University of Central Florida

UCF X

Tyler Mark, X, X

2022-08-16

Contest (1)

template.cpp 23 lines

```
#include <bits/stdc++.h>
#define all(x) begin(x), end(x)
using namespace std;

#define rep(i, a, b) for(int i = a; i < (b); ++i)
#define sz(x) (int) (x).size()
#define all(x) begin(x), end(x)

using ll = long long;
using ld = long double;
using pii = pair<int, int>;
using vi = vector<int>;
using vii = vector<pii>;
using vvi = vector<vi>;

int main() {
    cin.tie(0)->sync_with_stdio(0);
    cin.exceptions(cin.failbit);

    return 0;
}
```

Data structures (2)

Geometry (3)

point.h

Description: Handles vector/point operations in the cartesian plane. Works for any type. Ints may cause errors with integer division.

8aa3bf, 21 lines

```
const ld eps = 1e-9; //tolerance based on problem
const ld inf = 1e18;

template<class T> struct pnt {
    T x, y;
    pnt(T _x = 0, T _y = 0): x(_x), y(_y) {}
    bool operator<(pnt o) const { return tie(x, y) < tie(o.x, o
        .y); }
    pnt operator+(pnt o) { return pnt(x + o.x, y + o.y); }
    bool operator==(pnt o) const { return abs(x-o.x) < eps &&
        abs(y-o.y) < eps; }
    pnt operator-(pnt o) { return pnt(x - o.x, y - o.y); }
    pnt operator*(T c) { return pnt(x*c, y*c); }
    T dot(pnt o) { return x*o.x + y*o.y; }
    T cross(pnt o) { return x*o.y - y*o.x; }
    T cross(pnt o1, pnt o2) { return (o1-*this).cross(o2-*this)
        ; }
    T mag2() { return dot(*this); }
    ld mag() { return sqrtl(dist2()); }
    pnt unit() { return *this * (1/dist()); }
    pnt perp() { return pnt(-y, x); }
    pnt norm() { return perp().unit(); }
    ld ang() { return atan2l(y, x); }
};
```

3.1 Lines and Segments

sideOf.h

Description: Checks what side of the line a point is on. Returns -1 if the point is left, 1 if right, and 0 if on the line. Orientation is based off the unit vector of the line

6c1c12, 4 lines

```
template<class P> int sideOf(P s, P e, P p) {
    auto cp = s.cross(e, p);
    return (cp > eps) - (cp < -eps);
}
```

onSeg.h

Description: Returns whether or not a point is on a segment

c2edf6, 3 lines

```
template<class P> bool onSeg(P s, P e, P p) {
    return abs(s.cross(e, p)) < eps && (s-p).dot(e-p) < eps;
}
```

lineIntersection.h

Description: Returns the intersection point of two lines using cramers rule. If the lines are parallel or are the same, (inf, inf) is returned.

fe20f6, 6 lines

```
template<class P> P lineInter(P s1, P e1, P s2, P e2) {
    auto det = -(e1-s1).cross(e2-s2);
    if(abs(det) < eps) return P(inf, inf);
    auto t = (e2-s2).cross(s2-s1) / det;
    return s1 + (e1-s1) * t;
}
```

doSegIntersection.h

Description: Checks if two segments intersect (inclusive of intersections at endpoints)

59a2a4, 4 lines

```
template<class P> bool doSegInter(P s1, P e1, P s2, P e2) {
    return sideOf(s1, e1, s2) != sideOf(s1, e1, e2) &&
           sideOf(s2, e2, s1) != sideOf(s2, e2, e1);
}
```

segIntersection.h

Description: Returns the intersection point of two segments.

Usage: Returns a vector of points. If no points, there is no intersection. If 1 point, the segments intersect at a distinct point. If 2 points, the segments intersect at a segment of points, where the 2 points are the end points.

71bb02, 9 lines

```
template<class P> vector<P> segInter(P s1, P e1, P s2, P e2) {
    if(doSegInter(s1, e1, s2, e2)) return {lineInter(s1, e1, s2
        , e2)};
  set<P> seg;
    if(onSeg(s1, e1, s2)) seg.insert(s2);
    if(onSeg(s1, e1, e2)) seg.insert(e2);
    if(onSeg(s2, e2, s1)) seg.insert(s1);
    if(onSeg(s2, e2, e1)) seg.insert(e1);
    return {all(seg)};
}
```

lineDistance.h

Description: Gets the distance between a point and a line.

8f3d9d, 3 lines

```
template<class P> ld lineDist(P s, P e, P p) {
    return (e-s).cross(p-s) / (e-s).mag();
}
```

segDistance.h

Description: Gets the distance between a point and a segment.

d02cd5, 5 lines

```
template<class P> ld segDist(P s, P e, P p) {
    if(s == e) return (p - s).mag();
    auto d = (e-s).mag2(), t = min(d, max(0.0, (p-s).dot(e-s)))
        ;
    return ((p - s)*d - (e - s)*t).mag() / d;
}
```

3.2 Polygons

polygonArea.h

Description: Uses shoelace theorem to find the area of a polygon.

Usage: If area is negative, points are given cw, otherwise points are given ccw.

Memory: $\mathcal{O}(n)$

Time: $\mathcal{O}(n)$

8b5fb1, 6 lines

```
template<class T> ld polyArea(vector<pnt<T>> poly) {
    int n = sz(poly); T area = 0;
    for(int i = 0; i < n; i++)
        area += poly[i].cross(poly[(i+1)%n]);
    return area / 2.0L;
}
```

inPolygon.h

Description: Uses the cutting-ray test to see if a point is inside a polygon.

Usage: Returns 0 if outside, 1 if strictly inside, and 2 if on.

Memory: $\mathcal{O}(n)$

Time: $\mathcal{O}(n)$

1ff9f1, 11 lines

```
template<class P> int inPoly(vector<P> poly, P p) {
    bool good = false; int n = sz(poly);
    auto crosses = [](P s, P e, P p) {
        return ((e.y >= p.y) - (s.y >= p.y)) * p.cross(s, e) >
            0;
    };
    for(int i = 0; i < n; i++){
        if(onSeg(poly[i], poly[(i+1)%n], p)) return 2;
        good ^= crosses(poly[i], poly[(i+1)%n], p);
    }
    return good;
}
```

convexHull.h

Description: gets the smallest convex polygon containing all points using monotone chaining.

Memory: $\mathcal{O}(n)$

Time: $\mathcal{O}(n \log n)$

02776c, 16 lines

```
template<class P> vector<P> convexHull(vector<P> poly){
    int n = sz(poly);
    vector<P> hull(n+1);
    sort(all(poly));
    int k = 0;
    for(int i = 0; i < n; i++){
        while(k >= 2 && hull[k-2].cross(hull[k-1], poly[i]) <=
            0) k--;
        hull[k++] = poly[i];
    }
    for(int i = n-1, t = k+1; i > 0; i--){
        while(k >= t && hull[k-2].cross(hull[k-1], poly[i-1])
            <= 0) k--;
        hull[k++] = poly[i-1];
    }
    hull.resize(k-1);
    return hull;
}
```

Graphs (4)

Mathematics (5)

Numerical Methods (6)

Strings (7)