

Вычислительная геометрия

Вычислительная геометрия

Точки


Расстояние между точками

- Даны две точки на плоскости. Найти расстояние между ними:

$$r = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2}$$

A(x_a, y_a)

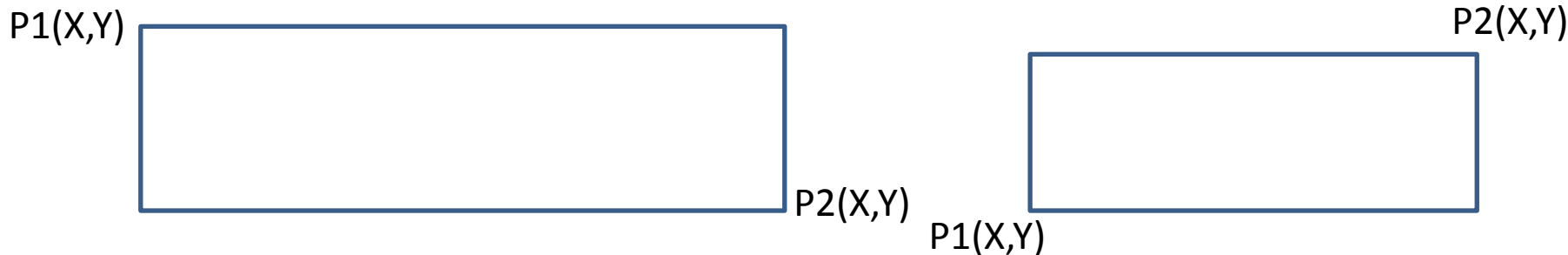
B(x_b, y_b)



Принадлежность точки прямоугольнику

```
boolean point_in_box (Point t, Point p1, Point p2) {  
    return  
        (abs (t.x - min(p1.x, p2.x)) <= eps || min(p1.x, p2.x) <= t.x) &&  
        (abs (max(p1.x, p2.x) - t.x) <= eps || max(p1.x, p2.x) >= t.x) &&  
        (abs (t.y - min(p1.y, p2.y)) <= eps || min(p1.y, p2.y) <= t.y) &&  
        (abs (max(p1.y, p2.y) - t.y) <= eps || max(p1.y, p2.y) >= t.y);  
}
```

Примечание. Нужно учитывать особенности арифметики с плавающей точкой.



Взаимное расположение точек

- Даны две точки.

Определить «самую левую», «самую правую», «самую верхнюю», «самую нижнюю»

```
// наиболее левая из двух точек
Point min_px (Point a, Point b) {
    return a.x < b.x || (abs (a.x - b.x) <= eps && a.y < b.y) ? a : b;
}
// наиболее правая из двух точек
Point max_px (Point a, Point b) {
    return a.x > b.x || (abs (a.x - b.x) <= eps && a.y > b.y) ? a : b;
}
// наиболее низкая из двух точек
Point min_py (Point a, Point b) {
    return a.y < b.y || (abs (a.y - b.y) <= eps && a.x < b.x) ? a : b;
}
// наиболее высокая из двух точек
Point max_py (Point a, Point b) {
    return a.y > b.y || (abs (a.y - b.y) <= eps && a.x > b.x) ? a : b;
}
```

Полярный угол точки

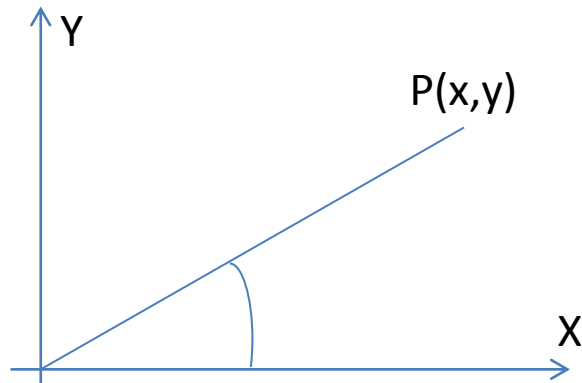
Дана точка

Необходимо вычислить угол между точкой и осью OX

Для вычисления используем формулу:

$$\alpha = \operatorname{arctg} (y / x)$$

В чем может быть проблема?



Полярный угол точки

Необходимо рассматривать отдельно случай когда $x = 0$, чтобы избежать этого, можно воспользоваться функцией **atan2**, которая вычисляет угол в диапазоне $(-\pi; \pi]$.

Наша функция возвращает этот угол в диапазоне $[0; 2\pi)$

```
double polar_angle (Point p) {  
    double alpha = atan2 (p.y, p.x) ;  
    if (alpha < 0) alpha += pi;  
    return alpha;  
}
```

Полярное расстояние

Необходимо найти расстояние между двумя точками в полярной системе координат

Переводим точки из полярной системы координат в декартову и вычисляем расстояние. Используем формулы:

$$x = r * \cos(\alpha)$$

$$y = r * \sin(\alpha)$$

```
double polar_dist (double alpha, double r1, double betta, double r2) {  
    Point p1 = point (r1 * cos (alpha), r1 * sin (alpha));  
    Point p2 = point (r2 * cos (betta), r2 * sin (betta));  
    return dist (p1, p2);  
}
```


Деление отрезка в заданном соотношении

*Даны координаты концов отрезка и два числа m и n .
Необходимо разделить отрезок на две части в
отношении длин $m:n$ и найти точку раздела.*



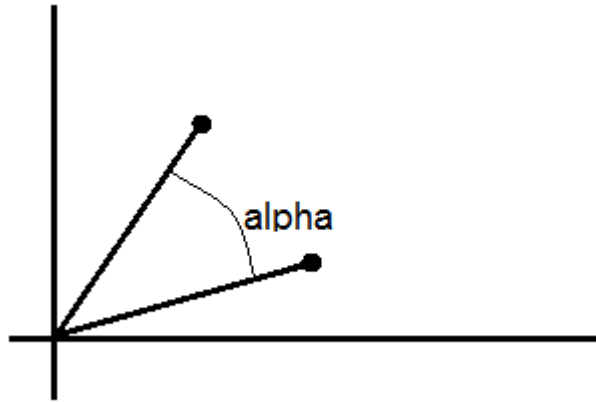
Воспользуемся следующими формулами:

$$x_0 = (x_1 * n + x_2 * m) / (m + n)$$

$$y_0 = (y_1 * n + y_2 * m) / (m + n)$$

Поворот точки против часовой стрелки

*Даны точка и угол поворота против часовой стрелки
Найти точку на которую наложится данная при повороте*



