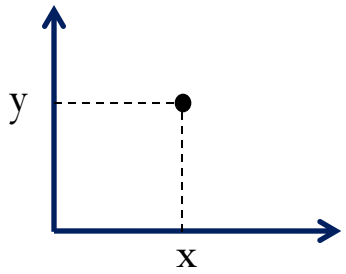
Geometría Computacional

Conceptos y algoritmos básicos para el ACM-ICPC

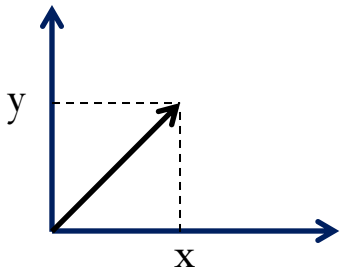
Punto y Vector

- Tus mejores amigos:

$\mathrm{P}(x, y)$



$\vec{V}(x, y)$



- ¿Y en 3 dimensiones?

$\mathrm{P}(x, y, z)$ $\vec{V}(x, y, z)$

- ¿Cómo los podemos representar en código?

Punto y Vector

- ¿Cómo los podemos representar en código?

```
#define Vector Point
struct Point
{
    double x, y;
    Point(){}
    Point(double a, double b) { x = a; y = b; }
};
```

- Declaración:

```
Point A(2, 4);

Point B;
B = Point(1, -1);

Vector V(0, 10);
```

Punto y Vector : Operaciones

- Suma de puntos:

$$P(x_1, y_1) + P(x_2, y_2) = P(x_1 + x_2, y_1 + y_2)$$

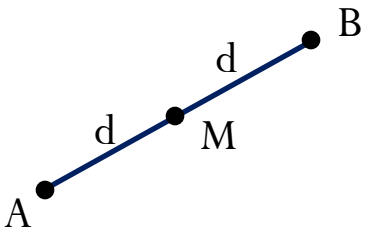
- Multiplicación y división de un punto y un escalar:

$$P(x, y).k = P(x.k, y.k)$$

$$\frac{P(x, y)}{k} = P\left(\frac{x}{k}, \frac{y}{k}\right)$$

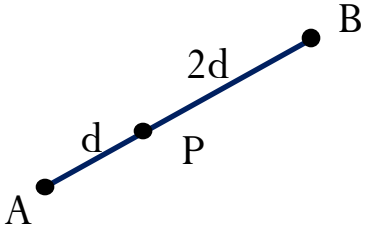
- No tienen interpretación geométrica, pero… :

Punto Medio



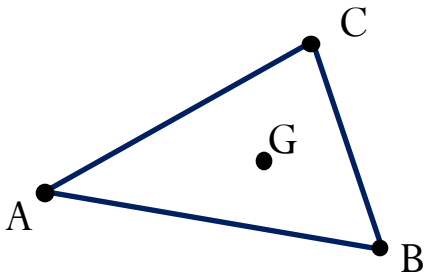
$$\mathrm{M} = \frac{A + B}{2}$$

"Promedio Ponderado"



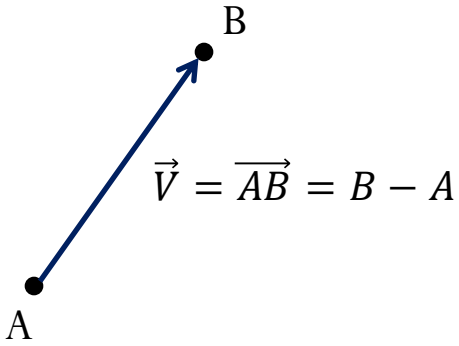
$$\mathrm{P} = \frac{2A + B}{3}$$

Baricentro



$$\mathrm{G} = \frac{A + B + C}{3}$$

Punto y Vector : Operaciones



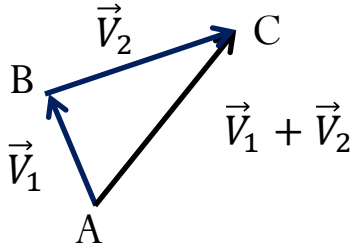
- Suma de punto y vector

$$A(x_1, y_1) + \vec{V}(x_2, y_2) = B(x_1 + x_2, y_1 + y_2)$$
$$A + \vec{V} = B$$

- Resta de puntos

$$B(x_1 + x_2, y_1 + y_2) - A(x_1, y_1) = \vec{V}(x_2, y_2)$$
$$B - A = \vec{V}$$

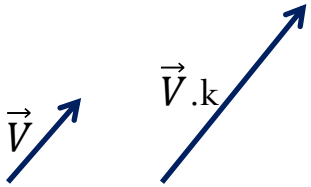
- Suma de vectores



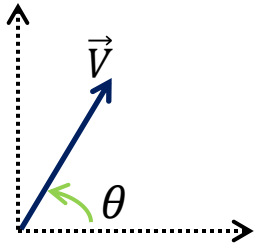
$$\vec{V}_1(x_1, y_1) + \vec{V}_2(x_2, y_2) = \vec{V}(x_1 + x_2, y_1 + y_2)$$
$$\overrightarrow{AB} + \overrightarrow{BC} = (B - A) + (C - B) = (C - A) = \overrightarrow{AC}$$

- Multiplicación y división de un vector y un escalar

$$\vec{V}(x, y).k = \vec{V}(x.k, y.k)$$
$$\frac{\vec{V}(x, y)}{k} = \vec{V}\left(\frac{x}{k}, \frac{y}{k}\right)$$



Punto y Vector : Operaciones



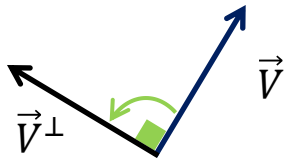
- Módulo

$$\vec{V} = (x, y) \rightarrow |\vec{V}| = \sqrt{x^2 + y^2}$$

- Argumento

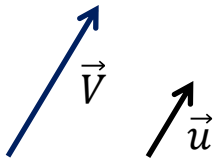
$$\theta = atan2(y, x)$$

- Vector ortogonal



$$\vec{V} = (x, y) \rightarrow \vec{V}^{\perp} = (-y, x)$$
$$(\vec{V}^{\perp})^{\perp} = -\vec{V}$$

- Vector unitario



$$\vec{V} = (x, y) \rightarrow \vec{u} = \frac{\vec{V}}{|\vec{V}|}$$

Punto y Vector : Operaciones

- Resumiendo…

```
#define EPS 1e-8
#define PI acos(-1)
#define Vector Point

struct Point
{
    double x, y;
    Point(){}
    Point(double a, double b) { x = a; y = b; }
    double mod2() { return x*x + y*y; }
    double mod()  { return sqrt(x*x + y*y); }
    double arg()  { return atan2(y, x); }
    Point ort()   { return Point(-y, x); }
    Point unit()  { double k = mod(); return Point(x/k, y/k); }
};
Point operator +(const Point &a, const Point &b) { return Point(a.x + b.x, a.y + b.y); }
Point operator -(const Point &a, const Point &b) { return Point(a.x - b.x, a.y - b.y); }
Point operator /(const Point &a, double k) { return Point(a.x/k, a.y/k); }
Point operator *(const Point &a, double k) { return Point(a.x*k, a.y*k); }

bool operator ==(const Point &a, const Point &b){
    return fabs(a.x - b.x) < EPS && fabs(a.y - b.y) < EPS;
}
bool operator !=(const Point &a, const Point &b){
    return !(a==b);
}
bool operator <(const Point &a, const Point &b){
    if(a.x != b.x) return a.x < b.x;
    return a.y < b.y;
}
```

Producto escalar y vectorial en $\mathbb{R}^2$

Sea $\vec{u} = (u_x, u_y), \vec{v} = (v_x, v_y)$, entonces:

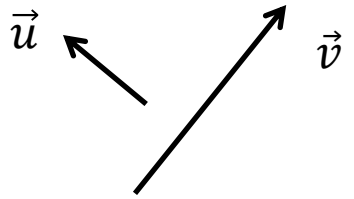
- **Producto Escalar**

$$\vec{u}.\vec{v} = u_x v_x + u_y v_y = |\vec{u}||\vec{v}| \cos\theta$$

$$\vec{u}.\vec{v} = \vec{v}.\vec{u}$$

Si $\vec{u}$ y $\vec{v}$ son perpendiculares:

$$\vec{u}.\vec{v} = 0$$

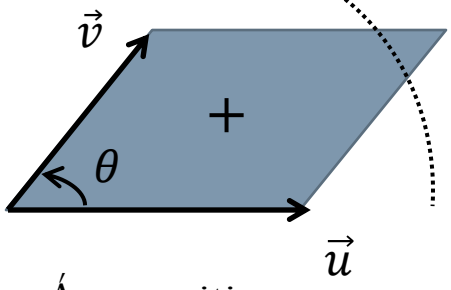


- **Producto Vectorial**

$$\vec{u} \times \vec{v} = u_x v_y - u_y v_x = |\vec{u}||\vec{v}| \, sen\,\theta$$

