

增加

fantasy

目录

1	圆的面积并	1
2	点双连通分量	3
3	边双连通分量	4
4	FFT	5
5	Millar Rabin and Rollard Rho	6
6	模方程	8
7	NTT	10
8	原根	11

1 圆的面积并

```
1 /*
2  * 圆的面积并模板 v1.0
3  */
4 #include <iostream>
5 #include <cstring>
6 #include <cstdio>
7 #include <cmath>
8 #include <vector>
9 #include <algorithm>
10 using namespace std;
11
12 const double pi = acos(-1);
13
14 const double eps = 1e-8;
15 int dcmp(double x) {
16     if(fabs(x) < eps) return 0;
17     else return x < 0 ? -1 : 1;
18 }
19
20 struct Point {
21     double x, y;
22     Point(double x=0, double y=0):x(x),y(y) { }
23 };
24
25 typedef Point Vector;
```

```

26
27 Vector operator + (const Vector& A, const Vector& B) { return Vector(A.x+B.x, A.y+B.y); }
28 Vector operator - (const Point& A, const Point& B) { return Vector(A.x-B.x, A.y-B.y); }
29 Vector operator * (const Vector& A, double p) { return Vector(A.x*p, A.y*p); }
30 bool operator == (const Point& a, const Point& b) { return dcmp(a.x-b.x) == 0 && dcmp(a.y-b.y) == 0; }
31 double Dot(const Vector& A, const Vector& B) { return A.x*B.x + A.y*B.y; }
32 double Length(const Vector& A) { return sqrt(Dot(A, A)); }
33 double Angle(const Vector& A, const Vector& B) { return acos(Dot(A, B) / Length(A) / Length(B)); }
34 double Cross(const Vector& A, const Vector& B) { return A.x*B.y - A.y*B.x; }
35 double angle(Vector v) { return atan2(v.y, v.x); }
36
37 struct Circle {
38     Point c;
39     double r;
40     Circle(Point c = Point(), double r = 0):c(c),r(r) {}
41     Point point(double a) {
42         return Point(c.x + cos(a)*r, c.y + sin(a)*r);
43     }
44 };
45 int getCircleCircleIntersection(Circle C1, Circle C2, vector<Point>& sol) {
46     double d = Length(C1.c - C2.c);
47     if(dcmp(d) == 0) {
48         if(dcmp(C1.r - C2.r) == 0) return -1; // 重合, 无穷多交点
49         return 0;
50     }
51     if(dcmp(C1.r + C2.r - d) < 0) return 0;
52     if(dcmp(fabs(C1.r-C2.r) - d) > 0) return 0;
53
54     double a = angle(C2.c - C1.c);
55     double da = acos((C1.r*C1.r + d*d - C2.r*C2.r) / (2*C1.r*d));
56     Point p1 = C1.point(a-da), p2 = C1.point(a+da);
57
58     sol.push_back(p1);
59     if(p1 == p2) return 1;
60     sol.push_back(p2);
61     return 2;
62 }
63
64 namespace Circle_Union{
65     const int maxn = 1005;
66
67     int n;
68     Circle c[maxn];
69     int vis[maxn];
70     double ans[maxn];
71     struct Node{
72         double ang;
73         int kind;
74         Node(double ang = 0, int kind = 0):ang(ang), kind(kind){}
75         friend bool operator < (const Node &a, const Node &b){
76             if(a.ang == b.ang) return a.kind > b.kind;
77             return a.ang < b.ang;
78         }
79     };
80     void work(int x){
81         for(int i = 1; i <= n; i++) if( i != x ){
82             if(Length(c[i].c - c[x].c) + c[x].r <= c[i].r) {
83                 vis[x]++;
84             }
85         }
86         vector<Node> vec; vec.clear();
87         vector<Point> sol;

