珍藏多年的计算几何模板

By whatbeg

发表于 2016-09-20 总阅读230次

文章目录
1．说明
2．模板

说明

珍藏多年的从前自己收集整理的一套计算几何的模板，如今也很少会用到了，不如公布出来供给广大ACMer学习参考之用，也算是没白费这一番功夫了。

模板包括二维计算几何的绝大多数封装，包括

- 点，线，圆的基础定义
- 向量操作
- 点到线段的距离
- 点到直线的距离
- 判断直线相交并求交点
- 判断线段相交并求交点
- 向量旋转
- 环顾法、射线法判断点是否在多边形内部
- 判断未知时针方向的多边形是否是凸包
- 求二维凸包
- 求凸包面积
- 求凸包周长
- 旋转卡壳求凸包最远两点
- 求两个凸包的最短距离
- 多边形的边转为直线
- 求半平面交
- 直线切割多边形形成新的多边形
- 判断点是否在圆内
- 求两个圆的交点
- 求线段与圆的交点

- 求三角形与圆的交点

等等，并且都通过了基础题的测试，形成一套可用的模板体系。

大家有需要即取，不过模板虽好，必也得自己对算法有一番深入的学习才好，才有见效。

文章目录
1．说明
2．模板

模板

C++语言

```
struct Point{
    double x,y;
    Point(double x=0, double y=0):x(x),y(y) {}
    void input() { scanf("%lf%lf",&x,&y); }
};
typedef Point Vector;
struct Circle{
    Point c;
    double r;
    Circle(){}
    Circle(Point c,double r):c(c),r(r) {}
    Point point(double a) { return Point(c.x + cos(a)*r, c.y + sin(a)*r
    void input() { scanf("%lf%lf%lf",&c.x,&c.y,&r); }
};
struct Line{
    Point p;
    Vector v;
    double ang;
    Line(){}
    Line(Point p, Vector v):p(p),v(v) { ang = atan2(v.y,v.x); }
    Point point(double t) { return Point(p.x + t*v.x, p.y + t*v.y); }
    bool operator < (const Line &L)const { return ang < L.ang; }
};

int dcmp(double x) {
    if(x < -eps) return -1;
    if(x > eps) return 1;
    return 0;
}
template <class T> T sqr(T x) { return x * x;}
Vector operator + (Vector A, Vector B) { return Vector(A.x + B.x, A.y +
Vector operator - (Vector A, Vector B) { return Vector(A.x - B.x, A.y -
Vector operator * (Vector A, double p) { return Vector(A.x*p, A.y*p); }
Vector operator / (Vector A, double p) { return Vector(A.x/p, A.y/p); }
bool operator < (const Point& a, const Point& b) { return a.x < b.x ||
bool operator >= (const Point& a, const Point& b) { return a.x >= b.x &
bool operator <= (const Point& a, const Point& b) { return a.x <= b.x &
bool operator == (const Point& a, const Point& b) { return dcmp(a.x-b.x
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 A, Vector B) { return acos(Dot(A, B) / Length(A) / 
double Cross(Vector A, Vector B) { return A.x*B.y - A.y*B.x; }
Vector VectorUnit(Vector x){ return x / Length(x);}
Vector Normal(Vector x) { return Point(-x.y, x.x) / Length(x);}
```

```
44  double angle(Vector v) { return atan2(v.y, v.x); }
45
46  bool OnSegment(Point P, Point A, Point B) {          //端点也算
47      return dcmp(Cross(A-P,B-P)) == 0 && dcmp(Dot(A-P,B-P)) <= 0;
48  }
49  double DistanceToSeg(Point P, Point A, Point B) {
50      if(A == B) return Length(P-A);
51      Vector v1 = B-A, v2 = P-A, v3 = P-B;
52      if(dcmp(Dot(v1, v2)) < 0) return Length(v2);
53      if(dcmp(Dot(v1, v3)) > 0) return Length(v3);
54      return fabs(Cross(v1, v2)) / Length(v1);
55  }
56  double DistanceToLine(Point P, Point A, Point B) {
57      Vector v1 = B-A, v2 = P-A;
58      return fabs(Cross(v1,v2)) / Length(v1);
59  }
60  Point GetLineIntersection(Line A, Line B) {
61      Vector u = A.p - B.p;
62      double t = Cross(B.v, u) / Cross(A.v, B.v);
63      return A.p + A.v*t;
64  }
65  double DisP(Point A,Point B) {
66      return Length(B-A);
67  }
68  bool SegmentIntersection(Point A,Point B,Point C,Point D) {
69      return max(A.x,B.x) >= min(C.x,D.x) &&
70             max(C.x,D.x) >= min(A.x,B.x) &&
71             max(A.y,B.y) >= min(C.y,D.y) &&
72             max(C.y,D.y) >= min(A.y,B.y) &&
73             dcmp(Cross(C-A,B-A)*Cross(D-A,B-A)) <= 0 &&
74             dcmp(Cross(A-C,D-C)*Cross(B-C,D-C)) <= 0;
75  }
76  bool LineSegmentIntersection(Point A,Point B,Point C,Point D) {
77      return dcmp(Cross(C-A,B-A)*Cross(D-A,B-A)) <= 0;
78  }
79  void SegIntersectionPoint(Point& P,Point a,Point b,Point c,Point d) {
80      P.x = (Cross(d-a,b-a)*c.x - Cross(c-a,b-a)*d.x)/(Cross(d-a,b-a)-Cros
81      P.y = (Cross(d-a,b-a)*c.y - Cross(c-a,b-a)*d.y)/(Cross(d-a,b-a)-Cros
82  }
83  Vector Rotate(Point P,Vector A,double rad){     //以P为基准点把向量A旋转rad
84      return Vector(P.x+A.x*cos(rad)-A.y*sin(rad),P.y+A.x*sin(rad)+A.y*cos
85  }
86  //点是否在多边形内部（环顾法）
87  int CheckPointInPolygon(Point A,Point* p,int n){
88      double TotalAngle = 0.0;
89      for(int i=0;i<n;i++) {
90          if(dcmp(Cross(p[i]-A,p[(i+1)%n]-A)) >= 0) TotalAngle += Angle(p
91          else TotalAngle -= Angle(p[i]-A,p[(i+1)%n]-A);
92      }
93      if(dcmp(TotalAngle) == 0)                  return 0;   //外部
94      else if(dcmp(fabs(TotalAngle)-2*pi) == 0) return 1;   //完全内部
95      else if(dcmp(fabs(TotalAngle)-pi) == 0)   return 2;   //边界上
96      else                                       return 3;   //多边形顶点
97  }
98  //射线法
99  int Ray_PointInPolygon(Point A,Point* p,int n) {
100     int wn = 0;
101     for(int i=0;i<n;i++) {
```