$$\vec{u} \times \vec{v} = -\vec{v} \times \vec{u}$$

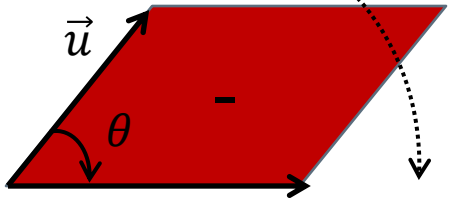
$\vec{u} \times \vec{v} > 0$



Área positiva

$\vec{v}$ a la 'izquierda' de $\vec{u}$

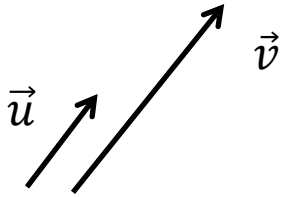
$\vec{u} \times \vec{v} < 0$



Área negativa

$\vec{v}$ a la 'derecha' de $\vec{u}$

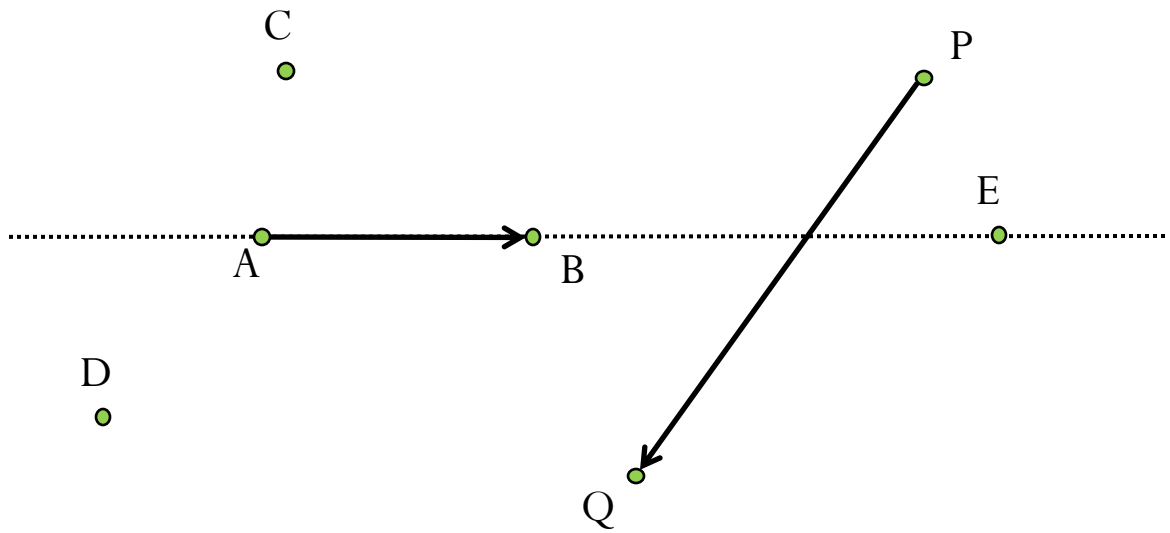
$\vec{u} \times \vec{v} = 0$



Área Nula

$\vec{u}$ y $\vec{v}$ paralelos

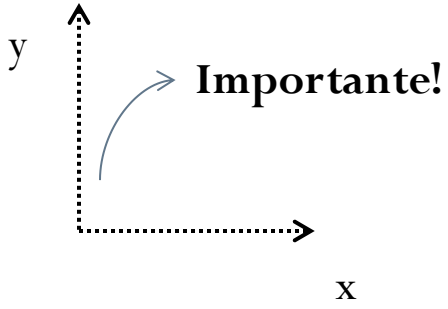
Orientación de Puntos



A la izquierda de $\overrightarrow{AB}$: C y P

A la derecha de $\overrightarrow{AB}$: D y Q

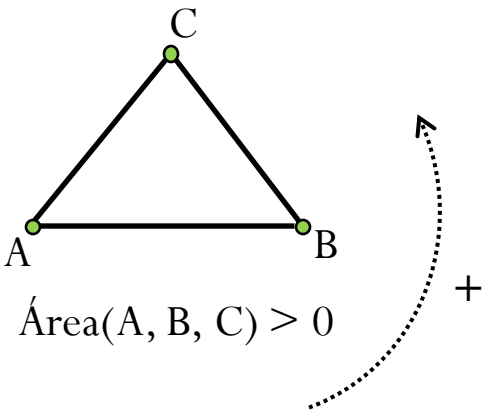
Co-lineal con el segmento $\overline{AB}$: E



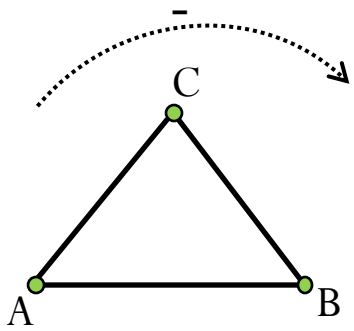
A la izquierda de $\overrightarrow{PQ}$: E

A la derecha de $\overrightarrow{PQ}$: A, B, C y D

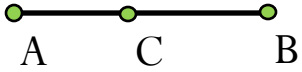
Área de Triángulos



Área(A, B, C) > 0

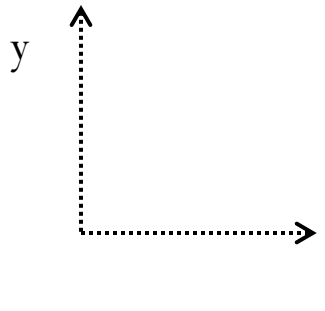


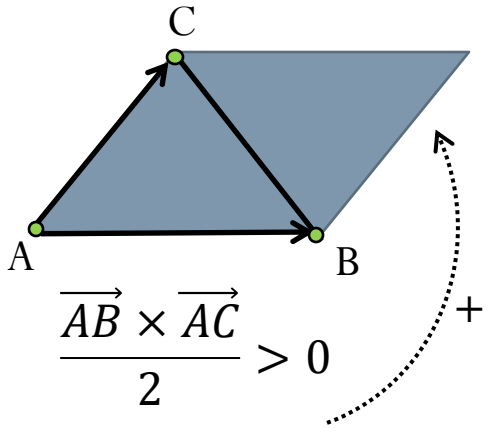
Área(A, C, B) < 0



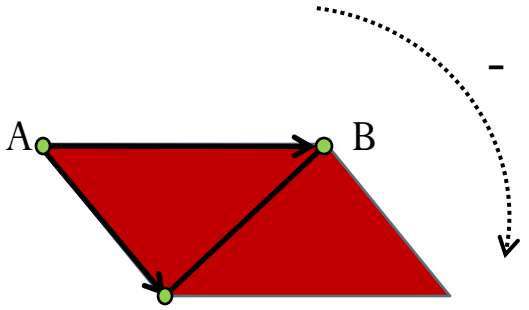
Área(A, B, C) = 0

Podemos definir: Área(A, B, C) = $\overrightarrow{AB} \times \overrightarrow{AC}$ = Doble del área del triangulo ABC, con signo.
= Área con signo del paralelogramo A,B,B+C-A,C





$$\frac{\overrightarrow{AB} \times \overrightarrow{AC}}{2} > 0$$



C

$$\frac{\overrightarrow{AB} \times \overrightarrow{AC}}{2} < 0$$

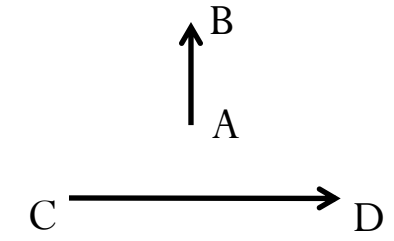
Producto escalar y vectorial, áreas

- Resumiendo…

```
// Distancia entre 2 puntos
double dist(const Point &A, const Point &B)    { return hypot(A.x - B.x, A.y - B.y); }

// Producto escalar
double dot(const Vector &A, const Vector &B)   { return A.x * B.x + A.y * B.y; }

// Producto vectorial
double cross(const Vector &A, const Vector &B) { return A.x * B.y - A.y * B.x; }

// Doble del área del triangulo ABC, con signo
double area(const Point &A, const Point &B, const Point &C) { return cross(B - A, C - A); }
```

Determinar si 2 rectas son perpendiculares o paralelas

Problema 1: Dadas 2 rectas no degeneradas AB y CD, determinar si son paralelas, perpendiculares o si se intersectan en algún punto.



