

Talisman

November 25, 2018

目录

1 代数	2	6.4 最小表示-gwx	18
1.1 FFT-wrz	2	6.5 回文树-wrz	18
1.2 FWT	2	6.6 后缀数组-wrz	18
1.3 高精度-wrz	2	6.7 后缀数组 SAIS	19
1.4 线性基-gwx	3	6.8 后缀自动机-wrz	19
1.5 单纯形	3	6.9 ex 后缀自动机-wrz	19
1.6 NTT-gwx	3	7 其他	19
2 计算几何	3	7.1 蔡勒公式	19
2.1 二维计算几何-wrz	3	7.2 dancing-links	19
2.2 basis	4	7.3 枚举子集	20
2.3 circles-intersections	5	7.4 梅森旋转	20
2.4 cirque-area-merge	6	7.5 大数乘法取模	20
2.5 凸包	6	8 提示	20
2.6 三角形内心, 外心, 垂心	6	8.1 make 支持 c++11	20
2.7 三维计算几何	7	8.2 Java	20
2.8 三维凸包	7	8.3 cout 输出小数	21
3 数据结构	7	8.4 释放容器内存	21
3.1 KD 树-wrz	7	8.5 tuple	21
3.2 KD 树-gwx	7	8.6 读入优化 & 手开 O3	21
3.3 LCT-wrz	8	8.7 手开栈	21
3.4 左偏树-wrz	8	9 附录-数学公式	21
3.5 splay-wrz	9		
3.6 treap-gwx	9		
3.7 可持久化平衡树	10		
4 图论	10		
4.1 树哈希	10		
4.2 匹配	10		
4.3 上下界网络流	10		
4.4 矩阵树定理	11		
4.5 边双联通-gwx	11		
4.6 带花树	11		
4.7 弦图	11		
4.8 支配树-gwx	12		
4.9 欧拉回路-wrz	12		
4.10 Hopcroft-Karp	12		
4.11 KM-truly-n3	12		
4.12 k 短路 a 星-gwx	13		
4.13 K 短路可并堆	13		
4.14 最大团	14		
4.15 SPFA 判负环-wrz	14		
4.16 斯坦纳树	14		
4.17 stoer-wagner 无向图最小割树	14		
4.18 tarjan-gwx	15		
4.19 朱刘算法-gwx	15		
4.20 zkw 费用流	15		
5 数论	15		
5.1 杜教筛	15		
5.2 直线下整点	16		
5.3 拉格朗日插值	16		
5.4 线性回归	16		
5.5 素数测试-gwx	16		
5.6 原根-gwx	16		
5.7 勾股数	17		
5.8 Pell 方程	17		
5.9 平方剩余	17		
5.10 博弈论	17		
6 字符串	17		
6.1 AC 自动机-wrz	17		
6.2 扩展 KMP-gwx	18		
6.3 马拉车-gwx	18		

1 代数

1.1 FFT-wrз

```

1 typedef complex<double> comp;
2 int len;
3 comp w[N<<1], a[N<<1], b[N<<1], c[N<<1]; // 数组记得至
    少开两倍
4 void init()
5 {
6     double pi = acos(-1.0);
7     for(int i = 0; i < len; i++) w[i] = (comp){cos(2*
        pi*i/len), sin(2*pi*i/len)};
8 }
9 void FFT(comp *a, comp *w)
10 {
11     for(int i = 0, j = 0; i < len; i++)
12     {
13         if(i < j) swap(a[i], a[j]);
14         for(int l = len >> 1; (j ^= l) < l; l >>= 1);
15     }
16     for(int i = 2; i <= len; i <= 1)
17     {
18         int m = i >> 1;
19         for(int j = 0; j < len; j += i)
20         {
21             for(int k = 0; k < m; k++)
22             {
23                 comp tmp = w[len/i*k] * a[j+k+m];
24                 a[j+k+m] = a[j+k] - tmp;
25                 a[j+k] = a[j+k] + tmp;
26             }
27         }
28     }
29 }
30 void mul(comp *a, comp *b, comp *c, int l) // 多项式乘
    法, c = a * b, c的长度为l
31 {
32     for(len = 1; len <= l; len <= 1);
33     init(); FFT(a, w); FFT(b, w);
34     for(int i = 0; i < len; i++) c[i] = a[i] * b[i];
35     reverse(c+1, c+len); FFT(c, w);
36     for(int i = 0; i < len; i++) c[i].r /= len; // 转
    化为int等时应加0.5, 如int(c[i].r+0.5)
37 }

```

1.2 FWT

```

1 void FWT(int a[], int n)
2 {
3     for (int d = 1; d < n; d <= 1)
4         for (int m = d << 1, i = 0; i < n; i += m)
5             for (int j = 0; j < d; j++) {
6                 int x = a[i + j], y = a[i + j + d];
7
8                 a[i + j] = (x + y) % mod,
9                 a[i + j + d] = (x - y + mod) % mod;
10
11                 //xor: a[i + j] = x + y, a[i + j + d]
12                 //      = (x - y + mod) % mod;
13                 //and: a[i + j] = x + y;
14                 //or:  a[i + j + d] = x + y;
15             }
16 }
17 void UFWT(int a[], int n)
18 {
19     for (int d = 1; d < n; d <= 1)
20         for (int m = d << 1, i = 0; i < n; i += m)
21             for (int j = 0; j < d; j++) {
22                 int x = a[i + j], y = a[i + j + d];
23
24                 a[i + j] = 1LL * (x + y) * rev % mod,
25                 a[i + j + d] = (1LL * (x - y) * rev %
26                     mod + mod) % mod;
27
28                 //xor: a[i + j] = (x + y) / 2, a[i + j
29                 //      + d] = (x - y) / 2;
30                 //and: a[i + j] = x - y;
31                 //or:  a[i + j + d] = y - x;
32             }
33 }
34 void solve(int a[], int b[], int n)
35 {
36     FWT(a, n);
37     FWT(b, n);
38     for (int i = 0; i < n; i++)
39         a[i] = 1LL * a[i] * b[i] % mod;
40     UFWT(a, n);

```

1.3 高精度-wrз

```

1 #include<bits/stdc++.h>
2 #define BASE 10000
3 #define L 20005
4 using namespace std;
5 int p; char s[10*L];
6 struct bigint

```

```

7 {
8     int num[L], len;
9     bigint(int x = 0){memset(num,0,sizeof(num)); len =
        1; num[0] = x;}
10     bigint operator + (bigint b)
11     {
12         bigint c;
13         c.len = max(b.len, len);
14         for(int i = 0; i < c.len; i++)
15         {
16             c.num[i] += num[i] + b.num[i];
17             c.num[i+1] = c.num[i] / BASE;
18             c.num[i] %= BASE;
19         }
20         if(c.num[c.len])c.len++;
21         return c;
22     }
23     bigint operator - (bigint b)
24     {
25         bigint c;
26         c.len = max(len, b.len);
27         for(int i = 0; i < c.len; i++)
28         {
29             c.num[i] += num[i] - b.num[i];
30             if(c.num[i] < 0) {c.num[i] += BASE; c.num[
31                 i+1]--;}
32             while(!c.num[c.len-1] && c.len > 1)c.len--;
33             return c;
34         }
35     }
36     void operator -- (int b)
37     {
38         num[0] -= b;
39         for(int i = 0; i < len; i++)
40         {
41             num[i+1] += num[i] / BASE;
42             num[i] %= BASE;
43             if(num[i] < 0)num[i] += BASE, num[i+1]--;
44         }
45         while(!num[len-1] && len > 1) len--;
46     }
47     bigint operator * (bigint b)
48     {
49         bigint c;
50         c.len = len + b.len;
51         for(int i = 0; i < len; i++)
52             for(int j = 0; j < b.len; j++)
53             {
54                 c.num[i+j] += num[i] * b.num[j];
55                 c.num[i+j+1] += c.num[i+j] / BASE;
56                 c.num[i+j] %= BASE;
57             }
58         if(!c.num[c.len-1] && c.len > 1)c.len--;
59         return c;
60     }
61     bigint operator * (int b)
62     {
63         bigint c;
64         for(int i = 0; i < len; i++) c.num[i] = num[i]
65             * b; // long long
66         for(int i = 0; i < len; i++){c.num[i+1] += c.
67             num[i] / BASE; c.num[i] %= BASE;}
68         c.len = len;
69         while(c.num[c.len])c.len++;
70         return c;
71     }
72     bool subtract(bigint b, int pos)
73     {
74         if(len < b.len - pos)return false;
75         else if(len == b.len-pos)
76             for(int i = len-1; i >= 0; i--)
77                 if(num[i] < b.num[i+pos])return false;
78                 else if(num[i] > b.num[i+pos])break;
79         for(int i = 0; i < len; i++)
80         {
81             num[i] -= b.num[i+pos];
82             if(num[i] < 0){num[i] += BASE; num[i+1]
83                 --;}
84         }
85         while(!num[len-1] && len > 1)len--;
86         return true;
87     }
88     // remember to change [BASE] to 10 !!!
89     // [this] is the remainder
90     bigint operator / (bigint b)
91     {
92         bigint c;
93         if(len < b.len)return c;
94         int k = len - b.len;
95         c.len = k + 1;
96         for(int i = len-1; i >= 0; i--)
97         {
98             if(i >= k)b.num[i] = b.num[i-k];
99             else b.num[i] = 0;
100         }
101         b.len = len;
102         for(int i = 0; i <= k; i++) while(this->
            subtract(b,i)) c.num[k-i]++;
103         for(int i = 0; i < c.len; i++)
104         {
105             c.num[i+1] += c.num[i] / BASE;
106             c.num[i] %= BASE;
107         }

```

```

103     }
104     while(!c.num[c.len-1] && c.len > 0) c.len--;
105     return c;
106 }
107 // [this] is not the remainder
108 bigint operator / (int b)
109 {
110     bigint c; int tmp = 0;
111     for(int i = len-1; i>=0; i--)
112     {
113         tmp = tmp * BASE + num[i];
114         c.num[i] = tmp / b;
115         tmp %= b;
116     }
117     for(c.len = len; !c.num[c.len-1] && c.len > 1;
118         c.len--);
119     return c;
120 }
121 bool scan()
122 {
123     int n = -1; char ch = getchar();
124     while(ch < '0' || ch > '9') if(ch == EOF)
125         return false; else ch = getchar();
126     while(ch >= '0' && ch <= '9') s[++n] = ch - '0', ch = getchar();
127     len = 0;
128     for(int i = n; i >= 0; i-=4)
129     {
130         num[len] += s[i];
131         if(i>=1) num[len] += s[i-1] * 10;
132         if(i>=2) num[len] += s[i-2] * 100;
133         if(i>=3) num[len] += s[i-3] * 1000;
134         ++len;
135     }
136     return true;
137 }
138 void clr(){memset(num,0,sizeof(num));}
139 void print()
140 {
141     printf("%d",num[len-1]);
142     for(int i = len-2; i>=0; i--) printf("%04d",
143         num[i]);
144     printf("\n");
145 }
146 int main(){}
```

1.4 线性基-gwx

```

1 ll solve()
2 {
3     ll res = 0;
4     memset(b, 0, sizeof(b));
5     for(int i = 1; i <= tot; i++)
6         for(int j = 60; j >= 0; j--)
7             if((a[i] >> j) & 1)
8             {
9                 if(!b[j])
10                 {
11                     b[j] = a[i];
12                     break;
13                 }
14                 a[i] ^= b[j];
15             }
16     for(int i = 60; i >= 0; i--)
17         res = max(res, res ^ b[i]);
18     return res;
19 }
```

1.5 单纯形

```

1 // max{c * x | Ax <= b, x >= 0}的解，无解返回空的
2 // vector，否则就是解。答案在an中
3 template <int MAXN = 100, int MAXM = 100>
4 struct simplex {
5     int n, m; double a[MAXN][MAXN], b[MAXN], c[MAXN];
6     bool infeasible, unbounded;
7     double v, an[MAXN + MAXM]; int q[MAXN + MAXM];
8     void pivot (int l, int e) {
9         std::swap (q[e], q[l + n]);
10         double t = a[l][e]; a[l][e] = 1; b[l] /= t;
11         for (int i = 0; i < n; ++i) a[l][i] /= t;
12         for (int i = 0; i < m; ++i) if (i != l && std
13             ::abs (a[i][e]) > EPS) {
14             t = a[i][e]; a[i][e] = 0; b[i] -= t * b[l]
15             ;
16             for (int j = 0; j < n; ++j) a[i][j] -= t *
17                 a[l][j]; }
18         if (std::abs (c[e]) > EPS) {
19             t = c[e]; c[e] = 0; v += t * b[l];
20             for (int j = 0; j < n; ++j) c[j] -= t * a
21                 [l][j]; } }
22     bool pre () {
23         for (int l, e; ) {
24             l = e = -1;
25             for (int i = 0; i < m; ++i) if (b[i] < -
26                 EPS && (!~l || rand () & 1)) l = i;
27             if (!~l) return false;
28             for (int i = 0; i < n; ++i) if (a[l][i] <
29                 -EPS && (!~e || rand () & 1)) e = i;
```

```

23         if (!~e) return infeasible = true;
24         pivot (l, e); } }
25     double solve () {
26         double p; std::fill (q, q + n + m, -1);
27         for (int i = 0; i < n; ++i) q[i] = i;
28         v = 0; infeasible = unbounded = false;
29         if (pre ()) return 0;
30         for (int l, e; ; pivot (l, e)) {
31             l = e = -1; for (int i = 0; i < n; ++i) if
32                 (c[i] > EPS) { e = i; break; }
33             if (!~e) break; p = INF;
34             for (int i = 0; i < m; ++i) if (a[i][e] >
35                 EPS && p > b[i] / a[i][e]) {
36                 p = b[i] / a[i][e], l = i;
37                 if (!~l) return unbounded = true, 0; }
38             for (int i = n; i < n + m; ++i) if (~q[i]) an
39                 [q[i]] = b[i - n];
40             return v; } };
```

1.6 NTT-gwx

```

1 const int G;
2 int n, m, inm;
3 int rev[maxn], a[maxn], b[maxn]; //maxn > 2 ^ k
4
5 ll power(ll b, int k)
6 {
7     ll res = 1;
8     for(; k; k >>= 1, b = b * b % mod)
9         if(k & 1)
10             res = res * b % mod;
11     return res;
12 }
13
14 void ntt(ll*a, int f)
15 {
16     for(int i = 0; i < m; i++)
17         if(rev[i] < i)
18             swap(a[i], a[rev[i]]);
19     for(int l = 2, h = 1; l <= m; h = l, l <= 1)
20     {
21         int ur;
22         if(f == 1)
23             ur = power(G, (mod - 1) / l);
24         else
25             ur = power(G, mod - 1 - (mod - 1) / l);
26         for(int i = 0; i < m; i += l)
27         {
28             ll w = 1;
29             for(int k = i; k < i + h; k++, w = w * ur
30                 % mod)
31             {
32                 int x = a[k], y = a[k + h] * w % mod;
33                 a[k] = (x + y) % mod;
34                 a[k + h] = (x - y + mod) % mod;
35             }
36         }
37         if(f == -1)
38             for(int i = 0; i < m; i++)
39                 a[i] = a[i] * inm % mod;
40     }
41 }
42
43 void multi()
44 {
45     ntt(a, 1); ntt(b, 1);
46     for(int i = 0; i < m; i++)
47         a[i] = a[i] * b[i] % mod;
48     ntt(a, -1);
49 }
50
51 void init()
52 {
53     for(m = 1; m <= 2 * n; m <= 1) ;
54     for(int i = 1; i < m; i++)
55     {
56         rev[i] = rev[i - 1];
57         for(int j = m >> 1; (rev[i] ^= j) < j; j >>=
58             1) ;
59     }
60 }
```

2 计算几何

2.1 二维计算几何-wrxz

```

1 #include <bits/stdc++.h>
2 using namespace std;
3 const double inf = 1e9;
4 const double eps = 1e-9;
5 const double pi = acos(-1.0);
6 bool le(double x, double y){return x < y - eps; } // x
7 // 严格小于y
8 bool leq(double x, double y){return x < y + eps; } // x
9 // 小于等于y
10 bool equ(double x, double y){return fabs(x - y) < eps
11     ;} // x等于y
12 double mysqrt(double x) {return x < eps ? 0 : sqrt(x)
13     ;} // 开根号
14 double sqr(double x) {return x * x;} // 平方
```

```

11 struct point // 点或向量
12 {
13     double x, y;
14     double operator * (point that){return x*that.x + y
15         *that.y;}
16     double operator ^ (point that){return x*that.y - y
17         *that.x;}
18     point operator * (double t){return (point){x*t, y*
19         t};}
20     point operator / (double t){return (point){x/t, y/
21         t};}
22     point operator + (point that) {return (point){x +
23         that.x, y + that.y};}
24     point operator - (point that) {return (point){x -
25         that.x, y - that.y};}
26     double len(){return mysqrt(x*x+y*y);} // 到原点距
27         离/向量长度
28     point reset_len(double t) // 改变向量长度为t, t为
29         正则方向不变, t为负则方向相反
30     {double p = len();return (point){x*t/p, y*t/p};}
31     point rot90() {return (point){-y, x};} // 逆时针旋
32         转90度
33     point rotate(double angle) // 使向量逆时针旋转
34         angle弧度
35     {double c = cos(angle), s = sin(angle);return (
36         point){c * x - s * y, s * x + c * y};}
37 };
38 struct line // 参数方程表示, p为线上一点, v为方向向量
39 {
40     point p, v; // p为线上一点, v为方向向量
41     double angle; // 半平面交用, 用atan2计算, 此时v的
42         左侧为表示的半平面。注意有的函数声明一个新的
43         line时没有初始化这个值!
44     bool operator < (const line &that) const {return
45         angle < that.angle;} // 半平面交用, 按与x轴夹
46         角排序
47 };
48 struct circle{point c; double r;};
49 double distance(point a, point b) // a, b两点距离
50 {return mysqrt(sqr(a.x - b.x) + sqr(a.y - b.y));}
51 circle make_circle(point a, point b) // 以a, b两点为直
52     径作圆
53 {double d = distance(a, b);return (circle){(a+b)/2, d
54     /2};}
55 double point_to_line(point a, line b) // 点a到直线b距
56     离
57 {return fabs((b.v ^ (a - b.p)) / b.v.len());}
58 point project_to_line(point a, line b) // 点a到直线b的
59     垂足/投影
60 {return b.v.reset_len((a - b.p) * b.v / b.v.len()) + b
61     .p;}
62 vector<point> circle_inter(circle a, circle b) // 圆a
63     和圆b的交点, 需保证两圆不重合, 圆的半径必须大于0
64 {
65     double d = distance(a.c, b.c);
66     vector<point> ret;
67     if(le(a.r + b.r, d) || le(a.r - d, b.r) || le(b.r
68         - d, a.r)) return vector<point>(); // 相离或内
69         含
70     point r = (b.c - a.c).reset_len(1);
71     double x = ((sqr(a.r) - sqr(b.r)) / d + d) / 2;
72     double h = mysqrt(sqr(a.r) - sqr(x));
73     if(equ(h, 0)) return vector<point>({a.c + r * x});
74         // 内切或外切
75     else return vector<point>({a.c + r*x + r.rot90()*h
76         , a.c + r*x - r.rot90()*h}); // 相交两点
77 }
78 vector<point> line_circle_inter(line a, circle b) //
79     直线a和圆b的交点
80 {
81     double d = point_to_line(b.c, a);
82     if(le(b.r, d)) return vector<point>(); // 不交
83     double x = mysqrt(sqr(b.r) - sqr(d));
84     point p = project_to_line(b.c, a);
85     if(equ(x, 0)) return vector<point> ({p}); // 相切
86     else return vector<point> ({p + a.v.reset_len(x),
87         p - a.v.reset_len(x)}); // 相交两点
88 }
89 point line_inter(line a, line b) // 直线a和直线b的交
90     点, 需保证两直线不平行
91 {double s1 = a.v ^ (b.p - a.p);double s2 = a.v ^ (b.p
92     + b.v - a.p);return (b.p * s2 - (b.p + b.v) * s1)
93     / (s2 - s1);}
94 vector<point> tangent(point p, circle a) // 过点p的圆a
95     的切线的切点, 圆的半径必须大于0
96 {circle c = make_circle(p, a.c);return circle_inter(a,
97     c);}
98 vector<line> intangent(circle a, circle b) // 圆a和圆b
99     的内公切线
100 {
101     point p = (b.c * a.r + a.c * b.r) / (a.r + b.r);
102     vector<point> va = tangent(p, a), vb = tangent(p,
103         b);
104     vector<line> ret;
105     if(va.size() == 2 && vb.size() == 2){ret.push_back
106         ((line){va[0], vb[0] - va[0]});ret.push_back((
107         line){va[1], vb[1] - va[1]});}
108 }

