + 1][m - 1] = -1;
17    for (double number; ; ) {
18        if (r < n) {
19            std::swap(index[s], index[r + m]);
20            value[r][s] = 1 / value[r][s];
21            for (int j = 0; j <= m; ++j)
22                if (j != s) value[r][j] *= -value[r][s];
23            for (int i = 0; i <= n + 1; ++i) {
24                if (i != r) {
25                    for (int j = 0; j <= m; ++j)
26                        if (j != s) value[i][j] += value[r][j] * value[i][s];
27                    value[i][s] *= value[r][s];
28                }
29            }
30        }
31        r = s = -1;
32        for (int j = 0; j < m; ++j) {
33            if (s < 0 || index[s] > index[j]) {
34                if (value[n + 1][j] > eps || value[n + 1][j] > -eps && value[n][j] > eps)
35                    s = j;
36            }
37        }
38    }
39    if (s < 0) break;
40    for (int i = 0; i < n; ++i) {
41        if (value[i][s] < -eps) {
42            if (r < 0)
43                || (number = value[r][m] / value[r][s] - value[i][m] / value[i][s]) < -eps
44                || number < eps && index[r + m] > index[i + m]) {
45                r = i;
46            }
47        }
48    }
49    if (r < 0) {
50        // Solution is unbounded.
51        return std::vector<double>();
52    }
53    if (value[n + 1][m] < -eps) {
54        // No solution.
55        return std::vector<double>();
56    }
57    std::vector<double> answer(m - 1);
58    for (int i = m; i < n + m; ++i)
59        if (index[i] < m - 1) answer[index[i]] = value[i - m][m];
60    return answer;
61 }
62 }
```

2.3 自适应辛普森

```
1 double area(const double &left, const double &right) {
2     double mid = (left + right) / 2;
3     return (right - left) * (calc(left) + 4 * calc(mid) + calc(right)) / 6;
4 }
5 double simpson(const double &left, const double &right,
6                const double &eps, const double &area_sum) {
7     double mid = (left + right) / 2;
8     double area_left = area(left, mid);
9     double area_right = area(mid, right);
10    double area_total = area_left + area_right;
11    if (std::abs(area_total - area_sum) < 15 * eps) {
```

```

12     return area_total + (area_total - area_sum) / 15;
13 }
14 return simpson(left, mid, eps / 2, area_left)
15     + simpson(mid, right, eps / 2, area_right);
16 }
17 double simpson(const double &left, const double &right, const double &eps) {
18     return simpson(left, right, eps, area(left, right));
19 }

```

第3章 数据结构

3.1 堆

```

1 int up(int pos){
2     for (; pos != 1; pos /= 2)
3         if (C()(array.getpq(pos), array.getpq(pos / 2))) swap(pos, pos / 2);
4         else break;
5     return pos;
6 }
7
8 void down(int pos){
9     for (; pos * 2 <= array.size(); ){
10         if (pos * 2 + 1 <= array.size()){
11             if (C()(array.getpq(pos * 2 + 1), array.getpq(pos * 2))){
12                 if (C()(array.getpq(pos * 2 + 1), array.getpq(pos))) swap(pos * 2 + 1, pos), pos =
13                     pos * 2 + 1;
14                 else break;
15             }
16             else{
17                 if (C()(array.getpq(pos * 2), array.getpq(pos))) swap(pos * 2, pos), pos = pos *
18                     2;
19                 else break;
20             }
21         }
22         else{
23             if (C()(array.getpq(pos * 2), array.getpq(pos))) swap(pos * 2, pos), pos = pos * 2;
24             else break;
25         }
26     }
27 }

```

3.2 左偏树

```

1 int merge(int a, int b) {
2     if (a == 0) return b;
3     if (b == 0) return a;
4     if (w[a] > w[b]) std::swap(a, b);
5     c[a][1] = merge(c[a][1], b);
6     if (d[c[a][1]] > d[c[a][0]]) std::swap(c[a][0], c[a][1]);
7     if (c[a][1] == 0) d[a] = 0;
8     else d[a] = d[c[a][1]] + 1;
9     return a;
10 }

```

3.3 Splay 普通操作版

使用条件及注意事项：

1. 插入 x 数
2. 删除 x 数 (若有多个相同的数, 因只删除一个)
3. 查询 x 数的排名 (若有多个相同的数, 因输出最小的排名)
4. 查询排名为 x 的数
5. 求 x 的前驱 (前驱定义为小于 x , 且最大的数)
6. 求 x 的后继 (后继定义为大于 x , 且最小的数)

```

1 int pred(int x) {
2     splay(x, -1);
3     for (x = c[x][0]; c[x][1]; x = c[x][1]);
4     return x;
5 }
6 int succ(int x) {
7     splay(x, -1);
8     for (x = c[x][1]; c[x][0]; x = c[x][0]);
9     return x;
10 }

```

```

10 }
11 void remove(int x) {
12     if (b[x] > 1) {b[x]--; splay(x, -1); return;}
13     splay(x, -1);
14     if (!c[x][0] && !c[x][1]) rt = 0;
15     else if (c[x][0] && !c[x][1]) f[rt = c[x][0]] = -1;
16     else if (!c[x][0] && c[x][1]) f[rt = c[x][1]] = -1;
17     else{
18         int t = pred(x); f[rt = c[x][0]] = -1;
19         c[t][1] = c[x][1]; f[c[x][1]] = t;
20         splay(c[x][1], -1);
21     }
22     c[x][0] = c[x][1] = f[x] = d[x] = s[x] = b[x] = 0;
23 }
24 int find(int z) {
25     int x=rt;
26     while (d[x]!=z)
27         if (c[x][d[x]<z]) x=c[x][d[x]<z];
28         else break;
29     return x;
30 }
31 void insert(int z) {
32     if (!rt) {
33         f[rt = ++size] = -1;
34         d[size] = z; b[size] = 1;
35         splay(size, -1);
36         return;
37     }
38     int x = find(z);
39     if (d[x] == z) {
40         b[x]++;
41         splay(x, -1);
42         return;
43     }
44     c[x][d[x]<z] = ++size; f[size] = x;
45     d[size] = z; b[size] = s[size] = 1;
46     splay(size, -1);
47 }
48 int select(int z) {
49     int x = rt;
50     while (z < s[c[x][0]] + 1 || z > s[c[x][0]] + b[x])
51         if (z > s[c[x][0]] + b[x]) {
52             z -= s[c[x][0]] + b[x];
53             x = c[x][1];
54         }
55         else x = c[x][0];
56     return x;
57 }
58 int main() {
59     scanf("%d",&n);
60     for (int i = 1; i <= n; i++) {
61         int opt, x;
62         scanf("%d%d", &opt, &x);
63         if (opt == 1) insert(x);
64         else if (opt == 2) remove(find(x)); //删除x数 (若有多个相同的数, 因只删除一个)
65         else if (opt == 3) { // 查询x数的排名 (若有多个相同的数, 因输出最小的排名)
66             insert(x);
67             printf("%d\n", s[c[find(x)][0]] + 1);
68             remove(find(x));
69         }
70         else if (opt == 4) printf("%d\n",d[select(x)]);
71         else if (opt == 5) {
72             insert(x);
73             printf("%d\n", d[pred(find(x))]);
74             remove(find(x));
75         }
76         else if (opt == 6) {
77             insert(x);
78             printf("%d\n", d[succ(find(x))]);
79             remove(find(x));
80         }
81     }
82     return 0;
83 }

```

3.4 Splay 区间操作版

使用条件及注意事项：

这是为 NOI2005 维修数列的代码, 仅供区间操作的 splay 参考。

```

1 const int INF = 100000000;

```

```

2  const int Maxspace = 500000;
3  struct SplayNode{
4      int ls, rs, zs, ms;
5      SplayNode() {
6          ms = 0;
7          ls = rs = zs = -INF;
8      }
9      SplayNode(int d) {
10         ms = zs = ls = rs = d;
11     }
12     SplayNode operator +(const SplayNode &p) const {
13         SplayNode ret;
14         ret.ls = max(ls, ms + p.ls);
15         ret.rs = max(rs + p.ms, p.rs);
16         ret.zs = max(rs + p.ls, max(zs, p.zs));
17         ret.ms = ms + p.ms;
18         return ret;
19     }
20 }t[MAXN], d[MAXN];
21 int n, m, rt, top, a[MAXN], f[MAXN], c[MAXN][2], g[MAXN], h[MAXN], z[MAXN];
22 bool r[MAXN], b[MAXN];
23 void makesame(int x, int s) {
24     if (!x) return;
25     b[x] = true;
26     d[x] = SplayNode(g[x] = s);
27     t[x].zs = t[x].ms = g[x] * h[x];
28     t[x].ls = t[x].rs = max(g[x], g[x] * h[x]);
29 }
30 void makerev(int x) {
31     if (!x) return;
32     r[x] ^= 1;
33     swap(c[x][0], c[x][1]);
34     swap(t[x].ls, t[x].rs);
35 }
36 void pushdown(int x) {
37     if (!x) return;
38     if (r[x]) {
39         makerev(c[x][0]);
40         makerev(c[x][1]);
41         r[x]=0;
42     }
43     if (b[x]) {
44         makesame(c[x][0],g[x]);
45         makesame(c[x][1],g[x]);
46         b[x]=g[x]=0;
47     }
48 }
49 void updata(int x) {
50     if (!x) return;
51     h[x]=h[c[x][0]]+h[c[x][1]]+1;
52     t[x]=t[c[x][0]]+d[x]+t[c[x][1]];
53 }
54 void rotate(int x,int k) {
55     pushdown(x);pushdown(c[x][k]);
56     int y = c[x][k]; c[x][k] = c[y][k^1]; c[y][k^1] = x;
57     if (f[x] != -1) c[f[x]][c[f[x]][1] == x] = y; else rt = y;
58     f[y] = f[x]; f[x] = y; f[c[x][k]] = x;
59     updata(x); updata(y);
60 }
61 void splay(int x, int s) {
62     while (f[x] != s) {
63         if (f[f[x]]!=s) {
64             pushdown(f[f[x]]);
65             rotate(f[f[x]], (c[f[f[x]]][1] == f[x]) ^ r[f[f[x]]]);
66         }
67         pushdown(f[x]);
68         rotate(f[x], (c[f[x]][1]==x) ^ r[f[x]]);
69     }
70 }
71 void build(int &x,int l,int r) {
72     if (l > r) {x = 0; return;}
73     x = z[top--];
74     if (l < r) {
75         build(c[x][0],l,(l+r>>1)-1);
76         build(c[x][1],(l+r>>1)+1,r);
77     }
78     f[c[x][0]] = f[c[x][1]] = x;
79     d[x] = SplayNode(a[l+r>>1]);
80     updata(x);
81 }
82 void init() {
83     d[0] = SplayNode();
84     f[rt=2] = -1;

```

```

85     f[1] = 2; c[2][0] = 1;
86     int x;
87     build(x,1,n);
88     c[1][1] = x; f[x] = 1;
89     splay(x, -1);
90 }
91 int find(int z) {
92     int x = rt; pushdown(x);
93     while (z != h[c[x][0]] + 1) {
94         if (z > h[c[x][0]] + 1) {
95             z -= h[c[x][0]] + 1;
96             x = c[x][1];
97         }
98         else x = c[x][0];
99         pushdown(x);
100     }
101     return x;
102 }
103 void getrange(int &x,int &y) {
104     y = x + y - 1;
105     x = find(x);
106     y = find(y + 2);
107     splay(y, -1);
108     splay(x, y);
109 }
110 void recycle(int x) {
111     if (!x) return;
112     recycle(c[x][0]);
113     recycle(c[x][1]);
114     z[++top]=x;
115     t[x] = d[x] = SplayNode();
116     r[x] = b[x] = g[x] = f[x] = h[x] = 0;
117     c[x][0] = c[x][1]=0;
118 }
119 int main() {
120     scanf("%d%d",&n,&m);
121     for (int i = 1; i <= n; i++) scanf("%d",a+i);
122     for (int i = Maxspace; i>=3; i--) z[++top] = i;
123     init();
124     for (int i = 1; i <= m; i++) {
125         char op[10];
126         int x, y, tmp;
127         scanf("%s", op);
128         if (!strcmp(op, "INSERT")) {
129             scanf("%d%d", &x, &y);
130             n += y;
131             if (!y) continue;
132             for (int i = 1; i <= y; i++) scanf("%d",a+i);
133             build(tmp, 1, y);
134             x = find(x + 1); pushdown(x);
135             if (!c[x][1]) {c[x][1] = tmp; f[tmp] = x;}
136             else{
137                 x = c[x][1]; pushdown(x);
138                 while (c[x][0]) {
139                     x = c[x][0];
140                     pushdown(x);
141                 }
142                 c[x][0] = tmp; f[tmp] = x;
143             }
144             splay(tmp, -1);
145         }
146         else if (!strcmp(op, "DELETE")) {
147             scanf("%d%d", &x, &y); n -= y;
148             if (!y) continue;
149             getrange(x, y);
150             int k = (c[y][0] == x);
151             recycle(c[x][k]);
152             f[c[x][k]] = 0;
153             c[x][k] = 0;
154             splay(x, -1);
155         }
156         else if (!strcmp(op, "REVERSE")) {
157             scanf("%d%d", &x, &y);
158             if (!y) continue;
159             getrange(x, y);
160             int k = (c[y][0]==x);
161             makerev(c[x][k]);
162             splay(c[x][k], -1);
163         }
164         else if (!strcmp(op, "GET-SUM")) {
165             scanf("%d%d", &x, &y);
166             if (!y) {
167                 printf("0\n");

```

```

168         continue;
169     }
170     getrange(x,y);
171     int k = (c[y][0] == x);
172     printf("%d\n", t[c[x][k]].ms);
173     splay(c[x][k], -1);
174 }
175 else if (!strcmp(op, "MAX-SUM")) {
176     x = 1; y = n;
177     getrange(x, y);
178     int k = (c[y][0] == x);
179     printf("%d\n", t[c[x][k]].zs);
180     splay(c[x][k], -1);
181 }
182 else if (!strcmp(op, "MAKE-SAME")) {
183     scanf("%d%d%d", &x, &y, &tmp);
184     if (!y) continue;
185     getrange(x, y);
186     int k = (c[y][0] == x);
187     makesame(c[x][k], tmp);
188     splay(c[x][k], -1);
189 }
190 }
191 return 0;
192 }

```

3.5 坚固的 Treap

使用条件及注意事项: 题目来源 UVA 12358

```

1  int ran() {
2      static int ret = 182381727;
3      return (ret += (ret << 1) + 717271723) & (~0u >> 1);
4  }
5  int alloc(int node = 0) {
6      size++;
7      if (node) {
8          c[size][0] = c[node][0];
9          c[size][1] = c[node][1];
10         s[size] = s[node];
11         d[size] = d[node];
12     }
13     else{
14         c[size][0] = 0;
15         c[size][1] = 0;
16         s[size] = 1;
17         d[size] = ' ';
18     }
19     return size;
20 }
21 void update(int x) {
22     s[x] = 1;
23     if (c[x][0]) s[x] += s[c[x][0]];
24     if (c[x][1]) s[x] += s[c[x][1]];
25 }
26 int merge(const std::pair<int, int> &a) {
27     if (!a.first) return a.second;
28     if (!a.second) return a.first;
29     if (ran() % (s[a.first] + s[a.second]) < s[a.first]) {
30         int newnode = alloc(a.first);
31         c[newnode][1] = merge(std::make_pair(c[newnode][1], a.second));
32         update(newnode);
33         return newnode;
34     }
35     else{
36         int newnode = alloc(a.second);
37         c[newnode][0] = merge(std::make_pair(a.first, c[newnode][0]));
38         update(newnode);
39         return newnode;
40     }
41 }
42 std::pair<int, int> split(int x, int k) {
43     if (!x || !k) return std::make_pair(0, x);
44     int newnode = alloc(x);
45     if (k <= s[c[x][0]]) {
46         std::pair<int, int> ret = split(c[newnode][0], k);
47         c[newnode][0] = ret.second;
48         update(newnode);
49         return std::make_pair(ret.first, newnode);
50     }
51     else{
52         std::pair<int, int> ret = split(c[newnode][1], k - s[c[x][0]] - 1);

```