Solución:

Recordamos que:

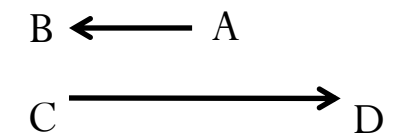
$$\vec{u}.\vec{v} = u_x v_x + u_y v_y = |\vec{u}||\vec{v}|\cos\theta$$

$$\vec{u} \times \vec{v} = u_x v_y - u_y v_x = |\vec{u}||\vec{v}|\, sen\,\theta$$

Por lo tanto, para $\vec{u}, \vec{v}$ no nulos:

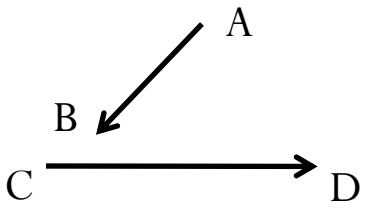
$$\theta = \pm 90° \leftrightarrow \vec{u}.\vec{v} = 0$$

$$\theta = 0° \leftrightarrow \vec{u} \times \vec{v} = 0$$



Entonces:

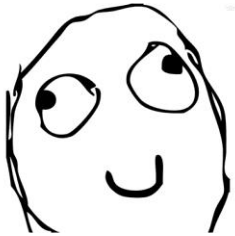
- Si $\overrightarrow{AB}.\overrightarrow{CD} = 0$, las rectas son perpendiculares.
- Si $\overrightarrow{AB} \times \overrightarrow{CD} = 0$, las rectas son paralelas.
- En caso contrario, se intersectan en algún punto.



Intersección de rectas

Problema 2: Dadas 2 rectas no paralelas AB y CD, hallar el punto de intersección.

Posibles soluciones: ¿Cual eligen?



Muy fácil! Hallo la ecuación de cada recta $a_1x + b_1y = c_1$, $a_2x + b_2y = c_2$, luego despejo y obtengo (x, y)

Puedo hacerlo resolviendo un sistema de ecuaciones con vectores:

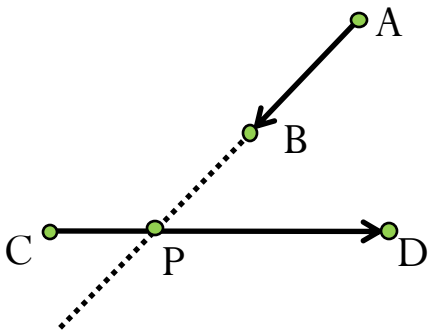
$$P = (x, y) = A + k_1\overrightarrow{AB} = C + k_2\overrightarrow{CD}$$

Puedo hacerlo gráficamente, no necesito resolver ninguna ecuación.

Intersección de rectas

Problema 2: Dadas 2 rectas no paralelas AB y CD, hallar el punto de intersección.

Solución analítica:



$$P = (x, y) = A + k_1\overrightarrow{AB} = C + k_2\overrightarrow{CD}$$

$$k_1\overrightarrow{AB} - k_2\overrightarrow{CD} = C - A$$

$$k_1\overrightarrow{AB} - k_2\overrightarrow{CD} = \overrightarrow{AC}$$

Haciendo producto vectorial con $\overrightarrow{CD}$:

$$(k_1\overrightarrow{AB} - k_2\overrightarrow{CD}) \times \overrightarrow{CD} = \overrightarrow{AC} \times \overrightarrow{CD}$$

$$k_1\overrightarrow{AB} \times \overrightarrow{CD} - k_2\overrightarrow{CD} \times \overrightarrow{CD} = \overrightarrow{AC} \times \overrightarrow{CD}$$

$$k_1\overrightarrow{AB} \times \overrightarrow{CD} = \overrightarrow{AC} \times \overrightarrow{CD}$$

$$k_1 = \frac{\overrightarrow{AC} \times \overrightarrow{CD}}{\overrightarrow{AB} \times \overrightarrow{CD}}$$

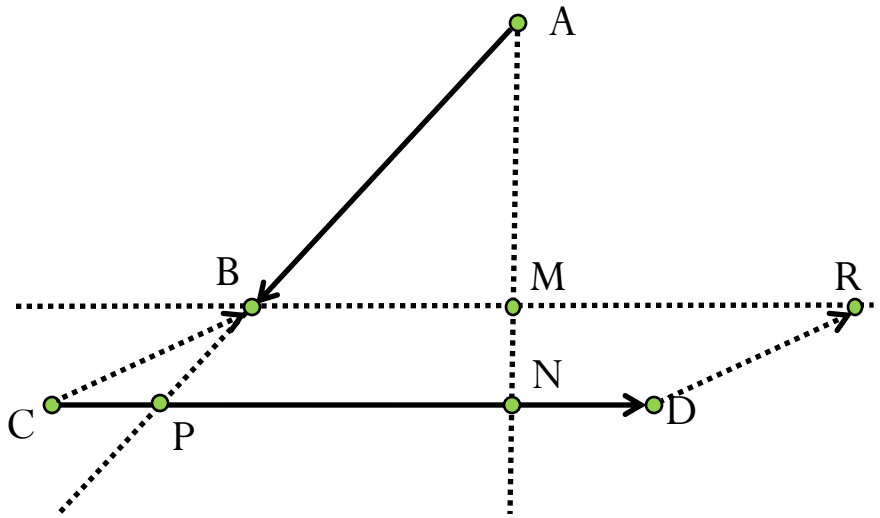
Por lo tanto:

$$P = (x, y) = A + \overrightarrow{AB}.\frac{\overrightarrow{AC} \times \overrightarrow{CD}}{\overrightarrow{AB} \times \overrightarrow{CD}}$$

Intersección de rectas

Problema 2: Dadas 2 rectas no paralelas AB y CD, hallar el punto de intersección.

Solución gráfica:



$$\mathrm{P} = (\mathrm{x}, \mathrm{y}) = A + \frac{|\overline{AP}|}{|\overline{AB}|}\overrightarrow{AB}$$

$$\frac{|\overline{AP}|}{|\overline{AB}|} = \frac{|\overline{AN}|}{|\overline{AM}|} = \frac{area(C, D, A)}{area(B, R, A)}$$

$$\frac{|\overline{AP}|}{|\overline{AB}|} = \frac{\overrightarrow{CD} \times \overrightarrow{CA}}{\overrightarrow{CD} \times \overrightarrow{BA}}$$

Por lo tanto:

$$\mathrm{P} = (\mathrm{x}, \mathrm{y}) = A + \overrightarrow{AB}.\frac{\overrightarrow{CD} \times \overrightarrow{CA}}{\overrightarrow{CD} \times \overrightarrow{BA}}$$

Intersección de rectas

$$P = (x, y) = A + \overrightarrow{AB}.\frac{\overrightarrow{AC} \times \overrightarrow{CD}}{\overrightarrow{AB} \times \overrightarrow{CD}}$$

$$P = (x, y) = A + \overrightarrow{AB}.\frac{\overrightarrow{CD} \times \overrightarrow{CA}}{\overrightarrow{CD} \times \overrightarrow{BA}}$$

Ejercicio: Demostrar que ambas formulas son equivalentes, y también demostrar que:

$$\vec{u} \times \vec{v} = -\vec{v} \times \vec{u}$$
$$\vec{u} \times \vec{v} = \vec{v} \times \overrightarrow{-u}$$
$$\vec{u} \times \vec{v} = -\vec{u} \times \overrightarrow{-v}$$
$$\vec{u} \times \vec{v} = \overrightarrow{-u} \times \overrightarrow{-v}$$
$$\vec{u} \times \vec{v} = -\overrightarrow{-u} \times \vec{v}$$

El signo del producto vectorial cambia al:

- Intercambiar los operandos
- Cambiar algún operando por su negativo

Intersección de rectas

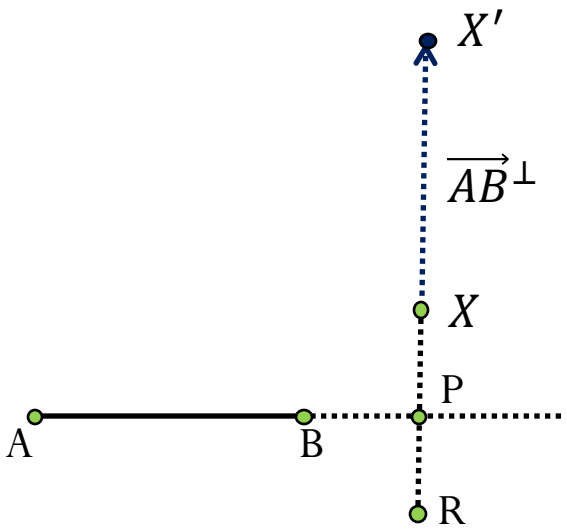
Problema 2: Dadas 2 rectas no paralelas AB y CD, hallar el punto de intersección.

```
double cross(const Vector &A, const Vector &B) { return A.x * B.y - A.y * B.x; }

Point lineIntersection(const Point &A, const Point &B, const Point &C, const Point &D)
{
    return A + (B - A) * (cross(C - A, D - C) / cross(B - A, D - C));
}
```

Proyección y reflexión de punto en recta

Problema 3: Dado un punto X y una recta AB, hallar la proyección y reflexión de X en AB

Solución:



Proyección de X en AB:

$$P = lineIntersection(A, B, X, X + \overrightarrow{AB}^{\perp})$$

Reflexión de X con respecto a AB:

$$R = X + 2\overrightarrow{XP} = X + 2(P - X)$$
$$R = 2P - X$$

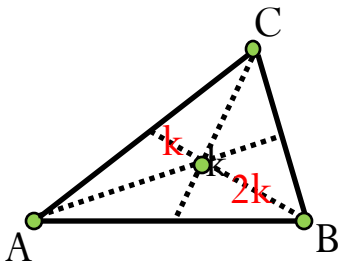
Donde P es la proyección de X en AB

Centros notables del triangulo

Problema 4: Dado un triangulo ABC, hallar el baricentro, ortocentro, circuncentro y el incentro.