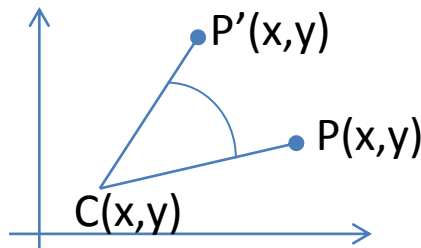
$$\begin{aligned}x &= x_1 * \cos(\alpha) - y_1 * \sin(\alpha) \\ y &= x_1 * \sin(\alpha) + y_1 * \cos(\alpha)\end{aligned}$$

Поворот точки вокруг заданной оси

*Дано точка, угол поворота против часовой стрелки,
и центр поворота*

Найти точку на которую наложится данная при повороте

```
Point turnof(Point p, double alpha, Point c) {  
    Point t = turn(point(p.x - c.x, p.y - c.y), alpha);  
    t.x += c.x;  
    t.y += c.y;  
    return t;  
}
```



Вычислительная геометрия

Прямые

Уравнение прямой

Даны две точки

*Необходимо найти уравнение прямой $a*x+b*y=c$,
проходящей через них*

Используем следующие формулы:

$$a = y_2 - y_1$$

$$b = x_1 - x_2$$

$$c = a * x_1 + b * y_1$$

$$\frac{x - x_1}{x_2 - x_1} = \frac{y - y_1}{y_2 - y_1}$$

Положение точки относительно прямой

- *Дана точка и прямая*

Необходимо определить знак, получаемый при подстановке точки в уравнение прямой

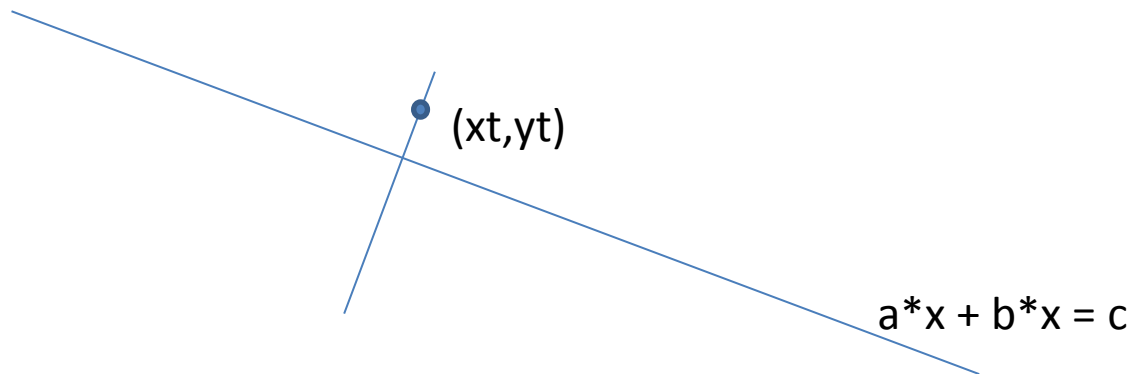
```
int point_in_line (Line l, Point p) {  
    double s = l.a * p.x + l.b * p.y - l.c;  
    return s < - eps ? -1 : s > eps ? 1 : 0;  
}
```

Эту функцию можно использовать при проверке принадлежности точки прямой (если знак равен 0, то прямая проходит через эту точку), также можно использовать при определении, лежат ли две точки с одной стороны от прямой.

Перпендикуляр

Дана прямая и точка

Необходимо найти уравнение прямой, перпендикулярной данной и проходящей через заданную точку



Перпендикуляр

Дана прямая и точка

Необходимо найти уравнение прямой, перпендикулярной данной и проходящей через заданную точку (x_t, y_t)

Для уравнения прямой $a \cdot x + b \cdot y = c$ существует единственный вектор перпендикулярный этой прямой с началом в точке $(0, 0)$. Это вектор нормали.

Второй конец вектора имеет координаты (a, b) .

Следовательно, у перпендикулярного ему вектора будут координаты $(b, -a)$. А значит и коэффициенты уравнения будут такими.

$$b \cdot x - a \cdot y = b \cdot x_t - a \cdot y_t$$

Расстояние от точки до прямой

- Дана точка (x_t, y_t) и прямая $a \cdot x + b \cdot y = c$
Необходимо найти длину перпендикуляра,
опущенного из точки к прямой

Расстояние от точки до прямой

- Дана точка (x_t, y_t) и прямая $a * x + b * y = c$
Необходимо найти длину перпендикуляра,
опущенного из точки к прямой

$$d = |a * x_t + b * y_t - c| / \sqrt{a^2 + b^2}$$

Проекция точки на прямую

*Дана точка (x_t, y_t) и прямая $a*x + b*y = c$*

*Необходимо найти точку на этой прямой,
расстояние до которой от данной минимально*

Проекция точки на прямую

*Дана точка (x_t, y_t) и прямая $a \cdot x + b \cdot y = c$
Необходимо найти точку на этой прямой,
расстояние до которой от данной минимально*

Найдём расстояние от точки до прямой,
разделим его на длину вектора нормали. И
прибавим соответственное количество раз
вектор нормали к точке.

Расстояние между двумя параллельными прямыми

Даны две параллельные прямые

$$a_1 * x + b_1 * y = c_1$$

$$a_2 * x + b_2 * y = c_2$$

Необходимо найти расстояние между ними

Расстояние между двумя параллельными прямыми

Даны две параллельные прямые

$$a_1 * x + b_1 * y = c_1$$

$$a_1 * x + b_1 * y = c_2$$

расстояние между ними

$$d = |c_1 - c_2| / \sqrt{a_1^2 + b_1^2}$$

Прямая, параллельная данной и лежащая на расстоянии d от неё

- *Дана прямая и число*
Необходимо найти прямую, параллельную данной и лежащую на расстоянии равном данному числу

Взаимное расположение прямых

- *Даны две прямые.*

$$a_1 * x + b_1 * y = c_1$$

$$a_2 * x + b_2 * y = c_2$$

Необходимо определить параллельны они или совпадают?

Если прямые параллельны, то

$$a_1 * b_2 = a_2 * b_1$$

Если прямые совпадают, то

$$a_1 * b_2 = a_2 * b_1$$

$$a_1 * c_2 = a_2 * c_1$$

$$b_1 * c_2 = b_2 * c_1$$

Пересечение прямых

- *Даны две прямые*

Необходимо найти точку их пересечения

Решаем систему уравнений:

$$a_1 * x + b_1 * y = c_1$$

$$a_2 * x + b_2 * y = c_2$$

Принадлежность точки отрезку

- *Дана точка и отрезок
Необходимо определить принадлежит ли
точка отрезку*

Принадлежность точки отрезку

- *Дана точка и отрезок
Необходимо определить принадлежит ли точка отрезку*

Точка принадлежит отрезку только если выполняются два условия:

- 1) Точка принадлежит прямой, проходящей через отрезок.
- 2) ?

