

Geometría en programación competitiva

Invierno de entrenamiento 2020

Alan Enrique Ontiveros Salazar

Club de Algoritmia ESCOM

8 de enero 2020



Contenido

1 Conceptos básicos

- Definición de vector
- Posición vs desplazamiento
- Operaciones básicas
- Magnitud y vector unitario
- Rotación de vectores y puntos
- Representación de un punto en C++

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos



Contenido

1 Conceptos básicos

- Definición de vector
- Posición vs desplazamiento
- Operaciones básicas
- Magnitud y vector unitario
- Rotación de vectores y puntos
- Representación de un punto en C++

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos



Definición de vector

- Desde el punto de vista geométrico, un **vector** es una flecha que apunta desde el origen del plano hasta un punto específico. De esa forma vemos que tiene una **longitud** (la distancia desde el origen hasta el punto), **dirección** (la recta que contiene tanto al origen como al punto) y **sentido** (hacia dónde apunta la flecha).



Definición de vector

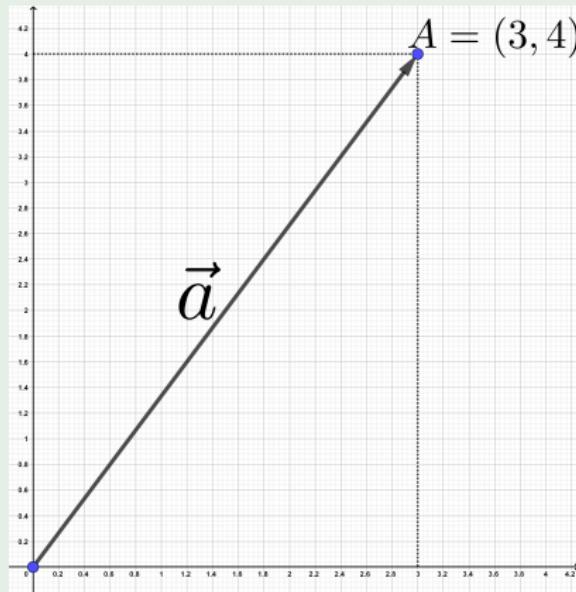
- Desde el punto de vista geométrico, un **vector** es una flecha que apunta desde el origen del plano hasta un punto específico. De esa forma vemos que tiene una **longitud** (la distancia desde el origen hasta el punto), **dirección** (la recta que contiene tanto al origen como al punto) y **sentido** (hacia dónde apunta la flecha).
- Desde el punto de vista algebráico, un **vector** \vec{a} es un elemento de \mathbb{R}^2 y escribimos $\vec{a} = (a_1, a_2)$, donde $a_1, a_2 \in \mathbb{R}$ son las **componentes** o **coordenadas** de \vec{a} .



Definición de vector

Ejemplo

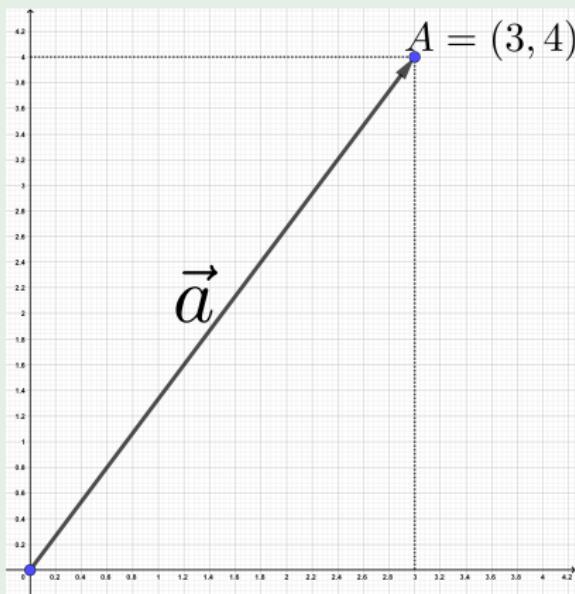
Si tenemos el punto $A = (3, 4)$, el vector \vec{a} sería el siguiente:



Definición de vector

Ejemplo

Si tenemos el punto $A = (3, 4)$, el vector \vec{a} sería el siguiente:



Vemos también que $\vec{a} = (3, 4)$ tiene las mismas coordenadas.

Contenido

1 Conceptos básicos

- Definición de vector
- Posición vs desplazamiento
- Operaciones básicas
- Magnitud y vector unitario
- Rotación de vectores y puntos
- Representación de un punto en C++

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos



Posición vs desplazamiento

En esencia un punto y un vector son lo mismo, pero un punto solo indica una posición en el plano, mientras que un vector indica un **desplazamiento** entre dos puntos.



Posición vs desplazamiento

En esencia un punto y un vector son lo mismo, pero un punto solo indica una posición en el plano, mientras que un vector indica un **desplazamiento** entre dos puntos.

- Dado un punto P , definimos al **vector de posición** de P respecto del origen O como el vector \overrightarrow{OP} que tendrá las mismas coordenadas del punto P . Por simplicidad, también podemos representar a ese vector como \vec{P} .



Posición vs desplazamiento

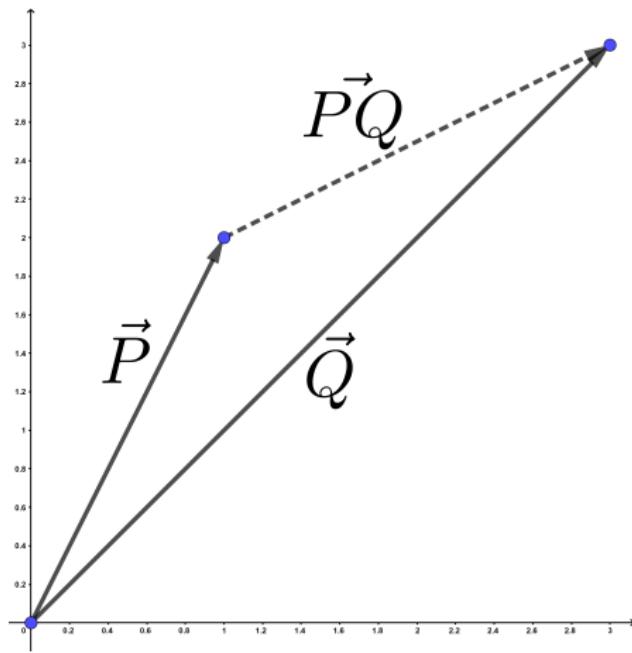
En esencia un punto y un vector son lo mismo, pero un punto solo indica una posición en el plano, mientras que un vector indica un **desplazamiento** entre dos puntos.

- Dado un punto P , definimos al **vector de posición** de P respecto del origen O como el vector \overrightarrow{OP} que tendrá las mismas coordenadas del punto P . Por simplicidad, también podemos representar a ese vector como \vec{P} .
- Dados dos puntos P y Q , definimos al **vector de desplazamiento** de P a Q como el vector \overrightarrow{PQ} , en donde el origen de la flecha está en P y la punta en Q . De esta forma vemos que los vectores no necesariamente comienzan en el origen, sino en donde queramos.



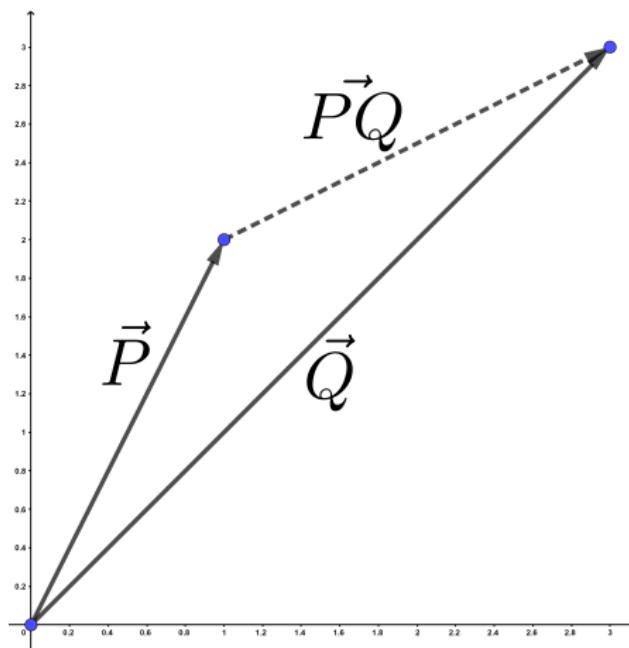
Posición vs desplazamiento

Por ejemplo, si tenemos que $P = (1, 2)$ y $Q = (3, 3)$, los vectores de posición y desplazamiento se verán así:



Posición vs desplazamiento

Por ejemplo, si tenemos que $P = (1, 2)$ y $Q = (3, 3)$, los vectores de posición y desplazamiento se verán así:



Pregunta: ¿Cuáles son las coordenadas de \overrightarrow{PQ} ?