```

```

88         for(int i = 1; i <= n; i++) if( i != x ){
89             sol.clear();
90             getCircleCircleIntersection(c[x], c[i], sol);
91             if(sol.size() < 2) continue;
92             if(angle(sol[1] - c[x].c) < angle(sol[0] - c[x].c)){
93                 vec.push_back(Node(pi, -1));
94                 vec.push_back(Node(-pi, 1));
95             }
96             vec.push_back(Node(angle(sol[0] - c[x].c), 1));
97             vec.push_back(Node(angle(sol[1] - c[x].c), -1));
98         }
99         vec.push_back(Node(-pi, 1));
100        vec.push_back(Node(pi, -1));
101        sort(vec.begin(), vec.end());
102        Node last = vec[0]; int cnt = 0;
103        for(int i = 0; i < vec.size(); i++){
104            ans[cnt + vis[x]] += 0.5 * Cross(c[x].point(last.ang) - Point(0,0), c[x].point(vec[i].ang) - Point
(0,0));
105
106            double del = vec[i].ang - last.ang;
107            ans[cnt + vis[x]] += 0.5 * c[x].r * c[x].r * (del - sin(del));
108            last = vec[i];
109            cnt += vec[i].kind;
110        }
111    }
112    void init(){
113        memset(vis, 0, sizeof(vis));
114        memset(ans, 0, sizeof(ans));
115    }
116    void WORK(){ for(int i = 1; i <= n; i++) work(i); }
117 }
118 //usage
119 //调用init()
120 //输入n个圆c[1...n]
121 //调用WORK()
122 //ans[i] 表示被覆盖i次及以上的面积。

```

2 点双连通分量

```

1  const int maxn = 20005;
2  const int maxm = 100005;
3  int n, m;
4  struct Data{
5      int from, to, next, id;
6  }data[maxm * 2];
7  int head[maxn];
8  int tot;
9  void insert(int a, int b, int id){
10     tot++;
11     data[tot].to = b;
12     data[tot].from = a;
13     data[tot].next = head[a];
14     data[tot].id = id;
15     head[a] = tot;
16 }
17 int pre[maxn], lowpt[maxn], time_flag;
18 stack<int> st;
19 void Vbcc(int v, int u){

```

```

20     pre[v] = lowpt[v] = ++ time_flag;
21     for(int k = head[v]; k; k = data[k].next){
22         int w = data[k].to;
23         if(!pre[w]){
24             st.push(k);
25             Vbcc(w, v);
26             lowpt[v] = min(lowpt[v], lowpt[w]);
27             if(lowpt[w] >= pre[v]){
28                 while(!st.empty() && pre[data[st.top()].from] >= pre[w]){
29                     // do what you want to do here
30                     st.pop();
31                 }
32                 // do what you want to do here
33                 st.pop();
34             }
35         } else if(pre[w] < pre[v] && w != u){
36             st.push(k);
37             lowpt[v] = min(lowpt[v], pre[w]);
38         }
39     }
40 }

```

3 边双连通分量

```

1  const int maxn = 20005;
2  const int maxm = 100005;
3  int n, m;
4  struct Data{
5      int from, to, next;
6  }data[maxm * 2];
7  int head[maxn];
8  int tot;
9  void insert(int a, int b){
10     tot++;
11     data[tot].to = b;
12     data[tot].from = a;
13     data[tot].next = head[a];
14     head[a] = tot;
15 }
16
17 int pre[maxn], lowpt[maxn], time_flag;
18 stack<int> st;
19 int ins[maxn];
20 void Ebcc(int v, int u){
21     pre[v] = lowpt[v] = ++time_flag;
22     st.push(v); ins[v] = 1;
23     for(int k = head[v]; k; k = data[k].next){
24         int w = data[k].to;
25         if(!pre[w]){
26             Ebcc(w, v);
27             lowpt[v] = min(lowpt[v], lowpt[w]);
28         } else if(pre[w] < pre[v] && ins[w] && w != u){
29             lowpt[v] = min(lowpt[v], pre[w]);
30         }
31     }
32     if(lowpt[v] == pre[v]){
33         while(!st.empty() && st.top() != v){
34             //do what you want to do here

```

```

35         ins[st.top()] = 0;
36         st.pop();
37     }
38     //do what you want to do here
39     ins[st.top()] = 0;
40     st.pop();
41 }
42 }