Принадлежность точки отрезку

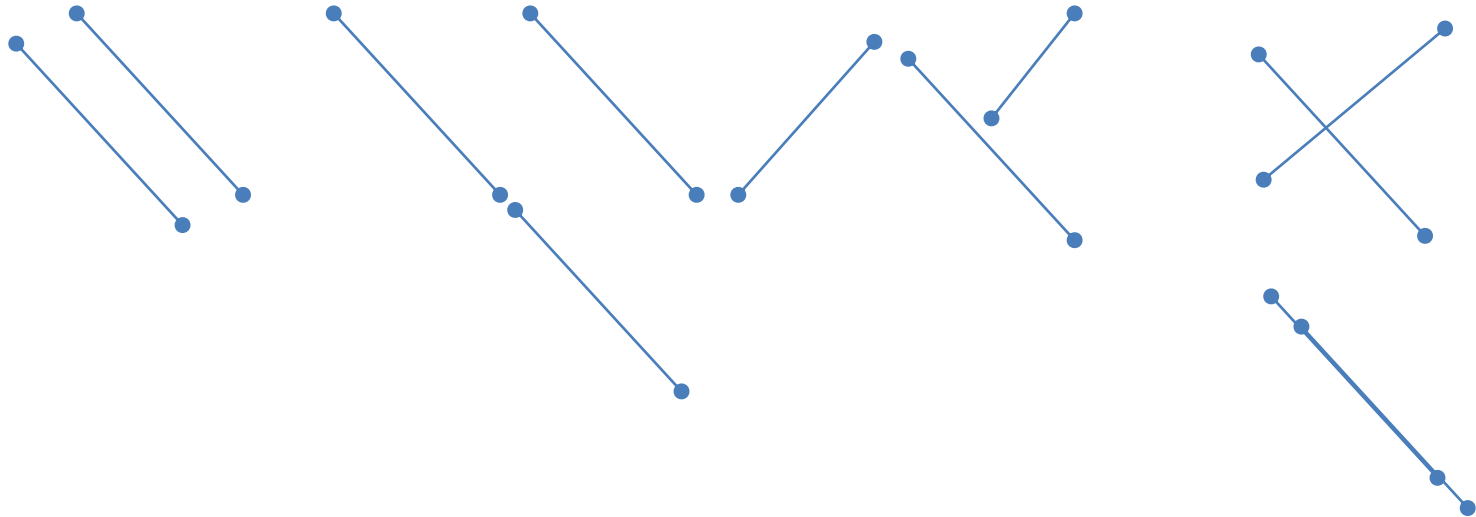
- *Дана точка и отрезок
Необходимо определить принадлежит ли точка отрезку*

Точка принадлежит отрезку только если выполняются два условия:

- 1) Точка принадлежит прямой, проходящей через отрезок.
- 2) Точка принадлежит прямоугольной области, образуемой отрезком.

Пересекаются ли отрезки?

- Даны два отрезка
Необходимо проверить, пересекаются ли они



Пересекаются ли отрезки?

- *Даны два отрезка*

Необходимо проверить, пересекаются ли они

- Если два конца первого отрезка имеют разные знаки положения относительно прямой, проходящей через второй отрезок, и аналогичное условие выполняется для второго отрезка, то отрезки пересекаются.

Пересекаются ли отрезки?

- *Даны два отрезка*
Необходимо проверить, пересекаются ли они
 - Если два конца первого отрезка имеют разные знаки положения относительно прямой, проходящей через второй отрезок, и аналогичное условие выполняется для второго отрезка, то отрезки пересекаются.
 - Если оба отрезка лежат на одной прямой, то отрезки будут пересекаться, если граница одного отрезка принадлежит другому.

Расстояние от точки до отрезка

- *Дан отрезок и точка*
Необходимо найти расстояние от точки до отрезка

Какие будут предложения?

Расстояние от точки до отрезка

- *Дан отрезок и точка*
Необходимо найти расстояние от точки до отрезка

Найдём прямую, проходящую через отрезок.

- Если проекция точки на эту прямую принадлежит нашему отрезку, то расстояние будет равно расстоянию до проекции.
- Иначе расстояние равно минимальному из расстояний между точкой и границами отрезка.

Пересечение отрезка с прямой

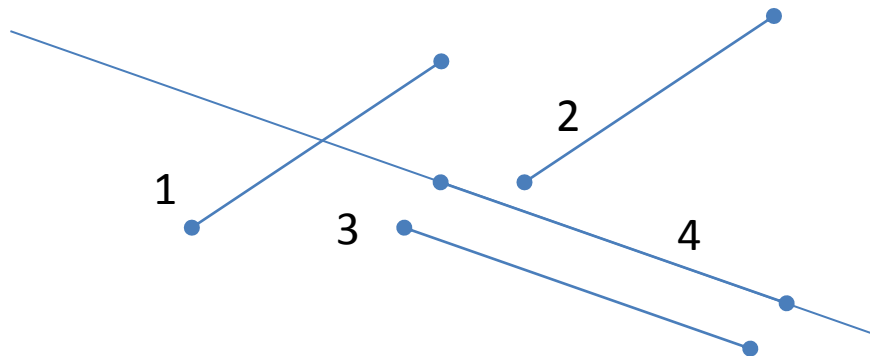
Дан отрезок и прямая

Необходимо определить пересекаются ли они

Пересечение отрезка с прямой

Дан отрезок и прямая

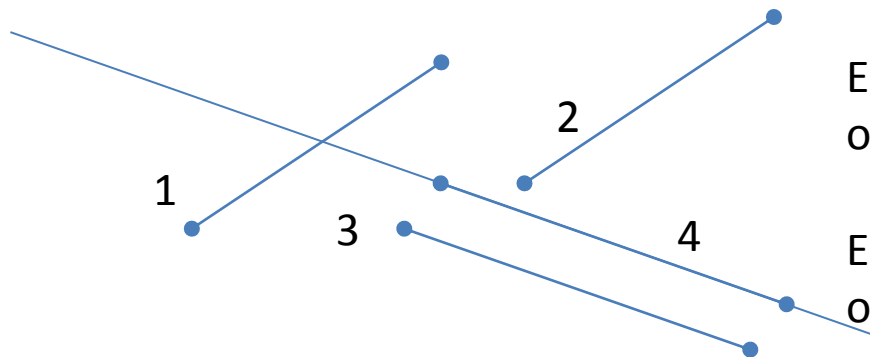
Необходимо определить пересекаются ли они



Пересечение отрезка с прямой

Дан отрезок и прямая

Необходимо определить пересекаются ли они



Если концы отрезка имеют разные знаки относительно прямой, то они пересекаются.