Contenido

1 Conceptos básicos

- Definición de vector
- Posición vs desplazamiento
- Operaciones básicas
- Magnitud y vector unitario
- Rotación de vectores y puntos
- Representación de un punto en C++

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos



Operaciones básicas

Sean $\vec{a} = (a_1, a_2)$ y $\vec{b} = (b_1, b_2)$ dos vectores. Veamos qué operaciones podemos hacer con ellos:



Operaciones básicas

Sean $\vec{a} = (a_1, a_2)$ y $\vec{b} = (b_1, b_2)$ dos vectores. Veamos qué operaciones podemos hacer con ellos:

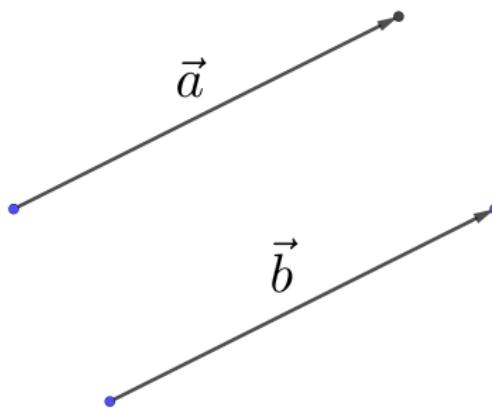
- **Igualdad:** decimos que $\vec{a} = \vec{b}$ si y solo si $a_1 = b_1$ y $a_2 = b_2$. Es decir, los vectores son iguales si y solo si sus coordenadas correspondientes son iguales.



Operaciones básicas

Sean $\vec{a} = (a_1, a_2)$ y $\vec{b} = (b_1, b_2)$ dos vectores. Veamos qué operaciones podemos hacer con ellos:

- **Igualdad:** decimos que $\vec{a} = \vec{b}$ si y solo si $a_1 = b_1$ y $a_2 = b_2$. Es decir, los vectores son iguales si y solo si sus coordenadas correspondientes son iguales.



En este caso, $\vec{a} = \vec{b} = (2, 1)$. Aunque \vec{a} y \vec{b} no tengan el mismo origen, si sus componentes son iguales, los vectores son iguales.



Operaciones básicas

Sean $\vec{a} = (a_1, a_2)$ y $\vec{b} = (b_1, b_2)$ dos vectores. Veamos qué operaciones podemos hacer con ellos:

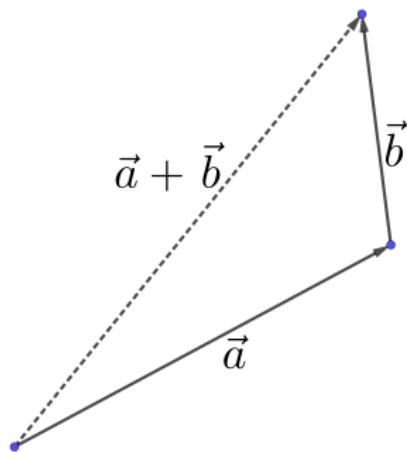
- **Suma:** $\vec{a} + \vec{b} = (a_1 + b_1, a_2 + b_2)$. Es decir, para sumar dos vectores solo sumamos sus componentes correspondientes.



Operaciones básicas

Sean $\vec{a} = (a_1, a_2)$ y $\vec{b} = (b_1, b_2)$ dos vectores. Veamos qué operaciones podemos hacer con ellos:

- **Suma:** $\vec{a} + \vec{b} = (a_1 + b_1, a_2 + b_2)$. Es decir, para sumar dos vectores solo sumamos sus componentes correspondientes.



Geométricamente, para sumar dos vectores colocamos el origen de \vec{b} junto a la flecha de \vec{a} y unimos desde el origen de \vec{a} hasta la flecha de \vec{b} .

Operaciones básicas

Sean $\vec{a} = (a_1, a_2)$ y $\vec{b} = (b_1, b_2)$ dos vectores. Veamos qué operaciones podemos hacer con ellos:

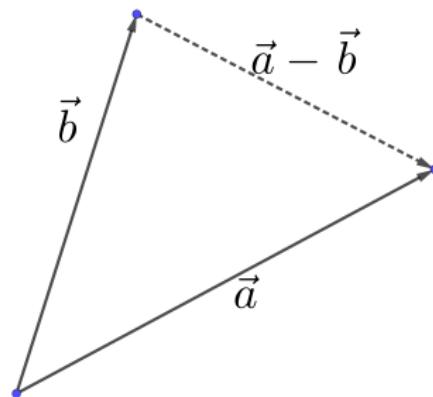
- **Resta:** $\vec{a} - \vec{b} = (a_1 - b_1, a_2 - b_2)$. Es decir, para restar dos vectores solo restamos sus componentes correspondientes.



Operaciones básicas

Sean $\vec{a} = (a_1, a_2)$ y $\vec{b} = (b_1, b_2)$ dos vectores. Veamos qué operaciones podemos hacer con ellos:

- **Resta:** $\vec{a} - \vec{b} = (a_1 - b_1, a_2 - b_2)$. Es decir, para restar dos vectores solo restamos sus componentes correspondientes.



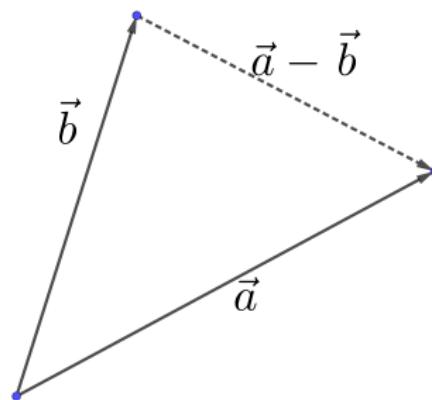
Geométricamente, para restar dos vectores los colocamos ambos en el mismo origen, y trazamos el vector resultante desde la flecha del segundo a la flecha del primero.



Operaciones básicas

Sean $\vec{a} = (a_1, a_2)$ y $\vec{b} = (b_1, b_2)$ dos vectores. Veamos qué operaciones podemos hacer con ellos:

- **Resta:** $\vec{a} - \vec{b} = (a_1 - b_1, a_2 - b_2)$. Es decir, para restar dos vectores solo restamos sus componentes correspondientes.



Y con esta definición, podemos decir que el vector de desplazamiento del punto P al punto Q es el vector $\overrightarrow{PQ} = \vec{Q} - \vec{P}$.



Operaciones básicas

Sean $\vec{a} = (a_1, a_2)$ y $\vec{b} = (b_1, b_2)$ dos vectores. Veamos qué operaciones podemos hacer con ellos:

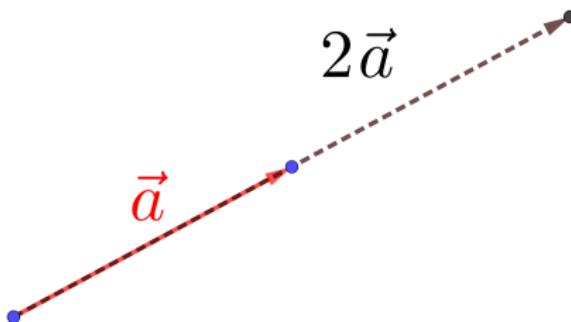
- **Multiplicación por escalar:** $k\vec{a} = (ka_1, ka_2, ka_3)$. Es decir, para multiplicar un vector por un escalar solo multiplicamos cada componente por dicho escalar.



Operaciones básicas

Sean $\vec{a} = (a_1, a_2)$ y $\vec{b} = (b_1, b_2)$ dos vectores. Veamos qué operaciones podemos hacer con ellos:

- **Multiplicación por escalar:** $k\vec{a} = (ka_1, ka_2, ka_3)$. Es decir, para multiplicar un vector por un escalar solo multiplicamos cada componente por dicho escalar.



Geométricamente, esta operación agranda o reduce el vector sin cambiar su dirección (su sentido se invierte si $k < 0$, se queda igual si $k > 0$ y obtenemos el cero vector si $k = 0$).



Contenido

1 Conceptos básicos

- Definición de vector
- Posición vs desplazamiento
- Operaciones básicas
- **Magnitud y vector unitario**
- Rotación de vectores y puntos
- Representación de un punto en C++

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos



Magnitud

Sea $\vec{a} = (x, y)$ un vector.

