圆的面积并问题和一些拓展

问题

给 N 个圆，求其面积并□ $N \le 1000$

解法 1

扫描线 + 辛普森积分。这里被积函数 $f(t)$ 为 $x=t$ 这条直线被圆覆盖的长度。

在实现时考虑辛普森积分的特点，需要预处理出圆在 x 轴的投影。

code

```
#include <bits/stdc++.h>
using namespace std;
const double eps = 1e-8;
const int maxn = 1e3 + 23;

struct Circle{
    double x, y, r;
}a[maxn];

int n;

double f(double x){
    vector<pair<double, int>> v;
    for(int i = 1; i <= n; i++){
        if (x >= a[i].x + a[i].r || x <= a[i].x - a[i].r) continue;
        double d = a[i].r * a[i].r - (x - a[i].x) * (x - a[i].x);
        d = sqrt(d);
        v.push_back({a[i].y - d, 1});
        v.push_back({a[i].y + d, -1});
    }
    sort(v.begin(), v.end());
    double ret = 0;
    for(int i = 0, cnt = 0; i < (int)v.size() - 1; i++){
        cnt += v[i].second;
        if(cnt) ret += v[i + 1].first - v[i].first;
    }
    return ret;
}

double calc(double l, double r){
    return (r - l) * (f(l) + f(r) + 4 * f((l + r) / 2)) / 6;
}
```

```
double Simpson(double l, double r, double ans){
    double mid = (l + r) /2, ans1 = calc(l, mid), ans2 = calc(mid, r);
    if (fabs(ans1 + ans2 - ans) <= eps) return ans;
    return Simpson(l, mid, ans1) + Simpson(mid, r, ans2);
}

int main(){
    cin >> n;
    vector<pair<double, int>> v;
    for (int i = 1; i <= n; i++){
        cin >> a[i].x >> a[i].y >> a[i].r;
        v.push_back({a[i].x - a[i].r, 1});
        v.push_back({a[i].x + a[i].r, -1});
    }
    sort(v.begin(), v.end());
    double ret = 0;
    int last = -1, cnt = 0;
    for (int i = 0; i < (int)v.size(); i++){
        cnt += v[i].second;
        if (cnt > 0 && last == -1) last = i;
        else if (cnt == 0 && last != -1) ret += Simpson(v[last].first,
v[i].first, 1e20), last = -1;
    }

    printf("%.4f\n", ret);
}
```

解法 2

考虑对最终形成的图形的轮廓线上应用格林公式进行曲线积分。

对圆 $Circle(x_0,y_0,r)$ 从 θ_1 到 θ_2 的一段圆弧，对最终答案的贡献为 $$ \begin{eqnarray}S&=&\iint_{D}1 dx dy\\&=&\frac{1}{2}\oint_{C}xdy-ydx\\&=&\frac{1}{2}\int_{\theta_1}^{\theta_2} (x_0+r\cdot cos\theta)d(y_0+r\cdot sin\theta)-(y_0+r\cdot sin\theta)d(x_0+r\cdot cos\theta)\\&=&\frac{r}{2}\int_{\theta_1}^{\theta_2} [(x_0+r\cdot cos\theta)\cdot cos\theta +(y_0+r\cdot sin\theta)\cdot sin\theta]d\theta\\&=&\frac{r}{2}\int_{\theta_1}^{\theta_2} (x_0\cdot cos\theta + y_0\cdot sin\theta+r)d\theta\\&=&\frac{1}{2}\cdot(f(r,x_0,y_0,\theta_2)-f(r,x_0,y_0,\theta_1))\end{eqnarray} $$ 其中$f(r,x,y,\theta)=r^2\cdot \theta+r\cdot x\cdot sin\theta-r\cdot y\cdot cos\theta$

因此我们需要求出每个圆在 N 个圆并的轮廓线上占的部分。

首先去重以及移去被内含的圆。之后对每个圆 i 求出所有其他圆 j 和其的交点 (u,v)在圆 i 的 (u,v) 这段劣弧上区间 $+1$。最终圆 i 上为 0 的部分会出现在轮廓线上。