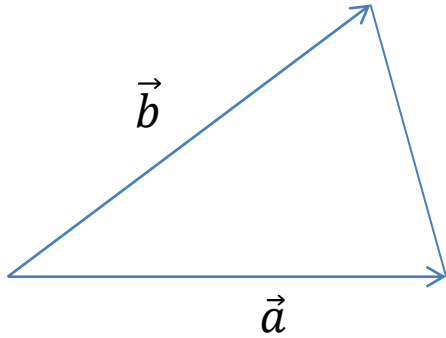
Если концы отрезка принадлежат прямой, то отрезок лежит на прямой.

Векторное произведение векторов

$$\vec{a}(x_a, y_a, z_a) \times \vec{b}(x_b, y_b, z_b) = \begin{vmatrix} \vec{i} & \vec{j} & \vec{k} \\ x_a & y_a & z_a \\ x_b & y_b & z_b \end{vmatrix}$$

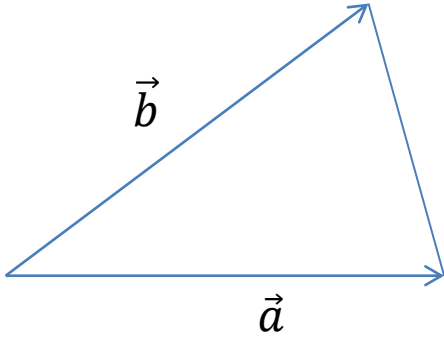
Если $z_a = z_b = 0$?

Как это поможет при вычислении площадей?



$$S = \frac{|\vec{a} \times \vec{b}|}{2}$$

Ориентированная площадь треугольника



А что если убрать знак модуля?

$$S = \frac{\vec{a} \times \vec{b}}{2}$$

Угол между тремя точками - через произведение векторов

Даны три точки.

Необходимо найти угол между ними.

Используем скалярное произведение
векторов:

$$\vec{a} * \vec{b} = |\vec{a}| * |\vec{b}| * \cos(\alpha) = x_a * x_b + y_a * y_b$$

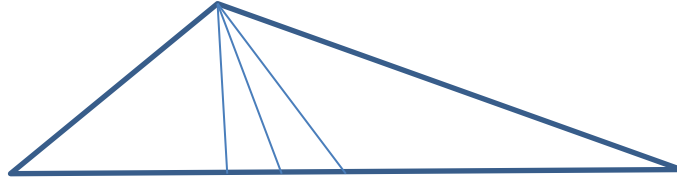
Положение точки относительно вектора в обходе против часовой стрелки

Даны координаты начала и конца вектора и координаты точки.

Необходимо определить лежит ли эта точка справа от этого вектора.

Высота, медиана, биссектриса

*Дан треугольник
Необходимо*



- *Найти уравнения прямой проходящей через высоту треугольника*
- *Найти уравнения прямой проходящей через медиану треугольника*
- *Найти уравнения прямой проходящей через биссектрису треугольника*

Вписанная в треугольник окружность

Дан треугольник

*Необходимо найти окружность, вписанную
в него*

- Центр окружности - это точка пересечения биссектрис углов треугольника.
- Радиус окружности равен расстоянию от центра до любой стороны треугольника.

Описанная окружность

Дан треугольник

Необходимо найти описанную вокруг него окружность

- Центр окружности - точка пересечения серединных перпендикуляров.
- Радиус окружности - расстояние от центра до вершины треугольника.

Положение точки относительно окружности

Дана точка и окружность

Необходимо выяснить, лежит ли точка внутри окружности, снаружи или на самой окружности?

- Если расстояние от точки до центра окружности больше радиуса, то точка лежит вне нашего круга.
- Если расстояние до центра меньше радиуса, то точка лежит внутри нашего круга.
- Если расстояние до центра равно радиусу, то точка лежит на окружности.

Минимальная окружность, покрывающая множество точек

Дано n точек

*Необходимо найти окружность
минимального радиуса такую, чтобы все
точки лежали либо внутри, либо на границе
этой окружности*

Минимальная окружность, покрывающая множество точек

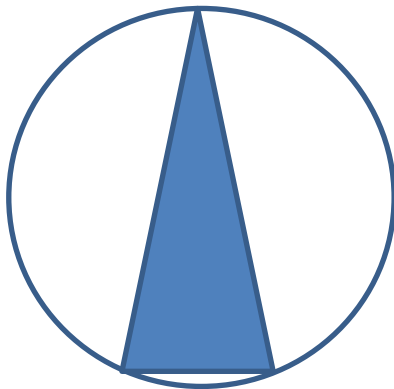
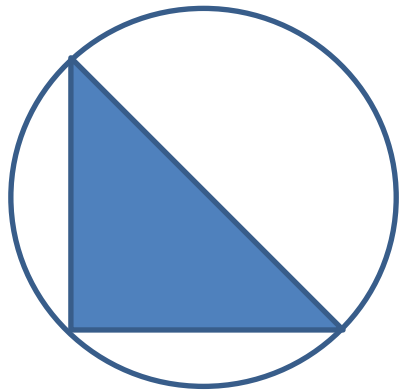
Очевидно, что на искомой окружности должны лежать точки, так как если они на ней не лежат, то радиус окружности можно уменьшить и, значит, эта окружность не была минимальной.

Минимальная окружность, покрывающая множество точек

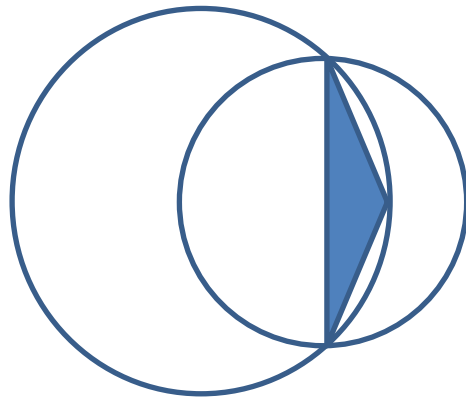
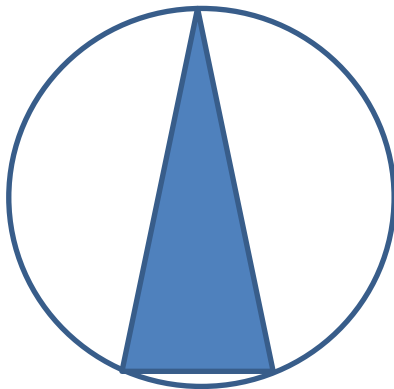
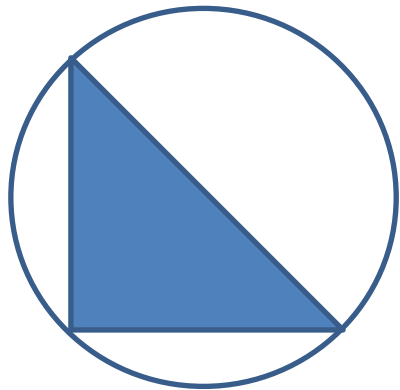
Очевидно, что на искомой окружности должны лежать точки, так как если они на ней не лежат, то радиус окружности можно уменьшить и, значит, эта окружность не была минимальной.