- La **magnitud, longitud o módulo** de \vec{a} es la distancia que hay del origen a la flecha de \vec{a} . Por el teorema de Pitágoras, la podemos definir como: $\|\vec{a}\| = \sqrt{x^2 + y^2}$.



Magnitud

Sea $\vec{a} = (x, y)$ un vector.

- La **magnitud, longitud o módulo** de \vec{a} es la distancia que hay del origen a la flecha de \vec{a} . Por el teorema de Pitágoras, la podemos definir como: $\|\vec{a}\| = \sqrt{x^2 + y^2}$.
- De esa definición podemos ver que $\|k\vec{a}\| = |k| \|\vec{a}\|$, es decir, al multiplicar el vector por un escalar, la longitud del vector también se multiplica por el valor absoluto del escalar, lo cual tiene sentido geométricamente.



Magnitud

Sea $\vec{a} = (x, y)$ un vector.

- La **magnitud, longitud o módulo** de \vec{a} es la distancia que hay del origen a la flecha de \vec{a} . Por el teorema de Pitágoras, la podemos definir como: $\|\vec{a}\| = \sqrt{x^2 + y^2}$.
- De esa definición podemos ver que $\|k\vec{a}\| = |k| \|\vec{a}\|$, es decir, al multiplicar el vector por un escalar, la longitud del vector también se multiplica por el valor absoluto del escalar, lo cual tiene sentido geométricamente.

Ejemplo

Si $\vec{a} = (3, 4)$, entonces $\|\vec{a}\| = \sqrt{3^2 + 4^2} = \sqrt{9 + 16} = \sqrt{25} = 5$.



Vector unitario

Sea $\vec{a} = (x, y)$ un vector.

- Si \vec{a} cumple que $\|\vec{a}\| = 1$, decimos que es un **vector unitario** y usualmente se denota como \hat{a} .



Vector unitario

Sea $\vec{a} = (x, y)$ un vector.

- Si \vec{a} cumple que $\|\vec{a}\| = 1$, decimos que es un **vector unitario** y usualmente se denota como \hat{a} .
- Para cualquier vector $\vec{a} \neq \vec{0}$ podemos obtener su equivalente unitario, es decir, otro vector \hat{a} cuya dirección y sentido sea el mismo que \vec{a} pero con longitud uno.



Vector unitario

Sea $\vec{a} = (x, y)$ un vector.

- Si \vec{a} cumple que $\|\vec{a}\| = 1$, decimos que es un **vector unitario** y usualmente se denota como \hat{a} .
- Para cualquier vector $\vec{a} \neq \vec{0}$ podemos obtener su equivalente unitario, es decir, otro vector \hat{a} cuya dirección y sentido sea el mismo que \vec{a} pero con longitud uno.

Para hacerlo, simplemente dividimos \vec{a} entre su magnitud, es decir,

$$\hat{a} = \frac{\vec{a}}{\|\vec{a}\|}.$$



Vector unitario

Sea $\vec{a} = (x, y)$ un vector.

- Si \vec{a} cumple que $\|\vec{a}\| = 1$, decimos que es un **vector unitario** y usualmente se denota como \hat{a} .
- Para cualquier vector $\vec{a} \neq \vec{0}$ podemos obtener su equivalente unitario, es decir, otro vector \hat{a} cuya dirección y sentido sea el mismo que \vec{a} pero con longitud uno.

Para hacerlo, simplemente dividimos \vec{a} entre su magnitud, es decir,

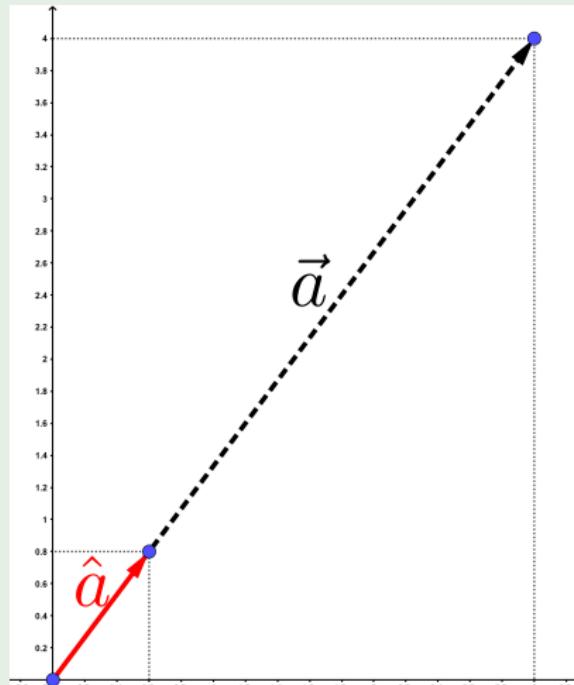
$$\hat{a} = \frac{\vec{a}}{\|\vec{a}\|}.$$



Vector unitario

Ejemplo

Si $\vec{a} = (3, 4)$, entonces $\hat{a} = \frac{(3,4)}{5} = \left(\frac{3}{5}, \frac{4}{5}\right)$.



Vector unitario

Algunas observaciones:

- Los vectores unitarios son muy útiles, pues en muchas ocasiones sabemos únicamente la dirección y sentido de movimiento, y con solo multiplicar dicho vector por un escalar, avanzamos esa cantidad en la dirección del vector.



Vector unitario

Algunas observaciones:

- Los vectores unitarios son muy útiles, pues en muchas ocasiones sabemos únicamente la dirección y sentido de movimiento, y con solo multiplicar dicho vector por un escalar, avanzamos esa cantidad en la dirección del vector.
- Si conocemos tanto la dirección y sentido como la longitud de un vector, podemos recuperar el vector original así: $\vec{a} = \|\vec{a}\| \hat{a}$.



Contenido

1 Conceptos básicos

- Definición de vector
- Posición vs desplazamiento
- Operaciones básicas
- Magnitud y vector unitario
- **Rotación de vectores y puntos**
- Representación de un punto en C++

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos



Rotación

- Sea $\vec{a} = (x, y)$ un vector. Para rotarlo un ángulo de θ alrededor de su origen en sentido antihorario, usamos la siguiente fórmula:

$$f(\vec{a}, \theta) = (x \cos \theta - y \sin \theta, x \sin \theta + y \cos \theta)$$



Rotación

- Sea $\vec{a} = (x, y)$ un vector. Para rotarlo un ángulo de θ alrededor de su origen en sentido antihorario, usamos la siguiente fórmula:

$$f(\vec{a}, \theta) = (x \cos \theta - y \sin \theta, x \sin \theta + y \cos \theta)$$

- Geométricamente, el origen del vector se mantiene en su lugar y la flecha es la que rota.



Rotación

- Sea $\vec{a} = (x, y)$ un vector. Para rotarlo un ángulo de θ alrededor de su origen en sentido antihorario, usamos la siguiente fórmula:

$$f(\vec{a}, \theta) = (x \cos \theta - y \sin \theta, x \sin \theta + y \cos \theta)$$

- Geométricamente, el origen del vector se mantiene en su lugar y la flecha es la que rota.
- Y si queremos rotar un punto Q en el plano alrededor de un punto P , lo que hay que hacer es rotar el vector de desplazamiento \overrightarrow{PQ} y al resultado sumarle el vector de posición \vec{P} . Es decir:

$$\vec{Q}' = \vec{P} + f(\overrightarrow{PQ}, \theta)$$



Ejemplo

Ejemplo: si $P = (1, 1)$ y $Q = (3, 2)$, entonces la rotación de Q alrededor de P con un ángulo de $\theta = 78^\circ$ es:

$$\overrightarrow{PQ} = \vec{Q} - \vec{P} = (3, 2) - (1, 1) = (2, 1)$$

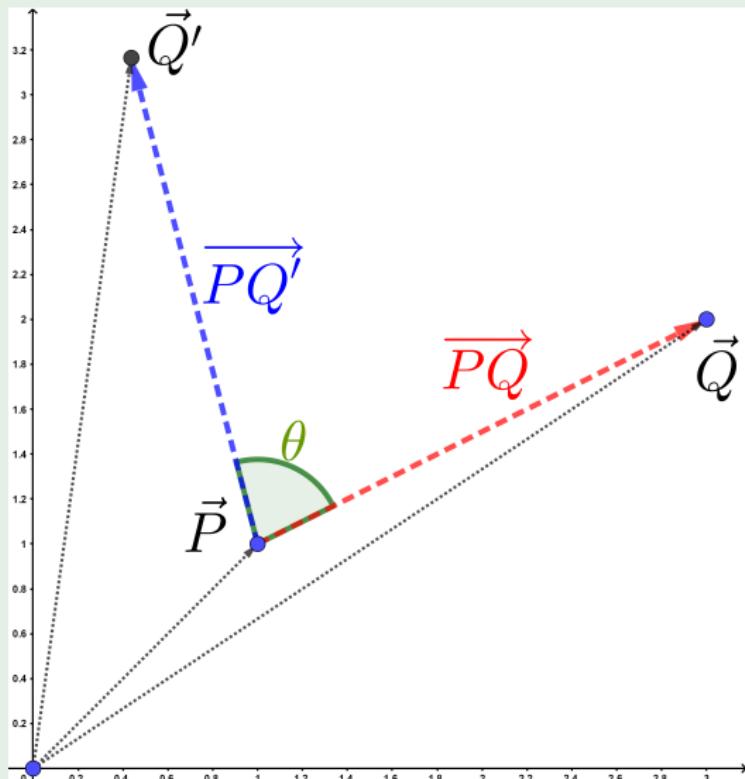
$$f(\overrightarrow{PQ}, \theta) = (2 \cos 78^\circ - 1 \sin 78^\circ, 2 \sin 78^\circ + 1 \cos 78^\circ) \approx (-0.5623, 2.1642)$$

$$Q' = \vec{P} + f(\overrightarrow{PQ}, \theta) \approx (1, 1) + (-0.5623, 2.1642) \approx (0.4377, 3.1642)$$