```

```

72     else if(va.size() == 1 && vb.size() == 1){ret.
73         push_back((line){p, (a.c - b.c).rot90()});}
74     return ret;
75 }
76 // 判断半平面交是否有解, 若有解需保证半平面交必须有
77     界, 可以通过外加四个大半平面解决
78 // lcnt为半平面数量, l为需要做的所有半平面的数组, p为
79     存交点的临时数组, h为时刻更新的合法的半平面数组,
80     下标均从1开始
81 bool HP(int lcnt, line *l, line *h, point *p)
82 {
83     sort(l+1, l+1+lcnt);
84     int head = 1, tail = 1;
85     h[1] = l[1];
86     for(int i = 2; i <= lcnt; i++)
87     {
88         line cur = l[i];
89         for(; head < tail && le(cur.v ^ (p[tail-1]-cur
90             .p), 0); tail--); // 先删队尾再删队头, 顺
91             序不能换
92         for(; head < tail && le(cur.v ^ (p[head]-cur.p
93             ), 0); head++);
94         h[++tail] = cur;
95         if(equ(h[tail].v ^ h[tail-1].v, 0)) // 平行
96         {
97             if(le(h[tail].v * h[tail-1].v, 0)) return
98                 false; // 方向相反的平行直线, 显然已经
99                 不可能围出有界半平面了
100             tail--;
101             if(le(h[tail+1].v ^ (h[tail].p - h[tail
102                 +1].p), 0)) h[tail] = h[tail+1];
103         }
104         if(head < tail) p[tail-1] = line_inter(h[tail
105             -1], h[tail]);
106     }
107     for(; head < tail && le(h[head].v ^ (p[tail-1]-h[
108         head].p), 0); tail--);
109     return tail - head > 1;
110 }
111 double calc(double X){return 0;} // 计算给定X坐标上的
112     覆盖的长度, 配合辛普森积分使用
113 // 自适应辛普森积分, 参数分别为(左端点x坐标, 中点x坐
114     标, 右端点x坐标, 左端点答案, 中点答案, 右端点答案)
115 // 改变计算深度应调整eps
116 double simpson(double l, double mid, double r, double
117     fl, double fm, double fr)
118 {
119     double lmid = (l+mid)/2, rmid = (r+mid)/2, flm =
120         calc(lmid), frm = calc(rmid);
121     double ans = (r-l) * (fl + 4*fm + fr), ans1 = (mid
122         -l) * (fl + 4*flm + fm), ansr = (r-mid) * (fm
123         + 4*frm + fr);
124     if(fabs(ans1 + ansr - ans) < eps) return ans / 6;
125     else return simpson(l,lmid,mid,fl,flm,fm) +
126         simpson(mid,rmid,r,frm,frm,fr);
127 }
128 int main(){

```

2.2 basis

```

1 struct Point {
2     DB x, y;
3     Point rotate(DB ang) const {return Point(cos(ang)
4         * x - sin(ang) * y, cos(ang) * y + sin(ang) *
5         x);} // 逆时针旋转 ang 弧度
6     Point turn90() const {return Point(-y, x);} // 逆
7         时针旋转 90 度
8     Point unit() const {return *this / len();}
9 };
10 DB dot(const Point& a, const Point& b) {return a.x * b
11     .x + a.y * b.y;}
12 DB det(const Point& a, const Point& b) {return a.x * b
13     .y - a.y * b.x;}
14 #define cross(p1,p2,p3) ((p2.x-p1.x)*(p3.y-p1.y)-(p3.x
15     -p1.x)*(p2.y-p1.y))
16 #define crossOp(p1,p2,p3) sign(cross(p1,p2,p3))
17 bool isLL(const Line& l1, const Line& l2, Point& p) {
18     // 直线与直线交点
19     DB s1 = det(l2.b - l2.a, l1.a - l2.a),
20         s2 = -det(l2.b - l2.a, l1.b - l2.a);
21     if (!sign(s1 + s2)) return false;
22     p = (l1.a * s2 + l1.b * s1) / (s1 + s2);
23     return true;
24 }
25 bool onSeg(const Line& l, const Point& p) { // 点在线
26     段上
27     return sign(det(p - l.a, l.b - l.a)) == 0 && sign(
28         dot(p - l.a, p - l.b)) <= 0;}
29 Point projection(const Line &l, const Point& p) {
30     return l.a + (l.b - l.a) * (dot(p - l.a, l.b - l.a)
31         / (l.b - l.a).len2());}
32 DB distoLine(const Line& l, const Point& p) { // 点到
33     直线距离
34     return fabs(det(p - l.a, l.b - l.a) / (l.b - l.a).
35         len());}
36 DB distoSeg(const Line& l, const Point& p) { // 点到
37     线段距离
38     return sign(dot(p - l.a, l.b - l.a)) * sign(dot(p

```

```

- 1.b, 1.a - 1.b)) == 1 ? disToLine(1, p) :
std::min((p - 1.a).len(), (p - 1.b).len());}
26 // 圆与直线交点
27 bool isCL(Circle a, Line l, Point& p1, Point& p2) {
28     DB x = dot(1.a - a.o, 1.b - 1.a),
29     y = (1.b - 1.a).len2(),
30     d = x * x - y * ((1.a - a.o).len2() - a.r * a.r);
31     if (sign(d) < 0) return false;
32     Point p = 1.a - ((1.b - 1.a) * (x / y)), delta = (
33         1.b - 1.a) * (msqrt(d) / y);
34     p1 = p + delta; p2 = p - delta;
35     return true;
36 }
37 // 圆与圆的交面积
38 DB areaCC(const Circle& c1, const Circle& c2) {
39     DB d = (c1.o - c2.o).len();
40     if (sign(d - (c1.r + c2.r)) >= 0) return 0;
41     if (sign(d - std::abs(c1.r - c2.r)) <= 0) {
42         DB r = std::min(c1.r, c2.r);
43         return r * r * PI;
44     }
45     DB x = (d * d + c1.r * c1.r - c2.r * c2.r) / (2 *
46         d),
47     t1 = acos(x / c1.r), t2 = acos((d - x) / c2.r);
48     return c1.r * c1.r * t1 + c2.r * c2.r * t2 - d *
49     c1.r * sin(t1);
50 }
51 // 圆与圆交点
52 bool isCC(Circle a, Circle b, P& p1, P& p2) {
53     DB s1 = (a.o - b.o).len();
54     if (sign(s1 - a.r - b.r) > 0 || sign(s1 - std::abs(
55         a.r - b.r)) < 0) return false;
56     DB s2 = (a.r * a.r - b.r * b.r) / s1;
57     DB aa = (s1 + s2) * 0.5, bb = (s1 - s2) * 0.5;
58     P o = (b.o - a.o) * (aa / (aa + bb)) + a.o;
59     P delta = (b.o - a.o).unit().turn90() * msqrt(a.r
60         * a.r - aa * aa);
61     p1 = o + delta, p2 = o - delta;
62     return true;
63 }
64 // 求点到圆的切点, 按关于点的顺时针方向返回两个点
65 bool tanCP(const Circle &c, const Point &p0, Point &p1
66     , Point &p2) {
67     double x = (p0 - c.o).len2(), d = x - c.r * c.r;
68     if (d < eps) return false; // 点在圆上认为没有切点
69     Point p = (p0 - c.o) * (c.r * c.r / x);
70     Point delta = ((p0 - c.o) * (-c.r * sqrt(d) / x)).
71     turn90();
72     p1 = c.o + p + delta;
73     p2 = c.o + p - delta;
74     return true;
75 }
76 // 求圆到圆的外共切线, 按关于 c1.o 的顺时针方向返回两
77 条线
78 vector<Line> extanCC(const Circle &c1, const Circle &
79     c2) {
80     vector<Line> ret;
81     if (sign(c1.r - c2.r) == 0) {
82         Point dir = c2.o - c1.o;
83         dir = (dir * (c1.r / dir.len())).turn90();
84         ret.push_back(Line(c1.o + dir, c2.o + dir));
85         ret.push_back(Line(c1.o - dir, c2.o - dir));
86     } else {
87         Point p = (c1.o * -c2.r + c2.o * c1.r) / (c1.r
88             - c2.r);
89         Point p1, p2, q1, q2;
90         if (tanCP(c1, p, p1, p2) && tanCP(c2, p, q1,
91             q2)) {
92             if (c1.r < c2.r) swap(p1, p2), swap(q1, q2);
93             ret.push_back(Line(p1, q1));
94             ret.push_back(Line(p2, q2));
95         }
96     }
97     return ret;
98 }
99 // 求圆到圆的内共切线, 按关于 c1.o 的顺时针方向返回两
100 条线
101 std::vector<Line> intanCC(const Circle &c1, const
102     Circle &c2) {
103     std::vector<Line> ret;
104     Point p = (c1.o * c2.r + c2.o * c1.r) / (c1.r + c2
105         .r);
106     Point p1, p2, q1, q2;
107     if (tanCP(c1, p, p1, p2) && tanCP(c2, p, q1, q2))
108     { // 两圆相切认为没有切线
109         ret.push_back(Line(p1, q1));
110         ret.push_back(Line(p2, q2));
111     }
112     return ret;
113 }
114 bool contain(vector<Point> polygon, Point p) { // 判断
115     点 p 是否被多边形包含, 包括落在边界上
116     int ret = 0, n = polygon.size();
117     for(int i = 0; i < n; ++ i) {
118         Point u = polygon[i], v = polygon[(i + 1) % n
119             ];
120         if (onSeg(Line(u, v), p)) return true; //
121         Here I guess.

```

```

103     if (sign(u.y - v.y) <= 0) swap(u, v);
104     if (sign(p.y - u.y) > 0 || sign(p.y - v.y) <=
105         0) continue;
106     ret += sign(det(p, v, u)) > 0;
107 }
108 return ret & 1;
109 }
110 // 用半平面 (q1,q2) 的逆时针方向去切凸多边形
111 std::vector<Point> convexCut(const std::vector<Point>&
112     ps, Point q1, Point q2) {
113     std::vector<Point> qs; int n = ps.size();
114     for (int i = 0; i < n; ++i) {
115         Point p1 = ps[i], p2 = ps[(i + 1) % n];
116         int d1 = crossOp(q1, q2, p1), d2 = crossOp(q1, q2
117             , p2);
118         if (d1 >= 0) qs.push_back(p1);
119         if (d1 * d2 < 0) qs.push_back(isSS(p1, p2, q1,
120             q2));
121     } return qs;
122 }

```

2.3 circles-intersections

```

1 struct Event {
2     Point p; double ang; int delta;
3     Event (Point p = Point(0, 0), double ang = 0,
4         double delta = 0) : p(p), ang(ang), delta(
5         delta) {}
6 };
7 bool operator < (const Event &a, const Event &b) {
8     return a.ang < b.ang;}
9 void addEvent(const Circle &a, const Circle &b, vector
10     <Event> &evt, int &cnt) {
11     double d2 = (a.o - b.o).len2(),
12     dRatio = ((a.r - b.r) * (a.r + b.r) / d2 +
13         1) / 2,
14     pRatio = sqrt(-(d2 - sqrt(a.r - b.r)) * (d2
15         - sqrt(a.r + b.r)) / (d2 * d2 * 4));
16     Point d = b.o - a.o, p = d.rotate(PI / 2),
17     q0 = a.o + d * dRatio + p * pRatio,
18     q1 = a.o + d * dRatio - p * pRatio;
19     double ang0 = (q0 - a.o).ang(),
20     ang1 = (q1 - a.o).ang();
21     evt.push_back(Event(q1, ang1, 1));
22     evt.push_back(Event(q0, ang0, -1));
23     cnt += ang1 > ang0;
24 }
25 bool issame(const Circle &a, const Circle &b) { return
26     sign((a.o - b.o).len()) == 0 && sign(a.r - b.r)
27     == 0; }
28 bool overlap(const Circle &a, const Circle &b) {
29     return sign(a.r - b.r - (a.o - b.o).len()) >= 0; }
30 bool intersect(const Circle &a, const Circle &b) {
31     return sign((a.o - b.o).len() - a.r - b.r) < 0; }
32 Circle c[N];
33 double area[N]; // area[k] -> area of intersections
34 >= k.
35 Point centroid[N];
36 bool keep[N];
37 void add(int cnt, DB a, Point c) {area[cnt] += a;
38     centroid[cnt] = centroid[cnt] + c * a;}
39 void solve(int C) {
40     for (int i = 1; i <= C; ++ i) {area[i] = 0;
41         centroid[i] = Point(0, 0);}
42     for (int i = 0; i < C; ++i) {
43         int cnt = 1;
44         vector<Event> evt;
45         for (int j = 0; j < i; ++j) if (issame(c[i], c
46             [j])) ++cnt;
47         for (int j = 0; j < C; ++j) {
48             if (j != i && !issame(c[i], c[j]) &&
49                 overlap(c[j], c[i])) ++cnt;}
50         for (int j = 0; j < C; ++j) {
51             if (j != i && !overlap(c[j], c[i]) && !
52                 overlap(c[i], c[j]) && intersect(c[i],
53                 c[j])) {addEvent(c[i], c[j], evt, cnt);}
54         }
55         if (evt.size() == 0u) { add(cnt, PI * c[i].r *
56             c[i].r, c[i].o);}
57         else {
58             sort(evt.begin(), evt.end());
59             evt.push_back(evt.front());
60             for (int j = 0; j + 1 < (int)evt.size();
61                 ++j) {
62                 cnt += evt[j].delta;
63                 add(cnt, det(evt[j].p, evt[j + 1].p) /
64                     2, (evt[j].p + evt[j + 1].p) / 3);
65             }
66             double ang = evt[j + 1].ang - evt[j].
67                 ang;
68             if (ang < 0) { ang += PI * 2;}
69             if (sign(ang) == 0) continue;
70             add(cnt, ang * c[i].r * c[i].r / 2, c[
71                 i].o +
72                 Point(sin(ang1) - sin(ang0), -cos(
73                     ang1) + cos(ang0)) * (2 / (3 *
74                         ang) * c[i].r));
75             add(cnt, -sin(ang) * c[i].r * c[i].r /
76                 2, (c[i].o + evt[j].p + evt[j +
77                     1].p) / 3);

```



```

51     }
52 }
53 }
54 for (int i = 1; i <= C; ++i) if (sign(area[i])) {
55     centroid[i] = centroid[i] / area[i];
56 }

```

2.4 cirque-area-merge

```

1 //n^2*logn
2 struct point {
3     point rotate(const double &ang) {return point(cos(
4         ang) * x - sin(ang) * y, cos(ang) * y + sin(
5         ang) * x);}
6     double ang() {return atan2(y, x);}
7 };
8 struct Circle {
9     point o; double r;
10    int tp; // 正圆为1 反向圆为-1
11    Circle (point o = point(0, 0), double r = 0, int
12        tp = 0) : o(o), r(r), tp(tp) {}
13 };
14 struct Event {
15     point p; double ang; int delta;
16     Event (point p = point(0, 0), double ang = 0,
17         double delta = 0) : p(p), ang(ang), delta(
18         delta) {}
19 };
20 bool operator < (const Event &a, const Event &b) {
21     return a.ang < b.ang;}
22 void addEvent(const Circle &a, const Circle &b, vector
23 <Event> &evt, int &cnt) {
24     double d2 = (a.o - b.o).len2(),
25     dRatio = ((a.r - b.r) * (a.r + b.r) / d2 + 1)
26     / 2,
27     pRatio = sqrt(-(d2 - sqr(a.r - b.r)) * (d2 -
28     sqr(a.r + b.r))) / d2 / 2;
29     point d = b.o - a.o, p = d.rotate(PI / 2),
30     q0 = a.o + d * dRatio + p * pRatio,
31     q1 = a.o + d * dRatio - p * pRatio;
32     double ang0 = (q0 - a.o).ang(),
33     ang1 = (q1 - a.o).ang();
34     evt.push_back(Event(q1, ang1, b.tp));
35     evt.push_back(Event(q0, ang0, -b.tp));
36     cnt += (ang1 > ang0) * b.tp;
37 }
38 bool issame(const Circle &a, const Circle &b) {return
39 sign((a.o - b.o).len()) == 0 && sign(a.r - b.r) ==
40 0;}
41 bool overlap(const Circle &a, const Circle &b) {return
42 sign(a.r - b.r - (a.o - b.o).len()) >= 0;}
43 bool intersect(const Circle &a, const Circle &b) {
44     return sign((a.o - b.o).len() - a.r - b.r) < 0;}
45 int C;
46 Circle c[N];
47 double area[N];
48 void solve() { // area[1]..area[C]
49     memset(area, 0, sizeof(double) * (C + 1));
50     for (int i = 0; i < C; ++i) {
51         int cnt = (c[i].tp > 0);
52         vector<Event> evt;
53         for (int j = 0; j < i; ++j) if (issame(c[i], c
54 [j])) cnt += c[j].tp;
55         for (int j = 0; j < C; ++j)
56             if (j != i && !issame(c[i], c[j]) &&
57                 overlap(c[j], c[i])) cnt += c[j].tp;
58         for (int j = 0; j < C; ++j)
59             if (j != i && !overlap(c[j], c[i]) && !
60                 overlap(c[i], c[j]) && intersect(c[i],
61                 c[j]))
62                 addEvent(c[i], c[j], evt, cnt);
63         if (evt.size() == 0) area[cnt] += c[i].tp * PI
64             * c[i].r * c[i].r;
65         else {
66             sort(evt.begin(), evt.end());
67             evt.push_back(evt.front());
68             for (int j = 0; j + 1 < (int)evt.size();
69                 ++j) {
70                 cnt += evt[j].delta;
71                 area[cnt] += c[i].tp * det(evt[j].p,
72                     evt[j + 1].p) / 2;
73                 double ang = evt[j + 1].ang - evt[j].
74                     ang;
75                 if (ang < 0) ang += PI * 2;
76                 area[cnt] += c[i].tp * (ang * c[i].r *
77                     c[i].r / 2 - sin(ang) * c[i].r *
78                     c[i].r / 2);
79             }
80         }
81     }
82 }

```

2.5 凸包

```

1 // 凸包中的点按逆时针方向
2 struct Convex {
3     int n;
4     std::vector<Point> a, upper, lower;
5     void make_shell(const std::vector<Point> &p,
6         std::vector<Point> &shell) { // p needs
7         to be sorted.