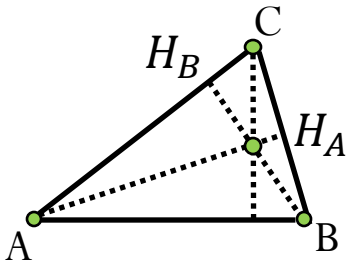
Solución:

A. Baricentro: Centro de gravedad o punto de intersección de medianas.



$$G = \frac{A + B + C}{3}$$

B. Ortocentro: Punto de intersección de alturas.



$$H = lineIntersection(A, H_A, B, H_B)$$

$$H_A = lineIntersection(A, A + \overrightarrow{BC}^{\perp}, B, C)$$

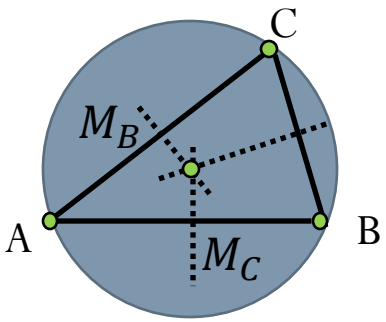
$$H_B = lineIntersection(B, B + \overrightarrow{AC}^{\perp}, A, C)$$

Centros notables del triangulo

Problema 4: Dado un triangulo ABC, hallar el baricentro, ortocentro, circuncentro y el incentro.

Solución:

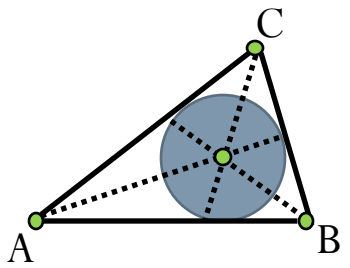
C. Circuncentro: Punto de intersección de mediatrices.



$$O = lineIntersection(M_C, M_C + \overrightarrow{AB}^{\perp}, M_B, M_B + \overrightarrow{AC}^{\perp})$$

$$M_C = \frac{A+B}{2} \qquad M_B = \frac{A+C}{2}$$

D. Incentro: Punto de intersección de bisectrices.



$$\mathrm{I} = lineIntersection(A, A + U_A, B, B + U_B)$$

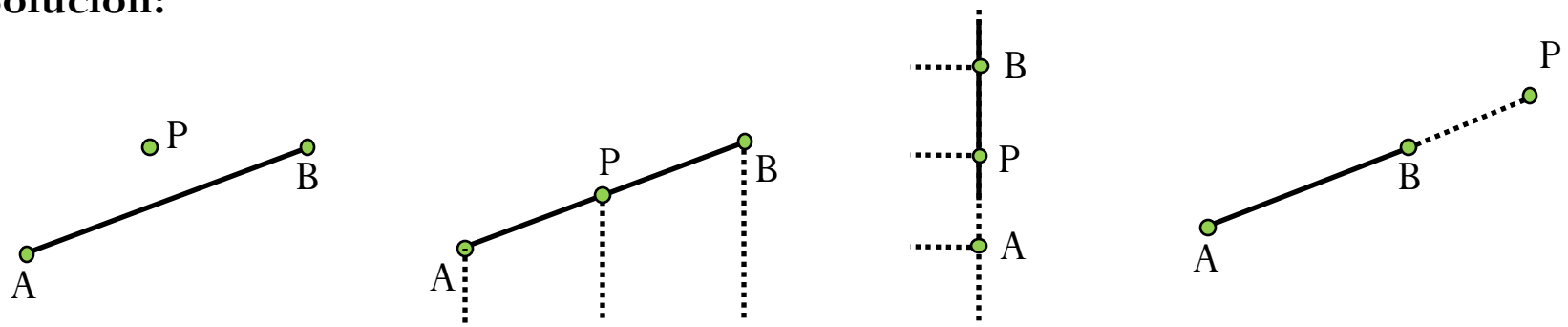
$$U_A = \overrightarrow{AC}.unit(\,) + \overrightarrow{AB}.unit(\,)$$

$$U_B = \overrightarrow{BA}.unit(\,) + \overrightarrow{BC}.unit(\,)$$

Punto en segmento

Problema 5: Dado un punto P y un segmentos AB, determinar si P pertenece a AB.

Solución:



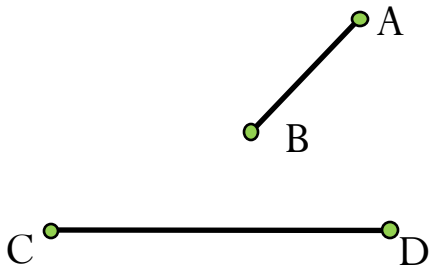
El área de ABP debe ser 0, y las coordenadas de P deben estar en el rango de AB:

```
bool onSegment(const Point &A, const Point &B, const Point &P)
{
    return abs(area(A, B, P)) < EPS &&
           P.x >= min(A.x, B.x) && P.x <= max(A.x, B.x) &&
           P.y >= min(A.y, B.y) && P.y <= max(A.y, B.y);
}
```

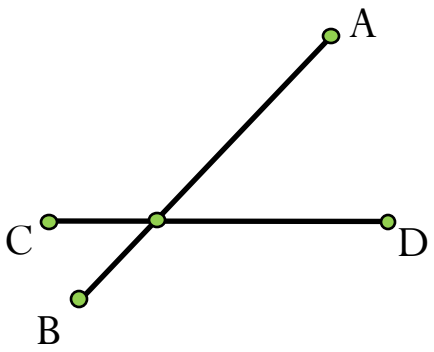
Intersección de segmentos

Problema 5: Dados 2 segmentos AB y CD, determinar si se intersectan o no.

Solución:



A y B a la izquierda de $\overrightarrow{CD}$

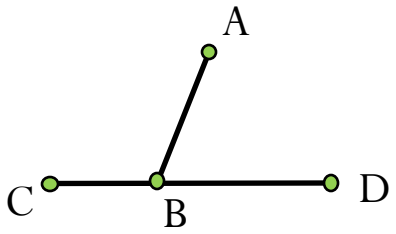


A a la izquierda de $\overrightarrow{CD}$

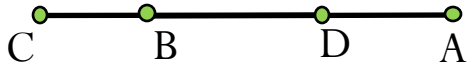
B a la derecha de $\overrightarrow{CD}$

C a la derecha de $\overrightarrow{AB}$

D a la izquierda de $\overrightarrow{AB}$



B en segmento $\overline{CD}$



B en segmento $\overline{CD}$

D en segmento $\overline{AB}$

Se intersectan ↔ Puntos en lados opuestos ∨ Algún punto dentro del segmento contrario

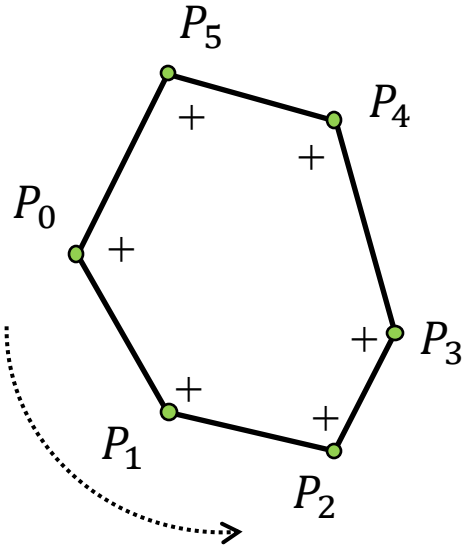
Intersección de segmentos

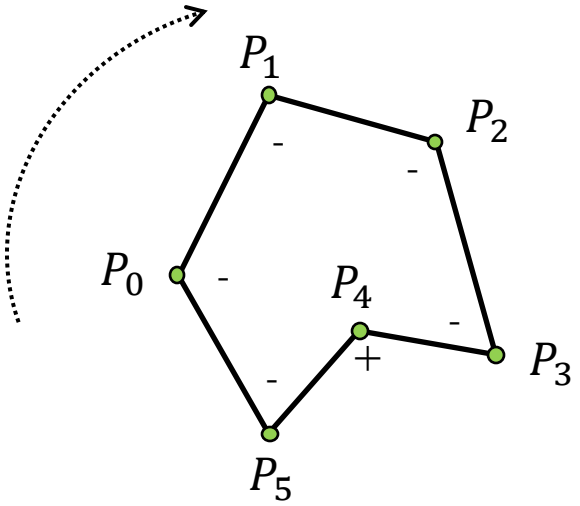
```
bool intersects(const Point &P1, const Point &P2, const Point &P3, const Point &P4)
{
    double A1 = area(P3, P4, P1);
    double A2 = area(P3, P4, P2);
    double A3 = area(P1, P2, P3);
    double A4 = area(P1, P2, P4);

    if( ((A1 > 0 && A2 < 0) || (A1 < 0 && A2 > 0)) &&
        ((A3 > 0 && A4 < 0) || (A3 < 0 && A4 > 0)))
            return true;

    else if(A1 == 0 && onSegment(P3, P4, P1)) return true;
    else if(A2 == 0 && onSegment(P3, P4, P2)) return true;
    else if(A3 == 0 && onSegment(P1, P2, P3)) return true;
    else if(A4 == 0 && onSegment(P1, P2, P4)) return true;
    else return false;
}
```

Polígonos convexos y no convexos

Problema 6: Dado un polígono simple P de n vértices P_i, determinar si es convexo o no convexo.