时间复杂度 $O(n^2\log n)$ 在实践中一般优于解法 1。

下面是一个可读性比较差的实现

```cpp
#include <bits/stdc++.h>
using namespace std;
typedef pair<double,int> pdd;
const double eps=1e-10;
const double pi=acos(-1.0);
const double pi2=2*pi;
const int maxn=1e3+23;
struct Point{
    double x,y;
    Point(double x=0,double y=0):x(x),y(y) {}
    bool operator < (const Point &b) const
    {
        return x<b.x||(x==b.x&&y<b.y);
    }
};
typedef Point Vector;
Point operator + (Point A,Point B){return Point(A.x+B.x,A.y+B.y);}
Point operator - (Point A,Point B) {return Point(A.x-B.x,A.y-B.y);}
Point operator * (Point A,double B) {return Point(A.x*B,A.y*B);}
Point operator / (Point A,double B) {return Point(A.x/B,A.y/B);}
int dcmp(double x)
{
    if(fabs(x)<eps) return 0;
    return x<0 ? -1 : 1;
}
bool operator == (const Point &a,const Point &b)
{
    return dcmp(a.x-b.x)==0&&dcmp(a.y-b.y)==0;
}

double dot(Vector a,Vector b)
{
    return a.x*b.x+a.y*b.y;
}
double length(Vector a)
{
    return sqrt(dot(a,a));
}
double angle(Vector v)
{
    return atan2(v.y,v.x);
}
double cross(Vector a,Vector b)
{
    return a.x*b.y-a.y*b.x;
}
struct Circle
{
    Point c;double r;
    Point point(double a)
    {
```

```
        return Point(c.x+cos(a)*r,c.y+sin(a)*r);
    }
    void read()
    {
        scanf("%lf%lf",&c.x,&c.y);
    }
    double oint(double t1,double t2)
    {
        return r*(r*(t2-t1)+c.x*(sin(t2)-sin(t1))-c.y*(cos(t2)-cos(t1)));
    }
    bool operator < (const Circle rhs) const
    {
        return c<rhs.c||(c==rhs.c&&r<rhs.r);
    }
    bool operator == (const Circle rhs) const
    {
        return c==rhs.c&&(dcmp(r-rhs.r)==0);
    }
}a[maxn];
pdd p[maxn<<2];int cnt,pb;
void getCircleIntersection(const Circle &C1,const Circle &C2)
{
    double d=length(C1.c-C2.c);
    if((C1.r+C2.r-d)<=0||C1.r-C2.r-d>=0){
        return ;
    }
    if(((C2.r-C1.r)-d)>=0) {pb++;return ;}
    double a=angle(C2.c-C1.c);
    double da=acos((C1.r*C1.r+d*d-C2.r*C2.r)/(2*C1.r*d));
    double l=a-da,r=a+da;
    if(l<-pi) l+=pi2;
    if(r>=pi) r-=pi2;
    if(l>r) pb++;
    p[cnt++]={l,1};
    p[cnt++]={r,-1};
}
int n;

double cal(int x)
{
    double ans=0;cnt=0;
    pb=0;
    for(int i=1;i<=n;i++){
        if(i==x) continue;
        getCircleIntersection(a[x],a[i]);
    }
    p[cnt++]={-pi,0};p[cnt++]={pi,0};
    sort(p,p+cnt);
    for(int i=1;i<cnt;i++){
        if(pb==0) ans+=a[x].oint(p[i-1].first,p[i].first);
```

```
        pb+=p[i].second;
    }
    return ans;
}

int main(){
    cin>>n;
    for(int i=1;i<=n;i++){
        scanf("%lf%lf%lf",&a[i].c.x,&a[i].c.y,&a[i].r);
    }
    double ans=0;
    for(int i=1;i<=n;i++) ans+=cal(i);
    printf("%.4f",ans/2);
}
```

一些拓展问题

问题 1

给 N 个圆，每个圆出现的概率为 p_i ，求面积并的期望， $N \le 1000$

解法

将扫描线的区间加转换为区间乘概率或轮廓上的区间加转为区间乘概率即可。

code

问题 2

给 N 个圆，求被至少 k 个圆覆盖部分的面积。

解法 1

扫描线求被覆盖至少 k 次的长度之后辛普森积分。

解法 2

被覆盖至少 k 次部分的轮廓线由恰好被覆盖 $k-1$ 次的圆弧组成。

code

PS：上面两题笔者的辛普森积分均不能通过，可能是辛普森积分的姿势太低了。