```

```

7         clear(shell); int n = p.size();
8         for (int i = 0, j = 0; i < n; i++, j++) {
9             for (; j >= 2 && sign(det(shell[j-1] -
10                 shell[j-2],
11                 p[i] - shell[j-2])) <= 0;
12                 --j) shell.pop_back();
13             shell.push_back(p[i]);
14         }
15     }
16     void make_convex() {
17         std::sort(a.begin(), a.end());
18         make_shell(a, lower);
19         std::reverse(a.begin(), a.end());
20         make_shell(a, upper);
21         a = lower; a.pop_back();
22         a.insert(a.end(), upper.begin(), upper.end());
23         if ((int)a.size() >= 2) a.pop_back();
24         n = a.size();
25     }
26     void init(const std::vector<Point> &a) {clear(a);
27         a = _a; n = a.size(); make_convex();}
28     void read(int _n) { // Won't make convex.
29         clear(a); n = _n; a.resize(n);
30         for (int i = 0; i < n; i++) a[i].read();
31     }
32     std::pair<DB, int> get_tangent(
33         const std::vector<Point> &convex, const
34         Point &vec) {
35         int l = 0, r = (int)convex.size() - 2;
36         assert(r >= 0);
37         for (; l + 1 < r; ) {
38             int mid = (l + r) / 2;
39             if (sign(det(convex[mid + 1] - convex[mid],
40                 vec)) > 0) r = mid;
41             else l = mid;
42         }
43         return std::max(std::make_pair(det(vec, convex
44             [r]), r),
45             std::make_pair(det(vec, convex[0]), 0)
46             );
47     }
48     int binary_search(Point u, Point v, int l, int r)
49     {
50         int s1 = sign(det(v - u, a[l % n] - u));
51         for (; l + 1 < r; ) {
52             int mid = (l + r) / 2;
53             int smid = sign(det(v - u, a[mid % n] - u)
54             );
55             if (smid == s1) l = mid;
56             else r = mid;
57         }
58         return l % n;
59     }
60     // 求凸包上和向量 vec 叉积最大的点, 返回编号, 共线
61     // 的多个切点返回任意一个
62     int get_tangent(Point vec) {
63         std::pair<DB, int> ret = get_tangent(upper,
64             vec);
65         ret.second = (ret.second + (int)lower.size() -
66             1) % n;
67         ret = std::max(ret, get_tangent(lower, vec));
68         return ret.second;
69     }
70     // 求凸包和直线 u, v 的交点, 如果不相交返回 false
71     // , 如果有则是和 (i, next(i)) 的交点, 交在点上不
72     // 确定返回前后两条边其中之一
73     bool get_intersection(Point u, Point v, int &i0,
74         int &i1) {
75         int p0 = get_tangent(u - v), p1 = get_tangent(
76             v - u);
77         if (sign(det(v - u, a[p0] - u)) * sign(det(v -
78             u, a[p1] - u)) <= 0) {
79             if (p0 > p1) std::swap(p0, p1);
80             i0 = binary_search(u, v, p0, p1);
81             i1 = binary_search(u, v, p1, p0 + n);
82             return true;
83         }
84         else return false;
85     }
86 }

```

2.6 三角形内心, 外心, 垂心

```

1 Point inCenter(const Point &A, const Point &B, const
2     Point &C) { // 内心
3     double a = (B - C).len(), b = (C - A).len(), c = (
4         A - B).len(),
5     s = fabs(det(B - A, C - A)),
6     r = s / p;
7     return (A * a + B * b + C * c) / (a + b + c);
8 }
9 Point circumCenter(const Point &a, const Point &b,
10     const Point &c) { // 外心
11     Point bb = b - a, cc = c - a;
12     double db = bb.len2(), dc = cc.len2(), d = 2 * det
13         (bb, cc);
14     return a - Point(bb.y * dc - cc.y * db, cc.x * db
15         - bb.x * dc) / d;
16 }
17 Point orthoCenter(const Point &a, const Point &b,

```

```

13     const Point &c) { // 垂心
14     Point ba = b - a, ca = c - a, bc = b - c;
15     double Y = ba.y * ca.y * bc.y,
16           A = ca.x * ba.y - ba.x * ca.y,
17           x0 = (Y + ca.x * ba.y * b.x - ba.x * ca.y *
18             c.x) / A,
19           y0 = -ba.x * (x0 - c.x) / ba.y + ca.y;
20     return Point(x0, y0);
21 }

```

2.7 三维计算几何

```

1 // 三维绕轴旋转, 大拇指指向 axis 向量方向, 四指弯曲方
2 // 向转 w 弧度
3 Point rotate(const Point& s, const Point& axis, DB w)
4 {
5     DB x = axis.x, y = axis.y, z = axis.z;
6     DB s1 = x * x + y * y + z * z, ss1 = msqrt(s1),
7       cosw = cos(w), sinw = sin(w);
8     DB a[4][4];
9     memset(a, 0, sizeof a);
10    a[3][3] = 1;
11    a[0][0] = ((y * y + z * z) * cosw + x * x) / s1;
12    a[0][1] = x * y * (1 - cosw) / s1 + z * sinw / ss1;
13    a[0][2] = x * z * (1 - cosw) / s1 - y * sinw / ss1;
14    a[1][0] = x * y * (1 - cosw) / s1 - z * sinw / ss1;
15    a[1][1] = ((x * x + z * z) * cosw + y * y) / s1;
16    a[1][2] = y * z * (1 - cosw) / s1 + x * sinw / ss1;
17    a[2][0] = x * z * (1 - cosw) / s1 + y * sinw / ss1;
18    a[2][1] = y * z * (1 - cosw) / s1 - x * sinw / ss1;
19    a[2][2] = ((x * x + y * y) * cosw + z * z) / s1;
20    DB ans[4] = {0, 0, 0, 0}, c[4] = {s.x, s.y, s.z,
21      1};
22    for (int i = 0; i < 4; ++i)
23      for (int j = 0; j < 4; ++j)
24        ans[i] += a[j][i] * c[j];
25    return Point(ans[0], ans[1], ans[2]);
26 }

```

2.8 三维凸包

```

1 __inline P cross(const P& a, const P& b) { return P(a.
2   y * b.z - a.z * b.y, a.z * b.x - a.x * b.z, a.x
3   * b.y - a.y * b.x);}
4 __inline DB mix(const P& a, const P& b, const P& c) {
5   return dot(cross(a, b), c);}
6 __inline DB volume(const P& a, const P& b, const P& c,
7   const P& d) { return mix(b - a, c - a, d - a);}
8 struct Face {
9   int a, b, c;
10  __inline Face() {}
11  __inline Face(int _a, int _b, int _c):a(_a), b(_b),
12    c(_c) {}
13  __inline DB area() const { return 0.5 * cross(p[b]
14    - p[a], p[c] - p[a]).len();}
15  __inline P normal() const { return cross(p[b] - p[
16    a], p[c] - p[a]).unit();}
17  __inline DB dis(const P& p0) const {return dot(
18    normal(), p0 - p[a]);}
19 };
20 std::vector<Face> face, tmp; // Should be O(n).
21 int mark[N][N], Time, n;
22 __inline void add(int v) {
23   ++ Time;
24   clear(tmp);
25   for (int i = 0; i < (int)face.size(); ++ i) {
26     int a = face[i].a, b = face[i].b, c = face[i].
27     c;
28     if (sign(volume(p[v], p[a], p[b], p[c])) > 0)
29       mark[a][b] = mark[b][a] = mark[a][c] =
30       mark[c][a] = mark[b][c] = mark[c][b] =
31       Time;
32     else {tmp.push_back(face[i]);}
33   }
34   clear(face); face = tmp;
35   for (int i = 0; i < (int)tmp.size(); ++ i) {
36     int a = face[i].a, b = face[i].b, c = face[i].
37     c;
38     if (mark[a][b] == Time) face.emplace_back(v, b
39       , a);
40     if (mark[b][c] == Time) face.emplace_back(v, c
41       , b);
42     if (mark[c][a] == Time) face.emplace_back(v, a
43       , c);
44     assert(face.size() < 500u);
45   }
46 }
47 void reorder() {
48   for (int i = 2; i < n; ++ i) {
49     P tmp = cross(p[i] - p[0], p[i] - p[1]);
50     if (sign(tmp.len()) > 0) {
51       std::swap(p[i], p[2]);
52       for (int j = 3; j < n; ++ j)

```

```

39     if (sign(volume(p[0], p[1], p[2], p[j]
40       ))) {
41       std::swap(p[j], p[3]); return;
42     }
43   }
44 }
45 void build_convex() {
46   reorder(); clear(face);
47   face.emplace_back(0, 1, 2);
48   face.emplace_back(0, 2, 1);
49   for (int i = 3; i < n; ++ i)add(i);
50 }

```

3 数据结构

3.1 KD 树-wrx

```

1 // 这里写的是询问三维空间中的一个长方体内有没有点
2 int D;
3 struct point
4 {
5     int d[3];
6     bool operator < (const point &that) const {return
7       d[D] < that.d[D];}
8 }p[N];
9 struct node
10 {
11     int d[2][3]; // 0 min 1 max
12 }t[N<<2];
13 void pushup(int x)
14 {
15     for(int i = 0; i < 3; i++)
16     {
17         t[x].d[0][i] = min(t[x<<1].d[0][i], t[x<<1|1].
18           d[0][i]);
19         t[x].d[1][i] = max(t[x<<1].d[1][i], t[x<<1|1].
20           d[1][i]);
21     }
22 }
23 void build(int x, int l, int r, int d)
24 {
25     if(l == r)
26     {
27         for(int i = 0; i < 3; i++)
28             t[x].d[0][i] = t[x].d[1][i] = p[l].d[i];
29         return;
30     }
31     D = d; int mid = (l+r) >> 1;
32     nth_element(p+l, p+mid, p+r+1);
33     (++d) %= 3;
34     build(x<<1, l, mid, d);
35     build(x<<1|1, mid+1, r, d);
36     pushup(x);
37 }
38 int d[2][3];
39 bool query(int x, int l, int r)
40 {
41     int in = 1, out = 0;
42     for(int i = 0; i < 3; i++)
43     {
44         if(!(d[0][i] <= t[x].d[0][i] && t[x].d[1][i]
45           <= d[1][i])) in = 0; // 包含
46         if(d[1][i] < t[x].d[0][i] || t[x].d[1][i] < d
47           [0][i]) out = 1; // 不交
48     }
49     if(in) return true; if(out) return false;
50     int mid = (l+r) >> 1;
51     return query(x<<1, l, mid) || query(x<<1|1, mid+1,
52       r);
53 }

```

3.2 KD 树-gwx

```

1 struct Point
2 {
3     double x, y;
4     int id;
5     Point operator - (const Point &a) const {
6       return (Point){x - a.x, y - a.y, id};
7     }
8 } b[maxn], c[maxn];
9 struct node
10 {
11     Point p;
12     int ch[2];
13 } a[maxn];
14 struct rev
15 {
16     int id;
17     double dis;
18     bool operator < (const rev &a) const{
19       int tmp = sign(dis - a.dis);
20       if(tmp)
21         return tmp < 0;
22       return id < a.id;
23     }
24 };
25 typedef pr priority_queue <rev>;
26 pr p0;

```



```

27 int build(int l, int r, int f)
28 {
29     if(l > r)
30         return 0;
31     int x = (l + r) >> 1;
32     if(f == 0)
33         nth_element(a + l, a + x, a + r + 1, cmp0); // 按x排序
34     else
35         nth_element(a + l, a + x, a + r + 1, Cmp1); // 按y排序
36     a[x].ch[0] = build(l, x - 1, f ^ 1);
37     a[x].ch[1] = build(x + 1, r, f ^ 1);
38     return x;
39 }
40 void update(pr &a, rev x)
41 {
42     if(a.size() < K)
43         a.push(x);
44     else if(x < a.top())
45     {
46         a.pop();
47         a.push(x);
48     }
49 }
50 pr merge(pr a, pr b)
51 {
52     int s1 = a.size(), s2 = b.size();
53     if(s1 < s2)
54     {
55         while(!a.empty())
56         {
57             update(b, a.top());
58             a.pop();
59         }
60         return b;
61     }
62     else
63     {
64         while(!b.empty())
65         {
66             update(a, b.top());
67             b.pop();
68         }
69         return a;
70     }
71 }
72 pr query(int u, Point x, int f)
73 {
74     if(!u)
75         return p0; // empty priority_queue
76     int d = (dis(a[u].ch[0].p, x) > dis(a[u].ch[1].p, x));
77     double dx;
78     pr res = query(a[u].ch[d], x, f ^ 1);
79     update(res, (rev){a[u].p.id, dis(a[u].p, x)});
80     if(f == 0)
81         dx = abs(x.x - a[u].p.x);
82     else
83         dx = abs(x.y - a[u].p.y);
84     if(dx > res.top().dis)
85         return res;
86     res = merge(res, query(a[u].ch[d ^ 1], x, f ^ 1));
87     return res;
88 }
89 pr solve(Point p) // 离p最近的K个点
90 {
91     int root = build(1, tot, 0);
92     return query(root, p, 0);
93 }
94 }

```

```

23 if(x->rev)
24 {
25     x->rev = 0, x->ch[0]->rev ^= 1, x->ch[1]->rev ^= 1;
26     mswap(x->ch[0]->ch[0], x->ch[0]->ch[1]);
27     mswap(x->ch[1]->ch[0], x->ch[1]->ch[1]);
28 }
29 for(int i = 0; i <= 1; i++)
30 {
31     x->ch[i]->v = (x->k * x->ch[i]->v % MOD + x->b) % MOD;
32     x->ch[i]->sum = (x->ch[i]->sum * x->k % MOD + x->b * x->ch[i]->siz % MOD) % MOD;
33     (x->ch[i]->k *= x->k) %= MOD;
34     (x->ch[i]->b *= x->k) %= MOD, (x->ch[i]->b += x->b) %= MOD;
35 }
36 x->k = 1; x->b = 0;
37 }
38 void update(node *x){if(!isroot(x))update(x->fa);
39     pushdown(x);}
40 void rotate(node *x)
41 {
42     node *f = x->fa; int d = type(x);
43     x->fa = f->fa, !isroot(f) ? x->fa->ch[type(f)] = x : 0;
44     (f->ch[d] = x->ch[d^1]) != null ? f->ch[d]->fa = f : 0;
45     f->fa = x, x->ch[d^1] = f; pushup(f);
46 }
47 void splay(node *x)
48 {
49     update(x);
50     for(; !isroot(x); )
51     {
52         if(isroot(x->fa)) rotate(x);
53         else if(type(x) == type(x->fa)) rotate(x->fa), rotate(x);
54         else rotate(x), rotate(x);
55     }
56     pushup(x);
57 }
58 void access(node *x)
59 {
60     node *tmp = null;
61     for(; x != null; )
62     {
63         splay(x);
64         x->ch[1] = tmp;
65         pushup(x);
66         tmp = x;
67         x = x->fa;
68     }
69 }
70 void makeroot(node *x)
71 {
72     access(x);
73     splay(x);
74     x->rev ^= 1;
75     swap(x->ch[0], x->ch[1]);
76 }
77 void link(node *x, node *y)
78 {
79     makeroot(x);
80     x->fa = y;
81 }
82 void cut(node *x, node *y)
83 {
84     makeroot(x); access(y);
85     splay(y); y->ch[0] = x->fa = null;
86     pushup(y);
87 }

```

3.3 LCT-wrz

```

1 struct node
2 {
3     node *ch[2], *fa;
4     uint v, sum, k, b; int rev, siz;
5 }mem[N], *tot, *null, *pos[N];
6 void init()
7 {
8     null = tot = mem;
9     null->ch[0] = null->ch[1] = null->fa = null;
10    null->v = null->sum = null->b = null->rev = null->siz = 0; null->k = 1;
11    for(int i = 1; i <= n; i++) pos[i] = ++tot, *pos[i] = *null, pos[i]->v = pos[i]->sum = 1;
12 }
13 int type(node *x){return x->fa->ch[1]==x?1:0;}
14 int isroot(node *x){return x->fa->ch[type(x)] != x;}
15 void mswap(node &x, node &y){node *t = x; x = y; y = t;}
16 void pushup(node *x)
17 {
18     x->sum = (x->v + x->ch[0]->sum + x->ch[1]->sum) % MOD;
19     x->siz = (x->ch[0]->siz + x->ch[1]->siz + 1) % MOD;
20 }
21 void pushdown(node *x)
22 {