```

53     c[newnode][1] = ret.first;
54     update(newnode);
55     return std::make_pair(newnode, ret.second);
56 }
57 }
58 void travel(int x) {
59     if (c[x][0]) travel(c[x][0]);
60     putchar(d[x]);
61     if (d[x] == 'c') cnt++;
62     if (c[x][1]) travel(c[x][1]);
63 }
64 int build(int l, int r) {
65     int newnode = alloc();
66     d[newnode] = tmp[l + r >> 1];
67     if (l <= (l + r >> 1) - 1) c[newnode][0] = build(l, (l + r >> 1) - 1);
68     if ((l + r >> 1) + 1 <= r) c[newnode][1] = build((l + r >> 1) + 1, r);
69     update(newnode);
70     return newnode;
71 }
72 int main() {
73     scanf("%d", &n);
74     for (int i = 1, last = 0; i <= n; i++) {
75         int op, v, p, l;
76         scanf("%d", &op);
77         if (op == 1) {
78             scanf("%d%s", &p, tmp + 1);
79             p -= cnt;
80             std::pair<int, int> ret = split(rt[last], p);
81             rt[last + 1] = merge(std::make_pair(ret.first, build(1, strlen(tmp + 1))));
82             rt[last + 1] = merge(std::make_pair(rt[last + 1], ret.second));
83             last++;
84         }
85         else if (op == 2) {
86             scanf("%d%d", &p, &l);
87             p -= cnt; l -= cnt;
88             std::pair<int, int> A = split(rt[last], p - 1);
89             std::pair<int, int> B = split(A.second, l);
90             rt[last + 1] = merge(std::make_pair(A.first, B.second));
91             last++;
92         }
93         else if (op == 3) {
94             scanf("%d%d%d", &v, &p, &l);
95             v -= cnt; p -= cnt; l -= cnt;
96             std::pair<int, int> A = split(rt[v], p - 1);
97             std::pair<int, int> B = split(A.second, l);
98             travel(B.first);
99             puts("");
100         }
101     }
102     return 0;
103 }

```

3.6 k-d 树

使用条件及注意事项: 这是求 k 远点的代码, 要求 k 近点的话把堆的比较函数改一改, 把朝左儿子或者是右儿子的方向改一改。

```

1  struct Heapnode{
2      long long d;
3      int pos;
4      bool operator <(const Heapnode &p) const {
5          return d > p.d || (d == p.d && pos < p.pos);
6      }
7  };
8  struct MsgNode{
9      int xmin, xmax, ymin, ymax;
10     MsgNode() {}
11     MsgNode(const Point &a) : xmin(a.x), xmax(a.x), ymin(a.y), ymax(a.y) {}
12     long long dist(const Point &a) {
13         int dx = std::max(std::abs(a.x - xmin), std::abs(a.x - xmax));
14         int dy = std::max(std::abs(a.y - ymin), std::abs(a.y - ymax));
15         return (long long)dx * dx + (long long)dy * dy;
16     }
17     MsgNode operator +(const MsgNode &rhs) const {
18         MsgNode ret;
19         ret.xmin = std::min(xmin, rhs.xmin);
20         ret.xmax = std::max(xmax, rhs.xmax);
21         ret.ymin = std::min(ymin, rhs.ymin);
22         ret.ymax = std::max(ymax, rhs.ymax);
23         return ret;
24     }
25 };

```

```

26 struct TNode{
27     int l, r;
28     Point p;
29     MsgNode d;
30 }tree[MAXN];
31 void buildtree(int &rt, int l, int r, int pivot) {
32     if (l > r) return;
33     rt = ++size;
34     int mid = l + r >> 1;
35     if (pivot == 1) std::nth_element(p + l, p + mid, p + r + 1, cmpx);
36     if (pivot == 0) std::nth_element(p + l, p + mid, p + r + 1, cmpy);
37     tree[rt].d = MsgNode(tree[rt].p = p[mid]);
38     buildtree(tree[rt].l, l, mid - 1, pivot ^ 1);
39     buildtree(tree[rt].r, mid + 1, r, pivot ^ 1);
40     if (tree[rt].l) tree[rt].d = tree[rt].d + tree[tree[rt].l].d;
41     if (tree[rt].r) tree[rt].d = tree[rt].d + tree[tree[rt].r].d;
42 }
43 void query(int rt, const Point &a, int k, int pivot) {
44     Heapnode now = (Heapnode){dist(a, tree[rt].p), tree[rt].p.pos};
45     if (heap.size() < k) heap.push(now);
46     else if (now < heap.top()) {heap.pop(); heap.push(now);}
47     int lson = tree[rt].l, rson = tree[rt].r;
48     if (pivot == 1 && cmpx(a, tree[rt].p)) std::swap(lson, rson);
49     if (pivot == 0 && cmpy(a, tree[rt].p)) std::swap(lson, rson);
50     if (lson && (heap.size() < k || tree[lson].d.dist(a) >= heap.top().d)) query(lson, a, k,
51         pivot ^ 1);
52     if (rson && (heap.size() < k || tree[rson].d.dist(a) >= heap.top().d)) query(rson, a, k,
53         pivot ^ 1);
54 }
55 int main() {
56     for (int i = 1; i <= q; i++) {
57         int k;
58         Point now;
59         now.read();
60         scanf("%d", &k);
61         while (!heap.empty()) heap.pop();
62         query(rt, now, k, 1);
63         printf("%d\n", heap.top().pos);
64     }
65     return 0;
66 }

```

3.7 树链剖分

3.7.1 点操作版本

使用条件及注意事项：树上最大（非空）子段和，注意一条路径询问的时候信息统计的顺序。

```

1 struct Node{
2     int asum, lsum, rsum, zsum;
3     Node() {
4         asum = 0;
5         lsum = -INF;
6         rsum = -INF;
7         zsum = -INF;
8     }
9     Node(int d) : asum(d), lsum(d), rsum(d), zsum(d) {}
10    Node operator +(const Node &rhs) const {
11        Node ret;
12        ret.asum = asum + rhs.asum;
13        ret.lsum = std::max(lsum, asum + rhs.lsum);
14        ret.rsum = std::max(rsum + rhs.asum, rhs.rsum);
15        ret.zsum = std::max(zsum, rhs.zsum);
16        ret.zsum = std::max(ret.zsum, rsum + rhs.lsum);
17        return ret;
18    }
19 }tree[MAXN * 6];
20 int n, q, cnt, tot, h[MAXN], d[MAXN], t[MAXN], f[MAXN], s[MAXN], z[MAXN], w[MAXN], o[MAXN]
21     , a[MAXN];
22 std::pair<bool, int> flag[MAXN * 6];
23 void addedge(int x, int y) {
24     cnt++; e[cnt] = (Edge){y, h[x]}; h[x] = cnt;
25     cnt++; e[cnt] = (Edge){x, h[y]}; h[y] = cnt;
26 }
27 void makesame(int n, int l, int r, int d) {
28     flag[n] = std::make_pair(true, d);
29     tree[n].asum = d * (r - l + 1);
30     if (d > 0) {
31         tree[n].lsum = d * (r - l + 1);
32         tree[n].rsum = d * (r - l + 1);
33         tree[n].zsum = d * (r - l + 1);
34     }
35 }

```

```

34     else{
35         tree[n].lsum = d;
36         tree[n].rsum = d;
37         tree[n].zsum = d;
38     }
39 }
40 void pushdown(int n, int l, int r) {
41     if (flag[n].first) {
42         makesame(n << 1, l, l + r >> 1, flag[n].second);
43         makesame(n << 1 ^ 1, (l + r >> 1) + 1, r, flag[n].second);
44         flag[n] = std::make_pair(false, 0);
45     }
46 }
47 void modify(int n, int l, int r, int x, int y, int d) {
48     if (x <= l && r <= y) {
49         makesame(n, l, r, d);
50         return;
51     }
52     pushdown(n, l, r);
53     if ((l + r >> 1) < x) modify(n << 1 ^ 1, (l + r >> 1) + 1, r, x, y, d);
54     else if ((l + r >> 1) + 1 > y) modify(n << 1, l, l + r >> 1, x, y, d);
55     else{
56         modify(n << 1, l, l + r >> 1, x, y, d);
57         modify(n << 1 ^ 1, (l + r >> 1) + 1, r, x, y, d);
58     }
59     tree[n] = tree[n << 1] + tree[n << 1 ^ 1];
60 }
61 Node query(int n, int l, int r, int x, int y) {
62     if (x <= l && r <= y) return tree[n];
63     pushdown(n, l, r);
64     if ((l + r >> 1) < x) return query(n << 1 ^ 1, (l + r >> 1) + 1, r, x, y);
65     else if ((l + r >> 1) + 1 > y) return query(n << 1, l, l + r >> 1, x, y);
66     else{
67         Node left = query(n << 1, l, l + r >> 1, x, y);
68         Node right = query(n << 1 ^ 1, (l + r >> 1) + 1, r, x, y);
69         return left + right;
70     }
71 }
72 void modify(int x, int y, int val) {
73     int fx = t[x], fy = t[y];
74     while (fx != fy) {
75         if (d[fx] > d[fy]) {
76             modify(1, 1, n, w[fx], w[x], val);
77             x = f[fx]; fx = t[x];
78         }
79         else{
80             modify(1, 1, n, w[fy], w[y], val);
81             y = f[fy]; fy = t[y];
82         }
83     }
84     if (d[x] < d[y]) modify(1, 1, n, w[x], w[y], val);
85     else modify(1, 1, n, w[y], w[x], val);
86 }
87 Node query(int x, int y) {
88     int fx = t[x], fy = t[y];
89     Node left = Node(), right = Node();
90     while (fx != fy) {
91         if (d[fx] > d[fy]) {
92             left = query(1, 1, n, w[fx], w[x]) + left;
93             x = f[fx]; fx = t[x];
94         }
95         else{
96             right = query(1, 1, n, w[fy], w[y]) + right;
97             y = f[fy]; fy = t[y];
98         }
99     }
100     if (d[x] < d[y]) {
101         right = query(1, 1, n, w[x], w[y]) + right;
102     }
103     else{
104         left = query(1, 1, n, w[y], w[x]) + left;
105     }
106     std::swap(left.lsum, left.rsum);
107     return left + right;
108 }
109 void predfs(int x) {
110     s[x] = 1; z[x] = 0;
111     for (int i = h[x]; i; i = e[i].next) {
112         if (e[i].node == f[x]) continue;
113         f[e[i].node] = x;
114         d[e[i].node] = d[x] + 1;
115         predfs(e[i].node);
116         s[x] += s[e[i].node];
117     }
118 }

```



```

117     if (s[z[x]] < s[e[i].node]) z[x] = e[i].node;
118 }
119
120 void getanc(int x, int anc) {
121     t[x] = anc; w[x] = ++tot; o[tot] = x;
122     if (z[x]) getanc(z[x], anc);
123     for (int i = h[x]; i; i = e[i].next) {
124         if (e[i].node == f[x] || e[i].node == z[x]) continue;
125         getanc(e[i].node, e[i].node);
126     }
127 }
128 void buildtree(int n, int l, int r) {
129     if (l == r) {
130         tree[n] = Node(a[o[l]]);
131         return;
132     }
133     buildtree(n << 1, l, l + r >> 1);
134     buildtree(n << 1 ^ 1, (l + r >> 1) + 1, r);
135     tree[n] = tree[n << 1] + tree[n << 1 ^ 1];
136 }
137 int main() {
138     scanf("%d", &n);
139     for (int i = 1; i <= n; i++) scanf("%d", a + i);
140     for (int i = 1; i < n; i++) {
141         int x, y; scanf("%d%d", &x, &y);
142         addedge(x, y);
143     }
144     predfs(1);
145     getanc(1, 1);
146     buildtree(1, 1, n);
147     scanf("%d", &q);
148     for (int i = 1; i <= q; i++) {
149         int op, x, y, c;
150         scanf("%d", &op);
151         if (op == 1) {
152             scanf("%d%d", &x, &y);
153             Node ret = query(x, y);
154             printf("%d\n", std::max(0, ret.zsum));
155         }
156         else{
157             scanf("%d%d%d", &x, &y, &c);
158             modify(x, y, c);
159         }
160     }
161     return 0;
162 }

```

3.7.2 链操作版本

```

1 void modify(int x, int y) {
2     int fx = t[x], fy = t[y];
3     while (fx != fy) {
4         if (d[fx] > d[fy]) {
5             modify(1, 1, n, w[fx], w[x]);
6             x = f[fx]; fx = t[x];
7         }
8         else{
9             modify(1, 1, n, w[fy], w[y]);
10            y = f[fy]; fy = t[y];
11        }
12    }
13    if (x != y) {
14        if (d[x] < d[y]) modify(1, 1, n, w[z[x]], w[y]);
15        else modify(1, 1, n, w[z[y]], w[x]);
16    }
17 }

```

3.8 Link-Cut-Tree

```

1 struct MsgNode{
2     int leftColor, rightColor, answer;
3     MsgNode() {
4         leftColor = -1;
5         rightColor = -1;
6         answer = 0;
7     }
8     MsgNode(int c) {
9         leftColor = rightColor = c;
10        answer = 1;
11    }

```

```

12     MsgNode operator +(const MsgNode &p) const {
13         if (answer == 0) return p;
14         if (p.answer == 0) return *this;
15         MsgNode ret;
16         ret.leftColor = leftColor;
17         ret.rightColor = p.rightColor;
18         ret.answer = answer + p.answer - (rightColor == p.leftColor);
19         return ret;
20     }
21 }d[MAXN], g[MAXN];
22 int n, m, c[MAXN][2], f[MAXN], p[MAXN], s[MAXN], flag[MAXN];
23 bool r[MAXN];
24 void init(int x, int value) {
25     d[x] = g[x] = MsgNode(value);
26     c[x][0] = c[x][1] = 0;
27     f[x] = p[x] = flag[x] = -1;
28     s[x] = 1;
29 }
30 void update(int x) {
31     s[x] = s[c[x][0]] + s[c[x][1]] + 1;
32     g[x] = MsgNode();
33     if (c[x][0 ^ r[x]]) g[x] = g[x] + g[c[x][0 ^ r[x]]];
34     g[x] = g[x] + d[x];
35     if (c[x][1 ^ r[x]]) g[x] = g[x] + g[c[x][1 ^ r[x]]];
36 }
37 void makesame(int x, int c) {
38     flag[x] = c;
39     d[x] = MsgNode(c);
40     g[x] = MsgNode(c);
41 }
42 void pushdown(int x) {
43     if (r[x]) {
44         std::swap(c[x][0], c[x][1]);
45         r[c[x][0]] ^= 1;
46         r[c[x][1]] ^= 1;
47         std::swap(g[c[x][0]].leftColor, g[c[x][0]].rightColor);
48         std::swap(g[c[x][1]].leftColor, g[c[x][1]].rightColor);
49         r[x] = false;
50     }
51     if (flag[x] != -1) {
52         if (c[x][0]) makesame(c[x][0], flag[x]);
53         if (c[x][1]) makesame(c[x][1], flag[x]);
54         flag[x] = -1;
55     }
56 }
57 void rotate(int x, int k) {
58     pushdown(x); pushdown(c[x][k]);
59     int y = c[x][k]; c[x][k] = c[y][k ^ 1]; c[y][k ^ 1] = x;
60     if (f[x] != -1) c[f[x]][c[f[x]][1] == x] = y;
61     f[y] = f[x]; f[x] = y; f[c[x][k]] = x; std::swap(p[x], p[y]);
62     update(x); update(y);
63 }
64 void splay(int x, int s = -1) {
65     pushdown(x);
66     while (f[x] != s) {
67         if (f[f[x]] != s) rotate(f[f[x]], (c[f[f[x]]][1] == f[x]) ^ r[f[f[x]]]);
68         rotate(f[x], (c[f[x]][1] == x) ^ r[f[x]]);
69     }
70     update(x);
71 }
72 void access(int x) {
73     int y = 0;
74     while (x != -1) {
75         splay(x); pushdown(x);
76         f[c[x][1]] = -1; p[c[x][1]] = x;
77         c[x][1] = y; f[y] = x; p[y] = -1;
78         update(x); x = p[y = x];
79     }
80 }
81 void setroot(int x) {
82     access(x);
83     splay(x);
84     r[x] ^= 1;
85     std::swap(g[x].leftColor, g[x].rightColor);
86 }
87 void link(int x, int y) {
88     setroot(x);
89     p[x] = y;
90 }

```

第4章 图论

4.1 点双连通分量

4.1.1 坚固的点双连通分量

```

1 int n, m, x, y, ans1, ans2, tot1, tot2, flag, size, ind2, dfn[N], low[N], block[M], vis[N]
2 ;
3 vector<int> a[N];
4 pair<int, int> stack[M];
5 void tarjan(int x, int p) {
6     dfn[x] = low[x] = ++ind2;
7     for (int i = 0; i < a[x].size(); ++i)
8         if (dfn[a[x][i]] > dfn[a[x][i]] && a[x][i] != p) {
9             stack[++size] = make_pair(x, a[x][i]);
10            if (i == a[x].size() - 1 || a[x][i] != a[x][i + 1])
11                if (!dfn[a[x][i]]) {
12                    tarjan(a[x][i], x);
13                    low[x] = min(low[x], low[a[x][i]]);
14                    if (low[a[x][i]] >= dfn[x]) {
15                        tot1 = tot2 = 0;
16                        ++flag;
17                        for (; ; ) {
18                            if (block[stack[size].first] != flag) {
19                                ++tot1;
20                                block[stack[size].first] = flag;
21                            }
22                            if (block[stack[size].second] != flag) {
23                                ++tot1;
24                                block[stack[size].second] = flag;
25                            }
26                            if (stack[size].first == x && stack[size].second == a[x][i])
27                                break;
28                            ++tot2;
29                            --size;
30                        }
31                        for (; stack[size].first == x && stack[size].second == a[x][i]; --size)
32                            ++tot2;
33                        if (tot2 < tot1)
34                            ans1 += tot2;
35                        if (tot2 > tot1)
36                            ans2 += tot2;
37                    }
38                }
39            else
40                low[x] = min(low[x], dfn[a[x][i]]);
41        }
42    }
43    int main() {
44        for (; ; ) {
45            scanf("%d%d", &n, &m);
46            if (n == 0 && m == 0) return 0;
47            for (int i = 1; i <= n; ++i) {
48                a[i].clear();
49                dfn[i] = 0;
50            }
51            for (int i = 1; i <= m; ++i) {
52                scanf("%d%d", &x, &y);
53                ++x, ++y;
54                a[x].push_back(y);
55                a[y].push_back(x);
56            }
57            sort(a[1].begin(), a[1].end());
58            ans1 = ans2 = ind2 = 0;
59            for (int i = 1; i <= n; ++i)
60                if (!dfn[i]) {
61                    size = 0;
62                    tarjan(i, 0);
63                }
64            printf("%d %d\n", ans1, ans2);
65        }
66        return 0;
67    }

```

4.1.2 朴素的点双连通分量