Будем перебирать все *тройки точек* и строить минимальную окружность, покрывающую их. После этого будем проверять, принадлежать ли все точки найденной окружности. Из всех подходящих окружностей выберем ту, у которой минимальный радиус.

Минимальная окружность, покрывающая множество точек



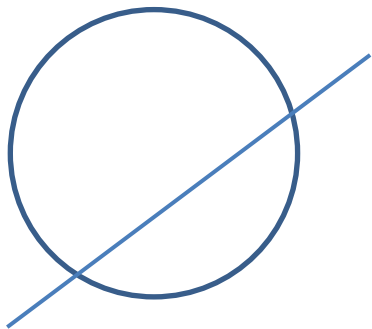
Минимальная окружность, покрывающая множество точек



Пересечение прямой с окружностью

Дано прямая и окружность

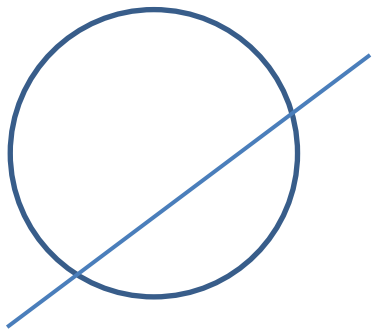
*Найти точки пересечения окружности и
прямой*



Пересечение прямой с окружностью

Дано прямая и окружность

Найти точки пересечения окружности и прямой



Идея заключается в повороте окружности и прямой, чтоб прямая совпала с осью абсцисс. Находим точки пересечения окружности и оси абсцисс, а потом поворачиваем их обратно.

```

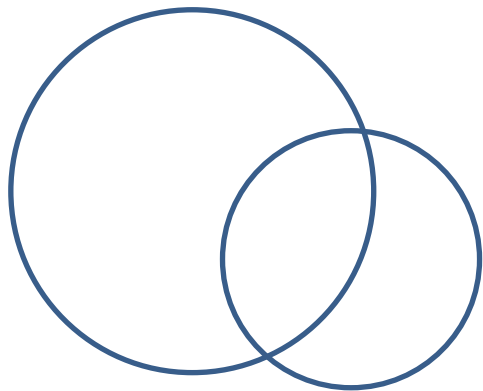
int cross_line_circle (Line l, Circle c, Point p1, Point p2) {
    // проекция центра окружности на прямую
    Point p = closest_point (l, c.c);
    // сколько всего решений?
    int flag = 0;
    double d = dist (c.c, p);
    if (abs (d - c.r) <= eps) flag = 1;
    else
        if (c.r > d) flag = 2;
        else return 0;
    // находим расстояние от проекции до точек пересечения
    double k = sqrt (c.r * c.r - d * d);
    double t = dist (Point (0, 0), Point (l.b, - l.a));
    // добавляем к проекции векторы направленные к точкам пересечения
    Point t1 = add_vector (p, point (0, 0), point (- l.b, l.a), k / t);
    Point t2 = add_vector (p, point (0, 0), point (l.b, - l.a), k / t);
    p1.x = t1.x; p1.y = t1.y; p2.x = t2.x; p2.y = t2.y;
    return flag;
}

```

Точки пересечения двух окружностей

Даны две окружности

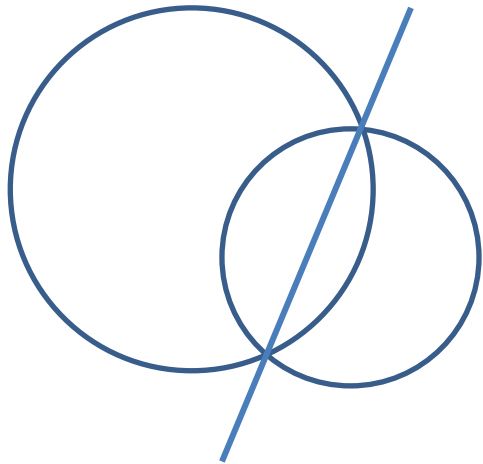
Найти точки их пересечения



Точки пересечения двух окружностей

Даны две окружности

Найти точки их пересечения



Если попробовать решить аналитически эту задачу, то после сокращений можно получить уравнение прямой, проходящей через две точки пересечения окружностей.

Далее находим точки пересечения этой прямой и любой окружности.

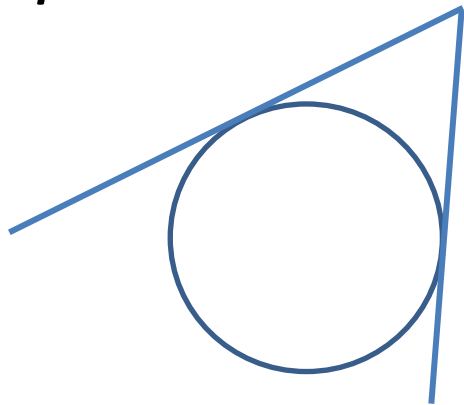
Точки пересечения двух окружностей

```
int cross_circle (double x1, double y1, double r1, double x2, double y2,  
                  double r2, Point p1, Point p2) {  
    if (abs (x1 - x2) <= eps && abs (y1 - y2) <= eps && abs (r1 - r2) <= eps)  
        return 3;  
    double a = 2.0 * (x2 - x1);  
    double b = 2.0 * (y2 - y1);  
    double c = x1 * x1 + y1 * y1 - r1 * r1 - (x2 * x2 + y2 * y2 - r2 * r2);  
    return cross_line_circle (line (a, b, c), circle (x1, y1, r1), p1, p2);  
}
```


Касательные к окружности

Дана окружность и точка.