```

3.4 左偏树-wrz

```

1 struct heap
2 {
3     heap *ch[2];
4     int dis, siz, v;
5 }mem[N*2], *h[N], *null, *tot;
6 heap* newheap()
7 {
8     heap *p = ++tot;
9     *p = *null;
10    return p;
11 }
12 void init()
13 {
14     null = tot = mem;
15     null->ch[0] = null->ch[1] = null;
16     null->v = null->dis = null->siz = 0;
17     for(int i = 1; i <= n; i++) h[i] = null;
18 }
19 heap* merge(heap *x, heap *y) // big
20 {
21     if(x == null) return y;
22     if(y == null) return x;
23     if(x->v < y->v) swap(x, y);
24     x->ch[1] = merge(x->ch[1], y);
25     if(x->ch[0]->dis < x->ch[1]->dis) swap(x->ch[0], x->ch[1]);

```

```

26     x->dis = x->ch[1]->dis + 1;
27     x->siz = x->ch[0]->siz + x->ch[1]->siz + 1;
28     return x;
29 }
30 heap *pop(heap *x){return merge(x->ch[0], x->ch[1]);}
31 int main()
32 {
33     init();
34     heap *a = newheap(); a->siz = 1; a->v = 233;
35     heap *b = newheap(); b->siz = 1; b->v = 233;
36     heap *c = merge(a, b);
37 }

```

3.5 splay-wrz

```

1 struct node
2 {
3     node *ch[2], *fa;
4     ll key; int siz, tag;
5 }mem[N*20], *tot, *null, *root;
6 void init()
7 {
8     root = null = tot = mem;
9     null->ch[0] = null->ch[1] = null->fa = null;
10    null->key = null->siz = null->tag = 0;
11 }
12 int type(node *x){return x->fa->ch[1]==x;}
13 node *newnode(ll key)
14 {
15     node *p = ++tot; *p = *null;
16     p->key = key; p->siz = 1;
17     return p;
18 }
19 void pushup(node *x)
20 {
21     x->siz = x->ch[0]->siz + x->ch[1]->siz + 1;
22 }
23 void rotate(node *x)
24 {
25     node *f = x->fa; int d = type(x);
26     (x->fa = f->fa) != null ? x->fa->ch[type(f)] = x :
27     0;
28     (f->ch[d] = x->ch[!d]) != null ? f->ch[d]->fa = f :
29     0;
30     x->ch[!d] = f, f->fa = x, pushup(f);
31 }
32 void pushdown(node *x)
33 {
34     if(x->tag)
35     {
36         int &tag = x->tag;
37         if(x->ch[0] != null) x->ch[0]->key += tag, x->
38         ch[0]->tag += tag;
39         if(x->ch[1] != null) x->ch[1]->key += tag, x->
40         ch[1]->tag += tag;
41         tag = 0;
42     }
43 }
44 void update(node *x)
45 {
46     if(x==null) return;
47     update(x->fa);
48     pushdown(x);
49 }
50 void splay(node *x, node *top)
51 {
52     update(x);
53     for(;x->fa!=top;){
54         if(x->fa->fa == top) rotate(x);
55         else if(type(x) == type(x->fa)) rotate(x->fa),
56             rotate(x);
57         else rotate(x), rotate(x);
58     }
59     if(top == null) root = x;
60     pushup(x);
61 }
62 void insert(node *x, node *f, node *p, int d)
63 {
64     if(x == null)
65     {
66         p->fa = f, f->ch[d] = p;
67         return;
68     }
69     pushdown(x);
70     if(p->key < x->key) insert(x->ch[0], x, p, 0);
71     else insert(x->ch[1], x, p, 1);
72     pushup(x);
73 }
74 void insert(node *p)
75 {
76     if(root == null) root = p, p->fa = p->ch[0] = p->
77     ch[1] = null;
78     else insert(root, null, p, 0), splay(p, null);
79 }
80 node *findl(node *x){return x->ch[0]==null?x:findl(x->
81 ch[0]);}
82 node *findr(node *x){return x->ch[1]==null?x:findr(x->
83 ch[1]);}
84 void insertlr()

```

```

79 {
80     insert(newnode(-INF));
81     insert(newnode(INF));
82 }
83 void delet(node *p)
84 {
85     splay(p, null);
86     node *lp = findr(p->ch[0]), *rp = findl(p->ch[1]);
87     if(lp == null && rp != null) root = p->ch[1], root
88     ->fa = null;
89     else if(lp != null && rp == null) root = p->ch[0],
90     root->fa = null;
91     else if(lp == null && rp == null) root = null;
92     else
93     {
94         splay(rp, null); splay(lp, rp);
95         lp->ch[1] = null; splay(lp, null);
96     }
97 }
98 node* findk(node *p, int k)
99 {
100    for(;;)
101    {
102        pushdown(p);
103        if(p->ch[0]->siz >= k) p = p->ch[0];
104        else if(p->ch[0]->siz + 1 == k) {splay(p, null
105        ); return p;}
106        else k -= p->ch[0]->siz + 1, p = p->ch[1];
107    }
108 }
109 node* findv(node *p, int v)
110 {
111     node* ret = null;
112     for(;; p!=null;){
113         pushdown(p);
114         if(p->key >= v) ret = p, p = p->ch[0];
115         else p = p->ch[1];
116     }
117     splay(ret, null);
118     return ret;
119 }
120 void addv(node *p, int v)
121 {
122     if(p == null) return;
123     p->key += v;
124     p->tag += v;
125 }

```

3.6 treap-gwx

```

1 //srand()
2 struct node
3 {
4     int pri, val, c, s; //pri: random value; c:
5     times of showing; s: size of subtree
6     int ch[2];
7     int cmp(int x) const {
8         if(x == val) return -1;
9         return x < val ? 0 : 1;
10    }
11 } a[maxn];
12 void maintain(int u) {
13     a[u].s = a[u].c + a[a[u].ch[0]].s + a[a[u].ch[1]].
14     s;
15 }
16 void rotate(int &u, int d)
17 {
18     int tmp = a[u].ch[d ^ 1];
19     a[u].ch[d ^ 1] = a[tmp].ch[d];
20     a[tmp].ch[d] = u;
21     maintain(u); maintain(tmp);
22     u = tmp;
23 }
24 void insert(int &u, int val)
25 {
26     if(!u)
27     {
28         u = ++cnt;
29         a[cnt] = (node){rand(), val, 1, 1};
30         return;
31     }
32     a[u].s++;
33     int d = a[u].cmp(val);
34     if(d == -1) {a[u].c++; return;}
35     insert(a[u].ch[d], val);
36     if(a[a[u].ch[d]].pri > a[u].pri) rotate(u, d ^ 1);
37 }
38 int find(int u, int val, int comp, int &res)
39 {
40     int d = a[u].cmp(val);
41     if(!u) return -1;
42     if(d == -1) return u;
43     if(d == comp)
44     {
45         if(d) res = max(res, a[u].val);
46         else res = min(res, a[u].val);
47     }
48     return find(a[u].ch[d], val, comp, res);
49 }
50 void remove(int &u)

```


有源汇的上下界最小流

1. 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。
2. 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上汇点 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_* 发出的边全部满流, 则 $T \rightarrow S$ 边上的流量即为原图的最小流, 否则无解。

4.4 矩阵树定理

1 C = 度数矩阵 - 邻接矩阵
2 无向图 G 的生成树个数 = C 的任意 $n - 1$ 阶主子式 (对角线的乘积)

4.5 边双联通-gwx

```
1 //G[i]: 第i个边双联通分量中有哪些点
2 void tarjan(int u, int pa)
3 {
4     d[u] = l[u] = ++timer;
5     for(int i = tail[u]; i; i = e[i].next)
6     {
7         if(!d[e[i].v])
8         {
9             st[++top] = i;
10            tarjan(e[i].v, u);
11            l[u] = min(l[u], l[e[i].v]);
12            if(l[e[i].v] >= d[u])
13            {
14                bcc++;
15                while(true)
16                {
17                    int now = st[top--];
18                    if(vst[e[now].u] != bcc)
19                    {
20                        vst[e[now].u] = bcc;
21                        G[bcc].push_back(e[now].u);
22                    }
23                    if(vst[e[now].v] != bcc)
24                    {
25                        vst[e[now].v] = bcc;
26                        G[bcc].push_back(e[now].v);
27                    }
28                    if(now == i) break;
29                }
30            }
31        }
32        else if(e[i].v != pa)
33            l[u] = min(l[u], d[e[i].v]);
34    }
35 }
```

4.6 带花树

```
1 vector<int> link[maxn];
2 int n, match[maxn], Queue[maxn], head, tail;
3 int pred[maxn], base[maxn], start, finish, newbase;
4 bool InQueue[maxn], InBlossom[maxn];
5 void push(int u){ Queue[tail++] = u; InQueue[u] = true; }
6 int pop(){ return Queue[head++]; }
7 int FindCommonAncestor(int u, int v){
8     bool InPath[maxn];
9     for(int i = 0; i < n; i++) InPath[i] = 0;
10    while(true){ u = base[u]; InPath[u] = true; if(u == start) break; u = pred[match[u]]; }
11    while(true){ v = base[v]; if(InPath[v]) break; v = pred[match[v]]; }
12    return v;
13 }
14 void ResetTrace(int u){
15     int v;
16     while(base[u] != newbase){
17         v = match[u];
18         InBlossom[base[u]] = InBlossom[base[v]] = true;
19         u = pred[v];
20         if(base[u] != newbase) pred[u] = v;
21     }
22 }
23 void BlossomContract(int u, int v){
24     newbase = FindCommonAncestor(u, v);
25     for(int i = 0; i < n; i++)
26         InBlossom[i] = 0;
27     ResetTrace(u); ResetTrace(v);
28     if(base[u] != newbase) pred[u] = v;
29     if(base[v] != newbase) pred[v] = u;
30     for(int i = 0; i < n; i++)
31         if(InBlossom[base[i]]){
32             base[i] = newbase;
33             if(!InQueue[i]) push(i);
34         }
35 }
36 bool FindAugmentingPath(int u){
37     bool found = false;
38     for(int i = 0; i < n; i++) pred[i] = -1, base[i] = i;
39     for(int i = 0; i < n; i++) InQueue[i] = 0;
40     start = u; finish = -1; head = tail = 0; push(start);
```

```
41 while(head < tail){
42     int u = pop();
43     for(int i = link[u].size() - 1; i >= 0; i--){
44         int v = link[u][i];
45         if(base[u] != base[v] && match[u] != v)
46             if(v == start || (match[v] >= 0 && pred[match[v]] >= 0))
47                 BlossomContract(u, v);
48         else if(pred[v] == -1){
49             pred[v] = u;
50             if(match[v] >= 0) push(match[v]);
51             else{ finish = v; return true; }
52         }
53     }
54 }
55 return found;
56 }
57 void AugmentPath(){
58     int u = finish, v, w;
59     while(u >= 0){ v = pred[u]; w = match[v]; match[v] = u; match[u] = v; u = w; }
60 }
61 void FindMaxMatching(){
62     for(int i = 0; i < n; i++) match[i] = -1;
63     for(int i = 0; i < n; i++) if(match[i] == -1) if(FindAugmentingPath(i)) AugmentPath();
64 }
```

4.7 弦图

定义 我们称连接环中不相邻的两个点的边为弦。一个无向图称为弦图, 当图中任意长度都大于 3 的环都至少有一个弦。弦图的每一个诱导子图一定是弦图。

单纯点 设 $N(v)$ 表示与点 v 相邻的点集。一个点称为单纯点当 $v + N(v)$ 的诱导子图为一个团。引理: 任何一个弦图都至少有一个单纯点, 不是完全图的弦图至少有两个不相邻的单纯点。

完美消除序列 一个序列 v_1, v_2, \dots, v_n 满足 v_i 在 v_i, v_{i+1}, \dots, v_n 的诱导子图中为一个单纯点。一个无向图是弦图当且仅当它有一个完美消除序列。

最大势算法 最大势算法能判断一个图是否是弦图。从 n 到 1 的顺序依次给点标号为 i 的点出现在完美消除序列的第 i 个。设 $label_i$ 表示第 i 个点与多少个已标号的点相邻, 每次选择 $label_i$ 最大的未标号的点进行标号。然后判断这个序列是否为完美序列。如果依次判断 v_i 在 v_{i+1}, \dots, v_n 中所有与 v_i 相邻的点是否构成一个团, 时间复杂度为 $O(nm)$ 。考虑优化, 设 v_{i+1}, \dots, v_n 中所有与 v_i 相邻的点依次为 v_{j_1}, \dots, v_{j_k} 。只需判断 v_{j_1} 是否与 v_{j_2}, \dots, v_{j_k} 相邻即可。时间复杂度 $O(n + m)$ 。

弦图的染色 按照完美消除序列中的点倒着给图中的点贪心染尽可能最小的颜色, 这样一定能用最少的颜色数给图中所有点染色。弦图的团数 = 染色数。

最大独立集 完美消除序列从前往后能选就选。最大独立集 = 最小团覆盖。

```
1 template <int MAXN = 100000, int MAXM = 100000>
2 struct chordal_graph {
3     int n; edge_list <MAXN, MAXM> e;
4     int id[MAXN], seq[MAXN];
5     void init () {
6         struct point {
7             int lab, u;
8             point (int lab = 0, int u = 0) : lab (lab), u (u) {}
9         };
10        bool operator < (const point &a) const { return lab < a.lab; } };
11        std::fill (id, id + n, -1);
12        static int label[MAXN]; std::fill (label, label + n, 0);
13        std::priority_queue <point> q;
14        for (int i = 0; i < n; ++i) q.push (point (0, i));
15        for (int i = n - 1; i >= 0; --i) {
16            for (; ~id[q.top().u]; ) q.pop ();
17            int u = q.top().u; q.pop (); id[u] = i;
18            for (int j = e.begin[u], v; ~j; j = e.next[j])
19                if (v = e.dest[j], !id[v]) ++label[v], q.push (point (label[v], v));
20            for (int i = 0; i < n; ++i) seq[id[i]] = i; }
21        bool is_chordal () {
22            static int vis[MAXN], q[MAXN]; std::fill (vis, vis + n, -1);
23            for (int i = n - 1; i >= 0; --i) {
24                int u = seq[i], t = 0, v;
25                for (int j = e.begin[u]; ~j; j = e.next[j])
26                    if (v = e.dest[j], id[v] > id[u]) q[t++] = v;
27                if (!t) continue; int w = q[0];
28                for (int j = 0; j < t; ++j) if (id[q[j]] < id[w]) w = q[j];
29                for (int j = e.begin[w]; ~j; j = e.next[j]) vis[e.dest[j]] = i;
30                for (int j = 0; j < t; ++j) if (q[j] != w && vis[q[j]] != i) return 0;
31            }
32            return 1; }
33        int min_color () {
34            int res = 0;
35            static int vis[MAXN], c[MAXN];
```



```

35     std::fill (vis, vis + n, -1);
36     std::fill (c, c + n, n);
37     for (int i = n - 1; i >= 0; --i) {
38         int u = seq[i];
39         for (int j = e.begin[u]; ~j; j = e.next[j])
40             vis[c[e.dest[j]]] = i;
41         int k; for (k = 0; vis[k] == i; ++k);
42         c[u] = k; res = std::max (res, k + 1);
43     }
44     return res; } };

```

4.8 支配树-gwx

```

1  /*
2   用ins()加边
3   build前设置n为点数，s为源点
4   树中的i号点对应原图的id[i]号点
5  */
6  struct Dominator_Tree {
7      int n, s, cnt;
8      int dfn[N], id[N], pa[N], semi[N], idom[N], p[N],
9          mn[N];
10     vector<int> e[N], dom[N], be[N];
11     void ins(int x, int y) {e[x].push_back(y);}
12     void dfs(int x) {
13         dfn[x] = ++cnt; id[cnt] = x;
14         for (int i : e[x]) {
15             if (!dfn[i]) dfs(i), pa[dfn[i]] = dfn[x];
16             be[dfn[i]].push_back(dfn[x]);
17         }
18     }
19     int get(int x) {
20         if (p[x] != p[p[x]]) {
21             if (semi[mn[x]] > semi[get(p[x])]) mn[x] =
22                 get(p[x]);
23             p[x] = p[p[x]];
24         }
25         return mn[x];
26     }
27     void LT() {
28         for (int i = cnt; i > 1; i--) {
29             for (int j : be[i]) semi[i] = min(semi[i],
30                 semi[get(j)]);
31             dom[semi[i]].push_back(i);
32             int x = p[i] = pa[i];
33             for (int j : dom[x]) idom[j] = (semi[get(j)]
34                 < x ? get(j) : x);
35             dom[x].clear();
36         }
37         for (int i = 2; i <= cnt; i++) {
38             if (idom[i] != semi[i]) idom[i] = idom[idom
39                 [i]];
40             dom[id[idom[i]]].push_back(id[i]);
41         }
42     }
43     void build() {
44         for (int i = 1; i <= n; ++i)
45             dfn[i] = 0, dom[i].clear(), be[i].clear(),
46             p[i] = mn[i] = semi[i] = i;
47         cnt = 0; dfs(s); LT();
48     }
49 };

```

4.9 欧拉回路-wrz

```

1  #include<cstdio>
2  #define N 100005
3  #define M 200005
4  using namespace std;
5  int last[N], ecnt = 1, cnt, ans[M], in_deg[N], out_deg
6  [N];
7  bool vis[M];
8  struct edge{int next,to;}e[M<<1];
9  void addedge(int a, int b)
10 {
11     e[++ecnt] = (edge){last[a], b};
12     last[a] = ecnt;
13 }
14 void dfs(int x)
15 {
16     for(int &i = last[x]; i; i = e[i].next)
17     {
18         int y = e[i].to, j = i;
19         if(!vis[j]>>1)
20         {
21             vis[j]>>1 = 1;
22             dfs(y);
23             ans[++cnt] = j;
24         }
25     }
26 }
27 int main()
28 {
29     int t, n, m, a, b;
30     scanf("%d%d%d",&t,&n,&m);
31     for(int i = 1; i <= m; i++)
32     {
33         scanf("%d%d",&a,&b);
34         addedge(a,b);