Solución: Recorremos el polígono en el orden que se encuentre y verificamos el sentido de giro en cada vértice. Si el polígono es convexo, todos los vértices deben tener el mismo sentido de giro.





El sentido de giro en el vértice P_i equivale al signo del área(P_{i-1}, P_i, P_{i+1})

Polígonos convexos y no convexos

Problema 6: Dado un polígono simple P de n vértices P_i, determinar si es convexo o no convexo.

Solución: Recorremos el polígono en el orden que se encuentre y verificamos el sentido de giro en cada vértice. Si el polígono es convexo, todos los vértices deben tener el mismo sentido de giro.

```
bool isConvex(const vector <Point> &P)
{
    int n = P.size(), pos = 0, neg = 0;
    for(int i=0; i<n; i++)
    {
        double A = area(P[i], P[(i+1)%n], P[(i+2)%n]);
        if(A < 0) neg++;
        else if(A > 0) pos++;
    }
    return neg == 0 || pos == 0;
}
```

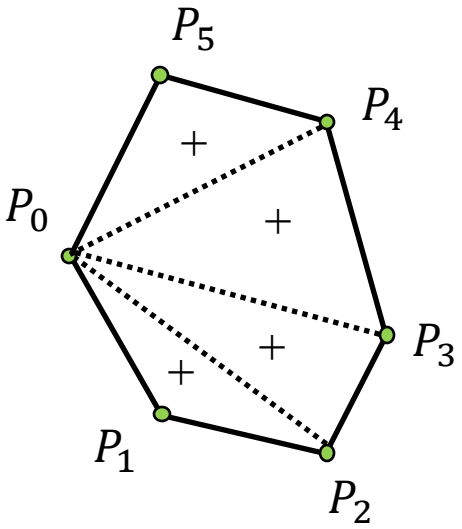
Área de Polígonos

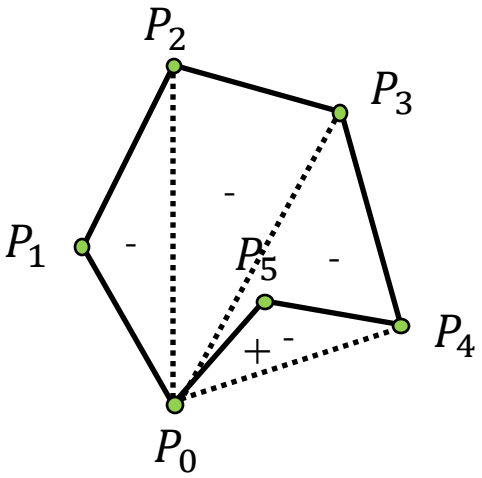
Problema 7: Dado un polígono simple P de n vértices P_i, el cual puede ser convexo o no convexo, determinar su área.

Solución:

Para un polígono **convexo** podemos triangularlo uniendo el primer vértice P_0 con todos los demás vértices y luego el área será la suma de áreas de todos los triángulos.

Para un polígono **no convexo**, también se puede hacer lo mismo!, pero se debe considerar el área con signo de cada triangulo.





$$Area(P) = \left| \sum_{i=1}^{n-2} area(P_0, P_i, P_{i+1}) \right|$$

Área de Polígonos

Problema 7: Dado un polígono simple P de n vértices P_i, el cual puede ser convexo o no convexo, determinar su área.

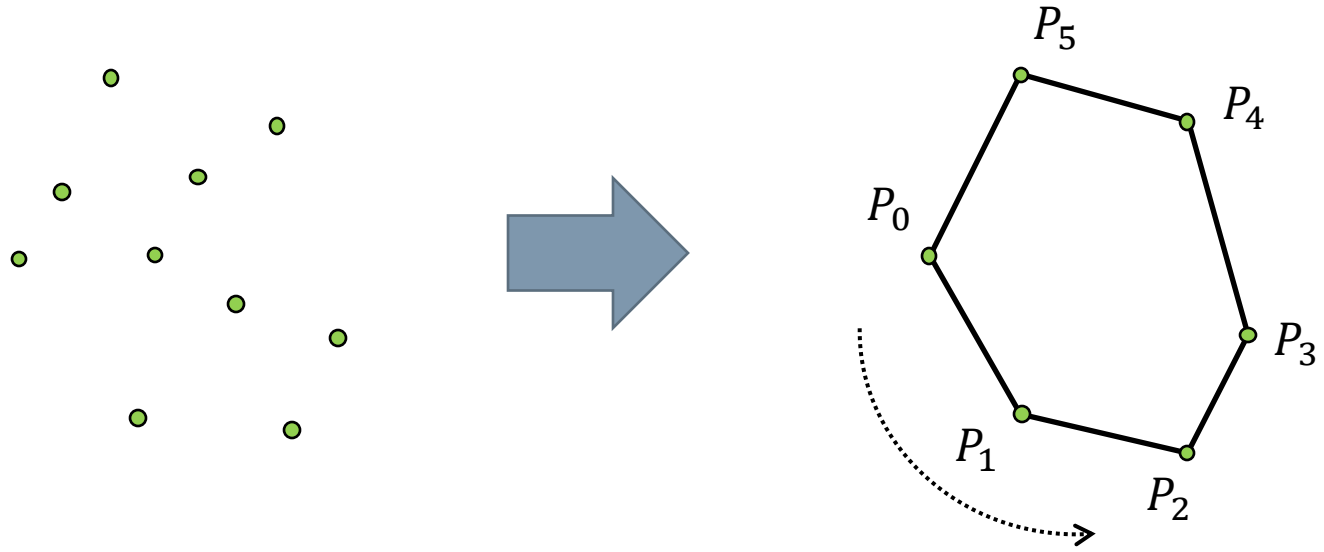
Solución:

$$Area(P) = \left| \sum_{i=1}^{n-2} area(P_0, P_i, P_{i+1}) \right|$$

```
double area(const vector <Point> &P)
{
    int n = P.size();
    double A = 0;
    for(int i=1; i<=n-2; i++)
        A += area(P[0], P[i], P[i+1]);
    return abs(A/2);
}
```

Convex Hull

Problema 8: Dado un conjunto de n puntos en el plano XY, hallar el polígono convexo de menor área que contiene todos los puntos en su interior o en sus bordes (convex hull).