```

4 FFT

```

1  /*
2  * FFT 模板v1.0, 注意调用时, 先需要保证  $n = 2^k$ , 并且搞好初始值。
3  * 调用DFT()进行FFT, 调用DFT_h()进行逆运算(插值)。
4  * A数组: [0,n)
5  */
6  #include <iostream>
7  #include <cstring>
8  #include <cstdio>
9  #include <complex>
10 #include <cmath>
11 using namespace std;
12 namespace FFT{
13     const double pi = acos(-1);
14     int inv_flag = 1;
15     complex<double> omega(int n){
16         double u = 2 * pi / n * inv_flag;
17         return complex<double>(cos(u), sin(u));
18     }
19     inline void BitReverseCopy(complex<double> A[], int n, int k){
20         for(int i = 0; i < n; i++){
21             int t = 0;
22             for(int j = 0; j < k; j++) if(i & (1<<j)){ // 二进制位[0,k)
23                 t |= 1 << (k - j - 1);
24             }
25             if(t > i) swap(A[i], A[t]);
26         }
27     }
28     inline void DFT(complex<double> A[], int &n){ // 这个算法会修改n, 不过这样正好
29         int k = 0; while((1LL<<k) < n) k++;
30         for(int i = n, ed = 1LL << k; i < ed; i++) A[i] = 0;
31         n = 1LL << k;
32
33         BitReverseCopy(A, n, k);
34         complex<double> wm, w, t, u;
35         for(int s = 1; s <= k; s++){
36             int m = 1 << s;
37             wm = omega(m);
38             for(int k = 0; k < n; k += m){
39                 w = 1;
40                 for(int j = 0, md = m/2; j < md; j++){
41                     t = w * A[k+j+md], u = A[k+j];
42                     A[k+j] = u + t, A[k+j+md] = u - t;
43                     w *= wm;
44                 }
45             }
46         }
47     }

```

```

48     inline void DFT_h(complex<double> A[], int n){
49         inv_flag = -1;
50         DFT(A,n);
51         inv_flag = 1;
52         for(int i = 0; i < n; i++) A[i] /= n;
53     }
54 }

```

5 Millar Rabin and Rollard Rho

```

1  /*
2   * Millar Rabin & Pollard Rho 模板 v1.0
3   */
4  #include <cstring>
5  #include <iostream>
6  #include <cstdio>
7  #include <vector>
8  #include <algorithm>
9  #include <ctime>
10 using namespace std;
11 typedef long long LL;
12 LL GCD, LCM;
13 inline LL multiply_mod(LL a, LL b, LL mod){
14     LL ans = 0;
15     while(b){
16         if(b & 1){
17             ans += a; ans %= mod;
18         }
19         a += a; a %= mod; b >>= 1;
20     }
21     return ans;
22 }
23 inline LL pow_mod(LL x, LL n, LL mod){
24     x %= mod;
25     LL ans = 1;
26     while(n){
27         if(n & 1){
28             ans = multiply_mod(ans, x, mod); ans %= mod;
29         }
30         x = multiply_mod(x, x, mod); x %= mod; n >>= 1;
31     }
32     return ans;
33 }
34 inline LL add_mod(LL a, LL b, LL mod){
35     return (a + b) % mod;
36 }
37 LL gcd(LL a, LL b){
38     if(b == 0) return a;
39     return gcd(b, a%b);
40 }
41
42 int prime[10000005];
43 int not_prime[10000005];
44 int cnt_prime;
45 const int PRIME_LIM = 10000000;
46 inline void init_prime(){
47     not_prime[1] = 1;
48     for(int i = 2; i <= PRIME_LIM; i++){

```

```

49         if(!not_prime[i]) prime[++cnt_prime] = i;
50         for(int k = 1; k <= cnt_prime; k++){
51             if(prime[k] * i > PRIME_LIM) break;
52             not_prime[prime[k] * i] = 1;
53             if(i % prime[k] == 0) break;
54         }
55     }
56 }
57 inline bool test(LL n, LL b){
58     LL m = n - 1;
59     LL counter = 0;
60     while(~m & 1){
61         m >>= 1;
62         counter++;
63     }
64     LL ret = pow_mod(b, m, n);
65     if(ret == 1 || ret == n - 1){
66         return true;
67     }
68     counter--;
69     while(counter >= 0){
70         ret = multiply_mod(ret, ret, n);
71         if(ret == n - 1){
72             return true;
73         }
74         counter--;
75     }
76     return false;
77 }
78 const int BASE[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43};
79 inline bool is_prime(LL n){
80     if(n < 2) return false;
81     if(n < 4) return true;
82     if((n & 1) == 0) return false;
83     if(n == 3215031751LL){
84         return false;
85     }
86     for(int i = 0; i < 14 && BASE[i] < n; ++i){
87         if(!test(n, BASE[i])){
88             return false;
89         }
90     }
91     return true;
92 }
93 LL pollard_rho(LL n, LL seed){
94     LL x, y, head = 1, tail = 2;
95     x = y = rand() % (n - 1) + 1;
96     while(1){
97         x = multiply_mod(x, x, n);
98         x = add_mod(x, seed, n);
99         if(x == y){
100             return n;
101         }
102         LL d = gcd(abs(x - y), n);
103         if(1 < d && d < n){
104             return d;
105         }
106         head++;
107         if(head == tail){
108             y = x;
109             tail <=> 1;
110         }