```

```

34         if(t == 1) addedge(b,a), in_deg[a]++, in_deg[b
35             ]++;
36         else ecnt++, in_deg[b]++, out_deg[a]++;
37     }
38     if(t == 1) // 无向
39     {
40         for(int i = 1; i <= n; i++)
41             if((in_deg[i]+out_deg[i]) & 1)
42                 return !printf("NO\n");
43     }
44     else // 有向
45     {
46         for(int i = 1; i <= n; i++)
47             if(in_deg[i] != out_deg[i])
48                 return !printf("NO\n");
49     }
50     dfs(a);
51     if(cnt != m)
52     {
53         puts("NO");
54     }
55     else
56     {
57         puts("YES");
58         for(int i = cnt; i; i--)
59         {
60             printf("%d",ans[i]&1?-(ans[i]>>1):(ans[i]
61                 >>1));
62         }
63     }

```

4.10 Hopcroft-Karp

```

1  // O(sqrt(n)m)
2  template <int MAXN = 100000, int MAXM = 100000>
3  struct hopcroft_karp {
4      int mx[MAXN], my[MAXM], lv[MAXN];
5      bool dfs (edge_list <MAXN, MAXM> &e, int x) {
6          for (int i = e.begin[x]; ~i; i = e.next[i]) {
7              int y = e.dest[i], w = my[y];
8              if (!w || (lv[x] + 1 == lv[w] && dfs (e,
9                  w))) {
10                 mx[x] = y; my[y] = x; return true; } }
11             lv[x] = -1; return false; }
12     int solve (edge_list <MAXN, MAXM> &e, int n, int m
13         ) {
14         std::fill (mx, mx + n, -1); std::fill (my, my
15             + m, -1);
16         for (int ans = 0; ; ) {
17             std::vector <int> q;
18             for (int i = 0; i < n; ++i)
19                 if (mx[i] == -1) {
20                     lv[i] = 0; q.push_back (i);
21                 } else lv[i] = -1;
22             for (int head = 0; head < (int) q.size();
23                 ++head) {
24                 int x = q[head];
25                 for (int i = e.begin[x]; ~i; i = e.
26                     next[i]) {
27                     int y = e.dest[i], w = my[y];
28                     if (~w && lv[w] < 0) { lv[w] = lv[
29                         x] + 1; q.push_back (w); } } }
30             int d = 0; for (int i = 0; i < n; ++i) if
31                 (!mx[i] && dfs (e, i)) ++d;
32             if (d == 0) return ans; else ans += d; } }

```

4.11 KM-truly-n3

```

1  struct KM {
2      // Truly O(n^3)
3      // 邻接矩阵，不能连的边设为 -INF，求最小权匹配时边
4      // 权取负，但不能连的还是 -INF，使用时先对 1 -> n
5      // 调用 hungary()，再 get_ans() 求值
6      int w[N][N];
7      int lx[N], ly[N], match[N], way[N], slack[N];
8      bool used[N];
9      void init() {
10         for (int i = 1; i <= n; i++) {
11             match[i] = 0;
12             lx[i] = 0;
13             ly[i] = 0;
14             way[i] = 0;
15         }
16     }
17     void hungary(int x) {
18         match[0] = x;
19         int j0 = 0;
20         for (int j = 0; j <= n; j++) {
21             slack[j] = INF;
22             used[j] = false;
23         }
24         do {
25             used[j0] = true;
26             int i0 = match[j0], delta = INF, j1 = 0;
27             for (int j = 1; j <= n; j++) {
28                 if (used[j] == false) {

```



```

28         int cur = -w[i0][j] - lx[i0] - ly[
29             j];
30         if (cur < slack[j]) {
31             slack[j] = cur;
32             way[j] = j0;
33         }
34         if (slack[j] < delta) {
35             delta = slack[j];
36             j1 = j;
37         }
38     }
39     for (int j = 0; j <= n; j++) {
40         if (used[j]) {
41             lx[match[j]] += delta;
42             ly[j] -= delta;
43         }
44         else slack[j] -= delta;
45     }
46     j0 = j1;
47     while (match[j0] != 0) {
48
49         do {
50             int j1 = way[j0];
51             match[j0] = match[j1];
52             j0 = j1;
53         } while (j0);
54     }
55
56     int get_ans() {
57         int sum = 0;
58         for(int i = 1; i <= n; i++) {
59             if (w[match[i]][i] == -INF); // 无解
60             if (match[i] > 0) sum += w[match[i]][i];
61         }
62         return sum;
63     }
64 } km;

```

4.12 k 短路 a 星-gwx

```

1 const int maxn = 1005;
2 int n, m;
3 int S, T, K;
4 int dist[maxn], cnt[maxn];
5 bool vst[maxn];
6 vector<pair<int, int>> G[maxn], H[maxn]; // 正图&反
7 // 图
8 struct node
9 {
10     ll d;
11     int id;
12     node(){}
13     node(ll d, int id): d(d), id(id) {}
14     bool operator< (const node &other) const{
15         return d + dist[id] > other.d + dist[other.id];
16     }
17 };
18 priority_queue<pair<ll, int>> q;
19 priority_queue<node> Q;
20
21 void init()
22 {
23     for(int i = 1; i <= n; ++i)
24         G[i].clear(), H[i].clear(), cnt[i] = 0;
25 }
26
27 void dijkstra(int S)
28 {
29     memset(dist, 127, sizeof(dist));
30     memset(vst, 0, sizeof(vst));
31     while(!q.empty()) q.pop();
32     dist[S] = 0;
33     q.push(make_pair(0, S));
34     for(int i = 1; i <= n; ++i)
35     {
36         if(q.empty()) break;
37         while(vst[q.top().second]) q.pop();
38         int u = q.top().second; q.pop();
39         vst[u] = 1;
40         for(auto i: H[u])
41         {
42             if(dist[i.first] > dist[u] + i.second)
43             {
44                 dist[i.first] = dist[u] + i.second;
45                 q.push(make_pair(-dist[i.first], i.first));
46             }
47         }
48     }
49 }
50
51 int solve()
52 {
53     while(!Q.empty()) Q.pop();
54     Q.push(node(0, S));
55     while(!Q.empty())
56     {
57         auto u = Q.top(); Q.pop();

```

```

58         if(++cnt[u.id] > K) continue;
59         if(u.d + dist[u.id] > ti) continue;
60         if(u.id == T && cnt[T] == K)
61             return u.d;
62         for(auto i: G[u.id])
63             Q.push(node(u.d + i.second, i.first));
64     }
65     return -1;
66 }

```

4.13 K 短路可并堆

```

1 //Kth Shortest Path via Persistable Mergeable Heap
2 //可持久化可并堆求k短路 O(SSSP+(m+k)\log n)
3 //By ysf
4 //通过题目: USACO Mar08 牛跑步 (板子题)
5
6 //注意这是个多项式算法, 在k比较大时很有优势, 但k比较小时
7 //最好还是用A*
8 //DAG和有环的情况都可以, 有重边或自环也无所谓, 但不能
9 //有零环
10 //以下代码以Dijkstra+可持久化左偏树为例
11
12 const int maxn=1005, maxe=10005, maxm=maxe*30; //点数, 边
13 //数, 左偏树结点数
14
15 //需要用到的结构体定义
16 struct A{//用来求最短路
17     int x,d;
18     A(int x,int d):x(x),d(d){}
19     bool operator<(const A &a)const{return d>a.d;}
20 };
21
22 struct node{//左偏树结点
23     int w,i,d; //i: 最后一条边的编号 d: 左偏树附加信息
24     node *lc,*rc;
25     node(){}
26     node(int w,int i):w(w),i(i),d(0){}
27     void refresh(){d=rc->d+1;}
28 }null[maxn],*ptr=null,*root[maxn];
29
30 struct B{//维护答案用
31     int x,w; //x是结点编号, w表示之前已经产生的权值
32     node *rt; //这个答案对应的堆顶, 注意可能不等于任何
33     //一个结点的堆
34     B(int x,node *rt,int w):x(x),w(w),rt(rt){}
35     bool operator<(const B &a)const{return w+rt->w>a.w
36         +a.rt->w;}
37 };
38
39 //全局变量和数组定义
40 vector<int>G[maxn],W[maxn],id[maxn]; //最开始要存反向
41 //图, 然后把G清空作为儿子列表
42 bool vis[maxn],used[maxe]; //used表示边是否在最短路上
43 int u[maxe],v[maxe],w[maxe]; //存下每条边, 注意是有向边
44 int d[maxn],p[maxn]; //p表示最短路上每个点的父边
45 int n,m,k,s,t; //s,t分别表示起点和终点
46
47 //以下是主函数中较关键的部分
48 for(int i=0;i<=n;i++)root[i]=null; //一定要加上!!!
49 //((读入&建反向图)
50 Dijkstra();
51 //((清空G,W,id)
52 for(int i=1;i<=m;i++)
53     if(p[i]){
54         used[p[i]]=true; //在最短路上
55         G[v[p[i]]].push_back(i);
56     }
57 for(int i=1;i<=m;i++){
58     w[i]=-d[u[i]]-d[v[i]]; //现在的w[i]表示这条边能使路
59     //径长度增加多少
60     if(!used[i])
61         root[u[i]]=merge(root[u[i]],newnode(w[i],i));
62 }
63 dfs(t);
64 priority_queue<B>heap;
65 heap.push(B(s,root[s],0)); //初始状态是找贡献最小的边加
66 //进去
67 printf("%d\n",d[s]); //第1短路需要特判
68 while(--k){ //其余k-1短路径用二叉堆维护
69     if(heap.empty())printf("-1\n");
70     else{
71         int x=heap.top().x,w=heap.top().w;
72         node *rt=heap.top().rt;
73         heap.pop();
74         printf("%d\n",d[s]+w+rt->w);
75         if(rt->lc!=null||rt->rc!=null)
76             heap.push(B(x,merge(rt->lc,rt->rc),w)); //
77             //pop掉当前边, 换成另一条贡献大一点的边
78         if(root[v[rt->i]]!=null)
79             heap.push(B(v[rt->i],root[v[rt->i]],w+rt->
80                 w)); //保留当前边, 往后面再接上另一条边
81     }
82 }
83 //主函数到此结束
84 //Dijkstra预处理最短路 O(m\log n)

```

```

76 void Dijkstra(){
77     memset(d,63,sizeof(d));
78     d[t]=0;
79     priority_queue<A>heap;
80     heap.push(A(t,0));
81     while(!heap.empty()){
82         int x=heap.top().x;
83         heap.pop();
84         if(vis[x])continue;
85         vis[x]=true;
86         for(int i=0;i<(int)G[x].size();i++){
87             if(!vis[G[x][i]]&&d[G[x][i]]>d[x]+W[x][i]){
88                 d[G[x][i]]=d[x]+W[x][i];
89                 p[G[x][i]]=id[x][i];
90                 heap.push(A(G[x][i],d[G[x][i]]));
91             }
92         }
93     }
94
95     //dfs 求出每个点的堆 总计O(m*log n)
96     //需要调用merge, 同时递归调用自身
97     void dfs(int x){
98         root[x]=merge(root[x],root[v[p[x]]]);
99         for(int i=0;i<(int)G[x].size();i++)
100             dfs(G[x][i]);
101     }
102
103     //包装过的new_node() O(1)
104     node *newnode(int w,int i){
105         ***ptr=node(w,i);
106         ptr->lc=ptr->rc=null;
107         return ptr;
108     }
109
110     //带可持久化的左偏树合并 总计O(n*log n)
111     //递归调用自身
112     node *merge(node *x,node *y){
113         if(x==null)return y;
114         if(y==null)return x;
115         if(x->w>y->w)swap(x,y);
116         node *z=newnode(x->w,x->i);
117         z->lc=x->lc;
118         z->rc=merge(x->rc,y);
119         if(z->lc->d>z->rc->d)swap(z->lc,z->rc);
120         z->refresh();
121         return z;
122     }

```

4.14 最大团

```

1  /*
2  Int g[] [] 为图的邻接矩阵。
3  MC(V)表示点集V的最大团
4  令Si={vi, vi+1, ..., vn}, mc[i]表示MC(Si)
5  倒着算mc[i], 那么显然MC(V)=mc[1]
6  此外有mc[i]=mc[i+1] or mc[i]=mc[i+1]+1
7  */
8  void init(){
9      int i, j;
10     for (i=1; i<=n; ++i) for (j=1; j<=n; ++j) scanf("%
11         d", &g[i][j]);
12 }
13 void dfs(int size){
14     int i, j, k;
15     if (len[size]==0) {
16         if (size>ans) {
17             ans=size; found=true;
18         }
19         return;
20     }
21     for (k=0; k<len[size] && !found; ++k) {
22         if (size+len[size]-k<=ans) break;
23         i=list[size][k];
24         if (size+mc[i]<=ans) break;
25         for (j=k+1; len[size+1]=0; j<len[size]; ++j)
26             if (g[i][list[size][j]]) list[size+1][len[size
27                 +1]++]=list[size][j];
28         dfs(size+1);
29     }
30 }
31 void work(){
32     int i, j;
33     mc[n]=ans=1;
34     for (i=n-1; i; --i) {
35         found=false;
36         len[i]=0;
37         for (j=i+1; j<=n; ++j) if (g[i][j]) list[i][len[i]
38             +1]++]=list[j][j];
39         dfs(i);
40         mc[i]=ans;
41     }
42 }

```

4.15 SPFA 判负环-wrz

```

1 int inq[N], inqt[N], dis[N];
2 bool SPFA()
3 {

```

```

4     queue<int> q;
5     for(int i = 1; i <= n; i++) dis[i] = 0, q.push(i),
6         inq[i] = 1; // 全部入队
7     for(; !q.empty(); )
8     {
9         int x = q.front(); q.pop(); inq[x] = 0;
10        for(int i = last[x]; i; i = e[i].next)
11        {
12            int y = e[i].to;
13            if(dis[x] + e[i].val < dis[y])
14            {
15                dis[y] = dis[x] + e[i].val;
16                if(!inq[y])
17                {
18                    if(++inqt[y] > n) return false; //
19                    入队n次即有负环
20                    inq[y] = 1;
21                    q.push(y);
22                }
23            }
24        }
25    }
26    return true;
27 }
28 /*
29 步骤:
30 1.建好原图
31 2.SPFA() // 若返回为true表示无负环, false表示有负
32 环
33 多次调用时记得清空inqt等数组
34 有负环时理论复杂度是O(n^2)的
35 */

```

4.16 斯坦纳树

```

1 //N点数, M边数, P关键点数
2 const int inf = 0x3f3f3f3f;
3 int n, m, p, status, idx[P], f[1 << P][N];
4 priority_queue<pair<int, int>> q; //int top, h[N];
5 void dijkstra(int dis[]) {}
6 void Steiner_Tree() {
7     for (int i = 1; i < status; i++) {
8         while (!q.empty()) q.pop(); //top = 0;
9         memset(vis, 0, sizeof(vis));
10        for (int j = 1; j <= n; j++) {
11            for (int k = i & (i - 1); k; (--k) &= i)
12                f[i][j] = min(f[i][j], f[k][j] + f[i ^
13                    k][j]);
14            if (f[i][j] != inf)
15                q.push(make_pair(-f[i][j], j)); //h[+
16                top] = j, vis[j] = 1;
17        }
18        dijkstra(f[i]); //SPFA(f[i]);
19    }
20 }
21 int main() {
22     scanf("%d%d%d", &n, &m, &p);
23     status = 1 << p;
24     tot = 0; memset(lst, 0, sizeof(lst));
25     /*求最小生成森林
26     每棵生成树中至少选择一个点, 点权为代价
27     新开一个空白关键点0作为源
28     Add(0, i, val[i]); Add(i, 0, val[i]);*/
29     for (int i = 1; i <= p; i++) scanf("%d", &idx[i]);
30     memset(f, 0x3f, sizeof(f));
31     for (int i = 1; i <= n; i++) f[0][i] = 0;
32     for (int i = 1; i <= p; i++) f[1 << (i - 1)][idx[i
33         ]] = 0;
34     Steiner_Tree();
35     int ans = inf;
36     for (int i = 1; i <= n; i++) ans = min(ans, f[
37         status - 1][i]);
38 }

```

4.17 stoer-wagner 无向图最小割树

```

1 int cost[maxn][maxn], seq[maxn], len[maxn], n, m, pop, ans;
2 bool used[maxn];
3 void Init(){
4     int i, j, a, b, c;
5     for(i=0; i<n; i++) for(j=0; j<n; j++) cost[i][j]=0;
6     for(i=0; i<m; i++){
7         scanf("%d%d%d", &a, &b, &c); cost[a][b]+=c;
8         cost[b][a]+=c;
9     }
10    pop=n; for(i=0; i<n; i++) seq[i]=i;
11 }
12 void Work(){
13     ans=inf; int i, j, k, l, mm, sum, pk;
14     while(pop > 1){
15         for(i=1; i<pop; i++) used[seq[i]]=0; used[seq
16             [0]]=1;
17         for(i=1; i<pop; i++) len[seq[i]]=cost[seq[0]][
18             seq[i]];
19         pk=0; mm=-inf; k=-1;
20         for(i=1; i<pop; i++) if(len[seq[i]] > mm){ mm=
21             len[seq[i]]; k=i; }
22         for(i=1; i<pop; i++){

```

```

19     used[seq[l=k]]=1;
20     if(i==pop-2) pk=k;
21     if(i==pop-1) break;
22     mm=-inf;
23     for(j=1;j<pop;j++) if(!used[seq[j]])
24         if((len[seq[j]]+=cost[seq[l]][seq[j]])
25             > mm)
26             mm=len[seq[j]], k=j;
27     sum=0;
28     for(i=0;i<pop;i++) if(i != k) sum+=cost[seq[k]
29         ][seq[i]];
30     ans=min(ans,sum);
31     for(i=0;i<pop;i++)
32         cost[seq[k]][seq[i]]=cost[seq[i]][seq[k]
33             ]+=cost[seq[pk]][seq[i]];
34     seq[pk]=seq[--pop];
35 }
printf("%d\n",ans);
}