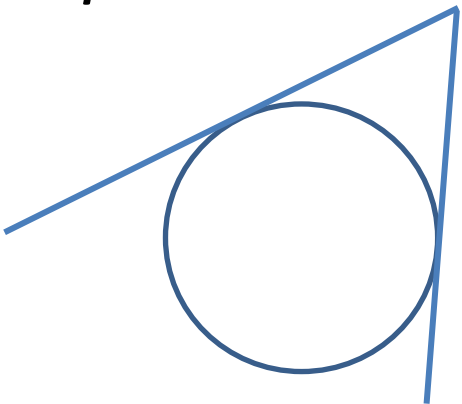
Необходимо найти точки касания касательных, проведённых к окружности из заданной точки.



Касательные к окружности

Дана окружность и точка.

Необходимо найти точки касания касательных, проведённых к окружности из заданной точки.

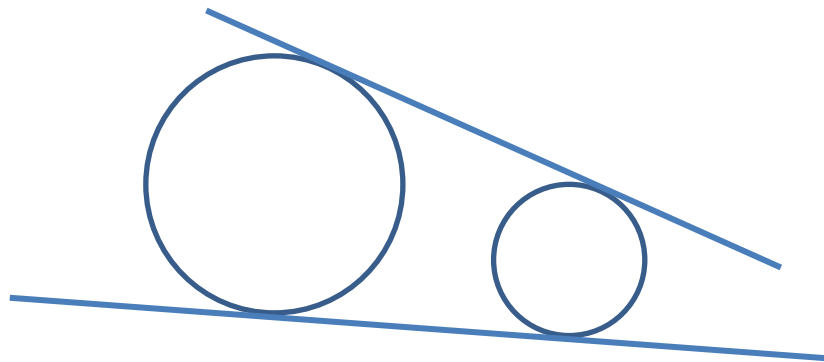


Найдём расстояния от заданной точки до точек касания. Построим окружность вокруг заданной точки с радиусом, равным расстоянию до точек касания. Точки пересечения полученной окружности с исходной окружностью являются искомыми точками касания.

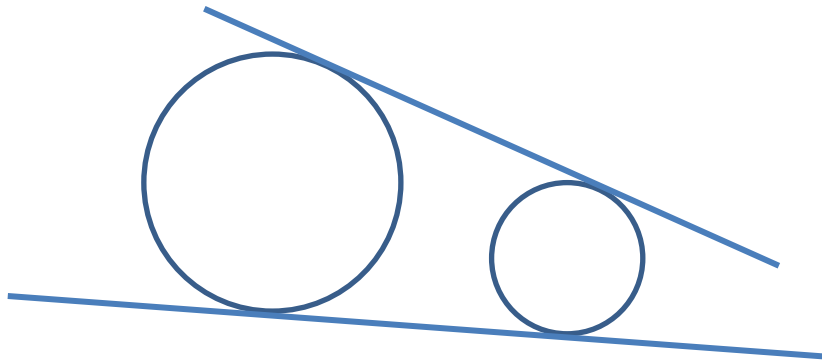
Касательные к двум окружностям

Даны две окружности

Необходимо найти две прямые, которые касаются обеих окружностей



Касательные к двум окружностям



Очевидно, что такие прямые существуют, и их всегда две. Если окружности имеют одинаковый радиус, то нам надо просто отложить две параллельные прямые на расстоянии равном радиусу от прямой, проходящей через центры окружностей.

Если окружности имеют разный радиус, то существует такая точка, в которой касательные к обеим окружностям будут пересекаться.

С помощью подобия треугольников найдём расстояние до этой точки, а зная векторы, направленные на неё, найдём саму точку.

После этого точки касания можно найти с помощью функций поворота точки вокруг найденной нами точки.

Луч, не пересекающий ни одну из заданных окружностей

Дана точка и множество окружностей.

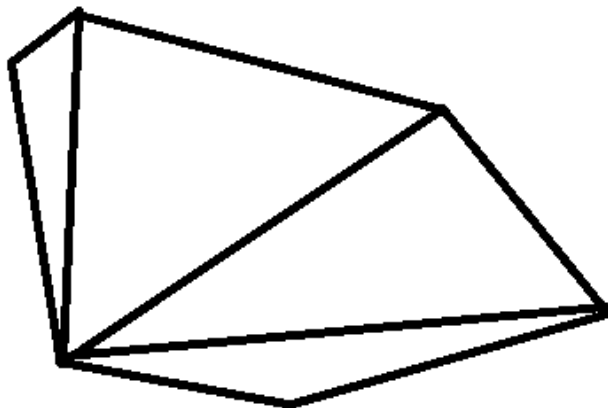
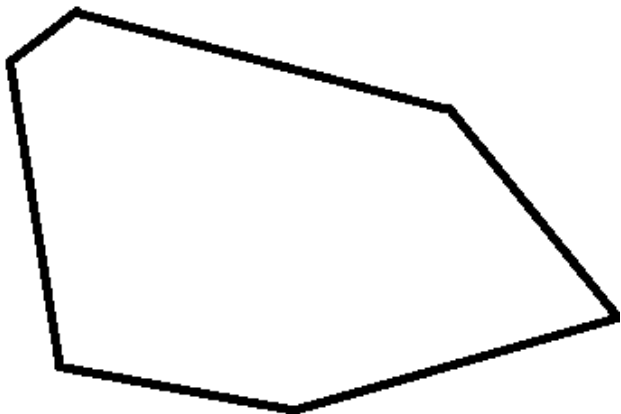
Необходимо найти точку, лежащую на луче (не равную исходной точке), такую, чтобы луч, выходящий из исходной точки и проходящий через заданную точку, не пересекал ни одну из заданных окружностей.

Луч, не пересекающий ни одну из заданных окружностей

- Такого луча не может быть в том и только в том случае, когда окружности образуют "забор для света" вокруг данной точки.
- Чтобы проверить так это или нет, упорядочим окружности вокруг исходной точки по полярному углу.
- После этого будем строить касательные от исходной точки к окружностям, если сдвинуть касательные на очень маленький угол вбок от окружности, то получатся два луча, не пересекающие эту окружность.
- Если такой луч не пересекает больше никакие окружности, то этот луч является искомым в нашей задаче.

Площадь многоугольника

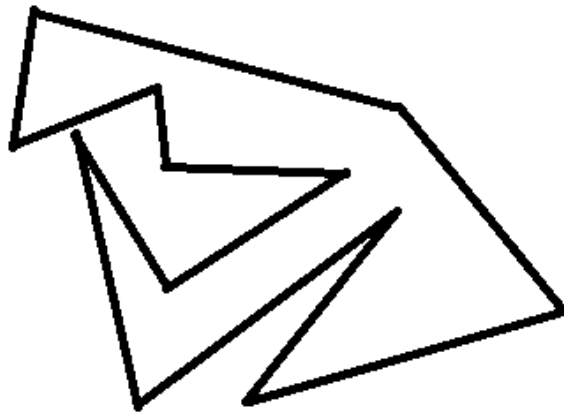
- Каким-либо образом разбить на непересекающиеся треугольники
- Вычислить сумму площадей треугольников



Площадь многоугольника

- Каким-либо образом разбить на непересекающиеся треугольники
- Вычислить сумму площадей треугольников

А если это сложно?