```

```

111     }
112 }
113 LL divisors[1005], mul[1005];
114 int tot;
115 void factorize(LL n){
116     if(n > 1){
117         if(is_prime(n)){
118             divisors[++tot] = n;
119         } else {
120             cerr << n << endl;
121             LL d = n;
122             while(d >= n){
123                 d = pollard_rho(n, rand() % (n - 1) + 1);
124             }
125             factorize(n / d);
126             factorize(d);
127         }
128     }
129 }
130 void Factorize(LL n){
131     tot = 0;
132     for(int i = 1; i <= cnt_prime && prime[i] <= n; i++){
133         while(n % prime[i] == 0){
134             divisors[++tot] = prime[i];
135             n /= prime[i];
136         }
137     }
138     factorize(n);
139 }
140
141 //注意 以上 模板要使用小素数先判断之后才能继续使用。

```

6 模方程

```

1 #include <iostream>
2 #include <cstring>
3 #include <cstdio>
4 #include <stack>
5 #include <vector>
6 using namespace std;
7 typedef long long LL;
8 namespace ModEquation{
9 void exgcd(LL a, LL b, LL &x, LL &y){ // calc: ax + by == gcd(a, b)
10     if(b == 0){ x = 1, y = 0; return; }
11     exgcd(b, a % b, y, x); y -= a / b * x;
12 }
13
14 LL gcd(LL a, LL b){ // clac GCD( a, b )
15     return b == 0 ? a : gcd(b, a % b);
16 }
17 bool mod_equation(LL a, LL &x, LL b, LL m) { // calc: ax == b ( mod m ) -> x % m (最小正整数解)
18     b = ((b % m) + m) % m;
19     LL g = gcd(a, m);
20     if(b % g != 0) return false;
21     LL y; exgcd(a, m, x, y);
22     LL md = m/g;
23     x = ((x % md) + md) % md;
24     x %= m;

```



```

25     x = x * (b/g % m) % m;
26     return true;
27 }
28 bool mod_equation(LL a, vector <LL> &ans, LL b, LL m){ // 可以算出模域内所有可能的解
29     ans.clear(); LL x;
30     b = ((b % m) + m) % m;
31     LL g = gcd(a, m);
32     if(b % g != 0) return false;
33     LL y; exgcd(a, m, x, y);
34     LL md = m/g;
35     x = ((x % md) + md) % md;
36     x %= m; x = x * (b/g % m) % m;
37     for(int i = 0; i < g; i++){
38         ans.push_back(x);
39         x = (x + md) % m;
40     }
41     return true;
42 }
43
44 // return the minnum non-negative x
45
46
47 /*
48     clac:
49     / x == b1 (mod m1)
50     | x == b2 (mod m2)
51     | ...
52     \ x == bn (mod mn)
53     return the minnum non-negative x
54 */
55
56 struct Equation{
57     LL b, m;
58     Equation(LL b = 0, LL m = 0):b(b), m(m){}
59 };
60 stack <Equation> st;
61 LL mod_equation_vector(){
62     while(st.size() >= 2){
63         Equation e1 = st.top(); st.pop();
64         Equation e2 = st.top(); st.pop();
65         LL g = gcd(e1.m, e2.m);
66         LL k1;
67         if((e2.b - e1.b) % g == 0 && mod_equation(e1.m/g, k1, (e2.b - e1.b)/g, e2.m/g)){
68             LL m = e1.m / g * e2.m;
69             k1 = ((k1 % m) + m)%m;
70             LL b = (k1 * e1.m) % m + e1.b;
71             b = ((b % m) + m) % m;
72             st.push(Equation(b, m));
73         } else {
74             return -1;
75         }
76     }
77     LL x; Equation e = st.top();
78     mod_equation(1, x, e.b, e.m);
79     return x;
80 }
81 };