Rotación

Ejemplo



Rotación

- Un caso particular de rotación es cuando rotamos el vector $\vec{a} = (x, y)$ un ángulo de 90° a la izquierda.



Rotación

- Un caso particular de rotación es cuando rotamos el vector $\vec{a} = (x, y)$ un ángulo de 90° a la izquierda.
- Si aplicamos la fórmula de rotación, el nuevo vector tendrá coordenadas $(-y, x)$.



Rotación

- Un caso particular de rotación es cuando rotamos el vector $\vec{a} = (x, y)$ un ángulo de 90° a la izquierda.
- Si aplicamos la fórmula de rotación, el nuevo vector tendrá coordenadas $(-y, x)$.
- Este vector se llama el **vector perpendicular** a \vec{a} , y usualmente se denota como \vec{a}^\perp .



Contenido

1 Conceptos básicos

- Definición de vector
- Posición vs desplazamiento
- Operaciones básicas
- Magnitud y vector unitario
- Rotación de vectores y puntos
- **Representación de un punto en C++**

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos



Representación de un punto en C++

Veamos cómo implementar las operaciones básicas que hemos visto hasta ahora en una estructura de datos de C++. Declararemos una estructura point con dos atributos x y y para las coordenadas, así como con métodos y sobrecarga de operadores para cada operación:



```

struct point{
    // Coordenadas del punto (x,y)
    double x, y;

    // Constructores
    point(): x(0), y(0) {}
    point(double x, double y): x(x), y(y) {}

    // Operaciones básicas
    point operator+(const point & p)const{return point(x + p.x, y + p.y);}
    point operator-(const point & p)const{return point(x - p.x, y - p.y);}
    point operator*(const double & k)const{return point(x * k, y * k);}
    point operator/(const double & k)const{return point(x / k, y / k);}

    // Propiedades del punto
    double length()const{return sqrt(x*x + y*y);}
    point unit()const{return (*this) / length();}
    point perp()const{return point(-y, x);}
    point rotate(const double & ang) const{
        return point(x*cos(ang) - y*sin(ang), x*sin(ang) + y*cos(ang));
    }
};


```

Detalles de la implementación

- En la mayoría de los casos, en los problemas geométricos los puntos en la entrada son todos con componentes enteras. ¿Qué operaciones de las anteriores siempre devuelven otro punto con coordenadas enteras, y cuáles no necesariamente?



Detalles de la implementación

- En la mayoría de los casos, en los problemas geométricos los puntos en la entrada son todos con componentes enteras. ¿Qué operaciones de las anteriores vistas siempre devuelven otro punto con coordenadas enteras, y cuáles no necesariamente?
- Para comparar dos puntos necesitamos que sus coordenadas sean iguales:

```
bool operator==(const point & p) const{return x == p.x && y == p.y;}  
bool operator!=(const point & p) const{return !(*this == p);}
```



Detalles de la implementación

- En la mayoría de los casos, en los problemas geométricos los puntos en la entrada son todos con componentes enteras. ¿Qué operaciones de las anteriores siempre devuelven otro punto con coordenadas enteras, y cuáles no necesariamente?
- Para comparar dos puntos necesitamos que sus coordenadas sean iguales:

```
bool operator==(const point & p) const{return x == p.x && y == p.y;}  
bool operator!=(const point & p) const{return !(*this == p);}
```

¿En qué casos podemos usar el operador == directamente y en cuáles no?



Detalles de la implementación

- Si ya tenemos los operadores de igualdad, convendría tener también los de comparación. ¿Cómo podemos decir que un punto \vec{a} es “mayor” o “menor” a un punto \vec{b} ?



Detalles de la implementación

- Si ya tenemos los operadores de igualdad, convendría tener también los de comparación. ¿Cómo podemos decir que un punto \vec{a} es “mayor” o “menor” a un punto \vec{b} ? Una solución es usar orden lexicográfico, y resulta que es la que más conviene en la mayoría de los problemas:

```
bool operator<(const point & p) const{
    if(x == p.x) return y < p.y;
    return x < p.x;
}
bool operator>(const point & p) const{
    if(x == p.x) return y > p.y;
    return x > p.x;
}
```



Detalles de la implementación

- Si ya tenemos los operadores de igualdad, convendría tener también los de comparación. ¿Cómo podemos decir que un punto \vec{a} es “mayor” o “menor” a un punto \vec{b} ? Una solución es usar orden lexicográfico, y resulta que es la que más conviene en la mayoría de los problemas:

```
bool operator<(const point & p) const{
    if(x == p.x) return y < p.y;
    return x < p.x;
}
bool operator>(const point & p) const{
    if(x == p.x) return y > p.y;
    return x > p.x;
}
```

Y de nuevo surge la pregunta, ¿en qué casos podemos usar los operadores $<$ y $>$ directamente y en cuáles no?



Detalles de la implementación

- Por último, también es útil tener sobrecargados los operadores `>>` y `<<`, para leer un punto de la entrada e imprimirlo a la salida, respectivamente:

```
istream &operator>>(istream &is, point & p)
    return is >> p.x >> p.y
}

ostream &operator<<(ostream &os, const point & p){
    return os << "(" << p.x << ", " << p.y << ")"
}
```



Ejemplo de uso

Veamos cómo quedaría el método `main` de un programa que lee dos puntos P y Q y un ángulo θ de la entrada, e imprime como salida el punto que resulta de rotar el punto Q alrededor de P con un ángulo de θ en sentido antihorario:



Ejemplo de uso

Veamos cómo quedaría el método `main` de un programa que lee dos puntos P y Q y un ángulo θ de la entrada, e imprime como salida el punto que resulta de rotar el punto Q alrededor de P con un ángulo de θ en sentido antihorario:

```
int main(){
    point p, q;
    double theta;
    cin >> p >> q >> theta;
    point q_rotated = p + (q - p).rotate(theta);
    cout << q_rotated << "\n";
    return 0;
}
```



Contenido

1 Conceptos básicos

2 Producto punto

- Definición
- Implementación
- Propiedades
- Ángulo entre vectores
- Proyección de un vector sobre otro
- Implementación parte 2

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos



Contenido

1 Conceptos básicos

2 Producto punto

- Definición
- Implementación
- Propiedades
- Ángulo entre vectores
- Proyección de un vector sobre otro
- Implementación parte 2

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos



Definición de producto punto

Además de la multiplicación por un escalar, existen otras dos operaciones básicas entre vectores que también tienen que ver con “productos”. De hecho, estos productos son igual o incluso más importantes que las operaciones que ya vimos.



Definición de producto punto

Además de la multiplicación por un escalar, existen otras dos operaciones básicas entre vectores que también tienen que ver con “productos”. De hecho, estos productos son igual o incluso más importantes que las operaciones que ya vimos.

Definición

Sean $\vec{a} = (a_x, a_y)$ y $\vec{b} = (b_x, b_y)$ dos vectores. Definimos al **producto punto** o **producto escalar** de \vec{a} con \vec{b} como:

$$\vec{a} \cdot \vec{b} = a_x b_x + a_y b_y$$



Definición de producto punto

Además de la multiplicación por un escalar, existen otras dos operaciones básicas entre vectores que también tienen que ver con “productos”. De hecho, estos productos son igual o incluso más importantes que las operaciones que ya vimos.