Периметр многоугольника

Дан многоугольник

Необходимо вычислить его периметр

Периметр многоугольника

Дан многоугольник

Необходимо вычислить его периметр

Складываем длины всех сторон 😊

Алгоритм «Ван Гога»

Основная идея его состоит в последовательном отрезании треугольников.

Функция имеет вид $VanGog(p, n, T)$.

T - массив результат - размер матрицы *count* $\times 3$, содержит индексы вершин триангуляции.

Функция возвращает количество треугольников.

```

boolean isTriangCut(Point[] p, int i, int j, int k) {
    if(cw(p[i],p[j],p[k])) return false;
    for(int m = 0; m < p.length; ++m)
        if(m != i && m != j && m != k && inTriangPoint(p[i],p[j],p[k],p[m]))
            return false;
    return true;
}

int vanGog(Point[] p, int T[][]) {
    int[] l=new int[p.length];
    int[] r=new int[p.length];
    int count=0;
    for(int i=0; i<p.length ; ++i) {
        l[i]=(i-1+n)%n;
        r[i]=(i+1+n)%n;
    }
    for(int i=r[n-1]; count < p.length-2; i=r[i])
        if(isTriangCut(p, l[i], i, r[i])) {
            T[count][0]=l[i]; T[count][1]=i; T[count][2]=r[i];
            l[r[i]]=l[i];      r[l[i]]=r[i];  ++count;
        }
    return count;
}

```

Триангуляция с количеством треугольников - $(n-2)$

Дано точки - вершины многоугольника в порядке обхода против часовой стрелки.

Разбить данный многоугольник на треугольники

Триангуляция с количеством треугольников - $(n-2)$

Используем стандартный рекурсивный алгоритм. Основная идея его состоит в отрезании многоугольника на две части по хорде если это не треугольник.

Функция имеет вид `triangulate(p,T)`.

T - массив результат - размер матрицы `count x 3`, содержит вершины триангуляции.

Функция разбивает многоугольник на $n-2$ треугольника.

Другие виды триангуляции

- **Триангуляция с минимальной суммой длин проведённых хорд**
- **Триангуляция многоугольника минимальной мощности**

Необходимо провести в нём некоторое количество отрезков, чтобы многоугольник распался на минимальное количество треугольников

Находится ли точка в многоугольнике?

Дан многоугольник и точка.

*Необходимо проверить, располагается ли
наша точка внутри многоугольника.*

Находится ли точка в многоугольнике?

Дан многоугольник и точка.

*Необходимо проверить, располагается ли
наша точка внутри многоугольника.*

- Если провести горизонтальную прямую проходящую через заданную точку, то могут возникнуть пересечения прямой с многоугольником.
- Рассмотрим все точки пересечения, у которых абсцисса меньше, чем у заданной точки.
- Если этих точек нечётное число, то исходная точка лежит внутри многоугольника, иначе нет.

Асимптотика $O(n)$.

Площадь многоугольника

Дан многоугольник.

Необходимо найти его площадь.

Площадь многоугольника

*Дан многоугольник.
Необходимо найти его площадь.*

Вычисляем площадь по следующей формуле:

$$S = 0.5 * \sum (x_i * y_{(i+1) \bmod n} - y_i * x_{(i+1) \bmod n})$$

(точки нумеруются от 0 до $n - 1$).

Так же стоит заметить, что значение площади получается со знаком +/-.
Этот знак можно использовать, чтобы узнать, по часовой стрелке заданы точки или нет.

Количество точек на границе многоугольника

Дан многоугольник с целочисленными координатами вершин.

Необходимо найти количество точек с целочисленными координатами на его границе.

Количество точек на границе многоугольника

Количество точек на отрезке с целочисленными координатами равно $\text{НОД}(dx, dy) + 1$.

Значит, количество точек на границе многоугольника равно

$$\text{SUM} (\text{НОД}(dx_i, dy_i) + 1) - n = \text{SUM} \text{НОД}(dx_i, dy_i).$$

НОД - Наименьший Общий Делитель.

Асимптотика $O(n)$.

Количество точек внутри многоугольника

Дан многоугольник с целочисленными координатами вершин.

Необходимо найти количество точек с целочисленными координатами, лежащих строго внутри многоугольника (не на границе).

Количество точек внутри многоугольника

Дан многоугольник с целочисленными координатами вершин.

Необходимо найти количество точек с целочисленными координатами, лежащих строго внутри многоугольника (не на границе).

Воспользуемся формулой Пика:

$$A = I + B/2 - 1$$

A - площадь многоугольника

I - количество точек внутри многоугольника

B - количество точек на границе многоугольника

Является ли многоугольник выпуклым

Дан многоугольник.

*Необходимо проверить, является ли он
выпуклым многоугольником.*

Является ли многоугольник выпуклым

- Если пройти по всем вершинам выпуклого многоугольника против часовой стрелки, то можно заметить, что каждая вершина лежит слева от предыдущей стороны.
- Значит, достаточно пройти по всем трём смежным вершинам многоугольника и установить, что знак ориентированной площади треугольника этих трёх вершин всегда одинаков.
- Если это так, то перед нами выпуклый многоугольник, иначе нет.

Является ли отрезок внутренней диагональю многоугольника

Дан многоугольник и индексы двух его вершин

*Необходимо узнать, является ли отрезок,
проведённый между этими вершинами,
внутренней диагональю данного
многоугольника*

Является ли отрезок внутренней диагональю многоугольника

Дан многоугольник и индексы двух его вершин

Необходимо узнать, является ли отрезок, проведённый между этими вершинами, внутренней диагональю данного многоугольника

Отрезок, соединяющий вершины многоугольника, является внутренней диагональю тогда и только тогда, когда он не пересекается ни с одной стороной многоугольника и все точки этого отрезка лежат внутри многоугольника.

Разрез многоугольника диагональю

*Дан многоугольник и индексы вершин,
образующих диагональ*

*Необходимо разделить многоугольник на
две части, лежащие с разных сторон
относительно диагонали*

Разрез многоугольника диагональю

*Дан многоугольник и индексы вершин, образующих диагональ
Необходимо разделить многоугольник на две части,
лежащие с разных сторон относительно диагонали*

Создадим два списка.