```

4.18 tarjan-gwx

```

1 //cut[i]: i是否为割点
2 //bridge[i]: e[i]是否为桥
3 void dfs(int u, int pa)
4 {
5     d[u] = l[u] = ++timer;
6     st.push(u); vst[u] = 1;
7     int child = 0;
8     for(int i = tail[u]; i; i = e[i].next)
9         if(!d[e[i].v])
10            {
11                child++;
12                dfs(e[i].v, u);
13                l[u] = min(l[u], l[e[i].v]);
14                if(l[e[i].v] >= d[u])
15                {
16                    cut[u] = 1;
17                    if(l[e[i].v] > d[u])
18                        bridge[i] = 1;
19                }
20            }
21     else if(vst[e[i].v]) l[u] = min(l[u], d[e[i].v]
22         );
23     if(!pa && child < 2) cut[u] = 0;
24     if(l[u] == d[u])
25     {
26         int v; scc++;
27         while(true)
28         {
29             v = st.top(); st.pop();
30             id[v] = scc; vst[v] = 0; size[scc]++;
31             if(u == v) break;
32         }
33     }
}

```

4.19 朱刘算法-gwx

```

1 //时间复杂度: O(nm)
2 int N, m;
3 int pre[maxn], in[maxn], f[maxn], id[maxn];
4 struct node {int u, v, w;} a[maxm * 2]; //边表
5
6 int find(int x)
7 {
8     return f[x] == x ? x : f[x] = find(f[x]);
9 }
10
11 int mst()
12 {
13     long long res = 0;
14     int root = 1;
15     int n = N;
16     while(true)
17     {
18         for(int i = 1; i <= n; i++) in[i] = INT_MAX,
19             pre[i] = 0;
20         for(int i = 1; i <= m; i++)
21             if(a[i].u != a[i].v && in[a[i].v] > a[i].
22                 w)
23                 in[a[i].v] = a[i].w, pre[a[i].v] = a[i]
24                     .u;
25         for(int i = 1; i <= n; i++)
26             if(in[i] == INT_MAX && i != root) return
27                 0;
28         int cnt = 0;
29         for(int i = 1; i <= n; i++) f[i] = i, id[i] =
30             0;
31         for(int i = 1; i <= n; i++)
32         {
33             if(i == root) continue;
34             res += in[i];
35             if(find(i) != find(pre[i])) f[f[i]] = f[
36                 pre[i]];
37             else
38             {
39                 cnt++;
40                 for(int j = i; j && !id[j]; j = pre[j]
41                     )

```

```

35         id[j] = cnt;
36     }
37     if(!cnt) break;
38     for(int i = 1; i <= n; i++)
39         if(!id[i]) id[i] = ++cnt;
40     for(int i = 1; i <= m; i++)
41     {
42         if(id[a[i].u] != id[a[i].v]) a[i].w -= in[
43             a[i].v];
44         a[i].u = id[a[i].u];
45         a[i].v = id[a[i].v];
46     }
47     n = cnt;
48     root = id[root];
49 }
50 return res;
51 }

```

4.20 zkw 费用流

```

1 //稠密图、二分图中较快, 稀疏图中不如SPFA
2 int flow, cost, price;
3
4 int dfs(int u, int f)
5 {
6     if(u == t)
7     {
8         flow += f;
9         cost += price * f;
10        return f;
11    }
12    vst[u] = 1;
13    int used = 0;
14    for(int i = tail[u]; i; i = e[i].next)
15        if(!vst[e[i].v] && e[i].c > 0 && e[i].w == 0)
16        {
17            int w = dfs(e[i].v, min(e[i].c, f - used))
18                ;
19            e[i].c -= w; e[i ^ 1].c += w; used += w;
20            if(used == f) return f;
21        }
22    return used;
23 }
24 bool modlabel()
25 {
26     int d = inf;
27     for(int u = s; u <= t; u++)
28         if(vst[u])
29             for(int i = tail[u]; i; i = e[i].next)
30                 if(e[i].c > 0 && !vst[e[i].v]) d = min
31                     (d, e[i].w);
32     if(d == inf) return 0;
33     for(int u = s; u <= t; u++)
34         if(vst[u])
35             for(int i = tail[u]; i; i = e[i].next)
36                 e[i].w -= d, e[i ^ 1].w += d;
37     price += d;
38     return 1;
39 }
40 void zkw()
41 {
42     do
43     {
44         do memset(vst, 0, sizeof(vst));
45         while(dfs(s, inf) > 0);
46         while(modlabel());
47     }
}

```

5 数论

5.1 杜教筛

```

1 // 用之前必须先init(); 如果n很大, 求和记得开long long;
2 // 如果有取模, 求和记得改取模
3 #define N 1000005 // (10^9)^(2/3)
4 #define M 3333331 // hash siz
5 int prime[N], notprime[N], pcnt, mu[N], pre[N];
6 int hash[M], nocnt; struct node{int id, f, next;}no
7     [1000000];
8 int F(int n) // calculate mu[1]+mu[2]+...+mu[n]
9 {
10     if(n<N) return pre[n];
11     int h = n/M; for(int i = hash[h]; i; i = no[i].
12         next) if(no[i].id == n) return no[i].f;
13     int ret = 1;
14     for(int i = 2, j; i <= n; i = j + 1)
15     {
16         j = n/(n/i);
17         ret -= F(n/i) * (j-i+1);
18     }
19     no[++nocnt] = (node){n, ret, hash[h]};
20     hash[h] = nocnt;
21     return ret;
22 }
23 void init()
24 {
25     mu[1] = 1;
26     for(int i = 2; i < N; i++)
27     {
28         if(!notprime[i]) prime[++pcnt] = i, mu[i] =
29             -1;

```

```

26     for(int j = 1; j <= pcnt && prime[j] * i < N; j++)
27     {
28         notprime[prime[j] * i] = 1;
29         if(i % prime[j]) mu[prime[j] * i] = -mu[i];
30         else {mu[prime[j] * i] = 0; break;}
31     }
32     for(int i = 1; i < N; i++) pre[i] = pre[i-1] + mu[i];
33 }
34

```

5.2 直线下整点

$$\sum_{i=0}^{n-1} \lfloor \frac{a+bi}{m} \rfloor, n, m, a, b > 0$$

```

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

```

5.3 拉格朗日插值

```

1 // 用之前必须先init(); 如果所有的逆元都能预处理就是O(n)
   // 的, 否则是O(nlogn)的
2 #define MOD 1000000007
3 int inv[N], invf[N], f[N];
4 int fpow(int a, int b)
5 {
6     int r = 1;
7     for(; b; b >>= 1)
8     {
9         if(b & 1) r = 1ll*r*a%MOD;
10        a = 1ll*a*a%MOD;
11    }
12    return r;
13 }
14 int la(int x, int k) // k次, 求f(x)
15 {
16     int lim = k+2, ff = 1;
17     for(int i = 1; i <= lim; i++)
18         ff = 1ll * ff * (x-i) % MOD;
19     for(int i = 1; i <= lim; i++)
20         f[i] = (f[i-1] + fpow(i, k)) % MOD; // 预处理
21         f(1), f(2), ..., f(lim), 注意修改
22     if(x <= lim) return f[x];
23     int ret = 0;
24     for(int i = 1; i <= lim; i++)
25     {
26         (ret += 1ll * f[i]
27             * ff % MOD * (x-i < N ? inv[x-i] :
28                 fpow(x-i, MOD-2)) % MOD // 复杂度
29             * invf[i-1] % MOD * invf[lim-i] % MOD
30             * ((lim-i) % 2 ? MOD-1 : 1) % MOD
31             ) %= MOD;
32     }
33     return ret;
34 }
35 void init()
36 {
37     inv[1] = 1;
38     for(int i = 2; i < N; i++) inv[i] = 1ll * (MOD -
39         MOD / i) * inv[MOD % i] % MOD;
40     invf[0] = 1;
41     for(int i = 1; i < N; i++) invf[i] = 1ll * invf[i-1] * inv[i] % MOD;
42 }

```

5.4 线性回归

```

1 // O(m^2 log n)
2 // Given a[0], a[1], ..., a[m-1]
3 // a[n] = c[0] * a[n-m] + ... + c[m-1] * a[n-1]
4 // Solve for a[n] = v[0] * a[0] + v[1] * a[1] + ... +
   // v[m-1] * a[m-1]
5 void linear_recurrence(long long n, int m, int a[],
6     int c[], int p) {
7     long long v[M] = {1 % p}, u[M << 1], msk = !n;
8     for(long long i(n); i > 1; i >= 1) {
9         msk <<= 1;
10    }
11    for(long long x(0); msk; msk >>= 1, x <= 1) {
12        fill_n(u, m << 1, 0);
13        int b(!n & msk);
14        x |= b;
15        if(x < m) {
16            u[x] = 1 % p;
17        } else {
18            for(int i(0); i < m; i++) {
19                for(int j(0), t(i + b); j < m; j++, t++) {
20                    u[t] = (u[t] + v[i] * v[j]) % p;
21                }
22            }
23            for(int i((m << 1) - 1); i >= m; i--) {
24                for(int j(0), t(i - m); j < m; j++, t++) {
25

```

```

26                u[t] = (u[t] + c[j] * u[i]) % p;
27            }
28        }
29        copy(u, u + m, v);
30    }
31    for(int i(m); i < 2 * m; i++) {
32        a[i] = 0;
33        for(int j(0); j < m; j++) {
34            a[i] = (a[i] + (long long)c[j] * a[i + j -
35                m]) % p;
36        }
37    }
38    for(int j(0); j < m; j++) {
39        b[j] = 0;
40        for(int i(0); i < m; i++) {
41            b[j] = (b[j] + v[i] * a[i + j]) % p;
42        }
43    }
44    for(int j(0); j < m; j++) {
45        a[j] = b[j];
46    }
47 }

```

5.5 素数测试-gwx

```

1 ll multi(ll x, ll y, ll M) {
2     ll res = 0;
3     for(; y; y >>= 1, x = (x + x) % M)
4         if(y & 1) res = (res + x) % M;
5     return res;
6 }
7 ll power(ll x, ll y, ll p)
8 {
9     ll res = 1;
10    for(; y; y >>= 1, x = multi(x, x, p))
11        if(y & 1) res = multi(res, x, p);
12    return res;
13 }
14 int primetest(ll n, int base)
15 {
16     ll n2 = n - 1, res;
17     int s = 0;
18     while(!(n2 & 1)) n2 >>= 1, s++;
19     res = power(base, n2, n);
20     if(res == 1 || res == n - 1) return 1;
21     s--;
22     while(s >= 0)
23     {
24         res = multi(res, res, n);
25         if(res == n - 1) return 1;
26         s--;
27     }
28     return 0; // n is not a strong pseudo prime
29 }
30 int isprime(ll n)
31 {
32     static ll testNum[] = {2, 3, 5, 7, 11, 13, 17, 19,
33         23, 29, 31, 37};
34     static ll lim[] = {4, 0, 137365311, 253260011,
35         2500000000011, 215230289874711, 3474749660383
36         11, 34155007172832111, 0, 0, 0, 0};
37     if(n < 2 || n == 321503175111) return 0;
38     for(int i = 0; i < 12; i++)
39     {
40         if(n < lim[i]) return 1;
41         if(!primetest(n, testNum[i])) return 0;
42     }
43     return 1;
44 }
45 ll pollard(ll n)
46 {
47     ll i, x, y, p;
48     if(isprime(n)) return n;
49     if(!(n & 1)) return 2;
50     for(i = 1; i < 20; i++)
51     {
52         x = i, y = func(x, n), p = gcd(y - x, n);
53         while(p == 1)
54         {
55             x = func(x, n);
56             y = func(func(y, n), n);
57             p = gcd((y - x + n) % n, n) % n;
58         }
59         if(p == 0 || p == n) continue;
60         return p;
61     }
62 }

```

5.6 原根-gwx

```

1 bool check_force(int g, int p)
2 {
3     int cnt = 0, prod = g;
4     for(int i = 1; i <= p - 1; ++i, prod = prod * g %
5         p)
6         if(prod == 1) if(++cnt > 1) return 0;
7     return 1;
8 }
9 //d[]: prime divisor of (p - 1)
10 bool check_fast(int g, int p)

```



```

10 {
11     for(int i = 1; i <= m; ++i)
12         if(power(g, (p - 1) / d[i], p) == 1) return 0;
13     return 1;
14 }
15 int primitive_root(int p)
16 {
17     for(int i = 2; i < p; ++i) if(check(i, p)) return
18         i;
19 }

```

5.7 勾股数

$a = m^2 - n^2, b = 2mn, c = m^2 + n^2$, 其中 m 和 n 中有一个是偶数, 则 (a, b, c) 是素勾股数

5.8 Pell 方程

Find the smallest integer root of $x^2 - ny^2 = 1$ when n is not a square number, with the solution set $x_{k+1} = x_0 x_k + n y_0 y_k, y_{k+1} = x_0 y_k + y_0 x_k$.

```

1 template <int MAXN = 100000>
2 struct pell {
3     std::pair <long long, long long> solve (long long
4         n) {
5         static long long p[MAXN], q[MAXN], g[MAXN], h[
6             MAXN], a[MAXN];
7         p[1] = q[0] = h[1] = 1; p[0] = q[1] = g[1] =
8             0;
9         a[2] = (long long) (floor (sqrt1 (n) + 1e-7L))
10             ;
11         for (int i = 2; ; ++i) {
12             g[i] = -g[i - 1] + a[i] * h[i - 1];
13             h[i] = (n - g[i] * g[i]) / h[i - 1];
14             a[i + 1] = (g[i] + a[2]) / h[i];
15             p[i] = a[i] * p[i - 1] + p[i - 2];
16             q[i] = a[i] * q[i - 1] + q[i - 2];
17             if (p[i] * p[i] - n * q[i] * q[i] == 1)
18                 return { p[i], q[i] }; } } };

```

5.9 平方剩余

Solve $x^2 \equiv n \pmod{p}$ ($0 \leq a < p$) where p is prime in $O(\log p)$.

```

1 struct quadric {
2     void multiply(long long &c, long long &d, long
3         long a, long long b, long long w, long long p)
4     {
5         int cc = (a * c + b * d % p * w) % p;
6         int dd = (a * d + b * c) % p; c = cc, d = dd;
7     }
8     bool solve(int n, int p, int &x) {
9         if (n == 0) return x = 0, true; if (p == 2)
10             return x = 1, true;
11         if (power (n, p / 2, p) == p - 1) return false
12             ;
13         long long c = 1, d = 0, b = 1, a, w;
14         do { a = rand() % p; w = (a * a - n + p) % p;
15             if (w == 0) return x = a, true;
16         } while (power (w, p / 2, p) != p - 1);
17         for (int times = (p + 1) / 2; times; times >>=
18             1) {
19             if (times & 1) multiply (c, d, a, b, w, p)
20                 ;
21             multiply (a, b, a, b, w, p); }
22         return x = c, true; } };

```

5.10 博弈论

Nim For simplicity, we denote a_i as the number of stones in the i -th pile, $M_i(S)$ as removing stones with the amount chosen in the set S from the i -th pile, and $M_i = M_i[1, a_i]$. Without further explanation, it is assumed that the SG function of a game $SG = \bigoplus_{i=1}^n SG(a_i)$.

Nim $M = \bigcup_{i=1}^n M_i$.

Normal: $SG(n) = n$.

Misere: The same, opposite if all piles are 1's.

Nim (powers) Given $k, M = \bigcup_{i=1}^n M_i \{k^m | m \geq 0\}$.

Normal: If k is odd, $SG(n) = n \% 2$. Otherwise,

$$SG(n) = \begin{cases} 2 & n \% (k+1) \% 2 \\ n \% (k+1) = k & \text{otherwise.} \end{cases}$$

Nim (no greater than half) $M = \bigcup_{i=1}^n M_i[1, \frac{a_i}{2}]$.

Normal: $SG(2n) = n, SG(2n+1) = SG(n)$.

Nim (always greater than half) $M = \bigcup_{i=1}^n M_i[\lceil \frac{a_i}{2} \rceil, a_i]$.

Normal: $SG(0) = 0, SG(n) = \lfloor \log_2 n \rfloor + 1$.

Nim (proper divisors) $M = \bigcup_{i=1}^n M_i \{x | x > 1 \wedge a_i \% x = 0\}$.

Normal: $SG(1) = 0, SG(n) = \max_x (n \% 2^x = 0)$.

Nim (divisors) $M = \bigcup_{i=1}^n M_i \{x | a_i \% x = 0\}$.

Normal: $SG(0) = 0, SG(n) = 1 + \max_x (n \% 2^x = 0)$.

Nim (fixed) Given a finite set $S, M = \bigcup_{i=1}^n M_i(S)$.

Normal: $SG_1(n)$ is eventually periodic.

Given a finite set $S, M = \bigcup_{i=1}^n M_i(S \cup a_i)$.

Normal: $SG_2(n) = SG_1(n) + 1$.

Moore's Nim Given $k, M = \bigcup \{M_{x_1} \times M_{x_2} \cdots \times M_{x_l} | l \leq k \wedge \forall i (x_i < x_{i+1})\}$.

Normal: Sum all $(a_i)_2$ in base $k+1$ without carry. Lose if the result is 0.

Misere: The same, except if all piles are 1's.

Staircase Nim One can take any number of objects from a_{i+1} to $a_i (i \geq 0)$.

Normal: Lose if $\bigoplus_{i=0}^{(n-1)/2} a_{2i+1} = 0$.

Lasker's Nim $M = \bigcup_{i=1}^n M_i \cup S_i$. (S_i : Split a pile into two non-empty piles.)

$$\text{Normal: } SG(n) = \begin{cases} n & n \% 4 = 1, 2 \\ n+1 & n \% 4 = 3 \\ n-1 & n \% 4 = 0. \end{cases}$$

Kayles $M = \bigcup_{i=1}^n M_i[1, 2] \cup MS_i[1, 2]$. (MS_i : Split a pile into two non-empty piles after removing stones.)

Normal: Periodic from the 72-th item with period length 12.

Dawson's chess n stones in a line. One can take a stone if its neighbours are not taken.

Normal: Periodic from the 52-th item with period length 34.

Ferguson game Two boxes with m stones and n stones. One can empty any one box and move any positive number of stones from another box to this box each step.

Normal: Lose if both m and n are odd.

Fibonacci game n stones. The first player may take any positive number of stones during the first move, but not all of them. After that, each player may take any positive number of stones, but less than twice the number of stones taken during the last turn.

Normal: Win if n is not a fibonacci number.

Wythoff's game Two piles of stones. Players take turns removing stones from one or both piles; when removing stones from both piles, the numbers of stones removed from each pile must be equal.