Definición

Sean $\vec{a} = (a_x, a_y)$ y $\vec{b} = (b_x, b_y)$ dos vectores. Definimos al **producto punto o producto escalar** de \vec{a} con \vec{b} como:

$$\vec{a} \cdot \vec{b} = a_x b_x + a_y b_y$$

Es decir, sumamos los productos de las coordenadas correspondientes entre los dos vectores. Hay que notar que **el resultado es un escalar, no otro vector**.



Definición de producto punto

Además de la multiplicación por un escalar, existen otras dos operaciones básicas entre vectores que también tienen que ver con “productos”. De hecho, estos productos son igual o incluso más importantes que las operaciones que ya vimos.

Definición

Sean $\vec{a} = (a_x, a_y)$ y $\vec{b} = (b_x, b_y)$ dos vectores. Definimos al **producto punto o producto escalar** de \vec{a} con \vec{b} como:

$$\vec{a} \cdot \vec{b} = a_x b_x + a_y b_y$$

Es decir, sumamos los productos de las coordenadas correspondientes entre los dos vectores. Hay que notar que **el resultado es un escalar, no otro vector**.

Pero, ¿para qué rayos me servirá tener un producto definido así? Ya lo veremos, pero primero veamos la implementación.



Contenido

1 Conceptos básicos

2 Producto punto

- Definición
- **Implementación**
- Propiedades
- Ángulo entre vectores
- Proyección de un vector sobre otro
- Implementación parte 2

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos



Implementación del producto punto

Vamos a agregar a nuestra estructura point un método para que calcule el producto punto con otro vector:

```
double dot(const point & p) const{return x * p.x + y * p.y;}
```



Implementación del producto punto

Vamos a agregar a nuestra estructura point un método para que calcule el producto punto con otro vector:

```
double dot(const point & p) const{return x * p.x + y * p.y;}
```

Preguntas:

- Si los dos vectores tienen coordenadas enteras, ¿su producto punto también lo será?



Implementación del producto punto

Vamos a agregar a nuestra estructura point un método para que calcule el producto punto con otro vector:

```
double dot(const point & p) const{return x * p.x + y * p.y;}
```

Preguntas:

- Si los dos vectores tienen coordenadas enteras, ¿su producto punto también lo será?
- Si el valor absoluto de las coordenadas de los vectores es menor o igual a M , ¿podemos poner también un límite en el producto punto entre ellos? ¿Esto qué consecuencias tiene al escoger el tipo de dato para las coordenadas?



Implementación del producto punto

Vamos a agregar a nuestra estructura point un método para que calcule el producto punto con otro vector:

```
double dot(const point & p) const{return x * p.x + y * p.y;}
```

Preguntas:

- Si los dos vectores tienen coordenadas enteras, ¿su producto punto también lo será?
- Si el valor absoluto de las coordenadas de los vectores es menor o igual a M , ¿podemos poner también un límite en el producto punto entre ellos? ¿Esto qué consecuencias tiene al escoger el tipo de dato para las coordenadas?

Ejemplo:

```
point a(2, 3), b(-1, 4);
cout << a.dot(b) << "\n"; // Debería imprimir 10
```

Contenido

1 Conceptos básicos

2 Producto punto

- Definición
- Implementación
- Propiedades
- Ángulo entre vectores
- Proyección de un vector sobre otro
- Implementación parte 2

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos



Propiedades del producto punto

El producto punto cumple con las siguientes propiedades:

- **Comutativa:** $\vec{a} \cdot \vec{b} = \vec{b} \cdot \vec{a}$



Propiedades del producto punto

El producto punto cumple con las siguientes propiedades:

- **Comutativa:** $\vec{a} \cdot \vec{b} = \vec{b} \cdot \vec{a}$
- **Distributiva:** $(s\vec{a} + t\vec{b}) \cdot \vec{c} = s(\vec{a} \cdot \vec{c}) + t(\vec{b} \cdot \vec{c})$



Propiedades del producto punto

El producto punto cumple con las siguientes propiedades:

- **Comutativa:** $\vec{a} \cdot \vec{b} = \vec{b} \cdot \vec{a}$
- **Distributiva:** $(s\vec{a} + t\vec{b}) \cdot \vec{c} = s(\vec{a} \cdot \vec{c}) + t(\vec{b} \cdot \vec{c})$
- $\vec{a} \cdot \vec{a} \geq 0$. Además $\vec{a} \cdot \vec{a} = 0$ si y solo si $\vec{a} = \vec{0}$.



Propiedades del producto punto

El producto punto cumple con las siguientes propiedades:

- **Comutativa:** $\vec{a} \cdot \vec{b} = \vec{b} \cdot \vec{a}$
- **Distributiva:** $(s\vec{a} + t\vec{b}) \cdot \vec{c} = s(\vec{a} \cdot \vec{c}) + t(\vec{b} \cdot \vec{c})$
- $\vec{a} \cdot \vec{a} \geq 0$. Además $\vec{a} \cdot \vec{a} = 0$ si y solo si $\vec{a} = \vec{0}$.
- $\vec{0} \cdot \vec{a} = 0$



Propiedades del producto punto

El producto punto cumple con las siguientes propiedades:

- **Comutativa:** $\vec{a} \cdot \vec{b} = \vec{b} \cdot \vec{a}$
- **Distributiva:** $(s\vec{a} + t\vec{b}) \cdot \vec{c} = s(\vec{a} \cdot \vec{c}) + t(\vec{b} \cdot \vec{c})$
- $\vec{a} \cdot \vec{a} \geq 0$. Además $\vec{a} \cdot \vec{a} = 0$ si y solo si $\vec{a} = \vec{0}$.
- $\vec{0} \cdot \vec{a} = 0$
- $\|\vec{a}\| = \sqrt{\vec{a} \cdot \vec{a}}$. Esta propiedad es muy útil y nos da un primer acercamiento a alguna interpretación geométrica del producto punto. Es muy común que en los problemas se use la propiedad equivalente: $\vec{a} \cdot \vec{a} = \|\vec{a}\|^2$.



Contenido

1 Conceptos básicos

2 Producto punto

- Definición
- Implementación
- Propiedades
- **Ángulo entre vectores**
- Proyección de un vector sobre otro
- Implementación parte 2

3 Producto cruz

4 Líneas

5 Segmentos

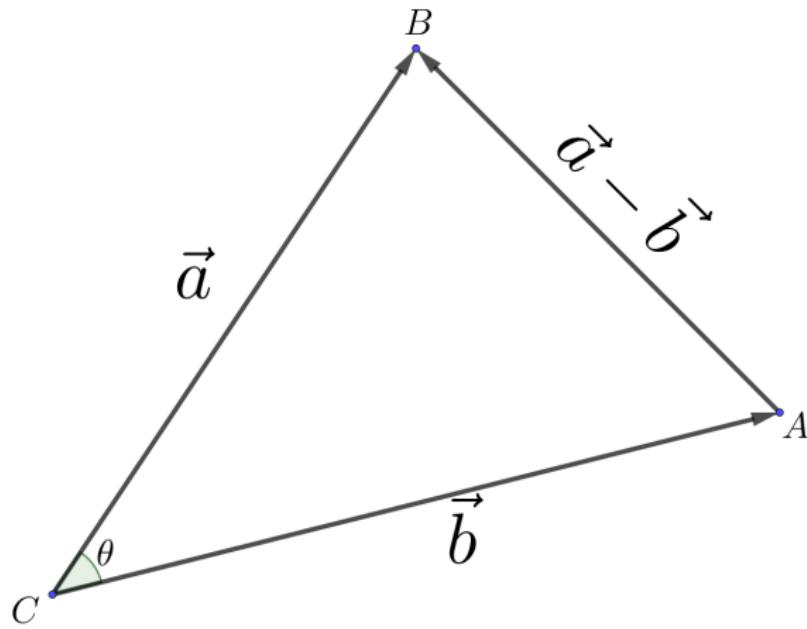
6 Polígonos

7 Círculos



Ángulo entre vectores

Sean \vec{a} y \vec{b} dos vectores. Consideremos el siguiente triángulo $\triangle ABC$ formado por los vectores \vec{a} , \vec{b} y $\vec{a} - \vec{b}$, donde θ es el ángulo entre los vectores \vec{a} y \vec{b} :



Ángulo entre vectores

Al aplicar ley de cosenos tenemos que:

$$\|\vec{a} - \vec{b}\|^2 = \|\vec{a}\|^2 + \|\vec{b}\|^2 - 2 \|\vec{a}\| \|\vec{b}\| \cos \theta$$

$$(\vec{a} - \vec{b}) \cdot (\vec{a} - \vec{b}) = \vec{a} \cdot \vec{a} + \vec{b} \cdot \vec{b} - 2 \|\vec{a}\| \|\vec{b}\| \cos \theta$$

$$\cancel{\vec{a} \cdot \vec{a}} - 2\vec{a} \cdot \vec{b} + \cancel{\vec{b} \cdot \vec{b}} = \cancel{\vec{a} \cdot \vec{a}} + \cancel{\vec{b} \cdot \vec{b}} - 2 \|\vec{a}\| \|\vec{b}\| \cos \theta$$

$$\cancel{-2\vec{a} \cdot \vec{b}} = \cancel{-2} \|\vec{a}\| \|\vec{b}\| \cos \theta$$