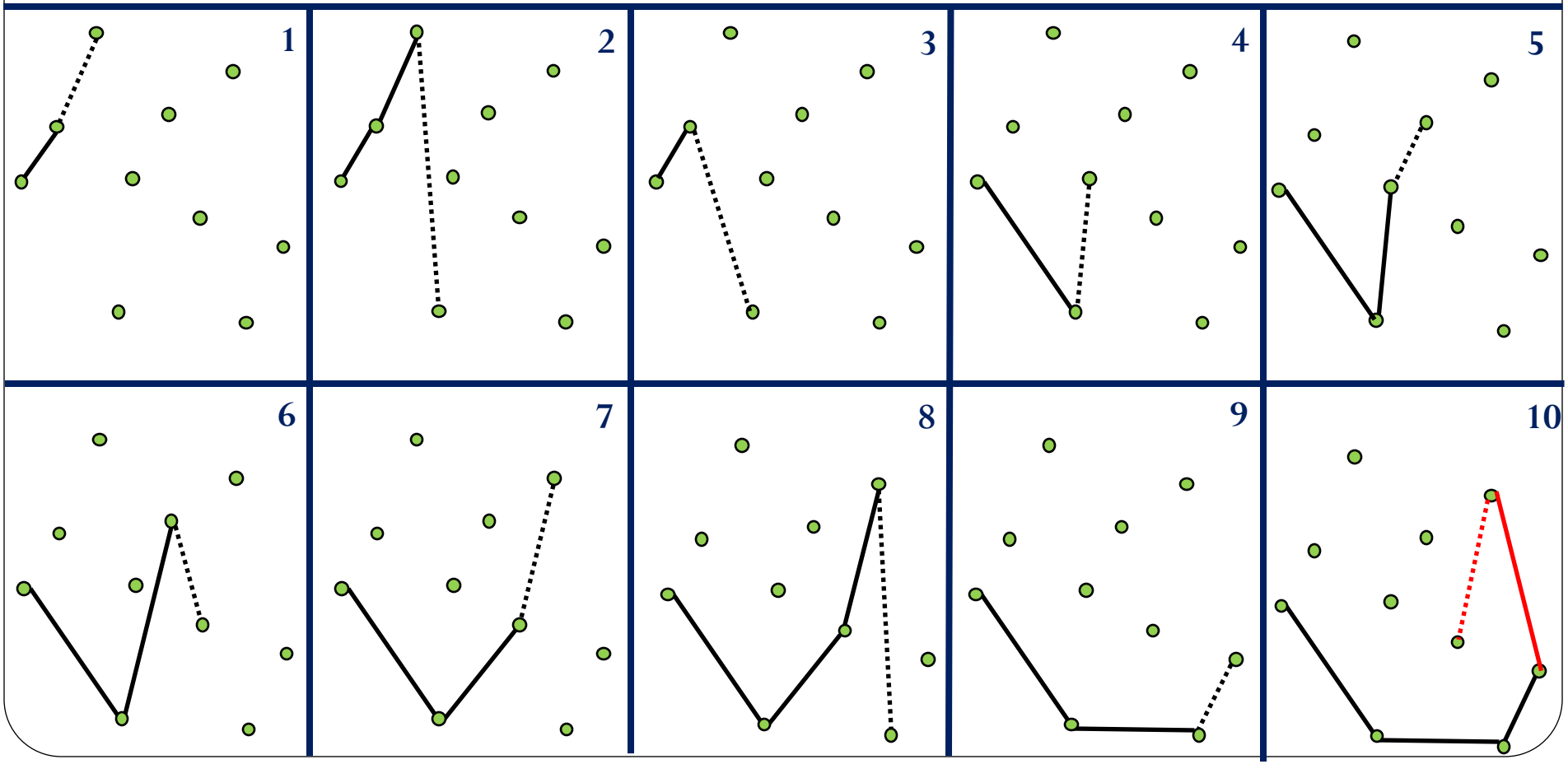


Convex Hull – Andrew's monotone chain - O(nlogn)

O(nlogn): Primero se ordenan los puntos en orden lexicográfico (primero por x, luego por y).

O(n): Luego encuentra la cadena inferior, empezando en el punto con menor x. La cadena se va expandiendo de modo que los giros sean únicamente en sentido positivo(antihorario).

O(n): Finalmente se continua formando la cadena superior, desde el punto con mayor x. Se mantiene el sentido de giro positivo.



Convex Hull – Andrew's monotone chain - O(nlogn)

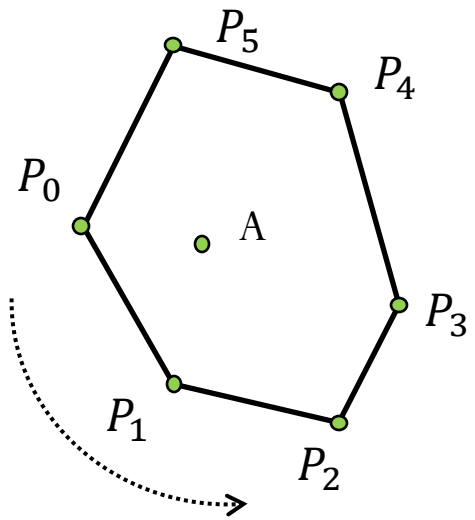
```
vector <Point> ConvexHull(vector <Point> P)
{
    //Ordenamiento lexicografico
    sort(P.begin(),P.end());

    int n = P.size(),k = 0;
    Point H[2*n];

    //Mitad inferior
    for(int i=0;i<n;++i){
        while(k>=2 && area(H[k-2],H[k-1],P[i]) <= 0) --k;
        H[k++] = P[i];
    }

    //Mitad superior
    for(int i=n-2,t=k;i>=0;--i){
        while(k>t && area(H[k-2],H[k-1],P[i]) <= 0) --k;
        H[k++] = P[i];
    }

    return vector <Point> (H,H+k-1);
}
```

Punto dentro de polígono convexo

Problema 9: Dado un punto A y un polígono **convexo** P, determinar si A se encuentra en el interior del polígono P.

Solución:

Basta con verificar que A se encuentre a un mismo lado (o a la izquierda o a la derecha) de todos los vectores $P_i\ P_{i+1}$. En otras palabras, todas las áreas(A, P_i, P_{i+1}) deben tener el mismo signo.



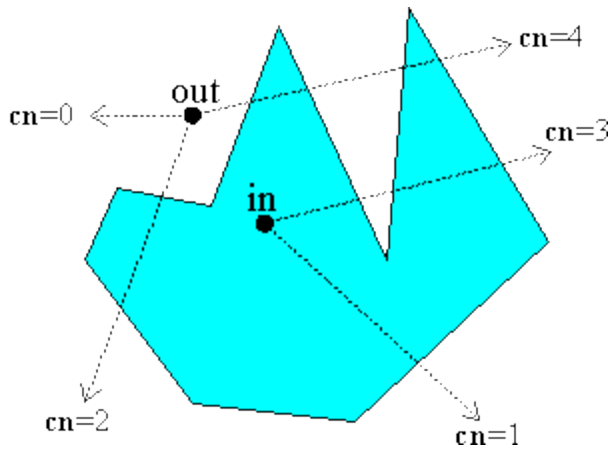

```
bool isInConvex(const vector <Point> &P, const Point &A)
{
    int n = P.size(), pos = 0, neg = 0;
    for(int i=0; i<n; i++)
    {
        double A = area(A, P[i], P[(i+1)%n]);
        if(A < 0) neg++;
        else if(A > 0) pos++;
    }
    return neg == 0 || pos == 0;
}
```

Punto dentro de polígono cualquiera

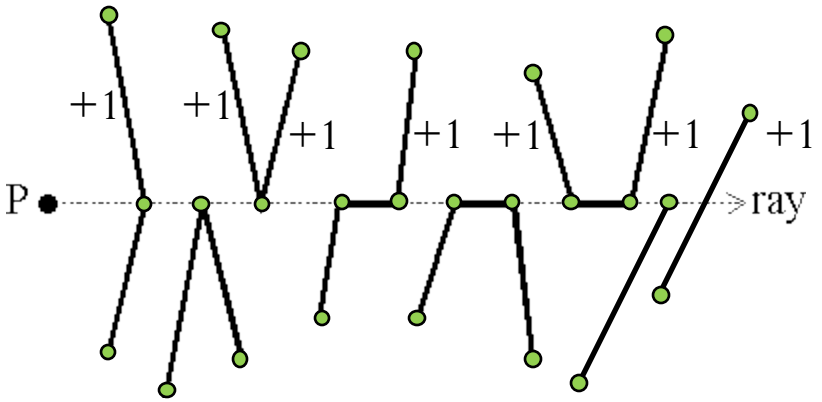
Problema 10: Dado un punto A y un polígono P, convexo o no convexo, determinar si A se encuentra en el interior del polígono P.

Solución:

Trazamos un rayo cualquiera desde el punto P y contamos el número de veces que cruza un lado. Si el número es impar el punto esta dentro del polígono, en caso contrario esta afuera.



Para un rayo horizontal P, P + Vector(INF, 0):

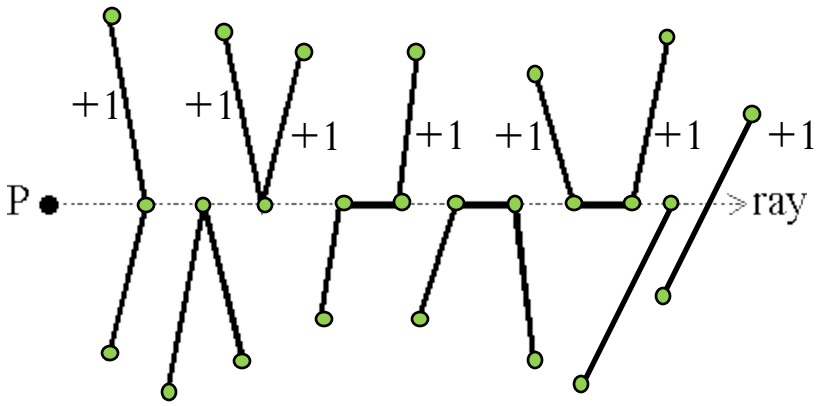


Se consideran solo los lados que tengan un extremo estrictamente por encima del rayo y el otro extremo por debajo o en el rayo.

Punto dentro de polígono cualquiera

Problema 10: Dado un punto A y un polígono P, convexo o no convexo, determinar si A se encuentra en el interior del polígono P.