- Пройдём по вершинам от первой исходной ко второй и добавим все рассматриваемые вершины в первый список.
- Пройдём от второй к первой и добавим все рассматриваемые вершины во второй список.

Асимптотика $O(n)$.

Расположение многоугольника относительно прямой

- *Дан многоугольник и прямая.
Необходимо определить с какой стороны
расположен многоугольник от прямой.*

Расположение многоугольника относительно прямой

- *Дан многоугольник и прямая.
Необходимо определить с какой стороны
расположен многоугольник от прямой.*

Подставляем вершины многоугольника в уравнение прямой и в зависимости от полученных знаков выдаём ответ.

Разрезание выпуклого многоугольника прямой

- *Дан выпуклый многоугольник и прямая. Необходимо найти два многоугольника, которые получатся при пересечении прямой исходного многоугольника.*

Разрезание выпуклого многоугольника прямой

- *Дан выпуклый многоугольник и прямая
Необходимо найти два многоугольника, которые
получатся при пересечении прямой исходного
многоугольника.*

Найдём точки пересечения прямой с многоугольником.
Добавим их в список вершин.

Теперь нужно разделить многоугольник на две части,
лежащие с разных сторон от «диагонали».

Асимптотика $O(n)$.

Разрезание выпуклого многоугольника в отношении площадей частей $m:n$

*Дан выпуклый многоугольник и два
натуральных числа m и n*

*Необходимо разделить заданный
многоугольник на два многоугольника, у
которых площади относятся как m к n*

Разрезание выпуклого многоугольника

в отношении площадей частей $m:n$

Рассмотрим решение, основанное на такой теореме:

Проведём в треугольнике отрезок соединяющий одну из вершин с противоположной стороной.

Пусть этот отрезок разобьёт треугольник на две части с отношением площадей $m:n$, тогда отрезки, являющиеся основаниями полученных частей и образующие основание всего треугольника, так же относятся как m к n .

Как мы можем этим воспользоваться?

Разделение выпуклого многоугольника на равные части

Дан выпуклый многоугольник и число k .

*Необходимо разделить наш многоугольник
на k равных частей.*

Разделение выпуклого

многоугольника на равные части

Дан выпуклый многоугольник и число k .

Необходимо разделить наш многоугольник на k равных частей.

Будем "отламывать" искомые многоугольники от исходного по одиночки. Сначала, нам надо разделить многоугольник в отношении $1:(k-1)$. Потом $2:(k-2)$ и так далее. У всех найденных многоугольников будет общая вершина - первая вершина в исходном многоугольнике.

Асимптотика $O(n)$.

Выпуклая оболочка

Дано множество точек

Найти минимальный многоугольник (по периметру), содержащий все эти точки

- **алгоритм Джарвиса**
- **алгоритм Грэхема**

Выпуклая оболочка – алг. Джарвиса

Очевидно, что искомый многоугольник будет являться выпуклым, и будет строиться из исходных точек.

Идея алгоритма Джарвиса заключается в последовательном построении выпуклой оболочки.

- Сначала нам необходимо найти точку, которая должна быть в оболочке. Для этого можно взять самую нижнюю из самых левых точек.
- Далее будем каждый раз добавлять такую точку, чтобы угол образуемый следующей, текущей и предыдущей точками был максимален. Для этого совсем не обязательно находить сам угол.
- Мы можем использовать ориентированную площадь треугольника: пусть p_0 - текущая точка, p_n - уже выбранная следующая точка и p_i - текущая точка в переборе точек, тогда если $\text{SignAreaTriangle}(p_0, p_i, p_n) > 0$, то точка p_i находится справа от вектора $p_0 \rightarrow p_n$ и, значит, она составляет больший угол в выпуклой оболочке, поэтому мы ей дадим большее предпочтение, чем точке p_n .

Выпуклая оболочка – алг. Джарвиса

-
- Мы можем использовать ориентированную площадь треугольника: пусть p_0 - текущая точка, p_n - уже выбранная следующая точка и p_i - текущая точка в переборе точек, тогда если $\text{SignAreaTriangle}(p_0, p_i, p_n) > 0$, то точка p_i находится справа от вектора $p_0 \rightarrow p_n$ и, значит, она составляет больший угол в выпуклой оболочке, поэтому мы ей дадим большее предпочтение, чем точке p_n .
- Также нам необходимо рассмотреть вырожденный случай - все три точки (p_0, p_i, p_n) лежат на одной прямой (ориентированная площадь треугольника равна 0).
- В таком случае нам нужно выбрать ту точку, которая расположена дальше от p_0 . Для этого тоже можно не использовать сложных вычислений, достаточно узнать, лежит ли p_n в прямоугольнике с диагональю p_0, p_i (это делается простым сравнением координат точек). Если p_n лежит в этом прямоугольнике, то p_i лежит дальше от p_0 и мы точке p_i вследствие этого отдадим большее предпочтение, чем p_n .

Выпуклая оболочка – алг. Джарвиса

- Сложность $O(nk)$.
k - количество точек в оболочке
n - количество заданных точек

Выпуклая оболочка – алг. Грэхема

- Идея алгоритма Грэхема заключается в последовательном отборе точек, упорядоченных ранее по полярному углу относительно какой-либо точки, которая обязательно будет входить в оболочку.
- Для этого можем использовать самую нижнюю из самых левых точек.

Выпуклая оболочка – алг. Грэхема

- Если посмотреть на полученный упорядоченный набор точек, то становится очевидно, что все точки, входящие в оболочку, будут располагаться в этом наборе в той же последовательности, в какой они образуют оболочку.
- Следовательно, мы можем последовательно их добавлять по одной из набора, но если получится, что в текущей оболочке три последовательные точки образуют угол больше либо равный 180° (ориентированная площадь треугольника меньше либо равна нулю), то необходимо удалить предыдущую точку.

Выпуклая оболочка – алг. Грэхема

- Если посмотреть на полученный упорядоченный набор точек, то становится очевидно, что все точки, входящие в оболочку, будут располагаться в этом наборе в той же последовательности, в какой они образуют оболочку.
- Следовательно, мы можем последовательно их добавлять по одной из набора, но если получится, что в текущей оболочке три последовательные точки образуют угол больше либо равный 180° (ориентированная площадь треугольника меньше либо равна нулю), то необходимо удалить предыдущую точку.
- Случай, когда две последовательные точки в оболочке лежат на одной прямой, можно не рассматривать отдельно, если в сортировке учесть, что чем дальше точка, тем она "приоритетнее".

Выпуклая оболочка – алг. Грэхема

- Сложность $O(n \log n)$

Спасибо!