Normal: Lose if $\lfloor \frac{\sqrt{5}+1}{2} |A-B| \rfloor = \min(A, B)$

Mock turtles n coins in a line. One can turn over any 1, 2, or 3 coins, but the rightmost coin turned must be from head to tail.

Normal: $SG(n) = 2n + [\text{popcount}(n) \text{ is even}]$.

Ruler n coins in a line. One can turn over any consecutive coins, but the rightmost coin turned must be from head to tail.

Normal: $SG(n) = \text{lowbit}(n)$.

Hackenbush The game starts with the players drawing a ground line (conventionally, but not necessarily, a horizontal line at the bottom of the paper or other playing area) and several line segments such that each line segment is connected to the ground, either directly at an endpoint, or indirectly, via a chain of other segments connected by endpoints. Any number of segments may meet at a point and thus there may be multiple paths to ground.

On his turn, a player cuts (erases) any line segment of his choice. Every line segment no longer connected to the ground by any path falls (i.e., gets erased). According to the normal play convention of combinatorial game theory, the first player who is unable to move loses.

Played exclusively with vertical stacks of line segments, also referred to as bamboo stalks, the game directly becomes Nim and can be directly analyzed as such. Divergent segments, or trees, add an additional wrinkle to the game and require use of the colon principle stating that when branches come together at a vertex, one may replace the branches by a non-branching stalk of length equal to their nim sum. This principle changes the representation of the game to the more basic version of the bamboo stalks. The last possible set of graphs that can be made are convergent ones, also known as arbitrarily rooted graphs. By using the fusion principle, we can state that all vertices on any cycle may be fused together without changing the value of the graph. Therefore, any convergent graph can also be interpreted as a simple bamboo stalk graph. By combining all three types of graphs we can add complexity to the game, without ever changing the Nim sum of the game, thereby allowing the game to take the strategies of Nim.

Joseph cycle n players are numbered with $0, 1, 2, \dots, n-1$. $f_{1,m} = 0, f_{n,m} = (f_{n-1,m} + m) \bmod n$.

6 字符串

6.1 AC 自动机-wrzs

```

1 struct ACAM
2 {
3     ACAM *next[S], *fail;
4     int ban;
5 } mem[N], *tot, *null, *root, *q[N];
6 ACAM *newACAM()
7 {
8     ACAM *p = ++tot;
9     *p = *null; return p;
10 }
11 void init()
12 {
13     null = tot = mem;
14     for(int i = 0; i < alpha; i++) null->next[i] =
15         null;
16     null->fail = null; null->ban = 0;
17     root = newACAM();
18 }
19 void inser(char *s)
20 {
21     ACAM *p = root;
22     for(int i = 0; s[i]; i++)
23     {
24         int w = s[i] - 'a';
25         if(p->next[w] == null) p->next[w] = newACAM();

```



```

25     p = p->next[w];
26 }
27 p->ban = 1;
28 }
29 void build()
30 {
31     root->fail = root; int head = 0, tail = 0;
32     for(int i = 0; i < alpha; i++)
33     {
34         if(root->next[i] == null) root->next[i] = root;
35         else root->next[i]->fail = root, q[tail++] = root->next[i];
36     }
37     for(; head < tail; head++)
38     {
39         ACAM *p = q[head];
40         p->ban |= p->fail->ban;
41         for(int i = 0; i < alpha; i++)
42         {
43             if(p->next[i] == null) p->next[i] = p->fail->next[i];
44             else p->next[i]->fail = p->fail->next[i], q[tail++] = p->next[i];
45         }
46     }
47 }

```

6.2 扩展 KMP-gwx

```

1 void get_next()
2 {
3     int a = 0, p = 0;
4     nxt[0] = m;
5     for(int i = 1; i < m; i++)
6     {
7         if(i >= p || i + nxt[i - a] >= p)
8         {
9             if(i >= p) p = i;
10            while(p < m && t[p] == t[p - i]) p++;
11            nxt[i] = p - i;
12            a = i;
13        }
14        else nxt[i] = nxt[i - a];
15    }
16 }
17 void exkmp()
18 {
19     int a = 0, p = 0;
20     get_next();
21     for(int i = 0; i < n; i++)
22     {
23         if(i >= p || i + nxt[i - a] >= p) // i >= p 的
24             作用：举个典型例子，s 和 t 无一字符相同
25         {
26             if(i >= p) p = i;
27             while(p < n && p - i < m && s[p] == t[p - i]) p++;
28             ext[i] = p - i;
29             a = i;
30         }
31         else ext[i] = nxt[i - a];
32     }
33 }

```

6.3 马拉车-gwx

```

1 //maxn = 2 * n
2 void manacher(int n)
3 {
4     int p = 0, r = 0;
5     for(int i = 1; i <= n; i++)
6     {
7         if(i <= r) len[i] = min(len[2 * p - i], r - i + 1);
8         else len[i] = 1;
9         while(b[i + len[i]] == b[i - len[i]]) len[i]++;
10        if(i + len[i] - 1 >= r) r = i + len[i] - 1, p = i;
11    }
12 }
13
14 int main()
15 {
16     scanf("%d\n%s", &n, a + 1);
17     b[++tot] = '@'; b[++tot] = '#';
18     for(int i = 1; i < n; i++)
19         b[++tot] = a[i], b[++tot] = '#';
20     b[++tot] = a[n];
21     b[++tot] = '#'; b[++tot] = '$';
22     manacher(tot);
23 }
24

```

6.4 最小表示-gwx

```

1 //不保证起始位置最靠前(?)
2 string find(int N, string s) {

```

```

3     int i, j, k, l;
4     for (i = 0, j = 1; j < N; ) {
5         for (k = 0; k < N && s[i + k] == s[j + k]; k++)
6             if (k >= N) break;
7         if (s[i + k] > s[j + k]) j += k + 1;
8         else l = i + k, i = j, j = max(l, j) + 1;
9     }
10    return s.substr(i, N);
11 }

```

6.5 回文树-wrz

```

1 char s[N], out[N];
2 struct PT
3 {
4     PT *fail, *next[A];
5     int len;
6 } mem[N], *tot, *null, *root1, *root0, *last;
7 PT *newPT()
8 {
9     PT *p = ++tot;
10    *p = *null; return p;
11 }
12 void init()
13 {
14     null = tot = mem;
15     null->fail = null;
16     for(int i = 0; i < A; i++) null->next[i] = null;
17     null->len = 0;
18     root1 = newPT(); root1->fail = root1; root1->len = -1;
19     root0 = newPT(); root0->fail = root1; last = root1;
20 }
21 int extend(int c, int i) // 返回这一次是否多了一个回文子串
22 {
23     PT *p = last;
24     for(; s[i-p->len-1] != c+'a'; p = p->fail);
25     if(p->next[c] != null) {last = p->next[c]; return 0;}
26     PT *np = p->next[c] = last = newPT(); np->len = p->len + 2;
27     if(p->len == -1) np->fail = root0;
28     else
29     {
30         for(p=p->fail; s[i-p->len-1] != c+'a'; p = p->fail);
31         np->fail = p->next[c];
32     }
33     return 1;
34 }
35 void main()
36 {
37     scanf("%s", s+1); init();
38     for(int i = 1, ii = strlen(s+1); i <= ii; i++)
39         out[i] = extend(s[i]-'a', i)?'1':'0';
40     puts(out+1);
41 }

```

6.6 后缀数组-wrz

```

1 // 对都是数字的数组做SA时要保证数组中没有0，否则height
2 // 等可能由于s[0]=s[n+1]=0出问题
3 // 多次使用要保证s[0]=s[n+1]=0
4 char s[N];
5 int n, t1[N], t2[N], sa[N], rank[N], sum[N], height[N],
6   ], lef, rig; // 数组开两倍
7 void SA_build()
8 {
9     int *x = t1, *y = t2, m = 30;
10    for(int i = 1; i <= n; i++) sum[x[i] = s[i] - 'a' + 1]++;
11    for(int i = 1; i <= m; i++) sum[i] += sum[i-1];
12    for(int i = n; i >= 1; i--) sa[sum[x[i]]--] = i;
13    for(int k = 1; k <= n; k <= 1)
14    {
15        int p = 0;
16        for(int i = n-k+1; i <= n; i++) y[++p] = i;
17        for(int i = 1; i <= n; i++) if(sa[i] - k > 0) y[++p] = sa[i] - k;
18
19        for(int i = 1; i <= m; i++) sum[i] = 0;
20        for(int i = 1; i <= n; i++) sum[x[i]]++;
21        for(int i = 1; i <= m; i++) sum[i] += sum[i-1];
22        for(int i = n; i >= 1; i--) sa[sum[x[y[i]]]--] = y[i];
23
24        swap(x, y);
25        for(int i = 1; i <= n; i++)
26            x[sa[i]] = x[sa[i-1]] + (y[sa[i]] == y[sa[i-1]] ? 0 : 1);
27        m = x[sa[n]];
28        if(m == n) break;
29    }
30    for(int i = 1; i <= n; i++) rank[sa[i]] = i;

```

```

29     for(int i = 1, k = 0; i <= n; height[rank[i++]] =
        k?k--:k)
30         for(; s[i+k] == s[sa[ran[i]-1]+k] && i+k <= n
            && sa[ran[i]-1]+k <= n; k++);
31 }

```

6.7 后缀数组 SAIS

```

1 // string is 0-based
2 // sa[] is 1-based
3 // s[n] < s[i] i = 0...n-1
4 namespace SA {
5     int sa[MAXN], rk[MAXN], ht[MAXN], s[MAXN << 1], t[
        MAX << 1], p[MAXN], cnt[MAXN], cur[MAXN];
6     #define pushS(x) sa[cur[s[x]]++] = x
7     #define pushL(x) sa[cur[s[x]]++] = x
8     #define inducedSort(v) std::fill_n(sa, n, -1); std::
        fill_n(cnt, m, 0);\
9         for (int i = 0; i < n; i++) cnt[s[i]]++;\
10        for (int i = 1; i < m; i++) cnt[i] += cnt[i-1];\
11        for (int i = 0; i < m; i++) cur[i] = cnt[i-1];\
12        for (int i = n1-1; ~i; i--) pushS(v[i]);\
13        for (int i = 1; i < m; i++) cur[i] = cnt[i-1];\
14        for (int i = 0; i < n; i++) if (sa[i] > 0 && t[sa
            [i]-1]) pushL(sa[i]-1);\
15        for (int i = 0; i < m; i++) cur[i] = cnt[i-1];\
16        for (int i = n-1; ~i; i--) if (sa[i] > 0 && t[sa
            [i]-1]) pushS(sa[i]-1)
17    void sais(int n, int m, int *s, int *t, int *p) {
18        int n1 = t[n-1] = 0, ch = rk[0] = -1, *s1 = s +
            n;
19        for (int i = n-2; ~i; i--) t[i] = s[i] == s[i
            +1] ? t[i+1] : s[i] > s[i+1];
20        for (int i = 1; i < n; i++) rk[i] = t[i-1] &&
            !t[i] ? (p[n1] = i, n1++) : -1;
21        inducedSort(p);
22        for (int i = 0, x, y; i < n; i++) if (~x = rk
            [sa[i]]) {
23            if (ch < 1 || p[x+1] - p[x] != p[y+1] - p[
                y]) ch++;
24            else for (int j = p[x], k = p[y]; j <= p[x
                +1]; j++, k++)
25                if ((s[j]<<1|t[j]) != (s[k]<<1|t[k]))
                    {ch++; break;}
26            s1[y = x] = ch; }
27        if (ch+1 < n1) sais(n1, ch+1, s1, t+n, p+n1);
28        else for (int i = 0; i < n1; i++) sa[s1[i]] =
            i;
29        for (int i = 0; i < n1; i++) s1[i] = p[sa[i]];
30        inducedSort(s1); }
31    int mapCharToInt(int n, const T *str) {
32        int m = *std::max_element(str, str+n);
33        std::fill_n(rk, m+1, 0);
34        for (int i = 0; i < n; i++) rk[str[i]] = 1;
35        for (int i = 0; i < m; i++) rk[i+1] += rk[i];
36        for (int i = 0; i < n; i++) s[i] = rk[str[i]]
            - 1;
37        return rk[m]; }
38    void suffixArray(int n, const T *str) {
39        int m = mapCharToInt(n, str);
40        sais(n, m, s, t, p);
41        for (int i = 0; i < n; i++) rk[sa[i]] = i;
42        for (int i = 0, h = ht[0] = 0; i < n-1; i++) {
43            int j = sa[rk[i]-1];
44            while (i+h < n && j+h < n && s[i+h] == s[j
                +h]) h++;
45            if (ht[rk[i]] = h) h--; } } }

```

6.8 后缀自动机-wrz

```

1 struct SAM
2 {
3     SAM *next[A], *fail;
4     int len, mi, mx;
5 } mem[N], *tot, *null, *root, *last, *q[N];
6 SAM *newSAM(int len)
7 {
8     SAM *p = ++tot;
9     *p = *null;
10    p->len = p->mi = len;
11    p->mx = 0;
12    return p;
13 }
14 void init()
15 {
16     null = tot = mem;
17     for(int i = 0; i < A; i++) null->next[i] = null;
18     null->fail = null;
19     null->len = null->mi = null->mx = 0;
20     root = last = newSAM(0);
21 }
22 void extend(int v)
23 {
24     SAM *p = last, *np = newSAM(p->len + 1); last = np
        ;
25     for(; p->next[v] == null && p != null; p = p->fail
        ) p->next[v] = np;
26     if(p==null) np->fail = root;
27     else
28     {

```

```

29     SAM *q = p->next[v];
30     if(q->len == p->len+1) np->fail = q;
31     else
32     {
33         SAM *nq = newSAM(p->len+1);
34         memcpy(nq->next, q->next, sizeof(nq->next)
            );
35         nq->fail = q->fail;
36         q->fail = np->fail = nq;
37         for(; p->next[v] == q && p != null; p = p
            ->fail) p->next[v] = nq;
38     }
39 }
40 }

```

6.9 ex 后缀自动机-wrz

```

1 struct sam
2 {
3     sam *fail, *next[A];
4     int len;
5 } mem[N<<1], *tot, *null, *root;
6 sam* newsam()
7 {
8     ***tot = *null;
9     return tot;
10 }
11 void init()
12 {
13     null = tot = mem; null->fail = null; null->len =
        0;
14     for(int i = 0; i < A; i++) null->next[i] = null;
15     root = newsam();
16 }
17 sam* extend(sam *p, int v)
18 {
19     if(p->next[v] != null)
20     {
21         sam *q = p->next[v];
22         if(p->len + 1 == q->len) return q;
23         else
24         {
25             sam *nq = newsam(); *nq = *q; nq->len = p
                ->len + 1;
26             q->fail = nq;
27             for(; p->next[v] == q && p != null; p = p
                ->fail) p->next[v] = nq;
28             return nq;
29         }
30     }
31     else
32     {
33         sam *np = newsam(); np->len = p->len + 1;
34         for(; p->next[v] == null && p != null; p = p-
            >fail) p->next[v] = np;
35         if(p == null) np->fail = root;
36         else
37         {
38             sam *q = p->next[v];
39             if(p->len + 1 == q->len) np->fail = q;
40             else
41             {
42                 sam *nq = newsam(); *nq = *q; nq->len
                    = p->len + 1;
43                 np->fail = q->fail = nq;
44                 for(; p->next[v] == q && p != null; p
                    = p->fail) p->next[v] = nq;
45             }
46         }
47         return np;
48     }
49 }
50 void build_tree()
51 {
52     for(sam *i = tot; i != mem; i--)
53         addedge(i->fail - mem, i - mem);
54 }

```

7 其他

7.1 蔡勒公式

```

1 int zeller(int y, int m, int d) {
2     if (m<=2) y--,m+=12; int c=y/100; y%=100;
3     int w=((c>>2)-(c<<1)+y+(y>>2)+(13*(m+1)/5)+d-1)%7;
4     if (w<0) w+=7; return(w);
5 }

```

7.2 dancing-links

```

1 struct Node {
2     Node *l, *r, *u, *d, *col;
3     int size, line_no;
4     Node() {
5         size = 0; line_no = -1;
6         l = r = u = d = col = NULL;
7     }
8 } *root;
9 void cover(Node *c) {
10    c->l->r = c->r; c->r->l = c->l;

```

```

11     for (Node *u = c->d; u != c; u = u->d)
12     for (Node *v = u->r; v != u; v = v->r) {
13         v->d->u = v->u;
14         v->u->d = v->d;
15         -- v->col->size;
16     }
17 }
18 void uncover(Node *c) {
19     for (Node *u = c->u; u != c; u = u->u) {
20         for (Node *v = u->l; v != u; v = v->l) {
21             ++ v->col->size;
22             v->u->d = v;
23             v->d->u = v;
24         }
25     }
26     c->l->r = c; c->r->l = c;
27 }
28 std::vector<int> answer;
29 bool search(int k) {
30     if (root->r == root) return true;
31     Node *r = NULL;
32     for (Node *u = root->r; u != root; u = u->r)
33         if (r == NULL || u->size < r->size)
34             r = u;
35     if (r == NULL || r->size == 0) return false;
36     else {
37         cover(r);
38         bool succ = false;
39         for (Node *u = r->d; u != r && !succ; u = u->d)
40             {
41                 answer.push_back(u->line_no);
42                 for (Node *v = u->r; v != u; v = v->r) //
43                     Cover row
44                     cover(v->col);
45                 succ |= search(k + 1);
46                 for (Node *v = u->l; v != u; v = v->l)
47                     uncover(v->col);
48                 if (!succ) answer.pop_back();
49             }
50         uncover(r);
51         return succ;
52     }
53 }
54 bool entry[CR][CC];
55 Node *who[CR][CC];
56 int cr, cc;
57 void construct() {
58     root = new Node();
59     Node *last = root;
60     for (int i = 0; i < cc; ++ i) {
61         Node *u = new Node();
62         last->r = u; u->l = last;
63         Node *v = u; u->line_no = i;
64         last = u;
65         for (int j = 0; j < cr; ++ j)
66             if (entry[j][i]) {
67                 ++ u->size;
68                 Node *cur = new Node();
69                 who[j][i] = cur;
70                 cur->line_no = j;
71                 cur->col = u;
72                 cur->u = v; v->d = cur;
73                 v = cur;
74             }
75         v->d = u; u->u = v;
76     }
77     last->r = root; root->l = last;
78     for (int j = 0; j < cr; ++ j) {
79         Node *last = NULL;
80         for (int i = cc - 1; i >= 0; -- i)
81             if (entry[j][i]) {
82                 last = who[j][i];
83                 break;
84             }
85         for (int i = 0; i < cc; ++ i)
86             if (entry[j][i]) {
87                 last->r = who[j][i];
88                 who[j][i]->l = last;
89                 last = who[j][i];
90             }
91     }
92 }
93 void destruct() {
94     for (Node *u = root->r; u != root; ) {
95         for (Node *v = u->d; v != u; ) {
96             Node *nxt = v->d;
97             delete(v);
98             v = nxt;
99         }
100         Node *nxt = u->r;
101         delete(u); u = nxt;
102     }
103     delete root;
104 }