Solución:



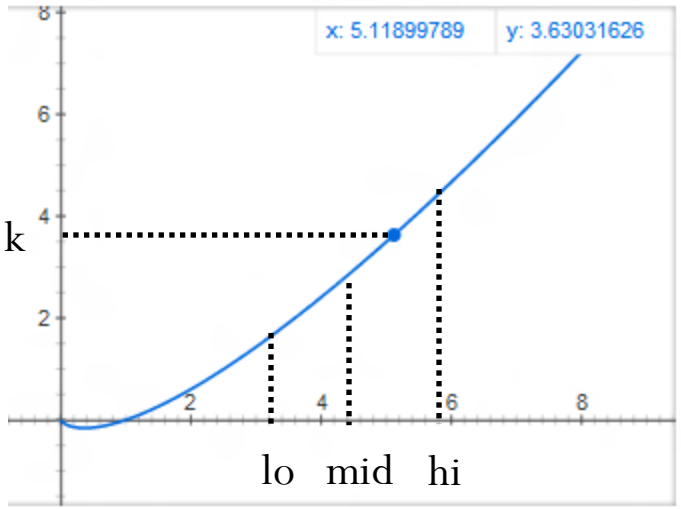

```
bool pointInPoly(const vector <Point> &P, const Point &A)
{
    int n = P.size(), cnt = 0;
    for(int i=0; i<n; i++)
    {
        int inf = i, sup = (i+1)%n;
        if(P[inf].y > P[sup].y) swap(inf, sup);
        if(P[inf].y <= A.y && A.y < P[sup].y)
            if(area(A, P[inf], P[sup]) > 0)
                cnt++;
    }
    return (cnt % 2) == 1;
}
```

Binary search

Binary search continuo

Dado un $k \geq 1$, encontrar un x positivo tal que $x\log(x) = k$

```
double f(double x) { return x * log10(x); }
double BS(double k){
    double lo = 1, hi = 1e8;
    for(int it=0; it<50; it++){
        double mid = (lo + hi) / 2;
        if(k < f(mid)) hi = mid;
        else lo = mid;
    }
}
```



Binary search discreto

Se debe cumplir la condición de monotonía. Es decir, podemos encontrar una función de la forma isValid(i) cuyos valores sean de las siguientes formas:

0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1
0, 0, 0, 0, 0, 0, 0, **0, 1**, 1, 1, 1, 1

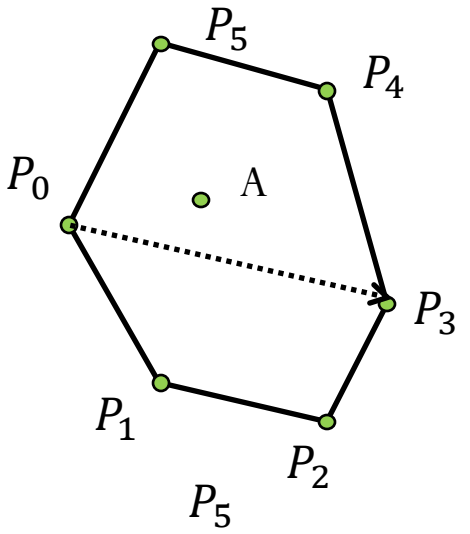
```
int BS(int n){
    int lo = 0, hi = n-1;
    if(!isValid(hi)) return -1;
    if(isValid(lo)) return lo;
    while(hi - lo > 1){
        int mid = lo + (hi - lo) / 2;
        if(isValid(mid)) hi = mid;
        else lo = mid;
    }
    return hi;
}
```

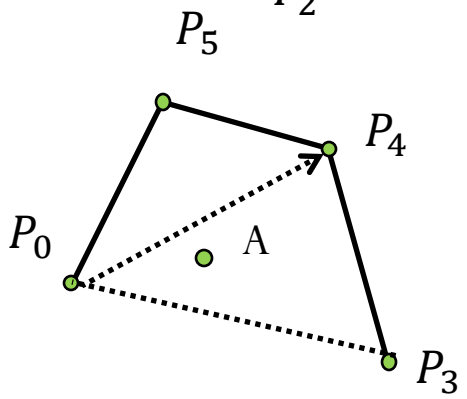
Punto dentro de polígono convexo

Problema 11: Dado un punto A y un polígono **convexo** P, determinar **eficientemente** si A se encuentra en el interior del polígono P.

Solución:

Podemos partir el polígono en 2 mitades y ver en que mitad debería estar el punto. Si hacemos esto varias veces resolveremos el problema en O(log n), porque en cada paso nos deshacemos de la mitad de vértices.

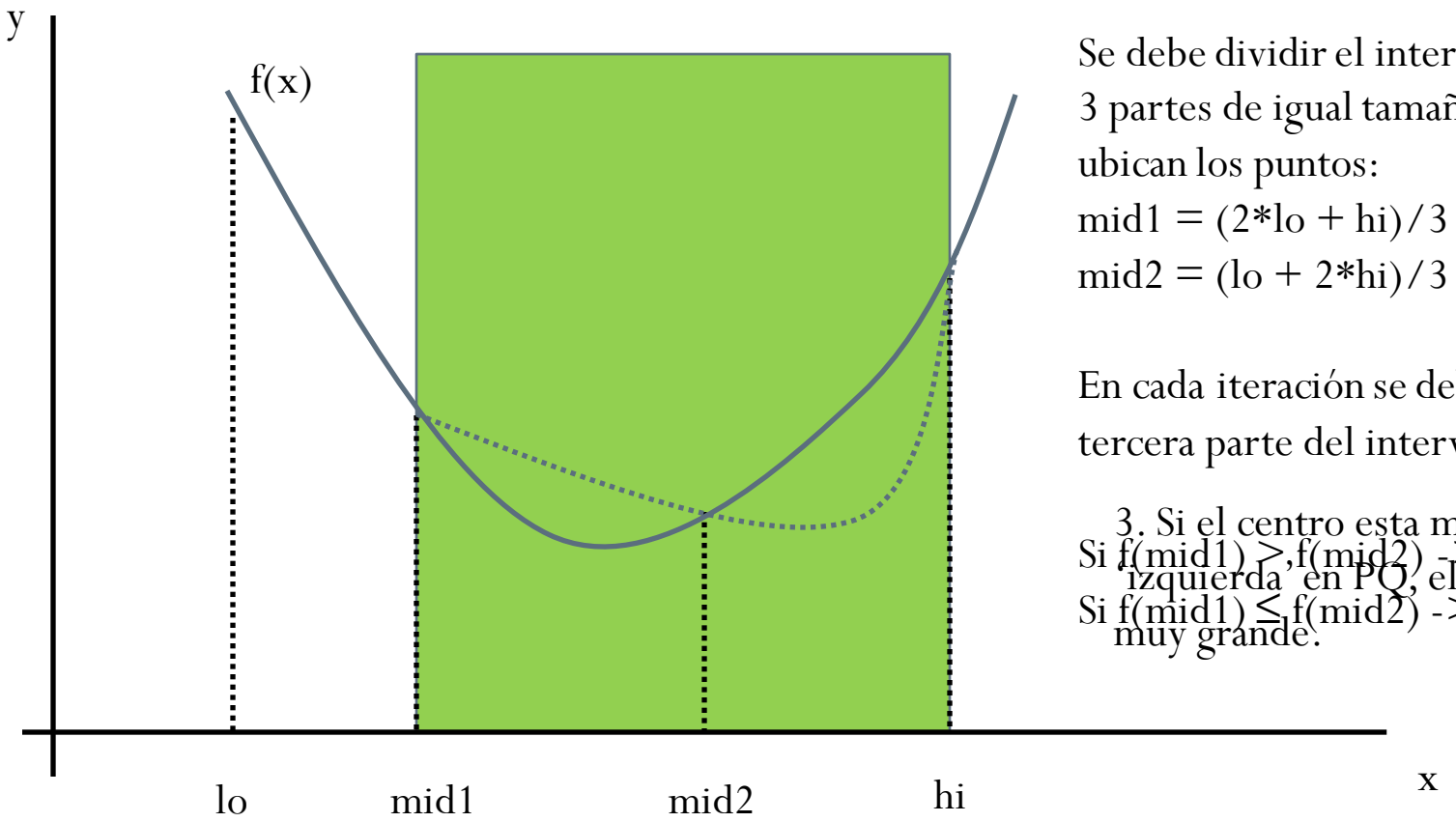





```
bool isInConvex(vector <Point> &P, const Point &A){
        int n = P.size(), lo = 1, hi = P.size() - 1;

        if(area(P[0], P[1], A) <= 0) return 0;
        if(area(P[n-1], P[0], A) <= 0) return 0;

        while(hi - lo > 1)
        {
                int mid = (lo + hi) / 2;

                if(area(P[0], P[mid], A) > 0) lo = mid;
                else hi = mid;
        }
        return area(P[lo], P[hi], A) > 0;
}
```

Ternary search

Sea f(x) una función unimodal(con forma similar a una parábola), tal como se indica en la figura, encontrar el menor valor de f(x).