```

7 NTT

```
1 #include <iostream>
2 #include <cstring>
3 #include <cstdio>
4 #include <cmath>
5 #include <algorithm>
6 using namespace std;
7 typedef long long LL;
8 namespace NTT{
9     const LL P = 998244353; // == 2^23 * 7 * 17 + 1
10    const LL G = 3; // G^Phi(P) == 1 (mod P)
11    LL g = 3;
12
13    inline LL pow_mod(LL x, LL n, LL mod){
14        LL ans = 1;
15        while(n){
16            if(n & 1){ ans *= x; ans %= mod; }
17            x *= x; x %= mod; n >>= 1;
18        }
19        return ans;
20    }
21    inline LL inv(LL x){
22        return pow_mod(x, P-2, P);
23    }
24    inline LL omega(int n){
25        LL u = (P-1) / n;
26        return pow_mod(g, u, P);
27    }
28    inline void BitReverseCopy(LL A[], int n, int k){
29        for(int i = 0; i < n; i++){
30            int t = 0;
31            for(int j = 0; j < k; j++) if(i & (1<<j)) // 二进制位[0,k)
32                t |= 1 << (k - j - 1);
33            if(t > i) swap(A[i], A[t]);
34        }
35    }
36    inline void DFT(LL A[], int &n){
37        int k = 0; while((1LL<<k) < n) k++;
38        for(int i = n, ed = 1LL << k; i < ed; i++) A[i] = 0;
39        n = 1LL << k;
40        BitReverseCopy(A, n, k);
41        LL wm, w, t, u;
42        for(int s = 1; s <= k; s++){
43            int m = 1 << s;
44            wm = omega(m);
45            for(int k = 0; k < n; k += m){
46                w = 1;
47                for(int j = 0, md = m/2; j < md; j++){
48                    t = w * A[k+j+md] % P, u = A[k+j];
49                    A[k+j] = u + t, A[k+j+md] = u - t + P;
50                    A[k+j] -= A[k+j] >= P ? P : 0;
51                    A[k+j+md] -= A[k+j+md] >= P ? P : 0;
52                    w = (w * wm) % P;
53                }
54            }
55        }
56    }
57    inline void DFT_h(LL A[], int n){
58        g = inv(G);
59        DFT(A, n);
60    }
```

```

60         g = G;
61         LL invN = inv(n);
62         for(int i = 0; i < n; i++) A[i] = A[i] * invN % P;
63     }
64 }
65 // 使用DFT进行NT变换
66 // 使用DFT_h进行逆变换
67 // 因为模数的关系，n不能超过1e6

```

8 原根

```

1 #include <iostream>
2 #include <cstring>
3 #include <cstdio>
4 #include <unordered_map>
5 #include <vector>
6 #include <cmath>
7 #include <algorithm>
8 using namespace std;
9 typedef long long LL;
10 namespace PrimitiveRoot{
11     LL pow_mod(LL x, LL n, LL mod){
12         LL ans = 1;
13         while(n){
14             if(n & 1){ ans *= x; ans %= mod; }
15             x *= x; x %= mod; n >>= 1;
16         }
17         return ans;
18     }
19     LL Phi(LL n){ // 计算n的欧拉函数
20         LL ans = n;
21         for(int i = 2; i * i <= n; i++) if(n % i == 0) {
22             ans = ans / i * (i-1);
23             while(n % i == 0) n /= i;
24         }
25         if(n > 0) ans = ans / n * (n-1);
26         return ans;
27     }
28     void exgcd(LL a, LL b, LL &x, LL &y){ // calc: ax + by == gcd(a, b)
29         if(b == 0){ x = 1, y = 0; return; }
30         exgcd(b, a % b, y, x); y -= a / b * x;
31     }
32
33     LL gcd(LL a, LL b){ // clac GCD( a, b )
34         return b == 0 ? a : gcd(b, a % b);
35     }
36     bool mod_equation(LL a, LL &x, LL b, LL m) { // calc: ax == b ( mod m ) -> x % m
37         b = ((b % m) + m) % m;
38         LL g = gcd(a, m);
39         if(b % g != 0) return false;
40         LL y; exgcd(a, m, x, y);
41         LL md = m/g;
42         x = ((x % md) + md) % md;
43         x %= m;
44         x = x * (b/g % m) % m;
45         return true;
46     }
47     bool mod_equation(LL a, vector <LL> &ans, LL b, LL m){ //计算模方程所有可能的解