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos \theta$$



Ángulo entre vectores

Al aplicar ley de cosenos tenemos que:

$$\|\vec{a} - \vec{b}\|^2 = \|\vec{a}\|^2 + \|\vec{b}\|^2 - 2 \|\vec{a}\| \|\vec{b}\| \cos \theta$$

$$(\vec{a} - \vec{b}) \cdot (\vec{a} - \vec{b}) = \vec{a} \cdot \vec{a} + \vec{b} \cdot \vec{b} - 2 \|\vec{a}\| \|\vec{b}\| \cos \theta$$

$$\cancel{\vec{a} \cdot \vec{a}} - 2\vec{a} \cdot \vec{b} + \cancel{\vec{b} \cdot \vec{b}} = \cancel{\vec{a} \cdot \vec{a}} + \cancel{\vec{b} \cdot \vec{b}} - 2 \|\vec{a}\| \|\vec{b}\| \cos \theta$$

$$\cancel{-2\vec{a} \cdot \vec{b}} = \cancel{-2} \|\vec{a}\| \|\vec{b}\| \cos \theta$$

$$\vec{a} \cdot \vec{b} = \|\vec{a}\| \|\vec{b}\| \cos \theta$$

La relación que obtuvimos ahora sí nos dice la verdadera interpretación geométrica del producto punto, pues relaciona sus longitudes y el ángulo entre ellos.



Ángulo entre vectores

- Podemos despejar el coseno del ángulo entre los vectores:

$$\cos \theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}. \text{ Y como el rango de la función } \arccos(x) \text{ es } [0, \pi],$$

usando esta fórmula siempre obtendremos el ángulo correcto (hay que notar que por lo general siempre hay dos ángulos entre cualquier par de vectores, pero siempre nos quedamos con el menor, que es menor o igual a 180°).



Ángulo entre vectores

- Podemos despejar el coseno del ángulo entre los vectores:

$$\cos \theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}. \text{ Y como el rango de la función } \arccos(x) \text{ es } [0, \pi],$$

usando esta fórmula siempre obtendremos el ángulo correcto (hay que notar que por lo general siempre hay dos ángulos entre cualquier par de vectores, pero siempre nos quedamos con el menor, que es menor o igual a 180°).

- Un caso especial es cuando $\theta = 90^\circ$, es decir, los dos vectores forman un ángulo recto entre ellos. En este caso $\vec{a} \cdot \vec{b} = 0$, y decimos que \vec{a} y \vec{b} son **ortogonales**.



Ángulo entre vectores

- Podemos despejar el coseno del ángulo entre los vectores:

$$\cos \theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \|\vec{b}\|}. \text{ Y como el rango de la función } \arccos(x) \text{ es } [0, \pi],$$

usando esta fórmula siempre obtendremos el ángulo correcto (hay que notar que por lo general siempre hay dos ángulos entre cualquier par de vectores, pero siempre nos quedamos con el menor, que es menor o igual a 180°).

- Un caso especial es cuando $\theta = 90^\circ$, es decir, los dos vectores forman un ángulo recto entre ellos. En este caso $\vec{a} \cdot \vec{b} = 0$, y decimos que \vec{a} y \vec{b} son **ortogonales**.
- Es por eso que el producto punto mide qué tan “paralelos” son dos vectores.



Contenido

1 Conceptos básicos

2 Producto punto

- Definición
- Implementación
- Propiedades
- Ángulo entre vectores
- **Proyección de un vector sobre otro**
- Implementación parte 2

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

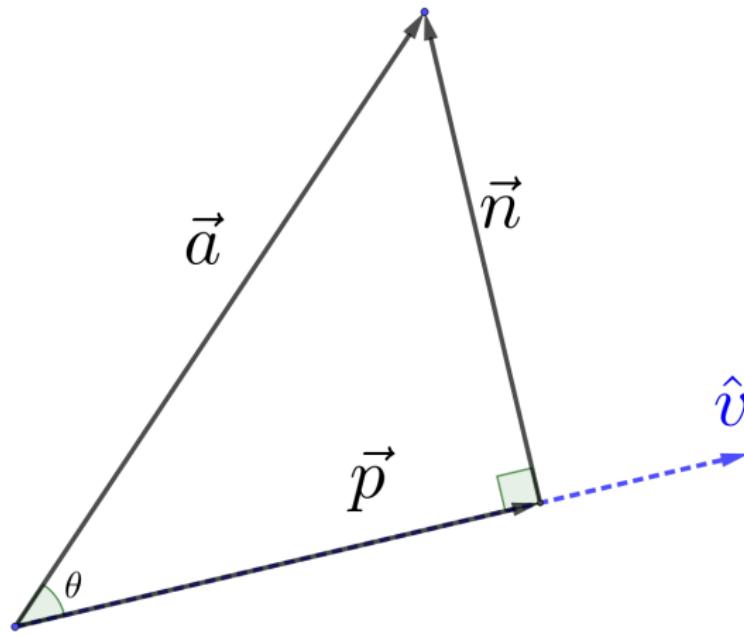
7 Círculos



Proyección de un vector sobre otro

Veamos otra aplicación del producto punto.

- Sea \vec{a} un vector y \hat{v} un vector unitario. Queremos hallar la proyección \vec{p} de \vec{a} sobre la línea que genera la dirección de \hat{v} .



Proyección de un vector sobre otro

- Primero vamos a hallar la longitud de \vec{p} . Por definición de coseno en el triángulo rectángulo que se forma, tenemos que $\|\vec{p}\| = \|\vec{a}\| \cos \theta$.



Proyección de un vector sobre otro

- Primero vamos a hallar la longitud de \vec{p} . Por definición de coseno en el triángulo rectángulo que se forma, tenemos que $\|\vec{p}\| = \|\vec{a}\| \cos \theta$.
- Pero por la propiedad del producto punto, sabemos que $\cos \theta = \frac{\vec{a} \cdot \hat{v}}{\|\vec{a}\| \|\hat{v}\|}$.



Proyección de un vector sobre otro

- Primero vamos a hallar la longitud de \vec{p} . Por definición de coseno en el triángulo rectángulo que se forma, tenemos que $\|\vec{p}\| = \|\vec{a}\| \cos \theta$.
- Pero por la propiedad del producto punto, sabemos que
$$\cos \theta = \frac{\vec{a} \cdot \hat{v}}{\|\vec{a}\| \|\hat{v}\|}.$$
- Combinando, tenemos que $\|\vec{p}\| = \vec{a} \cdot \hat{v}$.



Proyección de un vector sobre otro

- Primero vamos a hallar la longitud de \vec{p} . Por definición de coseno en el triángulo rectángulo que se forma, tenemos que $\|\vec{p}\| = \|\vec{a}\| \cos \theta$.
- Pero por la propiedad del producto punto, sabemos que
$$\cos \theta = \frac{\vec{a} \cdot \hat{v}}{\|\vec{a}\| \|\hat{v}\|}.$$
- Combinando, tenemos que $\|\vec{p}\| = \vec{a} \cdot \hat{v}$.
- Por último, de la figura vemos que la dirección de \vec{p} es la misma que la de \hat{v} , así que podemos determinar completamente el vector \vec{p} :
$$\vec{p} = \|\vec{p}\| \hat{v} = (\vec{a} \cdot \hat{v}) \hat{v}.$$



Proyección de un vector sobre otro

- Primero vamos a hallar la longitud de \vec{p} . Por definición de coseno en el triángulo rectángulo que se forma, tenemos que $\|\vec{p}\| = \|\vec{a}\| \cos \theta$.
- Pero por la propiedad del producto punto, sabemos que $\cos \theta = \frac{\vec{a} \cdot \hat{v}}{\|\vec{a}\| \|\hat{v}\|}$.
- Combinando, tenemos que $\|\vec{p}\| = \vec{a} \cdot \hat{v}$.
- Por último, de la figura vemos que la dirección de \vec{p} es la misma que la de \hat{v} , así que podemos determinar completamente el vector \vec{p} : $\vec{p} = \|\vec{p}\| \hat{v} = (\vec{a} \cdot \hat{v}) \hat{v}$.