文章目录
1．说明
2．模板

```
102          //if(OnSegment(A,p[i],p[(i+1)%n])) return -1;     //边界
103          int k = dcmp(Cross(p[(i+1)%n]-p[i], A-p[i]));
104          int d1 = dcmp(p[i].y-A.y);
105          int d2 = dcmp(p[(i+1)%n].y-A.y);
106          if(k > 0 && d1 <= 0 && d2 > 0) wn++;
107          if(k < 0 && d2 <= 0 && d1 > 0) wn--;
108      }
109      if(wn) return 1;      //内部
110      return 0;             //外部
111  }
112  //判断未知时针方向的多边形是否是凸包
113  bool CheckConvexHull(Point* p,int n){
114      int dir = 0;    //旋转方向
115      for(int i=0;i<n;i++) {
116          int nowdir = dcmp(Cross(p[(i+1)%n]-p[i],p[(i+2)%n]-p[i]));
117          if(!dir) dir = nowdir;
118          if(dir*nowdir < 0) return false;      //非凸包
119      }
120      return true;
121  }
122  //////凸包
123  int ConvexHull(Point* p, int n, Point* ch) {
124      sort(p,p+n);
125      int m = 0;
126      for(int i=0;i<n;i++) {
127          while(m > 1 && Cross(ch[m-1]-ch[m-2], p[i]-ch[m-2]) <= 0) m--;
128          ch[m++] = p[i];
129      }
130      int k = m;
131      for(int i=n-2;i>=0;i--) {
132          while(m > k && Cross(ch[m-1]-ch[m-2], p[i]-ch[m-2]) <= 0) m--;
133          ch[m++] = p[i];
134      }
135      if(n > 1) m--;
136      return m;
137  }
138  double CalcConvexArea(Point* p,int n) {         //凸包面积
139      double area = 0.0;
140      for(int i=1;i<n-1;i++)
141          area += Cross(p[i]-p[0],p[i+1]-p[0]);
142      return fabs(area*0.5);
143  }
144  double CalcConvexLength(Point* p,int n) {
145      double Len = 0.0;
146      for(int i=0;i<n;i++) Len += Length(p[(i+1)%n]-p[i]);
147      return Len;
148  }
149  ///////////旋转卡壳求凸包最远两点
150  double RotatingCalipers(Point* ch,int n) {  //旋转卡壳
151      int p,q = 1;
152      double ans = 0.0;
153      ch[n] = ch[0];
154      for(p=0;p<n;p++) {
155          while(dcmp(Cross(ch[p+1]-ch[p],ch[q+1]-ch[p])-Cross(ch[p+1]-ch[
156              q = (q+1)%n;
157          ans = max(ans,max(DisP(ch[p],ch[q]),DisP(ch[p+1],ch[q+1])));
158      }
159      return ans*ans;
```