```

```

48     ans.clear(); LL x;
49     b = ((b % m) + m) % m;
50     LL g = gcd(a, m);
51     if(b % g != 0) return false;
52     LL y; exgcd(a, m, x, y);
53     LL md = m/g;
54     x = ((x % md) + md) % md;
55     x %= m; x = x * (b/g % m) % m;
56     for(int i = 0; i < g; i++){
57         ans.push_back(x);
58         x = (x + md) % m;
59     }
60     return true;
61 }
62
63 LL root, phi, P;
64
65 LL inv(LL a){
66     LL x; mod_equation(a, x, 1, P);
67     return x;
68 }
69 void set_mod(LL P_) { // P = 2, 4, p^a, 2*p^a (其中p是奇素数)
70     P = P_;
71 }
72 vector <LL> factor;
73 LL get_primitive_root(){ //get x^{Phi(P)} == 1 (mod P) -> x是P的原根 0(玄)
74     phi = Phi(P); // P = 2, 4, p^a, 2*p^a (其中p是奇素数)
75     factor.clear(); int tmp = phi;
76     for(int i = 2; i * i <= phi; i++) if(tmp % i == 0){
77         factor.push_back(i);
78         while(tmp % i == 0) tmp /= i;
79     }
80     if(tmp > 1) factor.push_back(tmp);
81     for(int i = 1; i <= P; i++){
82         bool flag = pow_mod(i, phi, P) == 1;
83         if(!flag) continue;
84         for(int j = 0; j < factor.size(); j++){
85             if(pow_mod(i, phi/factor[j], P) == 1){
86                 flag = 0; break;
87             }
88         }
89         if(flag) return root = i;
90     }
91     return -1;
92 }
93 unordered_map <LL, LL> mp;
94 void D_log_prework(){ // 大步小步法的预处理
95     mp.clear(); LL q = sqrt(phi) + 1;
96     for(int i = 0; i * q <= phi - 1; i++)
97         mp[pow_mod(root, i * q, P)] = i * q;
98 }
99 LL D_log(LL x){ // 使用大步小步法计算x模P的离散对数
100     x %= P;
101     LL q = sqrt(phi) + 1;
102     LL inv_root = inv(root);
103     for(int i = 0; i < q; i++){
104         LL right = pow_mod(inv_root, i, P) * x % P;
105         if(mp.count(right)){
106             return mp[right] + i;
107         }
108     }
109     return -1;

```

```

110 }
111 bool n_times_remainder(vector <LL> x, LL K, LL A, LL P){ // clac:  $x^K \equiv A \pmod{P}$  的所有的x, 且按照升序
112     x.clear();
113     if(A == 0) { x.push_back(0); return true; } // log(0), 特判
114     else if(P == 2) { x.push_back(A); return true; } // 底为1, 特判
115     else {
116         set_mod(P);
117         get_primitive_root();
118         D_log_prework();
119         LL b = D_log(A);
120         if(b == -1){
121             return false;
122         }
123
124         mod_equation(K, x, b, phi);
125         for(int i = 0; i < x.size(); i++){
126             x[i] = pow_mod(root, x[i], P);
127         }
128         sort(x.begin(), x.end());
129         return true;
130     }
131 }
132 };
133
134 //usage:
135 //要使用n_times_remainder的话, 直接调用即可。
136 //
137 //只要求P的原根时
138 //set_mod(P); ans = get_primitive_root();
139 //
140 //只要求A关于P的离散对数时
141 //set_mod(P); get_primitive_root();
142 //D_log_prework(); ans = D_log(A);

```