```

1 void tarjan(int x) {
2     dfn[x] = low[x] = ++ind2;
3     v[x] = 1;
4     for (int i = nt[x]; pt[i]; i = nt[i])

```

```

5     if (!dfn[pt[i]]) {
6         tarjan(pt[i]);
7         low[x] = min(low[x], low[pt[i]]);
8         if (dfn[x] <= low[pt[i]])
9             ++v[x];
10    }
11    else
12        low[x] = min(low[x], dfn[pt[i]]);
13 }
14 int main() {
15     for (; ; ) {
16         scanf("%d%d", &n, &m);
17         if (n == 0 && m == 0)
18             return 0;
19         for (int i = 1; i <= ind; ++i)
20             nt[i] = pt[i] = 0;
21         ind = n;
22         for (int i = 1; i <= ind; ++i)
23             last[i] = i;
24         for (int i = 1; i <= m; ++i) {
25             scanf("%d%d", &x, &y);
26             ++x, ++y;
27             edge(x, y), edge(y, x);
28         }
29         memset(dfn, 0, sizeof(dfn));
30         memset(v, 0, sizeof(v));
31         ans = num = ind2 = 0;
32         for (int i = 1; i <= n; ++i)
33             if (!dfn[i]) {
34                 root = i;
35                 size = 0;
36                 ++num;
37                 tarjan(i);
38                 --v[root];
39             }
40         for (int i = 1; i <= n; ++i)
41             if (v[i] + num - 1 > ans)
42                 ans = v[i] + num - 1;
43         printf("%d\n", ans);
44     }
45     return 0;
46 }

```

4.2 2-SAT 问题

```

1 int stamp, comps, top;
2 int dfn[N], low[N], comp[N], stack[N];
3 void add(int x, int a, int y, int b) {
4     edge[x << 1 | a].push_back(y << 1 | b);
5 }
6 void tarjan(int x) {
7     dfn[x] = low[x] = ++stamp;
8     stack[top++] = x;
9     for (int i = 0; i < (int)edge[x].size(); ++i) {
10        int y = edge[x][i];
11        if (!dfn[y]) {
12            tarjan(y);
13            low[x] = std::min(low[x], low[y]);
14        } else if (!comp[y]) {
15            low[x] = std::min(low[x], dfn[y]);
16        }
17    }
18    if (low[x] == dfn[x]) {
19        comps++;
20        do {
21            int y = stack[--top];
22            comp[y] = comps;
23        } while (stack[top] != x);
24    }
25 }
26 bool solve() {
27     int counter = n + n + 1;
28     stamp = top = comps = 0;
29     std::fill(dfn, dfn + counter, 0);
30     std::fill(comp, comp + counter, 0);
31     for (int i = 0; i < counter; ++i) {
32         if (!dfn[i]) {
33             tarjan(i);
34         }
35     }
36     for (int i = 0; i < n; ++i) {

```

```

37     if (comp[i << 1] == comp[i << 1 | 1]) {
38         return false;
39     }
40     answer[i] = (comp[i << 1 | 1] < comp[i << 1]);
41 }
42 return true;
43 }

```

4.3 二分图最大匹配

4.3.1 Hopcroft Karp 算法

时间复杂度: $\mathcal{O}(\sqrt{V} \cdot E)$

```

1  int matchx[N], matchy[N], level[N];
2  bool dfs(int x) {
3      for (int i = 0; i < (int)edge[x].size(); ++i) {
4          int y = edge[x][i];
5          int w = matchy[y];
6          if (w == -1 || level[x] + 1 == level[w] && dfs(w)) {
7              matchx[x] = y;
8              matchy[y] = x;
9              return true;
10         }
11     }
12     level[x] = -1;
13     return false;
14 }
15 int solve() {
16     std::fill(matchx, matchx + n, -1);
17     std::fill(matchy, matchy + m, -1);
18     for (int answer = 0; ; ) {
19         std::vector<int> queue;
20         for (int i = 0; i < n; ++i) {
21             if (matchx[i] == -1) {
22                 level[i] = 0;
23                 queue.push_back(i);
24             } else {
25                 level[i] = -1;
26             }
27         }
28         for (int head = 0; head < (int)queue.size(); ++head) {
29             int x = queue[head];
30             for (int i = 0; i < (int)edge[x].size(); ++i) {
31                 int y = edge[x][i];
32                 int w = matchy[y];
33                 if (w != -1 && level[w] < 0) {
34                     level[w] = level[x] + 1;
35                     queue.push_back(w);
36                 }
37             }
38         }
39         int delta = 0;
40         for (int i = 0; i < n; ++i) {
41             if (matchx[i] == -1 && dfs(i)) {
42                 delta++;
43             }
44         }
45         if (delta == 0) {
46             return answer;
47         } else {
48             answer += delta;
49         }
50     }
51 }

```

4.4 二分图最大权匹配

时间复杂度: $\mathcal{O}(V^4)$

```

1  int DFS(int x){
2      visx[x] = 1;
3      for (int y = 1; y <= ny; y++){
4          if (visy[y]) continue;
5          int t = lx[x] + ly[y] - w[x][y];
6          if (t == 0) {
7              visy[y] = 1;
8              if (link[y] == -1 || DFS(link[y])){
9                  link[y] = x;
10                 return 1;

```

```

11         }
12     }
13     else slack[y] = min(slack[y], t);
14 }
15 return 0;
16 }
17 int KM() {
18     int i, j;
19     memset(link, -1, sizeof(link));
20     memset(ly, 0, sizeof(ly));
21     for (i = 1; i <= nx; i++)
22         for (j = 1, lx[i] = -inf; j <= ny; j++)
23             lx[i] = max(lx[i], w[i][j]);
24     for (int x = 1; x <= nx; x++){
25         for (i = 1; i <= ny; i++) slack[i] = inf;
26         while (true) {
27             memset(visx, 0, sizeof(visx));
28             memset(visy, 0, sizeof(visy));
29             if (DFS(x)) break;
30             int d = inf;
31             for (i = 1; i <= ny; i++)
32                 if (!visy[i] && d > slack[i]) d = slack[i];
33             for (i = 1; i <= nx; i++)
34                 if (visx[i]) lx[i] -= d;
35             for (i = 1; i <= ny; i++)
36                 if (visy[i]) ly[i] += d;
37             else slack[i] -= d;
38         }
39     }
40     int res = 0;
41     for (i = 1; i <= ny; i++)
42         if (link[i] > -1) res += w[link[i]][i];
43     return res;
44 }

```

时间复杂度: $\mathcal{O}(V^3)$

```

1  //最小权匹配
2  struct KM State {
3      int lx[N], ly[N], match[N], way[N];
4      KM State() {
5          for (int i = 1; i <= n; i++) {
6              match[i] = 0;
7              lx[i] = 0;
8              ly[i] = 0;
9              way[i] = 0;
10         }
11     }
12 };
13 struct KM Solver {
14     int w[N][N], slack[N];
15     KM State state;
16     bool used[N];
17     KM Solver() {
18         for (int i = 1; i <= n; i++) {
19             for (int j = 1; j <= n; j++) {
20                 w[i][j] = 0;
21             }
22         }
23     }
24     void hungary(int x) {
25         state.match[0] = x;
26         int j0 = 0;
27         for (int j = 0; j <= n; j++) {
28             slack[j] = INF;
29             used[j] = false;
30         }
31         do {
32             used[j0] = true;
33             int i0 = state.match[j0], delta = INF, j1;
34             for (int j = 1; j <= n; j++) {
35                 if (used[j] == false) {
36                     int cur = w[i0][j] - state.lx[i0] - state.ly[j];
37                     if (cur < slack[j]) {
38                         slack[j] = cur;
39                         state.way[j] = j0;
40                     }
41                     if (slack[j] < delta) {
42                         delta = slack[j];
43                         j1 = j;
44                     }
45                 }
46             }

```

```

47     for (int j = 0; j <= n; j++) {
48         if (used[j]) {
49             state.lx[state.match[j]] += delta;
50             state.ly[j] -= delta;
51         } else {
52             slack[j] -= delta;
53         }
54     }
55     j0 = j1;
56     while (state.match[j0] != 0) {
57         do {
58             int j1 = state.way[j0];
59             state.match[j0] = state.match[j1];
60             j0 = j1;
61         } while (j0);
62     }
63     int get_ans() {
64         int ret = 0;
65         for (int i = 1; i <= n; i++) {
66             if (state.match[i] > 0) {
67                 ret += w[state.match[i]][i];
68             }
69         }
70         return state.ly[0];
71     }
72 };

```

4.5 最大流

4.5.1 ISAP

时间复杂度: $\mathcal{O}(V^2 \cdot E)$

```

1 int Maxflow Isap(int s, int t, int n) {
2     std::fill(pre + 1, pre + n + 1, 0);
3     std::fill(d + 1, d + n + 1, 0);
4     std::fill(gap + 1, gap + n + 1, 0);
5     for (int i = 1; i <= n; i++) cur[i] = h[i];
6     gap[0] = n;
7     int u = pre[s] = s, v, maxflow = 0;
8     while (d[s] < n) {
9         v = n + 1;
10        for (int i = cur[u]; i; i = e[i].next)
11            if (e[i].flow && d[u] == d[e[i].node] + 1) {
12                v = e[i].node; cur[u] = i; break;
13            }
14        if (v <= n) {
15            pre[v] = u; u = v;
16            if (v == t) {
17                int dflow = INF, p = t; u = s;
18                while (p != s) {
19                    p = pre[p];
20                    dflow = std::min(dflow, e[cur[p]].flow);
21                }
22                maxflow += dflow; p = t;
23                while (p != s) {
24                    p = pre[p];
25                    e[cur[p]].flow -= dflow;
26                    e[e[cur[p]].opp].flow += dflow;
27                }
28            }
29        } else {
30            int mindist = n + 1;
31            for (int i = h[u]; i; i = e[i].next)
32                if (e[i].flow && mindist > d[e[i].node]) {
33                    mindist = d[e[i].node]; cur[u] = i;
34                }
35            if (!--gap[d[u]]) return maxflow;
36            gap[d[u] = mindist + 1]++; u = pre[u];
37        }
38    }
39    return maxflow;
40 }
41

```

4.5.2 SAP

时间复杂度: $\mathcal{O}(V^2 \cdot E)$

```

1 const int N = 110, M = 30110, INF = 1000000000; // 边表不要开小
2 int n, m, ind, S, T, flow, tot, pt[M], nt[M], last[N], size[M], num[N], h[N], now[N];
3 void edge(int x, int y, int z) {

```

```

4     last[x] = nt[last[x]] = ++ind;
5     pt[ind] = y, size[ind] = z;
6 }
7 int aug(int x, int y) {
8     if (x == T)
9         return y;
10    int f = y;
11    for (int i = now[x]; pt[i]; i = nt[i])
12        if (size[i] && h[pt[i]] + 1 == h[x]) {
13            int z = aug(pt[i], min(f, size[i]));
14            f -= z;
15            size[i] -= z;
16            size[i ^ 1] += z;
17            now[x] = i;
18            if (h[S] > tot || f == 0)
19                return y - f;
20        }
21    now[x] = nt[x];
22    if (--num[h[x]] == 0)
23        h[S] = tot + 1;
24    ++num[++h[x]];
25    return y - f;
26 }
27 int main() {
28     int np, nc;
29     for (; scanf("%d%d%d", &n, &np, &nc, &m) == 4; ) {
30         for (int i = 0; i <= ind; ++i)
31             pt[i] = nt[i] = last[i] = size[i] = 0;
32         ind = n + 2;
33         if (ind % 2 == 0)
34             ++ind;
35         S = n + 1, tot = T = n + 2;
36         for (int i = 0; i <= tot; ++i)
37             num[i] = h[i] = now[i] = 0;
38         for (int i = 1; i <= tot; ++i)
39             last[i] = i;
40         for (int i = 1; i <= m; ++i) {
41             int x, y, z;
42             for (; getchar() != '('; );
43             scanf("%d%c%d%c%d", &x, &y, &z);
44             ++x, ++y;
45             edge(x, y, z);
46             edge(y, x, 0);
47         }
48         for (int i = 1; i <= np; ++i) {
49             int y, z;
50             for (; getchar() != '('; );
51             scanf("%d%c%d", &y, &z);
52             ++y;
53             edge(S, y, z);
54             edge(y, S, 0);
55         }
56         for (int i = 1; i <= nc; ++i) {
57             int x, z;
58             for (; getchar() != '('; );
59             scanf("%d%c%d", &x, &z);
60             ++x;
61             edge(x, T, z);
62             edge(T, x, 0);
63         }
64         num[0] = tot;
65         for (int i = 1; i <= tot; ++i)
66             now[i] = nt[i];
67         flow = 0;
68         for (; h[S] <= T; )
69             flow += aug(S, INF);
70         printf("%d\n", flow);
71     }
72     return 0;
73 }

```

4.6 上下界网络流

$B(u, v)$ 表示边 (u, v) 流量的下界, $C(u, v)$ 表示边 (u, v) 流量的上界, $F(u, v)$ 表示边 (u, v) 的流量。设 $G(u, v) = F(u, v) - B(u, v)$, 显然有

$$0 \leq G(u, v) \leq C(u, v) - B(u, v)$$

4.6.1 无源汇的上下界可行流

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

4.6.2 有源汇的上下界可行流

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

4.6.3 有源汇的上下界最大流

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

4.6.4 有源汇的上下界最小流

- 在有源汇的上下界可行流中，从汇点 T 到源点 S 的边改为连一条上界为 x ，下界为 0 的边。 x 满足二分性质，找到最小的 x 使得新网络存在无源汇的上下界可行流即为原图的最小流。
- 按照无源汇的上下界可行流的方法，建立超级源点 S^* 与超级汇点 T^* ，求一遍 $S^* \rightarrow T^*$ 的最大流，但是注意这一次不加汇点 T 到源点 S 的这条边，即不使之改为无源汇的网络去求解。求完后，再加上那条汇点 T 到源点 S 上界 ∞ 的边。因为这条边下界为 0，所以 S^* ， T^* 无影响，再直接求一次 $S^* \rightarrow T^*$ 的最大流。若超级源点 S^* 出发的边全部满流，则 $T \rightarrow S$ 边上的流量即为原图的最小流，否则无解。

4.7 最小费用最大流

4.7.1 稀疏图

时间复杂度： $O(V \cdot E^2)$

```
1 struct EdgeList {
2     int size;
3     int last[N];
4     int succ[M], other[M], flow[M], cost[M];
5     void clear(int n) {
6         size = 0;
7         std::fill(last, last + n, -1);
8     }
9     void add(int x, int y, int c, int w) {
10         succ[size] = last[x];
11         last[x] = size;
12         other[size] = y;
13         flow[size] = c;
14         cost[size++] = w;
15     }
16 } e;
17 int n, source, target, prev[N];
18 void add(int x, int y, int c, int w) {
19     e.add(x, y, c, w);
20     e.add(y, x, 0, -w);
21 }
22 bool augment() {
23     static int dist[N], occur[N];
24     std::vector<int> queue;
25     std::fill(dist, dist + n, INT_MAX);
26     std::fill(occur, occur + n, 0);
27     dist[source] = 0;
28     occur[source] = true;
29     queue.push_back(source);
30     for (int head = 0; head < (int)queue.size(); ++head) {
31         int x = queue[head];
32         for (int i = e.last[x]; ~i; i = e.succ[i]) {
33             int y = e.other[i];
34             if (e.flow[i] && dist[y] > dist[x] + e.cost[i]) {
35                 dist[y] = dist[x] + e.cost[i];
36                 prev[y] = i;
37                 if (!occur[y]) {
38                     occur[y] = true;
39                     queue.push_back(y);
40                 }
41             }
42         }
43         occur[x] = false;
44     }
```

```
45     return dist[target] < INT_MAX;
46 }
47 std::pair<int, int> solve() {
48     std::pair<int, int> answer = std::make_pair(0, 0);
49     while (augment()) {
50         int number = INT_MAX;
51         for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
52             number = std::min(number, e.flow[prev[i]]);
53         }
54         answer.first += number;
55         for (int i = target; i != source; i = e.other[prev[i] ^ 1]) {
56             e.flow[prev[i]] -= number;
57             e.flow[prev[i] ^ 1] += number;
58             answer.second += number * e.cost[prev[i]];
59         }
60     }
61     return answer;
62 }
```

4.7.2 稠密图

使用条件：费用非负

时间复杂度： $O(V \cdot E^2)$