Definición

Sean \vec{a} y \hat{v} vectores. Definimos a la **proyección de \vec{a} sobre \hat{v}** como $\text{proj}_{\hat{v}}(\vec{a}) = (\vec{a} \cdot \hat{v}) \hat{v}$.



Proyección de un vector sobre otro

- Primero vamos a hallar la longitud de \vec{p} . Por definición de coseno en el triángulo rectángulo que se forma, tenemos que $\|\vec{p}\| = \|\vec{a}\| \cos \theta$.
- Pero por la propiedad del producto punto, sabemos que $\cos \theta = \frac{\vec{a} \cdot \hat{v}}{\|\vec{a}\| \|\hat{v}\|}$.
- Combinando, tenemos que $\|\vec{p}\| = \vec{a} \cdot \hat{v}$.
- Por último, de la figura vemos que la dirección de \vec{p} es la misma que la de \hat{v} , así que podemos determinar completamente el vector \vec{p} : $\vec{p} = \|\vec{p}\| \hat{v} = (\vec{a} \cdot \hat{v}) \hat{v}$.

Definición

Sean \vec{a} y \hat{v} vectores. Definimos a la **proyección de \vec{a} sobre \hat{v}** como $\text{proy}_{\hat{v}}(\vec{a}) = (\vec{a} \cdot \hat{v}) \hat{v}$.

Pregunta: ¿cómo sería la fórmula si \vec{v} no fuera unitario?



Contenido

1 Conceptos básicos

2 Producto punto

- Definición
- Implementación
- Propiedades
- Ángulo entre vectores
- Proyección de un vector sobre otro
- Implementación parte 2

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos



Implementación parte 2

```
double angleBetween(point a, point b){  
    double x = a.dot(b) / a.length() / b.length();  
    return acos(max(-1.0, min(1.0, x)));  
}
```



Implementación parte 2

```
double angleBetween(point a, point b){  
    double x = a.dot(b) / a.length() / b.length();  
    return acos(max(-1.0, min(1.0, x)));  
}
```

```
bool isPerp(point a, point b){  
    return a.dot(b) == 0;  
}
```



Implementación parte 2

```
double angleBetween(point a, point b){  
    double x = a.dot(b) / a.length() / b.length();  
    return acos(max(-1.0, min(1.0, x)));  
}
```

```
bool isPerp(point a, point b){  
    return a.dot(b) == 0;  
}
```

```
point proj(point a, point v){  
    v = v / v.unit();  
    return v * a.dot(v);  
}
```



Contenido

1 Conceptos básicos

- Áreas
- Resumen

2 Producto punto

4 Líneas

3 Producto cruz

- Definición
- Implementación
- Propiedades
- Orientación

5 Segmentos

6 Polígonos

7 Círculos



Contenido

1 Conceptos básicos

- Áreas
- Resumen

2 Producto punto

4 Líneas

3 Producto cruz

- Definición
- Implementación
- Propiedades
- Orientación

5 Segmentos

6 Polígonos

7 Círculos



Definición de producto cruz

El otro “producto” entre vectores es el producto cruz. Tiene muchas propiedades interesantes además de las que nos da el producto punto, pero primero hay que definirlo:

Definición

Sean $\vec{a} = (a_x, a_y)$ y $\vec{b} = (b_x, b_y)$ dos vectores. Definimos al **producto cruz** de \vec{a} con \vec{b} como:

$$\vec{a} \times \vec{b} = a_x b_y - a_y b_x$$



Definición de producto cruz

El otro “producto” entre vectores es el producto cruz. Tiene muchas propiedades interesantes además de las que nos da el producto punto, pero primero hay que definirlo:

Definición

Sean $\vec{a} = (a_x, a_y)$ y $\vec{b} = (b_x, b_y)$ dos vectores. Definimos al **producto cruz** de \vec{a} con \vec{b} como:

$$\vec{a} \times \vec{b} = a_x b_y - a_y b_x$$

De forma estricta, este producto sí es un vector, pero al trabajar en 2D y para lo que vamos a ver después, nos conviene que también sea un escalar.



Contenido

1 Conceptos básicos

- Áreas
- Resumen

2 Producto punto

4 Líneas

3 Producto cruz

- Definición
- **Implementación**
- Propiedades
- Orientación

5 Segmentos

6 Polígonos

7 Círculos



Implementación del producto cruz

Ahora vamos a agregar a nuestra estructura point un método para que calcule el producto cruz con otro vector:

```
double cross(const point & p) const{return x * p.y - y * p.x;}
```



Implementación del producto cruz

Ahora vamos a agregar a nuestra estructura point un método para que calcule el producto cruz con otro vector:

```
double cross(const point & p) const{return x * p.y - y * p.x;}
```

Ejemplo:

```
point a(-5, 2), b(3, 1);
cout << a.cross(b) << "\n"; // Debería imprimir -11
```



Contenido

1 Conceptos básicos

- Áreas
- Resumen

2 Producto punto

4 Líneas

3 Producto cruz

5 Segmentos

- Definición

- Implementación

• Propiedades

- Orientación

6 Polígonos

7 Círculos



Propiedades

El producto cruz cumple con las siguientes propiedades:

- **Anticonmutativa:** $\vec{a} \times \vec{b} = -\vec{b} \times \vec{a}$, mucho cuidado con esta propiedad.



Propiedades

El producto cruz cumple con las siguientes propiedades:

- **Anticonmutativa:** $\vec{a} \times \vec{b} = -\vec{b} \times \vec{a}$, mucho cuidado con esta propiedad.
- **Distributiva por la izquierda:** $\vec{a} \times (\vec{b} + \vec{c}) = \vec{a} \times \vec{b} + \vec{a} \times \vec{c}$.

Propiedades

El producto cruz cumple con las siguientes propiedades:

- **Anticonmutativa:** $\vec{a} \times \vec{b} = -\vec{b} \times \vec{a}$, mucho cuidado con esta propiedad.
- **Distributiva por la izquierda:** $\vec{a} \times (\vec{b} + \vec{c}) = \vec{a} \times \vec{b} + \vec{a} \times \vec{c}$.
- **Distributiva por la derecha:** $(\vec{a} + \vec{b}) \times \vec{c} = \vec{a} \times \vec{c} + \vec{b} \times \vec{c}$.



Propiedades

El producto cruz cumple con las siguientes propiedades:

- **Anticonmutativa:** $\vec{a} \times \vec{b} = -\vec{b} \times \vec{a}$, mucho cuidado con esta propiedad.
- **Distributiva por la izquierda:** $\vec{a} \times (\vec{b} + \vec{c}) = \vec{a} \times \vec{b} + \vec{a} \times \vec{c}$.
- **Distributiva por la derecha:** $(\vec{a} + \vec{b}) \times \vec{c} = \vec{a} \times \vec{c} + \vec{b} \times \vec{c}$.
- $\vec{a} \times \vec{a} = \vec{0} \times \vec{a} = \vec{0}$.



Contenido

1 Conceptos básicos

- Áreas
- Resumen

2 Producto punto

4 Líneas

3 Producto cruz

5 Segmentos

- Definición
- Implementación
- Propiedades
- Orientación

6 Polígonos

7 Círculos



Orientación

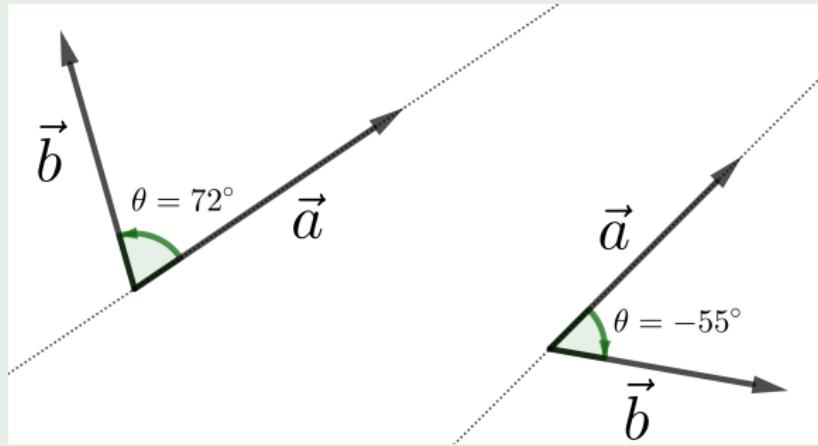
- Sean \vec{a} y \vec{b} dos vectores con un **ángulo dirigido** entre ellos igual a θ . Este ángulo es casi el mismo que definimos en el producto punto, con la diferencia que será positivo si para ir de \vec{a} a \vec{b} tuvimos que girar en sentido antihorario, y será negativo si tuvimos que girar en sentido horario. Hay que darse cuenta que el orden en que definamos \vec{a} y \vec{b} es importante.