```

7.3 枚举子集

```

1 for (int x = 1; x <= n; x++)
2     for (int y = x & (x - 1); y; (--y) &= x) {
3         //y is a subset of x
4     }

```

7.4 梅森旋转

```

1 #include <random>
2 int main() {
3     std::mt19937 g(seed); // std::mt19937_64
4     std::cout << g() << std::endl;
5 }

```

7.5 大数乘法取模

```

1 // 需要保证 x 和 y 非负
2 long long mult(long long x, long long y, long long
3     MODN) {
4     long long t = (x * y - (long long)((long double)x
5         / MODN * y + 1e-3) * MODN) % MODN;
6     return t < 0 ? t + MODN : t;
7 }

```

8 提示

8.1 make 支持 c++11

```

1 export CXXFLAGS='-std=c++11-Wall'
2 source .bashrc

```

8.2 Java

```

1 import java.util.*;
2 import java.math.*;
3 public class javaNote
4 {
5     static BigInteger q[] = new BigInteger[5000000];
6     // 定义数组的正确姿势，记得分配内存
7     public static void main(String[] args)
8     {
9
10         long currentTime = System.currentTimeMillis();
11         // 获取时间，单位是ms
12
13         Scanner sc = new Scanner(System.in); // 定义输入
14         int a = sc.nextInt(), b;
15         System.out.println("integer_ = " + a); // 输出
16
17         BigInteger x = new BigInteger("233"), y = new
18             BigInteger("666");
19         BigInteger.valueOf(1); // 将指定的表达式转化成
20             BigInteger类型
21         x.add(y); //x+y
22         x.subtract(y); //x-y
23         x.multiply(y); //x*y
24         x.divide(y);
25
26         x.pow(233); // x**233
27         x.compareTo(y); // 比较x和y, x < y : -1, x = y
28             : 0, x > y : 1
29
30         BigDecimal n = new BigDecimal("233"), m = new
31             BigDecimal("666");
32         n.divide(m,a,RoundingMode.DOWN); //n/m并精确到
33             小数点后第a位, a=0表示精确到个位, a为负数
34             表示精确到小数点前-a+1位, 可能变成科学计数
35             法
36
37         /*
38             取整方式
39             RoundingMode.CEILING: 取右边最近的整数, 即
40                 向正无穷取整
41             RoundingMode.FLOOR: 取左边最近的整数, 即向
42                 负无穷取整
43             RoundingMode.DOWN: 向0取整
44             RoundingMode.UP: 远离0取整
45             RoundingMode.HALF_UP: 上取整的四舍五入,
46                 >=0.5会进位, <0.5会舍去, 负数会先取绝
47                 对值再四舍五入再变回负数
48             RoundingMode.HALF_DOWN: 下取整的四舍五入,
49                 >0.5会进位, <=0.5会舍去, 负数原理同上
50             RoundingMode.HALF_EVEN: 分奇偶的四舍五入,
51                 >0.5会进位, <0.5会舍去, =0.5会向最近的
52                 偶数取整, 如2.5->2, (-2.5)->(-2)
53
54         */
55
56         Math.max(a, b); //取大
57         Math.min(a, b); //取小
58         Math.PI; //pi
59
60         HashSet<BigInteger> hash = new HashSet<
61             BigInteger>(); // hash table
62         hash.contains(x); // hash table中是否有a, 有则
63             返回true, 反之返回false
64         hash.add(x); // 把x加进hash table
65         hash.remove(x); // 从hash table中删去x
66
67         Arrays.sort(arr, 1, n+1); // arr 是需要排序的
68             数组, 后两个参数分别是排序的起始位置和结束
69             位置+1, 还可以有第四个参数是比较函数
70     }
71 }

```

```

49 // Arrays.sort(arr, a, b, cmp) = sort(arr+a,
50 // arr+b, cmp)
51 }
52 public static BigInteger sqrt (BigInteger x) {
53 if (x.equals (BigInteger.ZERO) || x.equals (
54 BigInteger.ONE)) return x;
55 BigInteger d = BigInteger.ZERO.setBit (x.bitLength
56 () / 2);
57 BigInteger d2 = d;
58 for (; ; ) {
59 BigInteger y = d.add (x.divide (d)).shiftRight
60 (1);
61 if (y.equals (d) || y.equals (d2)) return d.
62 min (d2);
63 d2 = d; d = y; } }

```

8.3 cout 输出小数

```
1 std::cout << std::fixed << std::setprecision(5);
```

8.4 释放容器内存

```

1 template <typename T>
2 _inline void clear(T& container) {
3 container.clear(); // 或者删除了一堆元素
4 T(container).swap(container);
5 }

```

8.5 tuple

```

1 mytuple = std::make_tuple (10, 2.6, 'a'); //
2 // packing values into tuple
3 std::tie (myint, std::ignore, mychar) = mytuple; //
4 // unpacking tuple into variables
5 std::get<I>(mytuple) = 20;
6 std::cout << std::get<I>(mytuple) << std::endl; //
7 // get the Ith(const) element

```

8.6 读入优化 & 手开 O3

```

1 // getchar() 读入优化 << 关同步 cin << 此优化
2 // 用 isdigit() 会小幅变慢
3 // 返回 false 表示读到文件尾
4 #define __attribute__ ((optimize ("-O3")))
5 #define __inline__ __attribute__((gnu_inline__,
6 __always_inline__, __artificial__))
7 namespace Reader {
8 const int L = (1 << 15) + 5;
9 char buffer[L], *S, *T;
10 _inline bool getchar(char &ch) {
11 if (S == T) {
12 T = (S = buffer) + fread(buffer, 1, L,
13 stdin);
14 if (S == T) {
15 ch = EOF;
16 return false;
17 }
18 }
19 ch = *S++;
20 return true;
21 }
22 _inline bool getint(int &x) {
23 char ch; bool neg = 0;
24 for (; getchar(ch) && (ch < '0' || ch > '9'); )
25 neg ^= ch == '-';
26 if (ch == EOF) return false;
27 x = ch - '0';
28 for (; getchar(ch), ch >= '0' && ch <= '9'; )
29 x = x * 10 + ch - '0';
30 if (neg) x = -x;
31 return true;
32 }
33 }

```

8.7 手开栈

The following lines allow the program to use larger stack memory.

```

1 //C++
2 #pragma comment (linker, "/STACK:36777216")
3 //G++
4 int __size__ = 256 << 20;
5 char *__p__ = (char*) malloc(__size__ + __size__);
6 __asm__ ("movl 0, %%esp\n" :: "r" (__p__));

```

9 附录-数学公式

Binomial coefficients

$$\binom{n}{k} = (-1)^k \binom{k-n-1}{k}, \quad \sum_{k \leq n} \binom{r+k}{k} = \binom{r+n+1}{n}$$

$$\sum_{k=0}^n \binom{k}{m} = \binom{n+1}{m+1}$$

$$\sqrt{1+z} = 1 + \sum_{k=1}^{\infty} \frac{(-1)^{k-1}}{k \times 2^{2k-1}} \binom{2k-2}{k-1} z^k$$

$$\sum_{k=0}^r \binom{r-k}{m} \binom{s+k}{n} = \binom{r+s+1}{m+n+1}$$

$$C_{n,m} = \binom{n+m}{m} - \binom{n+m}{m-1}, n \geq m$$

$$\binom{n}{k} \equiv [n \& k = k] \pmod{2}$$

$$\binom{n_1 + \dots + n_p}{m} = \sum_{k_1 + \dots + k_p = m} \binom{n_1}{k_1} \dots \binom{n_p}{k_p}$$

Fibonacci numbers

$$F(z) = \frac{z}{1-z-z^2}$$

$$f_n = \frac{\phi^n - \hat{\phi}^n}{\sqrt{5}}, \phi = \frac{1+\sqrt{5}}{2}, \hat{\phi} = \frac{1-\sqrt{5}}{2}$$

$$\sum_{k=1}^n f_k = f_{n+2} - 1, \quad \sum_{k=1}^n f_k^2 = f_n f_{n+1}$$

$$\sum_{k=0}^n f_k f_{n-k} = \frac{1}{5} (n-1) f_n + \frac{2}{5} n f_{n-1}$$

$$\frac{f_{2n}}{f_n} = f_{n-1} + f_{n+1}$$

$$f_1 + 2f_2 + 3f_3 + \dots + n f_n = n f_{n+2} - f_{n+3} + 2$$

$$\gcd(f_m, f_n) = f_{\gcd(m, n)}$$

$$f_n^2 + (-1)^n = f_{n+1} f_{n-1}$$

$$f_{n+k} = f_n f_{k+1} + f_{n-1} f_k$$

$$f_{2n+1} = f_n^2 + f_{n+1}^2$$

$$(-1)^k f_{n-k} = f_n f_{k-1} - f_{n-1} f_k$$

$$\text{Modulo } f_n, f_{mn+r} \equiv \begin{cases} f_r, & m \bmod 4 = 0; \\ (-1)^{r+1} f_{n-r}, & m \bmod 4 = 1; \\ (-1)^n f_r, & m \bmod 4 = 2; \\ (-1)^{r+1+n} f_{n-r}, & m \bmod 4 = 3. \end{cases}$$

Period modulo a prime p is a factor of $2p+2$ or $p-1$.

Only exception: $G(5) = 20$.

Period modulo the power of a prime p^k : $G(p^k) = G(p)p^{k-1}$.

Period modulo $n = p_1^{k_1} \dots p_m^{k_m}$: $G(n) = \text{lcm}(G(p_1^{k_1}), \dots, G(p_m^{k_m}))$.

Lucas numbers

$$L_0 = 2, L_1 = 1, L_n = L_{n-1} + L_{n-2} = \left(\frac{1+\sqrt{5}}{2}\right)^n + \left(\frac{1-\sqrt{5}}{2}\right)^n$$

$$L(x) = \frac{2-x}{1-x-x^2}$$

Catlan numbers

$$c_1 = 1, c_n = \sum_{i=0}^{n-1} c_i c_{n-1-i} = c_{n-1} \frac{4n-2}{n+1} = \frac{\binom{2n}{n}}{n+1}$$

$$= \binom{2n}{n} - \binom{2n}{n-1}, c(x) = \frac{1-\sqrt{1-4x}}{2x}$$

Stirling cycle numbers Divide n elements into k non-empty cycles.

$$s(n, 0) = 0, s(n, n) = 1, s(n+1, k) = s(n, k-1) - ns(n, k)$$

$$s(n, k) = (-1)^{n-k} \left[\begin{matrix} n \\ k \end{matrix} \right]$$

$$\left[\begin{matrix} n+1 \\ k \end{matrix} \right] = n \left[\begin{matrix} n \\ k \end{matrix} \right] + \left[\begin{matrix} n \\ k-1 \end{matrix} \right], \left[\begin{matrix} n+1 \\ 2 \end{matrix} \right] = n! H_n$$

$$x^n = x(x-1)\dots(x-n+1) = \sum_{k=0}^n \left[\begin{matrix} n \\ k \end{matrix} \right] (-1)^{n-k} x^k$$

$$x^{\overline{n}} = x(x+1)\dots(x+n-1) = \sum_{k=0}^n \left[\begin{matrix} n \\ k \end{matrix} \right] x^k$$

Stirling subset numbers Divide n elements into k non-empty subsets.

$$\left\{ \begin{matrix} n+1 \\ k \end{matrix} \right\} = k \left\{ \begin{matrix} n \\ k \end{matrix} \right\} + \left\{ \begin{matrix} n \\ k-1 \end{matrix} \right\}$$

$$x^n = \sum_{k=0}^n \left\{ \begin{matrix} n \\ k \end{matrix} \right\} x^{\underline{k}} = \sum_{k=0}^n \left\{ \begin{matrix} n \\ k \end{matrix} \right\} (-1)^{n-k} x^{\overline{k}}$$

$$m! \left\{ \begin{matrix} n \\ m \end{matrix} \right\} = \sum_{k=0}^m \binom{m}{k} k^n (-1)^{m-k}$$

$$\sum_{k=1}^n k^p = \sum_{k=0}^p \left\{ \begin{matrix} p \\ k \end{matrix} \right\} (n+1)^{\underline{k}}$$

For a fixed k , generating functions :

$$\sum_{n=0}^{\infty} \left\{ \begin{matrix} n \\ k \end{matrix} \right\} x^{n-k} = \prod_{r=1}^k \frac{1}{1-rx}$$

Motzkin numbers Draw non-intersecting chords between n points on a circle.

Pick n numbers $k_1, k_2, \dots, k_n \in \{-1, 0, 1\}$ so that $\sum_i^a k_i (1 \leq a \leq n)$ is non-negative and the sum of all numbers is 0.

$$M_{n+1} = M_n + \sum_i^{n-1} M_i M_{n-1-i} = \frac{(2n+3)M_n + 3nM_{n-1}}{n+3}$$

$$M_n = \sum_{i=0}^{\lfloor \frac{n}{2} \rfloor} \binom{n}{2k} Catalan(k)$$

$$M(X) = \frac{1-x-\sqrt{1-2x-3x^2}}{2x^2}$$

Eulerian numbers Permutations of the numbers 1 to n in which exactly k elements are greater than the previous element.

$$\left\langle \begin{matrix} n \\ k \end{matrix} \right\rangle = (k+1) \left\langle \begin{matrix} n-1 \\ k \end{matrix} \right\rangle + (n-k) \left\langle \begin{matrix} n-1 \\ k-1 \end{matrix} \right\rangle$$

$$x^n = \sum_k \left\langle \begin{matrix} n \\ k \end{matrix} \right\rangle \binom{x+k}{n}$$

$$\left\langle \begin{matrix} n \\ m \end{matrix} \right\rangle = \sum_{k=0}^m \binom{n+1}{k} (m+1-k)^n (-1)^k$$

Harmonic numbers Sum of the reciprocals of the first n natural numbers.

$$\begin{aligned} \sum_{k=1}^n H_k &= (n+1)H_n - n \\ \sum_{k=1}^n kH_k &= \frac{n(n+1)}{2}H_n - \frac{n(n-1)}{4} \\ \sum_{k=1}^n \binom{k}{m} H_k &= \binom{n+1}{m+1} \left(H_{n+1} - \frac{1}{m+1} \right) \end{aligned}$$

Pentagonal number theorem

$$\prod_{n=1}^{\infty} (1-x^n) = \sum_{n=-\infty}^{\infty} (-1)^k x^{k(3k-1)/2}$$

$$p(n) = p(n-1) + p(n-2) - p(n-5) - p(n-7) + \dots$$

$$f(n,k) = p(n) - p(n-k) - p(n-2k) + p(n-5k) + p(n-7k) - \dots$$

Bell numbers Divide a set that has exactly n elements.

$$B_n = \sum_{k=1}^n \left\{ \begin{matrix} n \\ k \end{matrix} \right\}, \quad B_{n+1} = \sum_{k=0}^n \binom{n}{k} B_k$$

$$B_{p^m+n} \equiv mB_n + B_{n+1} \pmod{p}$$

$$B(x) = \sum_{n=0}^{\infty} \frac{B_n}{n!} x^n = e^{e^x-1}$$

Bernoulli numbers

$$B_n = 1 - \sum_{k=0}^{n-1} \binom{n}{k} \frac{B_k}{n-k+1}$$

$$G(x) = \sum_{k=0}^{\infty} \frac{B_k}{k!} x^k = \frac{1}{\sum_{k=0}^{\infty} \frac{x^k}{(k+1)!}}$$

$$\sum_{k=1}^n k^m = \frac{1}{m+1} \sum_{k=0}^m \binom{m+1}{k} B_k n^{m-k+1}$$

Sum of powers

$$\sum_{i=1}^n i^2 = \frac{n(n+1)(2n+1)}{6}, \quad \sum_{i=1}^n i^3 = \left(\frac{n(n+1)}{2} \right)^2$$

$$\sum_{i=1}^n i^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$$

$$\sum_{i=1}^n i^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12}$$

Sum of squares Denote $r_k(n)$ the ways to form n with k squares. If :

$$n = 2^{a_0} p_1^{2a_1} \dots p_r^{2a_r} q_1 b_1 \dots q_s b_s$$

where $p_i \equiv 3 \pmod{4}$, $q_i \equiv 1 \pmod{4}$, then

$$r_2(n) = \begin{cases} 0 & \text{if any } a_i \text{ is a half-integer} \\ 4 \prod_1^r (b_i + 1) & \text{if all } a_i \text{ are integers} \end{cases}$$

$r_3(n) > 0$ when and only when n is not $4^a(8b+7)$.

Derangement

$$D_1 = 0, D_2 = 1, D_n = n! \left(\frac{1}{0!} - \frac{1}{1!} + \frac{1}{2!} - \frac{1}{3!} + \dots + \frac{(-1)^n}{n!} \right)$$

$$D_n = (n-1)(D_{n-1} + D_{n-2})$$

Tetrahedron volume If U, V, W, u, v, w are lengths of edges of the tetrahedron (first three form a triangle; u opposite to U and so on)

$$V = \frac{\sqrt{4u^2v^2w^2 - \sum_{cyc} u^2(v^2+w^2-U^2)^2 + \prod_{cyc} (v^2+w^2-U^2)}}{12}$$

Type	Width	Range
signed char	1	127
unsigned char	1	255
short	2	32 767
unsigned short	2	65 535
int	4	2 147 483 647
unsigned int	4	4 294 967 295
long long	8	9 223 372 036 854 775 807
unsigned long long	8	18 446 744 073 709 551 615
float	4	+/- 3.4e +/- 38 (7 digits)
double	8	+/- 1.7e +/- 308 (15 digits)
__float128	16	+/- 1.1e +/- 4932 (31 digits)