```
1 int aug(int no, int res) {
2     if (no == t) return cost += pil * res, res;
3     v[no] = true;
4     int flow = 0;
5     for (int i = h[no]; ~i; i = nxt[i])
6         if (cap[i] && !expense[i] && !v[p[i]]) {
7             int d = aug(p[i], min(res, cap[i]));
8             cap[i] -= d, cap[i ^ 1] += d, flow += d, res -= d;
9             if (!res) return flow;
10        }
11    return flow;
12 }
13 bool modlabel() {
14     int d = maxint;
15     for (int i = 1; i <= t; ++i)
16         if (v[i]) {
17             for (int j = h[i]; ~j; j = nxt[j])
18                 if (cap[j] && !v[p[j]] && expense[j] < d) d = expense[j];
19         }
20     if (d == maxint) return false;
21     for (int i = 1; i <= t; ++i)
22         if (v[i]) {
23             for (int j = h[i]; ~j; j = nxt[j])
24                 expense[j] -= d, expense[j ^ 1] += d;
25         }
26     pil += d;
27     return true;
28 }
29 void minimum_cost_flow_zkw() {
30     cost = 0;
31     do {
32         do {
33             memset(v, false, sizeof v);
34             while (aug(s, maxint));
35         } while (!modlabel());
36     }
```

4.8 一般图最大匹配

时间复杂度： $O(V^3)$

```
1 int match[N], belong[N], next[N], mark[N], visit[N];
2 std::vector<int> queue;
3 int find(int x) {
4     if (belong[x] != x) {
5         belong[x] = find(belong[x]);
6     }
7     return belong[x];
8 }
9 void merge(int x, int y) {
10     x = find(x);
11     y = find(y);
12     if (x != y) {
13         belong[x] = y;
14     }
15 }
16 int lca(int x, int y) {
```

```

17 static int stamp = 0;
18 stamp++;
19 while (true) {
20     if (x != -1) {
21         x = find(x);
22         if (visit[x] == stamp) {
23             return x;
24         }
25         visit[x] = stamp;
26         if (match[x] != -1) {
27             x = next[match[x]];
28         } else {
29             x = -1;
30         }
31     }
32     std::swap(x, y);
33 }
34
35 void group(int a, int p) {
36     while (a != p) {
37         int b = match[a], c = next[b];
38         if (find(c) != p) {
39             next[c] = b;
40         }
41         if (mark[b] == 2) {
42             mark[b] = 1;
43             queue.push_back(b);
44         }
45         if (mark[c] == 2) {
46             mark[c] = 1;
47             queue.push_back(c);
48         }
49         merge(a, b);
50         merge(b, c);
51         a = c;
52     }
53 }
54 void augment(int source) {
55     queue.clear();
56     for (int i = 0; i < n; ++i) {
57         next[i] = visit[i] = -1;
58         belong[i] = i;
59         mark[i] = 0;
60     }
61     mark[source] = 1;
62     queue.push_back(source);
63     for (int head = 0; head < (int)queue.size() && match[source] == -1; ++head) {
64         int x = queue[head];
65         for (int i = 0; i < (int)edge[x].size(); ++i) {
66             int y = edge[x][i];
67             if (match[x] == y || find(x) == find(y) || mark[y] == 2) {
68                 continue;
69             }
70             if (mark[y] == 1) {
71                 int r = lca(x, y);
72                 if (find(x) != r) {
73                     next[x] = y;
74                 }
75                 if (find(y) != r) {
76                     next[y] = x;
77                 }
78                 group(x, r);
79                 group(y, r);
80             } else if (match[y] == -1) {
81                 next[y] = x;
82                 for (int u = y; u != -1; ) {
83                     int v = next[u];
84                     int mv = match[v];
85                     match[v] = u;
86                     match[u] = v;
87                     u = mv;
88                 }
89                 break;
90             } else {
91                 next[y] = x;
92                 mark[y] = 2;
93                 mark[match[y]] = 1;
94                 queue.push_back(match[y]);
95             }
96         }
97     }
98 }
99 int solve() {

```

```

100 std::fill(match, match + n, -1);
101 for (int i = 0; i < n; ++i) {
102     if (match[i] == -1) {
103         augment(i);
104     }
105 }
106 int answer = 0;
107 for (int i = 0; i < n; ++i) {
108     answer += (match[i] != -1);
109 }
110 return answer;
111 }

```

4.9 无向图全局最小割

时间复杂度: $\mathcal{O}(V^3)$

注意事项: 处理重边时, 应该对边权累加

```

1 int node[N], dist[N];
2 bool visit[N];
3 int solve(int n) {
4     int answer = INT_MAX;
5     for (int i = 0; i < n; ++i) {
6         node[i] = i;
7     }
8     while (n > 1) {
9         int max = 1;
10        for (int i = 0; i < n; ++i) {
11            dist[node[i]] = graph[node[0]][node[i]];
12            if (dist[node[i]] > dist[node[max]]) {
13                max = i;
14            }
15        }
16        int prev = 0;
17        memset(visit, 0, sizeof(visit));
18        visit[node[0]] = true;
19        for (int i = 1; i < n; ++i) {
20            if (i == n - 1) {
21                answer = std::min(answer, dist[node[max]]);
22                for (int k = 0; k < n; ++k) {
23                    graph[node[k]][node[prev]] =
24                        (graph[node[prev]][node[k]] += graph[node[k]][node[max]]);
25                }
26                node[max] = node[prev];
27            }
28            visit[node[max]] = true;
29            prev = max;
30            max = -1;
31            for (int j = 1; j < n; ++j) {
32                if (!visit[node[j]]) {
33                    dist[node[j]] += graph[node[prev]][node[j]];
34                    if (max == -1 || dist[node[max]] < dist[node[j]]) {
35                        max = j;
36                    }
37                }
38            }
39        }
40    }
41    return answer;
42 }

```

4.10 最小树形图

```

1 int n, m, used[N], pass[N], eg[N], more, queue[N];
2 double g[N][N];
3 void combine(int id, double &sum) {
4     int tot = 0, from, i, j, k;
5     for (; id != 0 && !pass[id]; id = eg[id]) {
6         queue[tot++] = id;
7         pass[id] = 1;
8     }
9     for (from = 0; from < tot && queue[from] != id; from++);
10    if (from == tot) return;
11    more = 1;
12    for (i = from; i < tot; i++) {
13        sum += g[eg[queue[i]]][queue[i]];
14        if (i != from) {
15            used[queue[i]] = 1;
16            for (j = 1; j <= n; j++) if (!used[j]) {

```

```

17     if (g[queue[i]][j] < g[id][j]) g[id][j] = g[queue[i]][j];
18 }
19 }
20 }
21 for (i = 1; i <= n; i++) if (!used[i] && i != id) {
22     for (j = from; j < tot; j++) {
23         k = queue[j];
24         if (g[i][id] > g[i][k] - g[eg[k]][k]) g[i][id] = g[i][k] - g[eg[k]][k];
25     }
26 }
27 }
28 double mdst(int root) {
29     int i, j, k;
30     double sum = 0;
31     memset(used, 0, sizeof(used));
32     for (more = 1; more; ) {
33         more = 0;
34         memset(eg, 0, sizeof(eg));
35         for (i = 1; i <= n; i++) if (!used[i] && i != root) {
36             for (j = 1, k = 0; j <= n; j++) if (!used[j] && i != j)
37                 if (k == 0 || g[j][i] < g[k][i]) k = j;
38             eg[i] = k;
39         }
40         memset(pass, 0, sizeof(pass));
41         for (i = 1; i <= n; i++) if (!used[i] && !pass[i] && i != root) combine(i, sum);
42     }
43 }
44 for (i = 1; i <= n; i++) if (!used[i] && i != root) sum += g[eg[i]][i];
45 return sum;
46 }

```

4.11 有根树的同构

时间复杂度: $O(V \log V)$

```

1 const unsigned long long MAGIC = 4423;
2 unsigned long long magic[N];
3 std::pair<unsigned long long, int> hash[N];
4 void solve(int root) {
5     magic[0] = 1;
6     for (int i = 1; i <= n; ++i) {
7         magic[i] = magic[i - 1] * MAGIC;
8     }
9     std::vector<int> queue;
10    queue.push_back(root);
11    for (int head = 0; head < (int)queue.size(); ++head) {
12        int x = queue[head];
13        for (int i = 0; i < (int)son[x].size(); ++i) {
14            int y = son[x][i];
15            queue.push_back(y);
16        }
17    }
18    for (int index = n - 1; index >= 0; --index) {
19        int x = queue[index];
20        hash[x] = std::make_pair(0, 0);
21    }
22    std::vector<std::pair<unsigned long long, int>> value;
23    for (int i = 0; i < (int)son[x].size(); ++i) {
24        int y = son[x][i];
25        value.push_back(hash[y]);
26    }
27    std::sort(value.begin(), value.end());
28
29    hash[x].first = hash[x].first * magic[1] + 37;
30    hash[x].second++;
31    for (int i = 0; i < (int)value.size(); ++i) {
32        hash[x].first = hash[x].first * magic[value[i].second] + value[i].first;
33        hash[x].second += value[i].second;
34    }
35    hash[x].first = hash[x].first * magic[1] + 41;
36    hash[x].second++;
37 }
38 }

```

4.12 度限制生成树

```

1 int n, m, S, K, ans, cnt, Best[N], fa[N], FE[N];
2 int f[N], p[M], t[M], c[M], o, Cost[N];
3 bool u[M], d[M];

```

```

4 pair<int, int> MinCost[N];
5 struct Edge {
6     int a, b, c;
7     bool operator < (const Edge & E) const { return c < E.c; }
8 }E[M];
9 vector<int> SE;
10 inline int F(int x) {
11     return fa[x] == x ? x : fa[x] = F(fa[x]);
12 }
13 inline void AddEdge(int a, int b, int C) {
14     p[++o] = b; c[o] = C;
15     t[o] = f[a]; f[a] = o;
16 }
17 void dfs(int i, int father) {
18     fa[i] = father;
19     if (father == S) Best[i] = -1;
20     else {
21         Best[i] = i;
22         if (~Best[father] && Cost[Best[father]] > Cost[i]) Best[i] = Best[father];
23     }
24     for (int j = f[i]; j; j = t[j])
25         if (!d[j] && p[j] != father) {
26             Cost[p[j]] = c[j];
27             FE[p[j]] = j;
28             dfs(p[j], i);
29         }
30 }
31 inline bool Kruskal() {
32     cnt = n - 1, ans = 0; o = 1;
33     for (int i = 1; i <= n; i++) fa[i] = i, f[i] = 0;
34     sort(E + 1, E + m + 1);
35     for (int i = 1; i <= m; i++) {
36         if (E[i].b == S) swap(E[i].a, E[i].b);
37         if (E[i].a != S && F(E[i].a) != F(E[i].b)) {
38             fa[F(E[i].a)] = F(E[i].b);
39             ans += E[i].c;
40             cnt--;
41             u[i] = true;
42             AddEdge(E[i].a, E[i].b, E[i].c);
43             AddEdge(E[i].b, E[i].a, E[i].c);
44         }
45     }
46     for (int i = 1; i <= n; i++) MinCost[i] = make_pair(INF, INF);
47     for (int i = 1; i <= m; i++)
48         if (E[i].a == S) {
49             SE.push_back(i);
50             MinCost[F(E[i].b)] = min(MinCost[F(E[i].b)], make_pair(E[i].c, i));
51         }
52     int dif = 0;
53     for (int i = 1; i <= n; i++)
54         if (i != S && fa[i] == i) {
55             if (MinCost[i].second == INF) return false;
56             if (++dif > K) return false;
57             dfs(E[MinCost[i].second].b, S);
58             u[MinCost[i].second] = true;
59             ans += MinCost[i].first;
60         }
61     return true;
62 }
63 bool Solve() {
64     memset(d, false, sizeof d);
65     memset(u, false, sizeof u);
66     if (!Kruskal()) return false;
67     for (int i = cnt + 1; i <= K && i <= n; i++) {
68         int MinD = INF, MinID = -1;
69         for (int j = (int)SE.size() - 1; j >= 0; j--)
70             if (u[SE[j]])
71                 SE.erase(SE.begin() + j);
72         for (int j = 0; j < (int)SE.size(); j++) {
73             int tmp = E[SE[j]].c - Cost[Best[E[SE[j]].b]];
74             if (tmp < MinD) {
75                 MinD = tmp;
76                 MinID = SE[j];
77             }
78         }
79         if (MinID == -1) return true;
80         if (MinD >= 0) break;
81         ans += MinD;
82         u[MinID] = true;
83         d[FE[Best[E[MinID].b]]] = d[FE[Best[E[MinID].b]] ^ 1] = true;
84         dfs(E[MinID].b, S);
85     }
86     return true;

```

```

87 }
88 int main(){
89     Solve();
90     return 0;
91 }

```

4.13 弦图相关

4.13.1 弦图的判定

```

1  int n, m, first[1001], l, next[2000001], where[2000001], f[1001], a[1001], c[1001], L
    [1001], R[1001],
2  v[1001], idx[1001], pos[1001];
3  bool b[1001][1001];
4  inline void makelist(int x, int y){
5      where[++l] = y;
6      next[l] = first[x];
7      first[x] = l;
8  }
9  bool cmp(const int &x, const int &y){
10     return idx[x] < idx[y];
11 }
12 int main(){
13     for (;;) {
14         n = read(); m = read();
15         if (!n && !m) return 0;
16         memset(first, 0, sizeof(first)); l = 0;
17         memset(b, false, sizeof(b));
18         for (int i = 1; i <= m; i++) {
19             int x = read(), y = read();
20             if (x != y && !b[x][y]) {
21                 b[x][y] = true; b[y][x] = true;
22                 makelist(x, y); makelist(y, x);
23             }
24         }
25         memset(f, 0, sizeof(f));
26         memset(L, 0, sizeof(L));
27         memset(R, 255, sizeof(R));
28         L[0] = 1; R[0] = n;
29         for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
30         memset(idx, 0, sizeof(idx));
31         memset(v, 0, sizeof(v));
32         for (int i = n; i; --i) {
33             int now = c[i];
34             R[f[now]]--;
35             if (R[f[now]] < L[f[now]]) R[f[now]] = -1;
36             idx[now] = i; v[i] = now;
37             for (int x = first[now]; x; x = next[x])
38                 if (!idx[where[x]]) {
39                     swap(c[pos[where[x]]], c[R[f[where[x]]]]);
40                     pos[c[pos[where[x]]]] = pos[where[x]];
41                     pos[where[x]] = R[f[where[x]]];
42                     L[f[where[x]] + 1] = R[f[where[x]]]--;
43                     if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
44                     if (R[f[where[x]] + 1] == -1)
45                         R[f[where[x]] + 1] = L[f[where[x]] + 1];
46                     ++f[where[x]];
47                 }
48         }
49         bool ok = true;
50         //v是完美消除序列.
51         for (int i = 1; i <= n && ok; i++) {
52             int cnt = 0;
53             for (int x = first[v[i]]; x; x = next[x])
54                 if (idx[where[x]] > i) c[++cnt] = where[x];
55             sort(c + 1, c + cnt + 1, cmp);
56             bool can = true;
57             for (int j = 2; j <= cnt; j++)
58                 if (!b[c[1]][c[j]]) {
59                     ok = false;
60                     break;
61                 }
62         }
63         if (ok) printf("Perfect\n");
64         else printf("Imperfect\n");
65         printf("\n");
66     }
67 }

```

4.13.2 弦图的团数

```

1  int n, m, first[100001], next[2000001], where[2000001], l, L[100001], R[100001], c
    [100001], f[100001],
2  pos[100001], idx[100001], v[100001], ans;
3  inline void makelist(int x, int y){
4      where[++l] = y;
5      next[l] = first[x];
6      first[x] = l;
7  }
8  int main(){
9      memset(first, 0, sizeof(first)); l = 0;
10     n = read(); m = read();
11     for (int i = 1; i <= m; i++) {
12         int x, y;
13         x = read(); y = read();
14         makelist(x, y); makelist(y, x);
15     }
16     memset(L, 0, sizeof(L));
17     memset(R, 255, sizeof(R));
18     memset(f, 0, sizeof(f));
19     memset(idx, 0, sizeof(idx));
20     for (int i = 1; i <= n; i++) c[i] = i, pos[i] = i;
21     L[0] = 1; R[0] = n; ans = 0;
22     for (int i = n; i; --i) {
23         int now = c[i], cnt = 1;
24         idx[now] = i; v[i] = now;
25         if (R[f[now]] < L[f[now]]) R[f[now]] = -1;
26         for (int x = first[now]; x; x = next[x])
27             if (!idx[where[x]]) {
28                 swap(c[pos[where[x]]], c[R[f[where[x]]]]);
29                 pos[c[pos[where[x]]]] = pos[where[x]];
30                 pos[where[x]] = R[f[where[x]]];
31                 L[f[where[x]] + 1] = R[f[where[x]]]--;
32                 if (R[f[where[x]]] < L[f[where[x]]]) R[f[where[x]]] = -1;
33                 if (R[f[where[x]] + 1] == -1) R[f[where[x]] + 1] = L[f[where[x]] + 1];
34                 ++f[where[x]];
35             }
36         else ++cnt;
37         ans = max(ans, cnt);
38     }
39     printf("%d\n", ans);
40 }

```

4.14 最大团点数