文章目录
1．说明
2．模板

```
}
double MinDisOfTwoConvexHull(Point P[],int n,Point Q[],int m) {   //旋转
    int Pymin = 0, Qymax = 0, i,j;
    for(i=0;i<n;i++) if(dcmp(P[i].y-P[Pymin].y) < 0) Pymin = i;
    for(i=0;i<m;i++) if(dcmp(Q[i].y-Q[Qymax].y) > 0) Qymax = i;
    P[n] = P[0], Q[m] = Q[0];
    double Mindis = Mod, Tmp;
    for(i=0;i<n;i++) {
        while(dcmp(Tmp = Cross(P[Pymin+1]-P[Pymin],Q[Qymax+1]-P[Pymin])
            Qymax = (Qymax+1)%m;
        if(dcmp(Tmp) < 0) Mindis = min(Mindis,DistanceToSeg(Q[Qymax],P[
        else              Mindis = min(Mindis,SegDistancetoSeg(P[Pymin],
        Pymin = (Pymin+1)%n;
    }
    return Mindis;
}
bool OnLeft(Line L, Point p) { return dcmp(Cross(L.v,p-L.p)) > 0; }
Point* p;
bool CmpPolarPoint(Point a,Point b) {     //点极角排序
    int d = dcmp(Cross(a-p[0],b-p[0]));
    if(!d) return DisP(p[0],a) < DisP(p[0],b);
    return d > 0;
}
bool CmpPolarLine(Line a,Line b) {        //直线极角排序
    return angle(a.v) < angle(b.v);
}
void GetL(bool counter,Point* p,int n,Line* L) {  //多边形的边转为直线
    if(counter) { for(int i=n-1;i>=0;i--) L[n-i-1] = Line(p[(i+1)%n],p[
    else { for(int i=0;i<n;i++) L[i] = Line(p[i],p[(i+1)%n]-p[i]); }
}
int HalfPlaneIntersection(Line* L, int n, Point* poly) {    //半平面交点在
    sort(L,L+n,CmpPolarLine);
    int first,last;
    Point *p = new Point[n];
    Line  *q = new Line[n];
    q[first=last=0] = L[0];
    for(int i=1;i<n;i++) {
        while(first < last && !OnLeft(L[i],p[last-1])) last--;
        while(first < last && !OnLeft(L[i],p[first]))  first++;
        q[++last] = L[i];
        if(dcmp(Cross(q[last].v, q[last-1].v)) == 0) {
            last--;
            if(OnLeft(q[last], L[i].p)) q[last] = L[i];
        }
        if(first < last) p[last-1] = GetLineIntersection(q[last-1],q[las
    }
    while(first < last && !OnLeft(q[first],p[last-1])) last--;
    if(last-first <= 1) return 0;       //点或线或无界平面，返回0
    p[last] = GetLineIntersection(q[last],q[first]);
    int m = 0;
    for(int i=first;i<=last;i++) poly[m++] = p[i];
    delete p; delete q;
    return m;
}
int LineCrossPolygon(Point& L1,Point& L2,Point* p,int n,Point* poly) {
    int m = 0;
    for(int i=0,j;i<n;i++) {
        if(dcmp(Cross(L1-p[i],L2-p[i])) >= 0) { poly[m++] = p[i]; conti
```