Se debe dividir el intervalo [lo, hi] en 3 partes de igual tamaño. Para ello se ubican los puntos:

mid1 = (2*lo + hi)/3

mid2 = (lo + 2*hi)/3

En cada iteración se debe eliminar la tercera parte del intervalo:

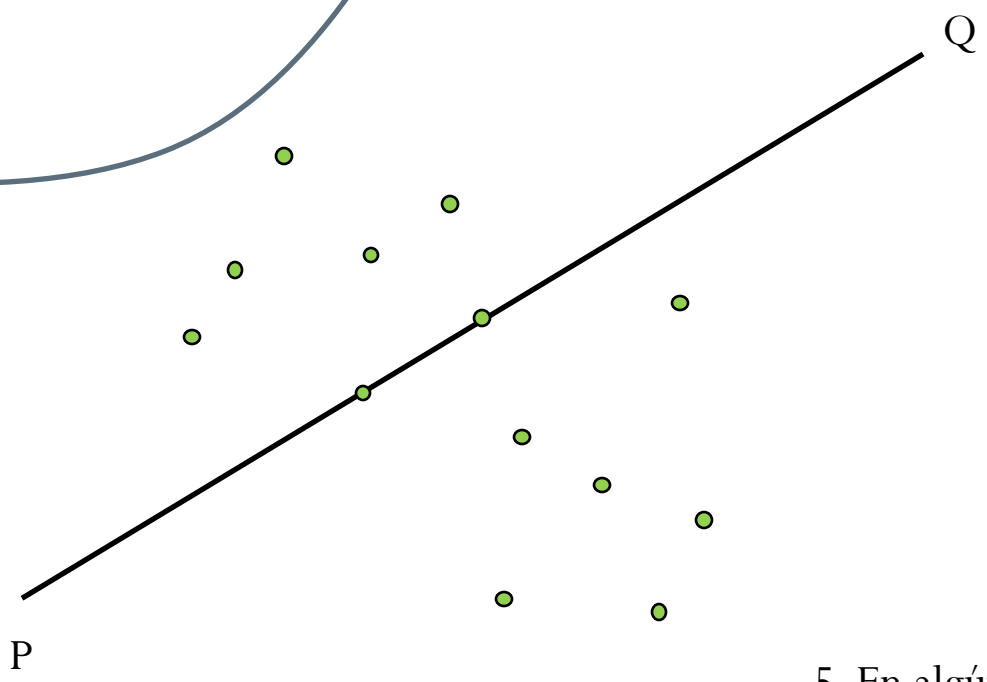
Si f(mid1) > f(mid2) -> lo = mid1

Si f(mid1) ≤ f(mid2) -> hi = mid2

3. Si el centro esta muy a la izquierda en PQ, el radio sera muy grande.

Ternary search

Problema 12: Dado un conjunto de n puntos en el plano XY, hallar el menor circulo que contenga a todos los puntos y cuyo centro este en la recta PQ.



1. Buscaremos el centro!

2. Queremos minimizar la distancia del centro al punto mas lejano. Esta distancia será el radio del circulo

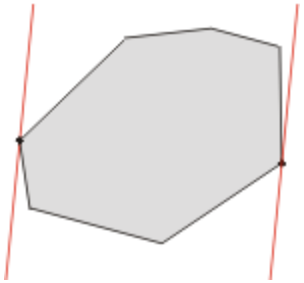
3. Si el centro esta muy a la 'derecha' en PQ, el radio será muy grande.

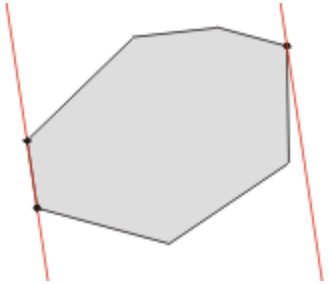
4. Si el centro esta muy a la 'izquierda' en PQ, el radio será muy grande.

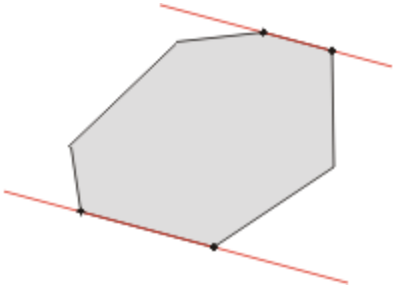
5. En algún lugar del 'medio', el radio será mínimo.

6. Muy grande – mínimo – muy grande -> Forma de Parábola -> Ternary search!!

Rotating Callipers: Anti-podal pairs







Dado un polígono convexo P, se dice que 2 vértices P_i y P_j forman un par antipodal si es que admiten líneas de soporte paralelas('tangentes' que pasan por dichos puntos).

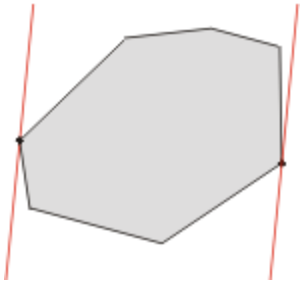
Existen O(n) pares antipodales.

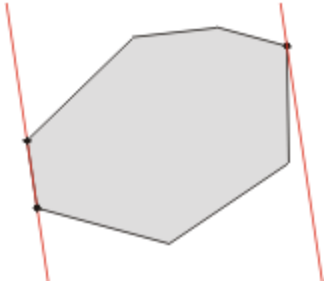
Muchos algoritmos requieren analizar únicamente pares antipodales. Si podemos hallarlos todos en O(n) entonces podemos resolver eficientemente estos problemas.

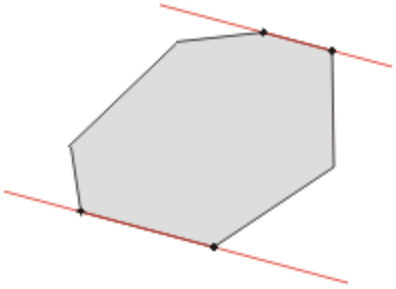
Como hallamos todos los pares antipodales en O(n)?

Rotating Callipers!

Rotating Callipers: The easy way





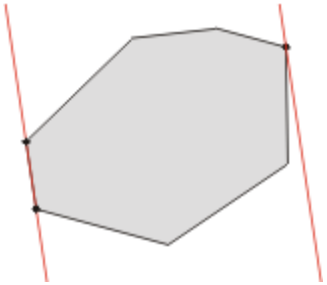


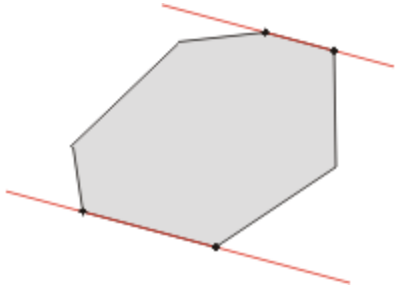
Obs1: Siempre podemos utilizar un lado como línea de soporte (si no fuera así podemos mover las líneas de soporte ligeramente hasta hacerlas coincidir con un lado).

Obs2: Si fijamos un lado como línea de soporte, el vértice mas lejano a esta línea formara un par antipodar con cada uno de los vértices del lado.

Obs3: Podemos iterar sobre todos los lados y hallar todos los antipodales, simplemente actualizando el punto mas lejano a cada lado.

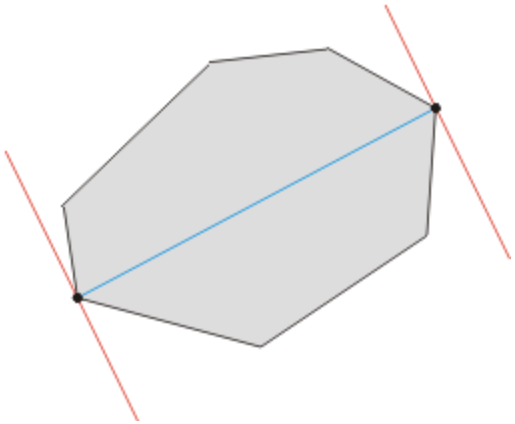
Rotating Callipers: The easy way

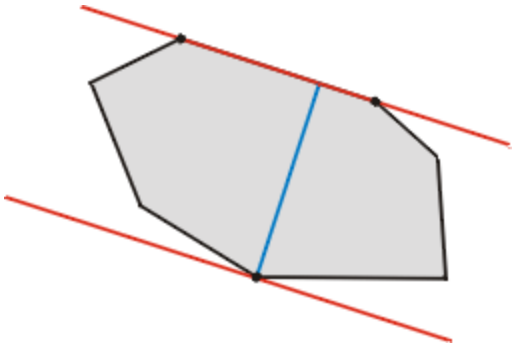




```
for(int i=0, j=2; i<n; i++)
{
    // P[j] debe ser el punto mas lejano a la linea P[i], P[(i+1)%n]:
    while(area(P[i], P[(i+1)%n], P[(j+1)%n]) > area(P[i], P[(i+1)%n], P[j])) j = (j+1)%n;

    // Par antipodal: i, j
    // Par antipodal: (i+1)%n, j
}
```

Rotating Callipers: Diámetro y ancho de polígono convexo

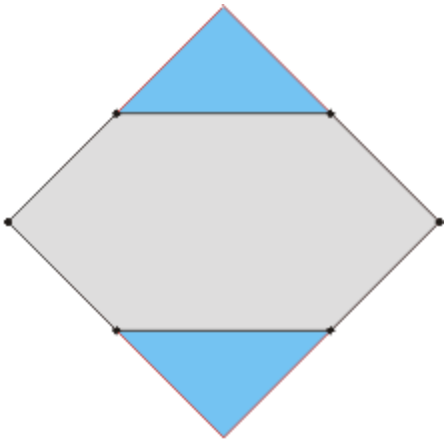




Solo se deben considerar pares antipodales!

Rotating Callipers: Enclosing rectangles

Área mínima



Perímetro mínimo

Podemos mantener 4 punteros!