```

1  namespace MaxClique { // 1-based
2      int g[MAXN][MAXN], len[MAXN], list[MAXN][MAXN], mc[MAXN], ans, found;
3      void DFS(int size) {
4          if (len[size] == 0) { if (size > ans) ans = size, found = true; return; }
5          for (int k = 0; k < len[size] && !found; ++k) {
6              if (size + len[size] - k <= ans) break;
7              int i = list[size][k]; if (size + mc[i] <= ans) break;
8              for (int j = k + 1, len[size + 1] = 0; j < len[size]; ++j) if (g[i][list[size][j]])
9                  list[size + 1][len[size + 1]++] = list[size][j];
10                 DFS(size + 1);
11             }
12     }
13 }
14 int work(int n) {
15     mc[n] = ans = 1;
16     for (int i = n - 1; i; --i) {
17         found = false; len[1] = 0;
18         for (int j = i + 1; j <= n; ++j) if (g[i][j]) list[1][len[1]++] = j;
19         DFS(1); mc[i] = ans;
20     }
21     return ans;
22 }

```

4.15 最大团计数

```

1  namespace MaxCliqueCounting {
2      int n, ans;
3      int ne[MAXN], ce[MAXN];
4      int g[MAXN][MAXN], list[MAXN][MAXN];
5      void dfs(int size) {
6          int i, j, k, t, cnt, best = 0;
7          bool bb;
8          if (ne[size] == ce[size]) {

```



```

9         if (ce[size] == 0)
10            ++ans;
11            return;
12    }
13    for (t = 0, i = 1; i <= ne[size]; ++i) {
14        for (cnt = 0, j = ne[size] + 1; j <= ce[size]; ++j)
15            if (!g[list[size][i]][list[size][j]])
16                ++cnt;
17            if (t == 0 || cnt < best)
18                t = i, best = cnt;
19    }
20    if (t && best <= 0)
21        return;
22    for (k = ne[size] + 1; k <= ce[size]; ++k) {
23        if (t > 0) {
24            for (i = k; i <= ce[size]; ++i)
25                if (!g[list[size][t]][list[size][i]])
26                    break;
27            swap(list[size][k], list[size][i]);
28        }
29        i = list[size][k];
30        ne[size + 1] = ce[size + 1] = 0;
31        for (j = 1; j < k; ++j)
32            if (g[i][list[size][j]])
33                list[size + 1][++ne[size + 1]] = list[size][j];
34        for (ce[size + 1] = ne[size + 1], j = k + 1; j <= ce[size]; ++j)
35            if (g[i][list[size][j]])
36                list[size + 1][++ce[size + 1]] = list[size][j];
37        dfs(size + 1);
38        ++ne[size];
39        --best;
40        for (j = k + 1, cnt = 0; j <= ce[size]; ++j)
41            if (!g[i][list[size][j]])
42                ++cnt;
43        if (t == 0 || cnt < best)
44            t = k, best = cnt;
45        if (t && best <= 0)
46            break;
47    }
48    }
49    void work () {
50        int i;
51        ne[0] = 0;
52        ce[0] = 0;
53        for (i = 1; i <= n; ++i)
54            list[0][++ce[0]] = i;
55        ans = 0;
56        dfs(0);
57    }
58 }

```

4.16 哈密顿回路 (ORE 性质的图)

ORE 性质:

$$\forall x, y \in V \wedge (x, y) \notin E \quad s.t. \quad deg_x + deg_y \geq n$$

返回结果: 从顶点 1 出发的一個哈密顿回路

使用条件: $n \geq 3$

```

1  int left[N], right[N], next[N], last[N];
2  void cover(int x) {
3      left[right[x]] = left[x];
4      right[left[x]] = right[x];
5  }
6  int adjacent(int x) {
7      for (int i = right[0]; i <= n; i = right[i]) {
8          if (graph[x][i]) {
9              return i;
10         }
11     }
12     return 0;
13 }
14 std::vector<int> solve() {
15     for (int i = 1; i <= n; ++i) {
16         left[i] = i - 1;
17         right[i] = i + 1;
18     }
19     int head, tail;
20     for (int i = 2; i <= n; ++i) {
21         if (graph[1][i]) {
22             head = 1;

```

```

23         tail = i;
24         cover(head);
25         cover(tail);
26         next[head] = tail;
27         break;
28     }
29 }
30 while (true) {
31     int x;
32     while (x = adjacent(head)) {
33         next[x] = head;
34         head = x;
35         cover(head);
36     }
37     while (x = adjacent(tail)) {
38         next[tail] = x;
39         tail = x;
40         cover(tail);
41     }
42     if (!graph[head][tail]) {
43         for (int i = head, j; i != tail; i = next[i]) {
44             if (graph[head][next[i]] && graph[tail][i]) {
45                 for (j = head; j != i; j = next[j]) {
46                     last[next[j]] = j;
47                 }
48                 j = next[head];
49                 next[head] = next[i];
50                 next[tail] = i;
51                 tail = j;
52                 for (j = i; j != head; j = last[j]) {
53                     next[j] = last[j];
54                 }
55                 break;
56             }
57         }
58     }
59     next[tail] = head;
60     if (right[0] > n) {
61         break;
62     }
63     for (int i = head; i != tail; i = next[i]) {
64         if (adjacent(i)) {
65             head = next[i];
66             tail = i;
67             next[tail] = 0;
68             break;
69         }
70     }
71 }
72 std::vector<int> answer;
73 for (int i = head; i = next[i]) {
74     if (i == 1) {
75         answer.push_back(i);
76         for (int j = next[i]; j != i; j = next[j]) {
77             answer.push_back(j);
78         }
79         answer.push_back(i);
80         break;
81     }
82     if (i == tail) {
83         break;
84     }
85 }
86 return answer;
87 }

```

第5章 字符串

5.1 模式串匹配

```

1  void build(char *pattern) {
2      int length = (int)strlen(pattern + 1);
3      fail[0] = -1;
4      for (int i = 1, j; i <= length; ++i) {
5          for (j = fail[i - 1]; j != -1 && pattern[i] != pattern[j + 1]; j = fail[j]);
6          fail[i] = j + 1;
7      }
8  }
9  void solve(char *text, char *pattern) {
10     int length = (int)strlen(text + 1);
11     for (int i = 1, j; i <= length; ++i) {
12         for (j = match[i - 1]; j != -1 && text[i] != pattern[j + 1]; j = fail[j]);

```

```

13     match[i] = j + 1;
14 }
15 }

```

5.2 坚固的模式串匹配

```

1 lenA = strlen(A); lenB = strlen(B);
2 nxt[0] = lenB, nxt[1] = lenB - 1;
3 for (int i = 0; i <= lenB; i++)
4     if (B[i] != B[i + 1]) {nxt[1] = i; break;}
5 int j, k = 1, p, L;
6 for (int i = 2; i < lenB; i++) {
7     p = k + nxt[k] - 1; L = nxt[i - k];
8     if (i + L <= p) nxt[i] = L;
9     else {
10         j = p - i + 1;
11         if (j < 0) j = 0;
12         while (i + j < lenB && B[i + j] == B[j]) j++;
13         nxt[i] = j; k = i;
14     }
15 }
16 int minlen = lenA <= lenB ? lenA : lenB; ex[0] = minlen;
17 for (int i = 0; i < minlen; i++)
18     if (A[i] != B[i]) {ex[0] = i; break;}
19 k = 0;
20 for (int i = 1; i < lenA; i++) {
21     p = k + ex[k] - 1; L = next[i - k];
22     if (i + L <= p) ex[i] = L;
23     else {
24         j = p - i + 1;
25         if (j < 0) j = 0;
26         while (i + j < lenA && j < lenB && A[i + j] == B[j]) j++;
27         ex[i] = j; k = i;
28     }
29 }

```

5.3 AC 自动机

```

1 int size, c[MAXT][26], f[MAXT], fail[MAXT], d[MAXT];
2 int alloc() {
3     size++;
4     std::fill(c[size], c[size] + 26, 0);
5     f[size] = fail[size] = d[size] = 0;
6     return size;
7 }
8 void insert(char *s) {
9     int len = strlen(s + 1), p = 1;
10    for (int i = 1; i <= len; i++) {
11        if (c[p][s[i] - 'a']) p = c[p][s[i] - 'a'];
12        else {
13            int newnode = alloc();
14            c[p][s[i] - 'a'] = newnode;
15            d[newnode] = s[i] - 'a';
16            f[newnode] = p;
17            p = newnode;
18        }
19    }
20 }
21 void buildfail() {
22     static int q[MAXT];
23     int left = 0, right = 0;
24     fail[1] = 0;
25     for (int i = 0; i < 26; i++) {
26         c[0][i] = 1;
27         if (c[1][i]) q[++right] = c[1][i];
28     }
29     while (left < right) {
30         left++;
31         int p = fail[f[q[left]]];
32         while (!c[p][d[q[left]]]) p = fail[p];
33         fail[q[left]] = c[p][d[q[left]]];
34         for (int i = 0; i < 26; i++) {
35             if (c[q[left]][i]) {
36                 q[++right] = c[q[left]][i];
37             }
38         }
39     }
40     for (int i = 1; i <= size; i++)

```

```

41     for (int j = 0; j < 26; j++) {
42         int p = i;
43         while (!c[p][j]) p = fail[p];
44         c[i][j] = c[p][j];
45     }
46 }

```

5.4 后缀数组

```

1 namespace suffix_array {
2     int wa[MAXN], wb[MAXN], ws[MAXN], wv[MAXN];
3     bool cmp(int *r, int a, int b, int l) {
4         return r[a] == r[b] && r[a + l] == r[b + l];
5     }
6     void DA(int *r, int *sa, int n, int m) {
7         int *x = wa, *y = wb, *t;
8         for (int i = 0; i < m; i++) ws[i] = 0;
9         for (int i = 0; i < n; i++) ws[x[i]] = r[i]++;
10        for (int i = 1; i < m; i++) ws[i] += ws[i - 1];
11        for (int i = n - 1; i >= 0; i--) sa[ws[x[i]]] = i;
12        for (int i, j = 1, p = 1; p < n; j <= 1, m = p) {
13            for (p = 0, i = n - j; i < n; i++) y[p++] = i;
14            for (i = 0; i < n; i++) if (sa[i] >= j) y[p++] = sa[i] - j;
15            for (i = 0; i < n; i++) wv[i] = x[y[i]];
16            for (i = 0; i < m; i++) ws[i] = 0;
17            for (i = 0; i < n; i++) ws[wv[i]]++;
18            for (i = 1; i < m; i++) ws[i] += ws[i - 1];
19            for (i = n - 1; i >= 0; i--) sa[ws[wv[i]]] = y[i];
20            for (t = x, x = y, y = t, p = 1, x[sa[0]] = 0, i = 1; i < n; i++)
21                x[sa[i]] = cmp(y, sa[i - 1], sa[i], j) ? p - 1 : p++;
22        }
23    }
24    void getheight(int *r, int *sa, int *rk, int *h, int n) {
25        for (int i = 1; i <= n; i++) rk[sa[i]] = i;
26        for (int i = 0, j, k = 0; i < n; h[rk[i++]] = k)
27            for (k ? k-- : 0, j = sa[rk[i] - 1]; r[i + k] == r[j + k]; k++);
28    }
29 }

```

5.5 广义后缀自动机

```

1 // Generalized Suffix Automaton
2 void add(int x, int &last) {
3     int lastnode = last;
4     if (c[lastnode][x]) {
5         int nownode = c[lastnode][x];
6         if (l[nownode] == l[lastnode] + 1) last = nownode;
7     }
8     else {
9         int auxnode = ++size; l[auxnode] = l[lastnode] + 1;
10        for (int i = 0; i < 26; i++) c[auxnode][i] = c[nownode][i];
11        f[auxnode] = f[nownode]; f[nownode] = auxnode;
12        for (; lastnode && c[lastnode][x] == nownode; lastnode = f[lastnode]) {
13            c[lastnode][x] = auxnode;
14        }
15        last = auxnode;
16    }
17 }
18 else {
19     int newnode = ++size; l[newnode] = l[lastnode] + 1;
20     for (; lastnode && !c[lastnode][x]; lastnode = f[lastnode]) c[lastnode][x] = newnode;
21     if (!lastnode) f[newnode] = 1;
22     else {
23         int nownode = c[lastnode][x];
24         if (l[lastnode] + 1 == l[nownode]) f[newnode] = nownode;
25         else {
26             int auxnode = ++size; l[auxnode] = l[lastnode] + 1;
27             for (int i = 0; i < 26; i++) c[auxnode][i] = c[nownode][i];
28             f[auxnode] = f[nownode]; f[nownode] = auxnode;
29             for (; lastnode && c[lastnode][x] == nownode; lastnode = f[lastnode]) {
30                 c[lastnode][x] = auxnode;
31             }
32         }
33     }
34     last = newnode;
35 }

```

5.6 Manacher 算法

```
1 void manacher(char *text, int length) {
2     palindrome[0] = 1;
3     for (int i = 1, j = 0; i < length; ++i) {
4         if (j + palindrome[j] <= i) {
5             palindrome[i] = 0;
6         } else {
7             palindrome[i] = std::min(palindrome[(j << 1) - i], j + palindrome[j] - i);
8         }
9         while (i - palindrome[i] >= 0 && i + palindrome[i] < length
10             && text[i - palindrome[i]] == text[i + palindrome[i]]) {
11             palindrome[i]++;
12         }
13         if (i + palindrome[i] > j + palindrome[j]) {
14             j = i;
15         }
16     }
17 }
```

5.7 回文树

```
1 struct Palindromic_Tree {
2     int nTree, nStr, last, c[MAXT][26], fail[MAXT], r[MAXN], l[MAXN], s[MAXN];
3     int allocate(int len) {
4         l[nTree] = len;
5         r[nTree] = 0;
6         fail[nTree] = 0;
7         memset(c[nTree], 0, sizeof(c[nTree]));
8         return nTree++;
9     }
10    void init() {
11        nTree = nStr = 0;
12        int newEven = allocate(0);
13        int newOdd = allocate(-1);
14        last = newEven;
15        fail[newEven] = newOdd;
16        fail[newOdd] = newEven;
17        s[0] = -1;
18    }
19    void add(int x) {
20        s[++nStr] = x;
21        int nownode = last;
22        while (s[nStr - 1][nownode] - 1 != s[nStr]) nownode = fail[nownode];
23        if (!c[nownode][x]) {
24            int newnode = allocate(l[nownode] + 2), &newfail = fail[newnode];
25            newfail = fail[nownode];
26            while (s[nStr - 1][newfail] - 1 != s[nStr]) newfail = fail[newfail];
27            newfail = c[newfail][x];
28            c[nownode][x] = newnode;
29        }
30        last = c[nownode][x];
31        r[last]++;
32    }
33    void count() {
34        for (int i = nTree - 1; i >= 0; i--) {
35            r[fail[i]] += r[i];
36        }
37    }
38 }
```

5.8 循环串最小表示

```
1 int solve(char *text, int length) {
2     int i = 0, j = 1, delta = 0;
3     while (i < length && j < length && delta < length) {
4         char tokeni = text[(i + delta) % length];
5         char tokenj = text[(j + delta) % length];
6         if (tokeni == tokenj) {
7             delta++;
8         } else {
9             if (tokeni > tokenj) {
10                i += delta + 1;
11            } else {
12                j += delta + 1;
13            }
14            if (i == j) {
15                j++;
16            }
17        }
18    }
19 }
```

```
16         }
17         delta = 0;
18     }
19     }
20     return std::min(i, j);
21 }
```

第6章 计算几何

6.1 二维基础

6.1.1 凸包

```
1 bool Pair_Comp(const Point &a, const Point &b) {
2     if (dcmp(a.x - b.x) < 0) return true;
3     if (dcmp(a.x - b.x) > 0) return false;
4     return dcmp(a.y - b.y) < 0;
5 }
6 int Convex_Hull(int n, Point *P, Point *C) {
7     sort(P, P + n, Pair_Comp);
8     int top = 0;
9     for (int i = 0; i < n; i++) {
10        while (top >= 2 && dcmp(det(C[top - 1] - C[top - 2], P[i] - C[top - 2])) <= 0) top--;
11        C[top++] = P[i];
12    }
13    int lasttop = top;
14    for (int i = n - 1; i >= 0; i--) {
15        while (top > lasttop && dcmp(det(C[top - 1] - C[top - 2], P[i] - C[top - 2])) <= 0)
16            top--;
17        C[top++] = P[i];
18    }
19    return top;
20 }
```

6.1.2 半平面交

```
1 bool isOnLeft(const Point &x, const Line &l) {
2     double d = det(x - l.a, l.b - l.a);
3     return dcmp(d) <= 0;
4 }
5 // 传入一个线段的集合L, 传出A, 并且返回A的大小
6 int getIntersectionOfHalfPlane(int n, Line *L, Line *A) {
7     Line *q = new Line[n + 1];
8     Point *p = new Point[n + 1];
9     sort(L, L + n, Polar_Angle_Comp_Line);
10    int l = 1, r = 0;
11    for (int i = 0; i < n; i++) {
12        while (l < r && !isOnLeft(p[r - 1], L[i])) r--;
13        while (l < r && !isOnLeft(p[l], L[i])) l++;
14        q[++r] = L[i];
15        if (l < r && is_Colinear(q[r], q[r - 1])) {
16            r--;
17            if (isOnLeft(L[i].a, q[r])) q[r] = L[i];
18        }
19        if (l < r) p[r - 1] = getIntersection(q[r - 1], q[r]);
20    }
21    while (l < r && !isOnLeft(p[r - 1], q[l])) r--;
22    if (r - l + 1 <= 2) return 0;
23    int tot = 0;
24    for (int i = 1; i <= r; i++) A[tot++] = q[i];
25    return tot;
26 }
```

6.1.3 最近点对