文章目录

1. 说明
2. 模板

```
218          j = (i-1+n)%n;
219          if(dcmp(Cross(L1-p[j],L2-p[j])) > 0) poly[m++] = GetLineIntersec
220          j = (i+1+n)%n;
221          if(dcmp(Cross(L1-p[j],L2-p[j])) > 0) poly[m++] = GetLineIntersec
222      }
223      return m;
224  }
225  //////圆
226  bool InCircle(Point x, Circle c) { return dcmp(c.r - Length(c.c-x)) > 0;
227  int GetCircleCircleIntersection(Circle C1, Circle C2, vector<Point>& sol
228  {
229      double d = Length(C1.c - C2.c);
230      if(dcmp(d) == 0){
231          if(dcmp(C1.r - C2.r) == 0) return -1;  //两圆重合
232          return 0;
233      }
234      if(dcmp(C1.r + C2.r - d) < 0) return 0;
235      if(dcmp(fabs(C1.r - C2.r) - d) > 0) return 0;
236
237      double a = angle(C2.c - C1.c);              //向量C1C2的极角
238      double da = acos((sqr(C1.r) + sqr(d) - sqr(C2.r)) / (2*C1.r*d)); //c
239
240      Point p1 = C1.point(a-da), p2 = C1.point(a+da);
241      sol.push_back(p1);
242      if(p1 == p2) return 1;
243      sol.push_back(p2);
244      return 2;
245  }
246  int GetSegCircleIntersection(Line L, Circle C, Point* sol)
247  {
248      Vector Noml = Normal(L.v);
249      Line PL = Line(C.c, Noml);
250      Point IP = GetLineIntersection(PL, L); //弦的中点
251      double Dis = Length(IP - C.c);
252      if(dcmp(Dis-C.r) > 0) return 0;        //在圆外
253      Vector HalfChord = VectorUnit(L.v)*sqrt(sqr(C.r)-sqr(Dis));
254      int ind = 0;
255      sol[ind] = IP + HalfChord;
256      if(OnSegment(sol[ind],L.p,L.point(1))) ind++;
257      sol[ind] = IP - HalfChord;
258      if(OnSegment(sol[ind],L.p,L.point(1))) ind++;
259      return ind;
260  }
261
262  Point Zero = Point(0,0);
263  double TriAngleCircleInsection(Circle C, Point A, Point B)
264  {
265      Vector OA = A-C.c, OB = B-C.c;
266      Vector BA = A-B, BC = C.c-B;
267      Vector AB = B-A, AC = C.c-A;
268      double DOA = Length(OA), DOB = Length(OB),DAB = Length(AB), r = C.r;
269      if(dcmp(Cross(OA,OB)) == 0) return 0;
270      if(dcmp(DOA-C.r) < 0 && dcmp(DOB-C.r) < 0) return Cross(OA,OB)*0.5;
271      else if(DOB < r && DOA >= r) {
272          double x = (Dot(BA,BC) + sqrt(r*r*DAB*DAB-Cross(BA,BC)*Cross(BA,
273          double TS = Cross(OA,OB)*0.5;
274          return asin(TS*(1-x/DAB)*2/r/DOA)*r*r*0.5+TS*x/DAB;
275      }
```

文章目录
1. 说明
2. 模板

```
276       else if(DOB >= r && DOA < r) {
277           double y = (Dot(AB,AC)+sqrt(r*r*DAB*DAB-Cross(AB,AC)*Cross(AB,A(
278           double TS = Cross(OA,OB)*0.5;
279           return asin(TS*(1-y/DAB)*2/r/DOB)*r*r*0.5+TS*y/DAB;
280       }
281       else if(fabs(Cross(OA,OB)) >= r*DAB || Dot(AB,AC) <= 0 || Dot      3C)
282           if(Dot(OA,OB) < 0) {
283               if(Cross(OA,OB) < 0) return (-acos(-1.0)-asin(Cross(OA,OB)/I
284               else                 return ( acos(-1.0)-asin(Cross(OA,OB)/I
285           }
286           else                     return asin(Cross(OA,OB)/DOA/DOB)*r*r*(
287       }
288       else {
289           double x = (Dot(BA,BC)+sqrt(r*r*DAB*DAB-Cross(BA,BC)*Cross(BA,B(
290           double y = (Dot(AB,AC)+sqrt(r*r*DAB*DAB-Cross(AB,AC)*Cross(AB,A(
291           double TS = Cross(OA,OB)*0.5;
292           return (asin(TS*(1-x/DAB)*2/r/DOA)+asin(TS*(1-y/DAB)*2/r/DOB))*
293       }
294   }
```

文章目录
1. 说明
2. 模板

我要小额赞助，助作者写出更好的文章!

算法 | Algorithm 算法

上一篇:

多台虚拟机搭建模拟网络环境

下一篇:

Hexo部署环境迁移--多台电脑搭建Hexo环境