```
1 bool comparex(const Point &a, const Point &b) {
2     return sgn(a.x - b.x) < 0;
3 }
4 bool comparey(const Point &a, const Point &b) {
5     return sgn(a.y - b.y) < 0;
6 }
7 double solve(const std::vector<Point> &point, int left, int right) {
8     if (left == right) {
9         return INF;
10    }
11    if (left + 1 == right) {
12        return dist(point[left], point[right]);
13    }
14 }
```

```

13     }
14     int mid = left + right >> 1;
15     double result = std::min(solve(left, mid), solve(mid + 1, right));
16     std::vector<Point> candidate;
17     for (int i = left; i <= right; ++i) {
18         if (std::abs(point[i].x - point[mid].x) <= result) {
19             candidate.push_back(point[i]);
20         }
21     }
22     std::sort(candidate.begin(), candidate.end(), comparey);
23     for (int i = 0; i < (int)candidate.size(); ++i) {
24         for (int j = i + 1; j < (int)candidate.size(); ++j) {
25             if (std::abs(candidate[i].y - candidate[j].y) >= result) {
26                 break;
27             }
28             result = std::min(result, dist(candidate[i], candidate[j]));
29         }
30     }
31     return result;
32 }
33 double solve(std::vector<Point> point) {
34     std::sort(point.begin(), point.end(), comparex);
35     return solve(point, 0, (int)point.size() - 1);
36 }

```

6.1.4 三角形的心

```

1 Point incenter(const Point &a, const Point &b, const Point &c) {
2     double p = (a - b).length() + (b - c).length() + (c - a).length();
3     return (a * (b - c).length() + b * (c - a).length() + c * (a - b).length()) / p;
4 }
5 Point circumcenter(const Point &a, const Point &b, const Point &c) {
6     Point p = b - a, q = c - a, s(dot(p, p) / 2, dot(q, q) / 2);
7     double d = det(p, q);
8     return a + Point(det(s, Point(p.y, q.y)), det(Point(p.x, q.x), s)) / d;
9 }
10 Point circumcenter(const Point &a, const Point &b, const Point &c) {
11     Point p = b - a, q = c - a, s(dot(p, p) / 2, dot(q, q) / 2);
12     double d = det(p, q);
13     return a + Point(det(s, Point(p.y, q.y)), det(Point(p.x, q.x), s)) / d;
14 }

```

6.2 三维基础

6.2.1 凸包

```

1 struct Triangle{
2     TPoint a, b, c;
3     Triangle() {}
4     Triangle(TPoint a, TPoint b, TPoint c) : a(a), b(b), c(c) {}
5     double getArea() {
6         TPoint ret = det(b - a, c - a);
7         return dist(ret) / 2.0;
8     }
9 };
10 namespace Convex_Hull {
11     struct Face{
12         int a, b, c;
13         bool isOnConvex;
14         Face() {}
15         Face(int a, int b, int c) : a(a), b(b), c(c) {}
16     };
17     int nFace, left, right, whe[MAXN][MAXN];
18     Face queue[MAXF], tmp[MAXF];
19     bool isVisible(const std::vector<TPoint> &p, const Face &f, const TPoint &a) {
20         return dcmp(detdot(p[f.a], p[f.b], p[f.c], a)) > 0;
21     }
22     bool init(std::vector<TPoint> &p) {
23         bool check = false;
24         for (int i = 1; i < (int)p.size(); ++i) {
25             if (dcmp(sqrdist(p[0], p[i])) > 0) {
26                 std::swap(p[1], p[i]);
27                 check = true;
28                 break;
29             }
30         }
31         if (!check) return false;
32         check = false;
33         for (int i = 2; i < (int)p.size(); ++i) {

```

```

34             if (dcmp(sqrdist(det(p[i] - p[0], p[1] - p[0]))) > 0) {
35                 std::swap(p[2], p[i]);
36                 check = true;
37                 break;
38             }
39         }
40         if (!check) return false;
41         check = false;
42         for (int i = 3; i < (int)p.size(); ++i) {
43             if (dcmp(detdot(p[0], p[1], p[2], p[i])) > 0) {
44                 std::swap(p[3], p[i]);
45                 check = true;
46                 break;
47             }
48         }
49         if (!check) return false;
50         for (int i = 0; i < (int)p.size(); ++i)
51             for (int j = 0; j < (int)p.size(); ++j) {
52                 whe[i][j] = -1;
53             }
54         return true;
55     }
56     void pushface(const int &a, const int &b, const int &c) {
57         nFace++;
58         tmp[nFace] = Face(a, b, c);
59         tmp[nFace].isOnConvex = true;
60         whe[a][b] = nFace;
61         whe[b][c] = nFace;
62         whe[c][a] = nFace;
63     }
64     bool deal(const std::vector<TPoint> &p, const std::pair<int, int> &now, const TPoint &
        base) {
65         int id = whe[now.second][now.first];
66         if (!tmp[id].isOnConvex) return true;
67         if (isVisible(p, tmp[id], base)) {
68             queue[++right] = tmp[id];
69             tmp[id].isOnConvex = false;
70             return true;
71         }
72         return false;
73     }
74     std::vector<Triangle> getConvex(std::vector<TPoint> &p) {
75         static std::vector<Triangle> ret;
76         ret.clear();
77         if (!init(p)) return ret;
78         if (!isVisible(p, Face(0, 1, 2), p[3])) pushface(0, 1, 2); else pushface(0, 2, 1);
79         if (!isVisible(p, Face(0, 1, 3), p[2])) pushface(0, 1, 3); else pushface(0, 3, 1);
80         if (!isVisible(p, Face(0, 2, 3), p[1])) pushface(0, 2, 3); else pushface(0, 3, 2);
81         if (!isVisible(p, Face(1, 2, 3), p[0])) pushface(1, 2, 3); else pushface(1, 3, 2);
82         for (int a = 4; a < (int)p.size(); ++a) {
83             TPoint base = p[a];
84             for (int i = 1; i <= nFace; ++i) {
85                 if (tmp[i].isOnConvex && isVisible(p, tmp[i], base)) {
86                     left = 0, right = 0;
87                     queue[++right] = tmp[i];
88                     tmp[i].isOnConvex = false;
89                     while (left < right) {
90                         Face now = queue[++left];
91                         if (!deal(p, std::make_pair(now.a, now.b), base)) pushface(now.a, now.b, a);
92                         if (!deal(p, std::make_pair(now.b, now.c), base)) pushface(now.b, now.c, a);
93                         if (!deal(p, std::make_pair(now.c, now.a), base)) pushface(now.c, now.a, a);
94                     }
95                     break;
96                 }
97             }
98         }
99         for (int i = 1; i <= nFace; ++i) {
100             Face now = tmp[i];
101             if (now.isOnConvex) {
102                 ret.push_back(Triangle(p[now.a], p[now.b], p[now.c]));
103             }
104         }
105         return ret;
106     }
107 };
108 int n;
109 std::vector<TPoint> p;
110 std::vector<Triangle> answer;
111 int main() {
112     scanf("%d", &n);
113     for (int i = 1; i <= n; ++i) {
114         TPoint a;
115         a.read();

```

```

116     p.push_back(a);
117 }
118 answer = Convex_Hull::getConvex(p);
119 double areaCounter = 0.0;
120 for (int i = 0; i < (int)answer.size(); i++) {
121     areaCounter += answer[i].getArea();
122 }
123 printf("%.3f\n", areaCounter);
124 return 0;
125 }

```

6.2.2 绕轴旋转

使用方法及注意事项：逆时针绕轴 AB 旋转 θ 角

```

1 Matrix getTrans(const double &a, const double &b, const double &c) {
2     Matrix ret;
3     ret.a[0][0] = 1; ret.a[0][1] = 0; ret.a[0][2] = 0; ret.a[0][3] = 0;
4     ret.a[1][0] = 0; ret.a[1][1] = 1; ret.a[1][2] = 0; ret.a[1][3] = 0;
5     ret.a[2][0] = 0; ret.a[2][1] = 0; ret.a[2][2] = 1; ret.a[2][3] = 0;
6     ret.a[3][0] = a; ret.a[3][1] = b; ret.a[3][2] = c; ret.a[3][3] = 1;
7     return ret;
8 }
9 Matrix getRotate(const double &a, const double &b, const double &c, const double &theta) {
10     Matrix ret;
11     ret.a[0][0] = a * a * (1 - cos(theta)) + cos(theta);
12     ret.a[0][1] = a * b * (1 - cos(theta)) + c * sin(theta);
13     ret.a[0][2] = a * c * (1 - cos(theta)) - b * sin(theta);
14     ret.a[0][3] = 0;
15     ret.a[1][0] = b * a * (1 - cos(theta)) - c * sin(theta);
16     ret.a[1][1] = b * b * (1 - cos(theta)) + cos(theta);
17     ret.a[1][2] = b * c * (1 - cos(theta)) + a * sin(theta);
18     ret.a[1][3] = 0;
19     ret.a[2][0] = c * a * (1 - cos(theta)) + b * sin(theta);
20     ret.a[2][1] = c * b * (1 - cos(theta)) - a * sin(theta);
21     ret.a[2][2] = c * c * (1 - cos(theta)) + cos(theta);
22     ret.a[2][3] = 0;
23     ret.a[3][0] = 0;
24     ret.a[3][1] = 0;
25     ret.a[3][2] = 0;
26     ret.a[3][3] = 1;
27     return ret;
28 }
29 Matrix getRotate(const double &ax, const double &ay, const double &az, const double &bx,
30                 const double &by, const double &bz, const double &theta) {
31     double l = dist(Point(0, 0, 0), Point(bx, by, bz));
32     Matrix ret = getTrans(-ax, -ay, -az);
33     ret = ret * getRotate(bx / l, by / l, bz / l, theta);
34     ret = ret * getTrans(ax, ay, az);
35     return ret;
36 }

```

6.3 多边形

6.3.1 判断点在多边形内部

```

1 bool point_on_line(const Point &p, const Point &a, const Point &b) {
2     return sgn(det(p, a, b)) == 0 && sgn(dot(p, a, b)) <= 0;
3 }
4 bool point_in_polygon(const Point &p, const std::vector<Point> &polygon) {
5     int counter = 0;
6     for (int i = 0; i < (int)polygon.size(); ++i) {
7         Point a = polygon[i], b = polygon[(i + 1) % (int)polygon.size()];
8         if (point_on_line(p, a, b)) {
9             // Point on the boundary are excluded.
10            return false;
11        }
12        int x = sgn(det(a, p, b));
13        int y = sgn(a.y - p.y);
14        int z = sgn(b.y - p.y);
15        counter += (x > 0 && y <= 0 && z > 0);
16        counter -= (x < 0 && z <= 0 && y > 0);
17    }
18    return counter;
19 }

```

6.3.2 多边形内整点计数

```

1 int getInside(int n, Point *P) { // 求多边形P内有多少个整数点
2     int OnEdge = n;
3     double area = getArea(n, P);
4     for (int i = 0; i < n - 1; i++) {
5         Point now = P[i + 1] - P[i];
6         int y = (int)now.y, x = (int)now.x;
7         OnEdge += abs(gcd(x, y)) - 1;
8     }
9     Point now = P[0] - P[n - 1];
10    int y = (int)now.y, x = (int)now.x;
11    OnEdge += abs(gcd(x, y)) - 1;
12    double ret = area - (double)OnEdge / 2 + 1;
13    return (int)ret;
14 }

```

6.4 圆

6.4.1 最小覆盖圆

```

1 Point getmid(Point a, Point b) {
2     return Point((a.x + b.x) / 2, (a.y + b.y) / 2);
3 }
4 Point getcross(Point a, Point vA, Point b, Point vB) {
5     Point u = a - b;
6     double t = det(vB, u) / det(vA, vB);
7     return a + vA * t;
8 }
9 Point getcir(Point a, Point b, Point c) {
10    Point midA = getmid(a, b), vA = Point(-(b - a).y, (b - a).x);
11    Point midB = getmid(b, c), vB = Point(-(c - b).y, (c - b).x);
12    return getcross(midA, vA, midB, vB);
13 }
14 double mincir(Point *p, int n) {
15    std::random_shuffle(p + 1, p + n + 1);
16    Point O = p[1];
17    double r = 0;
18    for (int i = 2; i <= n; i++) {
19        if (dist(O, p[i]) <= r) continue;
20        O = p[i]; r = 0;
21        for (int j = 1; j < i; j++) {
22            if (dist(O, p[j]) <= r) continue;
23            O = getmid(p[i], p[j]); r = dist(O, p[i]);
24            for (int k = 1; k < j; k++) {
25                if (dist(O, p[k]) <= r) continue;
26                O = getcir(p[i], p[j], p[k]);
27                r = dist(O, p[i]);
28            }
29        }
30    }
31    return r;
32 }

```

6.4.2 最小覆盖球

```

1 double eps(1e-8);
2 int sign(const double &x) {
3     return (x > eps) - (x + eps < 0);
4 }
5 bool equal(const double &x, const double &y) {
6     return x + eps > y and y + eps > x;
7 }
8 struct Point {
9     double x, y, z;
10    Point() {}
11    Point(const double &x, const double &y, const double &z) : x(x), y(y), z(z) {}
12 }
13 void scan() {
14     scanf("%lf%lf%lf", &x, &y, &z);
15 }
16 double sqrlen() const {
17     return x * x + y * y + z * z;
18 }
19 double len() const {
20     return sqrt(sqrlen());
21 }
22 }

```

```

23 void print() const {
24     printf("%lf %lf %lf\n", x, y, z);
25 }
26 } a[33];
27 Point operator + (const Point & a, const Point & b) {
28     return Point(a.x + b.x, a.y + b.y, a.z + b.z);
29 }
30 Point operator - (const Point & a, const Point & b) {
31     return Point(a.x - b.x, a.y - b.y, a.z - b.z);
32 }
33 Point operator * (const double & x, const Point & a) {
34     return Point(x * a.x, x * a.y, x * a.z);
35 }
36 double operator % (const Point & a, const Point & b) {
37     return a.x * b.x + a.y * b.y + a.z * b.z;
38 }
39 Point operator * (const Point & a, const Point & b) {
40     return Point(a.y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x * b.y - a.y * b.x);
41 }
42 struct Circle {
43     double r;
44     Point o;
45     Circle() {
46         o.x = o.y = o.z = r = 0;
47     }
48     Circle(const Point & o, const double & r) : o(o), r(r) {
49     }
50     void scan() {
51         o.scan();
52         scanf("%lf", &r);
53     }
54     void print() const {
55         o.print();
56         printf("%lf\n", r);
57     }
58 };
59 struct Plane {
60     Point nor;
61     double m;
62     Plane(const Point & nor, const Point & a) : nor(nor) {
63         m = nor % a;
64     }
65 };
66 Point intersect(const Plane & a, const Plane & b, const Plane & c) {
67     Point c1(a.nor.x, b.nor.x, c.nor.x), c2(a.nor.y, b.nor.y, c.nor.y), c3(a.nor.z, b.nor.z,
68         c.nor.z), c4(a.m, b.m, c.m);
69     return 1 / ((c1 * c2) % c3) * Point((c4 * c2) % c3, (c1 * c4) % c3, (c1 * c2) % c4);
70 }
71 bool in(const Point & a, const Circle & b) {
72     return sign((a - b.o).len() - b.r) <= 0;
73 }
74 bool operator < (const Point & a, const Point & b) {
75     if(!equal(a.x, b.x)) {
76         return a.x < b.x;
77     }
78     if(!equal(a.y, b.y)) {
79         return a.y < b.y;
80     }
81     if(!equal(a.z, b.z)) {
82         return a.z < b.z;
83     }
84     return false;
85 }
86 bool operator == (const Point & a, const Point & b) {
87     return equal(a.x, b.x) and equal(a.y, b.y) and equal(a.z, b.z);
88 }
89 vector<Point> vec;
90 Circle calc() {
91     if(vec.empty()) {
92         return Circle(Point(0, 0, 0), 0);
93     } else if(1 == (int)vec.size()) {
94         return Circle(vec[0], 0);
95     } else if(2 == (int)vec.size()) {
96         return Circle(0.5 * (vec[0] + vec[1]), 0.5 * (vec[0] - vec[1]).len());
97     } else if(3 == (int)vec.size()) {
98         double r((vec[0] - vec[1]).len() * (vec[1] - vec[2]).len() * (vec[2] - vec[0]).len() /
99             2 / fabs((vec[0] - vec[2]) * (vec[1] - vec[2]).len()));
100         return Circle(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])),
101             Plane(vec[2] - vec[1], 0.5 * (vec[2] + vec[1])),
102             Plane((vec[1] - vec[0]) * (vec[2] - vec[0]), vec[0])), r);
103     } else {
104         Point o(intersect(Plane(vec[1] - vec[0], 0.5 * (vec[1] + vec[0])),
105             Plane(vec[2] - vec[0], 0.5 * (vec[2] + vec[0])),

```

```

104         Plane(vec[3] - vec[0], 0.5 * (vec[3] + vec[0]))));
105         return Circle(o, (o - vec[0]).len());
106     }
107 }
108 Circle miniBall(int n) {
109     Circle res(calc());
110     for(int i(0); i < n; i++) {
111         if(!in(a[i], res)) {
112             vec.push_back(a[i]);
113             res = miniBall(i);
114             vec.pop_back();
115             if(i) {
116                 Point tmp(a[i]);
117                 memmove(a + 1, a, sizeof(Point) * i);
118                 a[0] = tmp;
119             }
120         }
121     }
122     return res;
123 }
124 int main() {
125     int n;
126     for(;;) {
127         scanf("%d", &n);
128         if(!n) {
129             break;
130         }
131         for(int i(0); i < n; i++) {
132             a[i].scan();
133         }
134         sort(a, a + n);
135         n = unique(a, a + n) - a;
136         vec.clear();
137         printf("%.10f\n", miniBall(n).r);
138     }

```

6.4.3 多边形与圆的交面积

```

1 // 求扇形面积
2 double getSectorArea(const Point &a, const Point &b, const double &r) {
3     double c = (2.0 * r * r - sqrdist(a, b)) / (2.0 * r * r);
4     double alpha = acos(c);
5     return r * r * alpha / 2.0;
6 }
7 // 求二次方程  $ax^2 + bx + c = 0$  的解
8 std::pair<double, double> getSolution(const double &a, const double &b, const double &c) {
9     double delta = b * b - 4.0 * a * c;
10    if (dcmp(delta) < 0) return std::make_pair(0, 0);
11    else return std::make_pair((-b - sqrt(delta)) / (2.0 * a), (-b + sqrt(delta)) / (2.0 * a));
12 }
13 // 直线与圆的交点
14 std::pair<Point, Point> getIntersection(const Point &a, const Point &b, const double &r) {
15     Point d = b - a;
16     double A = dot(d, d);
17     double B = 2.0 * dot(d, a);
18     double C = dot(a, a) - r * r;
19     std::pair<double, double> s = getSolution(A, B, C);
20     return std::make_pair(a + d * s.first, a + d * s.second);
21 }
22 // 原点到线段AB的距离
23 double getPointDist(const Point &a, const Point &b) {
24     Point d = b - a;
25     int sA = dcmp(dot(a, d)), sB = dcmp(dot(b, d));
26     if (sA * sB <= 0) return det(a, b) / dist(a, b);
27     else return std::min(dist(a), dist(b));
28 }
29 // a和b和原点组成的三角形与半径为r的圆的交的面积
30 double getArea(const Point &a, const Point &b, const double &r) {
31     double dA = dot(a, a), dB = dot(b, b), dC = getPointDist(a, b), ans = 0.0;
32     if (dcmp(dA - r * r) <= 0 && dcmp(dB - r * r) <= 0) return det(a, b) / 2.0;
33     Point tA = a / dist(a) * r;
34     Point tB = b / dist(b) * r;
35     if (dcmp(dC - r) > 0) return getSectorArea(tA, tB, r);
36     std::pair<Point, Point> ret = getIntersection(a, b, r);
37     if (dcmp(dA - r * r) > 0 && dcmp(dB - r * r) > 0) {
38         ans += getSectorArea(tA, ret.first, r);
39         ans += det(ret.first, ret.second) / 2.0;
40         ans += getSectorArea(ret.second, tB, r);
41     }

```

```

42 }
43 if (dcmp(dA - r * r) > 0) return det(ret.first, b) / 2.0 + getSectorArea(tA, ret.first,
    r);
44 else return det(a, ret.second) / 2.0 + getSectorArea(ret.second, tB, r);
45 }
46 // 求圆与多边形的交的主过程
47 double getArea(int n, Point *p, const Point &c, const double r) {
48     double ret = 0.0;
49     for (int i = 0; i < n; i++) {
50         int sgn = dcmp(det(p[i] - c, p[(i + 1) % n] - c));
51         if (sgn > 0) ret += getArea(p[i] - c, p[(i + 1) % n] - c, r);
52         else ret -= getArea(p[(i + 1) % n] - c, p[i] - c, r);
53     }
54     return fabs(ret);
55 }

```

第7章 其它

7.1 STL 使用方法

7.1.1 nth_element

用法: `nth_element(a + 1, a + id, a + n + 1);`

作用: 将排名为 `id` 的元素放在第 `id` 个位置。

7.1.2 next_permutation

用法: `next_permutation(a + 1, a + n + 1);`

作用: 以 `a` 中从小到大排序后为第一个排列, 求得当期数组 `a` 中的下一个排列, 返回值为当期排列是否为最后一个排列。

7.2 博弈论相关

7.2.1 巴什博弈

- 只有一堆 n 个物品, 两个人轮流从这堆物品中取物, 规定每次至少取一个, 最多取 m 个。最后取光者得胜。
- 显然, 如果 $n = m + 1$, 那么由于一次最多只能取 m 个, 所以, 无论先取者拿走多少个, 后取者都能够一次拿走剩余的物品, 后者取胜。因此我们发现了如何取胜的法则: 如果 $n = m + 1 + r + s$, (r 为任意自然数, $s \leq m$), 那么先取者要拿走 s 个物品, 如果后取者拿走 k ($k \leq m$) 个, 那么先取者再拿走 $m + 1 - k$ 个, 结果剩下 $(m + 1)(r + 1)$ 个, 以后保持这样的取法, 那么先取者肯定获胜。总之, 要保持给对手留下 $(m + 1)$ 的倍数, 就能最后获胜。

7.2.2 威佐夫博弈

- 有两堆各若干个物品, 两个人轮流从某一堆或同时从两堆中取同样多的物品, 规定每次至少取一个, 多者不限, 最后取光者得胜。
- 判断一个局势 (a, b) 为奇异局势 (必败态) 的方法:

$$a_k = [k(1 + \sqrt{5})/2] \quad b_k = a_k + k$$

7.2.3 阶梯博弈

- 博弈在一列阶梯上进行, 每个阶梯上放着自然数个点, 两个人进行阶梯博弈, 每一步则是将一个阶梯上的若干个 (至少一个) 移到前面去, 最后没有点可以移动的人输。
- 解决方法: 把所有奇数阶梯看成 N 堆石子, 做 NIM。(把石子从奇数堆移动到偶数堆可以理解为拿走石子, 就相当于几个奇数堆的石子在做 Nim)

7.2.4 图上删边游戏

链的删边游戏

- 游戏规则: 对于一条链, 其中一个端点是根, 两人轮流删边, 脱离根的部分也算被删去, 最后没边可删的人输。
- 做法: $sg[i] = n - dist(i) - 1$ (其中 n 表示总点数, $dist(i)$ 表示离根的距离)

树的删边游戏

- 游戏规则: 对于一棵有根树, 两人轮流删边, 脱离根的部分也算被删去, 没边可删的人输。
- 做法: 叶子结点的 $sg = 0$, 其他节点的 sg 等于儿子结点的 $sg + 1$ 的异或和。

局部连通图的删边游戏

- 游戏规则: 在一个局部连通图上, 两人轮流删边, 脱离根的部分也算被删去, 没边可删的人输。局部连通图的构图规则是, 在一棵基础树上加边得到, 所有形成的环保证不共用边, 且只与基础树有一个公共点。
- 做法: 去掉所有的偶环, 将所有的奇环变为长度为 1 的链, 然后做树的删边游戏。

7.3 Java Reference

```

1 import java.io.*;
2 import java.util.*;
3 import java.math.*;
4 public class Main {
5     static int get(char c) {
6         if (c <= '9')
7             return c - '0';
8         else if (c <= 'Z')
9             return c - 'A' + 10;
10        else
11            return c - 'a' + 36;
12    }
13    static char get(int x) {
14        if (x <= 9)
15            return (char)(x + '0');
16        else if (x <= 35)
17            return (char)(x - 10 + 'A');
18        else
19            return (char)(x - 36 + 'a');
20    }
21    static BigInteger get(String s, BigInteger x) {
22        BigInteger ans = BigInteger.valueOf(0), now = BigInteger.valueOf(1);
23        for (int i = s.length() - 1; i >= 0; i--) {
24            ans = ans.add(now.multiply(BigInteger.valueOf(get(s.charAt(i)))));
25            now = now.multiply(x);
26        }
27        return ans;
28    }
29    public static void main(String[] args) {
30        Scanner cin = new Scanner(new BufferedInputStream(System.in));
31        for (; ) {
32            BigInteger x = cin.nextBigInteger();
33            if (x.compareTo(BigInteger.valueOf(0)) == 0)
34                break;
35            String s = cin.next(), t = cin.next(), r = "";
36            BigInteger ans = get(s, x).mod(get(t, x));
37            if (ans.compareTo(BigInteger.valueOf(0)) == 0)
38                r = "0";
39            for (; ans.compareTo(BigInteger.valueOf(0)) > 0; ) {
40                r = get(ans.mod(x).intValue() + r);
41                ans = ans.divide(x);
42            }
43            System.out.println(r);
44        }
45    }
46    // Arrays
47    int a[];
48    .fill(a[, int fromIndex, int toIndex],val); | .sort(a[, int fromIndex, int toIndex])
49    // String
50    String s;
51    .charAt(int i); | compareTo(String) | compareToIgnoreCase () | contains(String) |
52    length () | substring(int l, int len)
53    // BigInteger
54    .abs() | .add() | bitLength () | subtract () | divide () | remainder () |
55    divideAndRemainder () | modPow(b, c) |
56    pow(int) | multiply () | compareTo () |
57    gcd() | intValue () | longValue () | isProbablePrime(int c) (1 - 1/2^c) |
58    nextProbablePrime () | shiftLeft(int) | valueOf ()
59    // BigDecimal
60    .ROUND_CEILING | ROUND_DOWN_FLOOR | ROUND_HALF_DOWN | ROUND_HALF_EVEN | ROUND_HALF_UP |
    ROUND_UP
61    .divide(BigDecimal b, int scale , int round_mode) | doubleValue () | movePointLeft(int) |
    pow(int) |
62    setScale(int scale , int round_mode) | stripTrailingZeros ()
63    // StringBuilder
64    StringBuilder sb = new StringBuilder ();
65    sb.append(elem) | out.println(sb)

```

7.4 Bug List

- 题意有毒时, 要耐心仔细读题
- 对于模拟题, 要注意可能会出现细节 case
- 当做到题号为 G 的题目时, be careful
- 对于题意/算法发生变动修改代码, 必须小心谨慎
- 可能多解时看清输出哪一个解
- 注意 Case 格式在一场中可能不同
- 看样例解释

- 手滑：循环的终止条件/nmij 打混/函数重载默认参数/多层数组嵌套/数据范围/复制的代码/struct 成员初始化
- 要 define 的常见名：left,right,next,hash,log
- 对 bitset 的常数认识不够。
- 对于诶过常数还 TLE 的题目，没有注意到是做法不够优越。
- 对于讨论题目，陷入打补丁的死回圈。
- 没有注意到不合理的数据范围而导致得出错误的算法
- 当意识到程序的逻辑问题时（比如大小于号打反），注意其他位置是不是也犯了类似的错误（也打反了）。
- 分块的大小要考虑常数谨慎估计。
- 网络流的数组开成 V 不要开成 N
- 看机时空的时候要冷静，否则容易导致不优的写法算法上位。
- 欧拉路注意判断连通性。
- 常识缺乏，一个空的 vector 空间约为 10 个 int。
- 写 splay/LCT 时，当需要自顶向下访问时（如求前驱/后继/K 值）忘记一边走一边 relax 标记。
- 忘记了变量已经修改，试图访问其原始值。
- Farmland 为了偷懒每次直接暴力找后继，挂在 star 上。
- Farmland 为了判断挖掘完毕的条件要使用边而不是点。
- 几何旋转（比如为了使得 x distinct）之前先看清楚题目到底限定了哪些点无重点
- 几何整数旋转时估计好数值范围。
- 判断直线与圆交点，当两点都在圆外时不一定无交点。
- 注意题目 N,M 的读入顺序
- 注意题目条件可能会隐藏在样例解释之中
- 大代码查错时先检查下标，取模等错误
- 注意检查同种错误重复发生
- 要考虑输入数据或许不合理要是程序有足够的容错性
- 弄混了局部变量和全局变量这个写程序的时候要注意，局部变量和全局变量不要重名
- 爆 long long 处理比较大的数的时候要注意
- 树 hash 这个以后不会再写错了
- MLE 交代码之前要算一算内存
- 板子错要验板子
- 上界设小上界要小心确定
- 没有必要的操作导致 TLE 写题之前要想一想是否有无谓的操作
- 数组下标越界写代码的时候要集中注意力
- 我没法一开始就给出完整的算法，或许要边写边改进我尽量想清楚再说算法
- 函数没有返回值这个犯了一次以后应该都会注意到的
- 没有想到题目会卡 SPFA，以为算法错这个以后也不会再犯
- 没有大胆写暴力有队过了，看上去又没有别的方法，就可以试一试
- 没有考虑重边和自环这个以后不会再犯
- 题目读错题应该一边读一边划，特别要注意转述题意很容易出错。每个人写题之前必须读题目的输入格式和输出格式

第8章 数学公式

8.1 常用数学公式

8.1.1 求和公式

- $\sum_{k=1}^n (2k-1)^2 = \frac{n(4n^2-1)}{3}$
- $\sum_{k=1}^n k^3 = [\frac{n(n+1)}{2}]^2$
- $\sum_{k=1}^n (2k-1)^3 = n^2(2n^2-1)$
- $\sum_{k=1}^n k^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$
- $\sum_{k=1}^n k^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12}$
- $\sum_{k=1}^n k(k+1) = \frac{n(n+1)(n+2)}{3}$
- $\sum_{k=1}^n k(k+1)(k+2) = \frac{n(n+1)(n+2)(n+3)}{4}$
- $\sum_{k=1}^n k(k+1)(k+2)(k+3) = \frac{n(n+1)(n+2)(n+3)(n+4)}{5}$

8.1.2 斐波那契数列

- $fib_0 = 0, fib_1 = 1, fib_n = fib_{n-1} + fib_{n-2}$
- $fib_{n+2} \cdot fib_n - fib_{n+1}^2 = (-1)^{n+1}$
- $fib_{-n} = (-1)^{n-1} fib_n$
- $fib_{n+k} = fib_k \cdot fib_{n+1} + fib_{k-1} \cdot fib_n$
- $gcd(fib_m, fib_n) = fib_{gcd(m,n)}$
- $fib_m | fib_n^2 \Leftrightarrow n | fib_m$

8.1.3 错排公式

- $D_n = (n-1)(D_{n-2} + D_{n-1})$
- $D_n = n! \cdot (1 - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \dots + \frac{(-1)^n}{n!})$

8.1.4 莫比乌斯函数

$$\mu(n) = \begin{cases} 1 & \text{若 } n = 1 \\ (-1)^k & \text{若 } n \text{ 无平方数因子, 且 } n = p_1 p_2 \dots p_k \\ 0 & \text{若 } n \text{ 有大于 } 1 \text{ 的平方数因数} \end{cases}$$

$$\sum_{d|n} \mu(d) = \begin{cases} 1 & \text{若 } n = 1 \\ 0 & \text{其他情况} \end{cases}$$

$$g(n) = \sum_{d|n} f(d) \Leftrightarrow f(n) = \sum_{d|n} \mu(d) g(\frac{n}{d})$$

$$g(x) = \sum_{n=1}^{[x]} f(\frac{x}{n}) \Leftrightarrow f(x) = \sum_{n=1}^{[x]} \mu(n) g(\frac{x}{n})$$

8.1.5 Burnside 引理

设 G 是一个有限群，作用在集合 X 上。对每个 g 属于 G ，令 X^g 表示 X 中在 g 作用下的不动元素，轨道数（记作 $|X/G|$ ）由如下公式给出： $|X/G| = \frac{1}{|G|} \sum_{g \in G} |X^g|$ 。

8.1.6 五边形数定理

设 $p(n)$ 是 n 的拆分数，有 $p(n) = \sum_{k \in \mathbb{Z} \setminus \{0\}} (-1)^{k-1} p\left(n - \frac{k(3k-1)}{2}\right)$

8.1.7 树的计数

- 有根树计数： $n+1$ 个结点的有根树的个数为 $a_{n+1} = \frac{\sum_{j=1}^n j \cdot a_j \cdot S_{n,j}}{n}$ 其中， $S_{n,j} = \sum_{i=1}^{n/j} a_{n+1-i,j} = S_{n-j,j} + a_{n+1-j}$
- 无根树计数：当 n 为奇数时， n 个结点的无根树的个数为 $a_n - \sum_{i=1}^{n/2} a_i a_{n-i}$ 当 n 为偶数时， n 个结点的无根树的个数为 $a_n - \sum_{i=1}^{n/2} a_i a_{n-i} + \frac{1}{2} a_{\frac{n}{2}} (a_{\frac{n}{2}} + 1)$
- n 个结点的完全图的生成树个数为 n^{n-2}
- 矩阵 - 树定理：图 G 由 n 个结点构成，设 $A[G]$ 为图 G 的邻接矩阵、 $D[G]$ 为图 G 的度数矩阵，则图 G 的不同生成树的个数为 $C[G] = D[G] - A[G]$ 的任意一个 $n-1$ 阶主子式的行列式值。

8.1.8 欧拉公式

平面图的顶点个数、边数和面的个数有如下关系： $V - E + F = C + 1$ 其中， V 是顶点的数目， E 是边的数目， F 是面的数目， C 是组成图形的连通部分的数目。当图是单连通图的时候，公式简化为： $V - E + F = 2$

8.1.9 皮克定理

给定顶点坐标均是整点（或正方形格点）的简单多边形，其面积 A 和内部格点数目 i 、边上格点数目 b 的关系： $A = i + \frac{b}{2} - 1$

8.1.10 牛顿恒等式

设 $\prod_{i=1}^n (x - x_i) = a_n + a_{n-1}x + \dots + a_1x^{n-1} + a_0x^n$ $p_k = \sum_{i=1}^n x_i^k$ 则 $a_0p_k + a_1p_{k-1} + \dots + a_{k-1}p_1 + ka_k = 0$

特别地，对于 $|A - \lambda E| = (-1)^n (a_n + a_{n-1}\lambda + \dots + a_1\lambda^{n-1} + a_0\lambda^n)$ 有 $p_k = Tr(A^k)$

8.2 平面几何公式

8.2.1 三角形

- 半周长 $p = \frac{a+b+c}{2}$
- 面积 $S = \frac{a \cdot H_a}{2} = \frac{ab \cdot \sin C}{2} = \sqrt{p(p-a)(p-b)(p-c)}$
- 中线 $M_a = \frac{\sqrt{2(b^2+c^2)-a^2}}{2} = \frac{\sqrt{b^2+c^2+2bc \cdot \cos A}}{2}$
- 角平分线 $T_a = \frac{\sqrt{bc \cdot [(b+c)^2 - a^2]}}{b+c} = \frac{2bc \cos \frac{A}{2}}{b+c}$
- 高线 $H_a = b \sin C = c \sin B = \sqrt{b^2 - (\frac{a^2+b^2-c^2}{2a})^2}$
- 内切圆半径

$$r = \frac{S}{p} = \frac{\arcsin \frac{B}{2} \cdot \sin \frac{C}{2}}{\sin \frac{B+C}{2}} = 4R \cdot \sin \frac{A}{2} \sin \frac{B}{2} \sin \frac{C}{2}$$

$$= \sqrt{\frac{(p-a)(p-b)(p-c)}{p}} = p \cdot \tan \frac{A}{2} \tan \frac{B}{2} \tan \frac{C}{2}$$

$$7. \text{ 外接圆半径 } R = \frac{abc}{4S} = \frac{a}{2 \sin A} = \frac{b}{2 \sin B} = \frac{c}{2 \sin C}$$

8.2.2 四边形

D_1, D_2 为对角线， M 为对角线中点连线， A 为对角线夹角， p 为半周长

- $a^2 + b^2 + c^2 + d^2 = D_1^2 + D_2^2 + 4M^2$
- $S = \frac{1}{2} D_1 D_2 \sin A$
- 对于圆内接四边形 $ac + bd = D_1 D_2$
- 对于圆内接四边形 $S = \sqrt{(p-a)(p-b)(p-c)(p-d)}$

8.2.3 正 n 边形

R 为外接圆半径, r 为内切圆半径

1. 中心角 $A = \frac{2\pi}{n}$
2. 内角 $C = \frac{n-2}{n}\pi$
3. 边长 $a = 2\sqrt{R^2 - r^2} = 2R \cdot \sin \frac{A}{2} = 2r \cdot \tan \frac{A}{2}$
4. 面积 $S = \frac{nar}{2} = nr^2 \cdot \tan \frac{A}{2} = \frac{nR^2}{2} \cdot \sin A = \frac{na^2}{4 \cdot \tan \frac{A}{2}}$

8.2.4 圆

1. 弧长 $l = rA$
2. 弦长 $a = 2\sqrt{2hr - h^2} = 2r \cdot \sin \frac{A}{2}$
3. 弓形高 $h = r - \sqrt{r^2 - \frac{a^2}{4}} = r(1 - \cos \frac{A}{2}) = \frac{1}{2} \cdot \arctan \frac{A}{4}$
4. 扇形面积 $S_1 = \frac{rl}{2} = \frac{r^2 A}{2}$
5. 弓形面积 $S_2 = \frac{rl - a(r-h)}{2} = \frac{r^2}{2}(A - \sin A)$

8.2.5 棱柱

1. 体积 $V = Ah$ A 为底面积, h 为高
2. 侧面积 $S = lp$ l 为棱长, p 为直截面周长
3. 全面积 $T = S + 2A$

8.2.6 棱锥

1. 体积 $V = \frac{1}{3}Ah$ A 为底面积, h 为高
2. 正棱锥侧面积 $S = lp$ l 为棱长, p 为直截面周长
3. 正棱锥全面积 $T = S + 2A$

8.2.7 棱台

1. 体积 $V = (A_1 + A_2 + \sqrt{A_1 A_2}) \cdot \frac{h}{3}$ A_1, A_2 为上下底面积, h 为高
2. 正棱台侧面积 $S = \frac{p_1 + p_2}{2} l$ p_1, p_2 为上下底面周长, l 为斜高
3. 正棱台全面积 $T = S + A_1 + A_2$

8.2.8 圆柱

1. 侧面积 $S = 2\pi rh$
2. 全面积 $T = 2\pi r(h + r)$
3. 体积 $V = \pi r^2 h$

8.2.9 圆锥

1. 母线 $l = \sqrt{h^2 + r^2}$
2. 侧面积 $S = \pi rl$
3. 全面积 $T = \pi r(l + r)$
4. 体积 $V = \frac{\pi}{3} r^2 h$

8.2.10 圆台

1. 母线 $l = \sqrt{h^2 + (r_1 - r_2)^2}$
2. 侧面积 $S = \pi(r_1 + r_2)l$
3. 全面积 $T = \pi r_1(l + r_1) + \pi r_2(l + r_2)$
4. 体积 $V = \frac{\pi}{3}(r_1^2 + r_2^2 + r_1 r_2)h$

8.2.11 球

1. 全面积 $T = 4\pi r^2$
2. 体积 $V = \frac{4}{3}\pi r^3$

8.2.12 球台

1. 侧面积 $S = 2\pi rh$
2. 全面积 $T = \pi(2rh + r_1^2 + r_2^2)$
3. 体积 $V = \frac{\pi h[3(r_1^2 + r_2^2) + h^2]}{6}$

8.2.13 球扇形

1. 全面积 $T = \pi r(2h + r_0)$ h 为球冠高, r_0 为球冠底面半径
2. 体积 $V = \frac{2}{3}\pi r^2 h$

8.3 立体几何公式

8.3.1 球面三角公式

设 a, b, c 是边长, A, B, C 是所对的二面角, 有余弦定理 $\cos a = \cos b \cdot \cos c + \sin b \cdot \sin c \cdot \cos A$ 正弦定理 $\frac{\sin A}{\sin a} = \frac{\sin B}{\sin b} = \frac{\sin C}{\sin c}$ 三角形面积是 $A + B + C - \pi$

8.3.2 四面体体积公式

U, V, W, u, v, w 是四面体的 6 条棱, U, V, W 构成三角形, $(U, u), (V, v), (W, w)$ 互为对棱, 则 $V =$

$$\frac{\sqrt{(s-2a)(s-2b)(s-2c)(s-2d)}}{192uvw} \quad \text{其中} \quad \begin{cases} a = \sqrt{xYZ}, \\ b = \sqrt{yZX}, \\ c = \sqrt{zXY}, \\ d = \sqrt{xyz}, \\ s = a + b + c + d, \\ X = (w - U + v)(U + v + w), \\ x = (U - v + w)(v - w + U), \\ Y = (u - V + w)(V + w + u), \\ y = (V - w + u)(w - u + V), \\ Z = (v - W + u)(W + u + v), \\ z = (W - u + v)(u - v + W) \end{cases}$$

8.4 积分表

$$\arcsin x \rightarrow \frac{1}{\sqrt{1-x^2}}$$

$$\arccos x \rightarrow -\frac{1}{\sqrt{1-x^2}}$$

$$\arctan x \rightarrow \frac{1}{1+x^2}$$

$$a^x \rightarrow \frac{a^x}{\ln a}$$

$$\sin x \rightarrow -\cos x$$

$$\cos x \rightarrow \sin x$$

$$\tan x \rightarrow -\ln \cos x$$

$$\sec x \rightarrow \ln \tan(\frac{x}{2} + \frac{\pi}{4})$$

$$\tan^2 x \rightarrow \tan x - x$$

$$\csc x \rightarrow \ln \tan \frac{x}{2}$$

$$\sin^2 x \rightarrow \frac{x}{2} - \frac{1}{2} \sin x \cos x$$

$$\cos^2 x \rightarrow \frac{x}{2} + \frac{1}{2} \sin x \cos x$$

$$\sec^2 x \rightarrow \tan x$$

$$\frac{1}{\sqrt{a^2-x^2}} \rightarrow \arcsin \frac{x}{a}$$

$$\csc^2 x \rightarrow -\cot x$$

$$\frac{1}{a^2-x^2} (|x| < |a|) \rightarrow \frac{1}{2a} \ln \frac{a+x}{a-x}$$

$$\frac{1}{x^2-a^2} (|x| > |a|) \rightarrow \frac{1}{2a} \ln \frac{x-a}{x+a}$$

$$\sqrt{a^2-x^2} \rightarrow \frac{x}{2} \sqrt{a^2-x^2} + \frac{a^2}{2} \arcsin \frac{x}{a}$$

$$\frac{1}{\sqrt{x^2+a^2}} \rightarrow \ln(x + \sqrt{a^2+x^2})$$

$$\sqrt{a^2+x^2} \rightarrow \frac{x}{2} \sqrt{a^2+x^2} + \frac{a^2}{2} \ln(x + \sqrt{a^2+x^2})$$

$$\frac{1}{\sqrt{x^2-a^2}} \rightarrow \ln(x + \sqrt{x^2-a^2})$$

$$\sqrt{x^2-a^2} \rightarrow \frac{x}{2} \sqrt{x^2-a^2} - \frac{a^2}{2} \ln(x + \sqrt{x^2-a^2})$$

$$\frac{1}{x\sqrt{a^2-x^2}} \rightarrow -\frac{1}{a} \ln \frac{a+\sqrt{a^2-x^2}}{x}$$

$$\frac{1}{x\sqrt{x^2-a^2}} \rightarrow \frac{1}{a} \arccos \frac{a}{x}$$

$$\frac{1}{x\sqrt{a^2+x^2}} \rightarrow -\frac{1}{a} \ln \frac{a+\sqrt{a^2+x^2}}{x}$$

$$\frac{1}{\sqrt{2ax-x^2}} \rightarrow \arccos(1 - \frac{x}{a})$$

$$\frac{x}{ax+b} \rightarrow \frac{x}{a} - \frac{b}{a^2} \ln(ax+b)$$

$$\sqrt{2ax-x^2} \rightarrow \frac{x-a}{2} \sqrt{2ax-x^2} + \frac{a^2}{2} \arcsin(\frac{x}{a} - 1)$$

$$\frac{1}{x\sqrt{ax+b}} (b < 0) \rightarrow \frac{2}{\sqrt{-b}} \arctan \sqrt{\frac{ax+b}{-b}}$$

$$x\sqrt{ax+b} \rightarrow \frac{2(3ax-2b)}{15a^2} (ax+b)^{\frac{3}{2}}$$

$$\frac{1}{x\sqrt{ax+b}} (b > 0) \rightarrow \frac{1}{\sqrt{b}} \ln \frac{\sqrt{ax+b}-\sqrt{b}}{\sqrt{ax+b}+\sqrt{b}}$$

$$\frac{x}{\sqrt{ax+b}} \rightarrow \frac{2(ax-2b)}{3a^2} \sqrt{ax+b}$$

$$\frac{1}{x^2\sqrt{ax+b}} \rightarrow -\frac{\sqrt{ax+b}}{bx} - \frac{a}{2b} \int \frac{dx}{x\sqrt{ax+b}}$$

$$\frac{\sqrt{ax+b}}{x} \rightarrow 2\sqrt{ax+b} + b \int \frac{dx}{x\sqrt{ax+b}}$$

$$\begin{aligned}
\frac{1}{\sqrt{(ax+b)^n}} (n > 2) &\rightarrow \frac{-2}{a(n-2)} \cdot \frac{1}{\sqrt{(ax+b)^{n-2}}} \\
\frac{1}{ax^2+c} (a > 0, c > 0) &\rightarrow \frac{1}{\sqrt{ac}} \arctan(x\sqrt{\frac{a}{c}}) \\
\frac{x}{ax^2+c} &\rightarrow \frac{1}{2a} \ln(ax^2+c) \\
\frac{1}{ax^2+c} (a+, c-) &\rightarrow \frac{1}{2\sqrt{-ac}} \ln \frac{x\sqrt{a}-\sqrt{-c}}{x\sqrt{a}+\sqrt{-c}} \\
\frac{1}{x(ax^2+c)} &\rightarrow \frac{1}{2c} \ln \frac{x^2}{ax^2+c} \\
\frac{1}{ax^2+c} (a-, c+) &\rightarrow \frac{1}{2\sqrt{-ac}} \ln \frac{\sqrt{c}+x\sqrt{-a}}{\sqrt{c}-x\sqrt{-a}} \\
x\sqrt{ax^2+c} &\rightarrow \frac{1}{3a} \sqrt{(ax^2+c)^3} \\
\frac{1}{(ax^2+c)^n} (n > 1) &\rightarrow \frac{x}{2c(n-1)(ax^2+c)^{n-1}} + \frac{2n-3}{2c(n-1)} \int \frac{dx}{(ax^2+c)^{n-1}} \\
\frac{x^n}{ax^2+c} (n \neq 1) &\rightarrow \frac{x^{n-1}}{a(n-1)} - \frac{c}{a} \int \frac{x^{n-2}}{ax^2+c} dx \\
\frac{1}{x^2(ax^2+c)} &\rightarrow \frac{-1}{cx} - \frac{a}{c} \int \frac{dx}{ax^2+c} \\
\frac{1}{x^2(ax^2+c)^n} (n \geq 2) &\rightarrow \frac{1}{c} \int \frac{dx}{x^2(ax^2+c)^{n-1}} - \frac{a}{c} \int \frac{dx}{(ax^2+c)^n} \\
\sqrt{ax^2+c} (a > 0) &\rightarrow \frac{x}{2} \sqrt{ax^2+c} + \frac{c}{2\sqrt{a}} \ln(x\sqrt{a} + \sqrt{ax^2+c}) \\
\sqrt{ax^2+c} (a < 0) &\rightarrow \frac{x}{2} \sqrt{ax^2+c} + \frac{c}{2\sqrt{-a}} \arcsin(x\sqrt{\frac{-a}{c}}) \\
\frac{1}{\sqrt{ax^2+c}} (a > 0) &\rightarrow \frac{1}{\sqrt{a}} \ln(x\sqrt{a} + \sqrt{ax^2+c})
\end{aligned}$$

$$\begin{aligned}
\frac{1}{\sqrt{ax^2+c}} (a < 0) &\rightarrow \frac{1}{\sqrt{-a}} \arcsin(x\sqrt{\frac{-a}{c}}) \\
\sin^2 ax &\rightarrow \frac{x}{2} - \frac{1}{4a} \sin 2ax \\
\cos^2 ax &\rightarrow \frac{x}{2} + \frac{1}{4a} \sin 2ax \\
\frac{1}{\sin ax} &\rightarrow \frac{1}{a} \ln \tan \frac{ax}{2} \\
\frac{1}{\cos^2 ax} &\rightarrow \frac{1}{a} \tan ax \\
\frac{1}{\cos ax} &\rightarrow \frac{1}{a} \ln \tan(\frac{\pi}{4} + \frac{ax}{2}) \\
\ln(ax) &\rightarrow x \ln(ax) - x \\
\sin^3 ax &\rightarrow \frac{-1}{a} \cos ax + \frac{1}{3a} \cos^3 ax \\
\cos^3 ax &\rightarrow \frac{1}{a} \sin ax - \frac{1}{3a} \sin^3 ax \\
\frac{1}{\sin^2 ax} &\rightarrow -\frac{1}{a} \cot ax \\
x \ln(ax) &\rightarrow \frac{x^2}{2} \ln(ax) - \frac{x^2}{4} \\
\cos ax &\rightarrow \frac{1}{a} \sin ax \\
x^2 e^{ax} &\rightarrow \frac{e^{ax}}{a^3} (a^2 x^2 - 2ax + 2) \\
(\ln(ax))^2 &\rightarrow x(\ln(ax))^2 - 2x \ln(ax) + 2x \\
x^2 \ln(ax) &\rightarrow \frac{x^3}{3} \ln(ax) - \frac{x^3}{9} \\
x^n \ln(ax) &\rightarrow \frac{x^{n+1}}{n+1} \ln(ax) - \frac{x^{n+1}}{(n+1)^2} \\
\sin(\ln ax) &\rightarrow \frac{x}{2} [\sin(\ln ax) - \cos(\ln ax)] \\
\cos(\ln ax) &\rightarrow \frac{x}{2} [\sin(\ln ax) + \cos(\ln ax)]
\end{aligned}$$