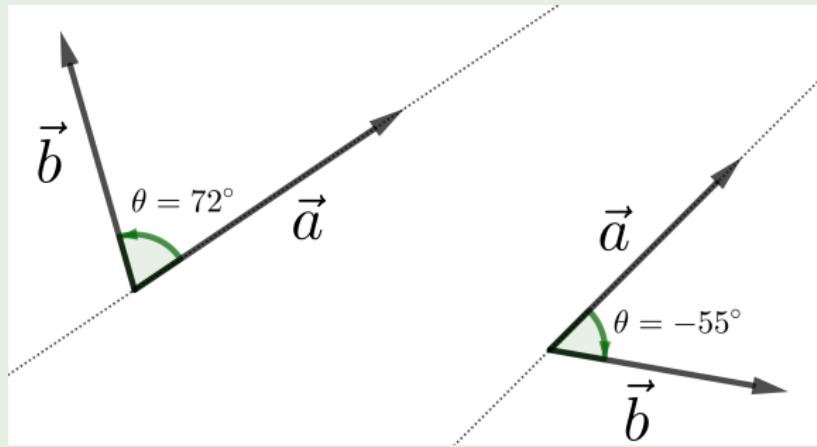
- Sean \vec{a} y \vec{b} dos vectores con un **ángulo dirigido** entre ellos igual a θ . Este ángulo es casi el mismo que definimos en el producto punto, con la diferencia que será positivo si para ir de \vec{a} a \vec{b} tuvimos que girar en sentido antihorario, y será negativo si tuvimos que girar en sentido horario. Hay que darse cuenta que el orden en que definamos \vec{a} y \vec{b} es importante.
- Lo anterior es equivalente a decir que θ será positivo si \vec{b} está a la izquierda de \vec{a} , y θ será negativo si \vec{b} está a la derecha de \vec{a} .
- A esta regla se le conoce como la **regla de la mano derecha**.



Ejemplo

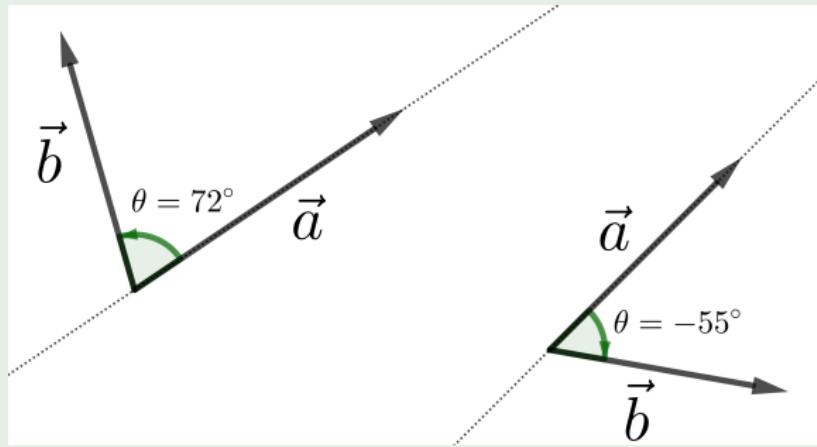


Ejemplo



- En el primer caso, para ir de \vec{a} a \vec{b} tuvimos que girar en sentido antihorario un ángulo de 72° , por eso le toca un signo positivo.

Ejemplo



- En el primer caso, para ir de \vec{a} a \vec{b} tuvimos que girar en sentido antihorario un ángulo de 72° , por eso le toca un signo positivo.
- En el segundo caso, para ir de \vec{a} a \vec{b} tuvimos que girar en sentido horario un ángulo de 55° , por eso le toca un signo negativo.

Orientación

Con lo anterior, se cumple la siguiente propiedad del producto cruz:

$$\vec{a} \times \vec{b} = \|\vec{a}\| \|\vec{b}\| \sin \theta$$



Orientación

Con lo anterior, se cumple la siguiente propiedad del producto cruz:

$$\vec{a} \times \vec{b} = \|\vec{a}\| \|\vec{b}\| \sin \theta$$

Demostración.

$$\begin{aligned}(\vec{a} \times \vec{b})^2 &= (a_x b_y - a_y b_x)^2 \\&= a_x^2 b_y^2 - 2a_x b_x a_y b_y + a_y^2 b_x^2 \\&= \color{blue}{a_x^2 b_x^2} + a_x^2 b_y^2 + a_y^2 b_x^2 + \color{red}{a_y^2 b_y^2} - \color{blue}{a_x^2 b_x^2} - 2a_x b_x a_y b_y - \color{red}{a_y^2 b_y^2} \\&= (a_x^2 + a_y^2)(b_x^2 + b_y^2) - (a_x b_x + a_y b_y)^2 \\&= \|\vec{a}\|^2 \|\vec{b}\|^2 - (\vec{a} \cdot \vec{b})^2 = \|\vec{a}\|^2 \|\vec{b}\|^2 - (\|\vec{a}\| \|\vec{b}\| \cos \theta)^2 \\&= \|\vec{a}\|^2 \|\vec{b}\|^2 (1 - \cos^2 \theta) = \|\vec{a}\|^2 \|\vec{b}\|^2 \sin^2 \theta\end{aligned}$$

Por lo tanto, $\vec{a} \times \vec{b} = \|\vec{a}\| \|\vec{b}\| \sin (\pm \theta)$, donde el signo correcto para θ se escoge con la regla de la mano derecha. □

Orientación

- Del resultado anterior, vemos que el signo de θ siempre es el mismo que el de $\vec{a} \times \vec{b}$ debido a que el seno es una función impar. Entonces, con solo fijarnos en el signo de $\vec{a} \times \vec{b}$ podemos saber la orientación relativa de \vec{b} respecto a \vec{a} .



- Del resultado anterior, vemos que el signo de θ siempre es el mismo que el de $\vec{a} \times \vec{b}$ debido a que el seno es una función impar. Entonces, con solo fijarnos en el signo de $\vec{a} \times \vec{b}$ podemos saber la orientación relativa de \vec{b} respecto a \vec{a} .
- Ahora consideremos el caso en el que tenemos tres puntos A , B y C , y queremos saber si C se encuentra a la izquierda o a la derecha de la línea generada por \overrightarrow{AB} . Basta con hallar el producto $\overrightarrow{AB} \times \overrightarrow{AC} = (\vec{B} - \vec{A}) \times (\vec{C} - \vec{A})$.



- Del resultado anterior, vemos que el signo de θ siempre es el mismo que el de $\vec{a} \times \vec{b}$ debido a que el seno es una función impar. Entonces, con solo fijarnos en el signo de $\vec{a} \times \vec{b}$ podemos saber la orientación relativa de \vec{b} respecto a \vec{a} .
- Ahora consideremos el caso en el que tenemos tres puntos A , B y C , y queremos saber si C se encuentra a la izquierda o a la derecha de la línea generada por \overrightarrow{AB} . Basta con hallar el producto $\overrightarrow{AB} \times \overrightarrow{AC} = (\vec{B} - \vec{A}) \times (\vec{C} - \vec{A})$.
- Lo anterior tiene otra interpretación:
 - El producto será positivo si para ir de A a B y luego a C tuvimos que girar a la izquierda o en sentido antihorario.



- Del resultado anterior, vemos que el signo de θ siempre es el mismo que el de $\vec{a} \times \vec{b}$ debido a que el seno es una función impar. Entonces, con solo fijarnos en el signo de $\vec{a} \times \vec{b}$ podemos saber la orientación relativa de \vec{b} respecto a \vec{a} .
- Ahora consideremos el caso en el que tenemos tres puntos A , B y C , y queremos saber si C se encuentra a la izquierda o a la derecha de la línea generada por \overrightarrow{AB} . Basta con hallar el producto $\overrightarrow{AB} \times \overrightarrow{AC} = (\vec{B} - \vec{A}) \times (\vec{C} - \vec{A})$.
- Lo anterior tiene otra interpretación:
 - El producto será positivo si para ir de A a B y luego a C tuvimos que girar a la izquierda o en sentido antihorario.
 - El producto será negativo si para ir de A a B y luego a C tuvimos que girar a la derecha o en sentido horario.

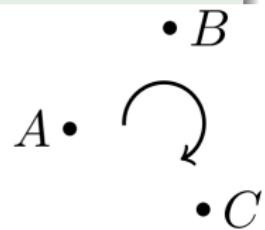
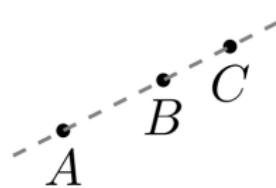
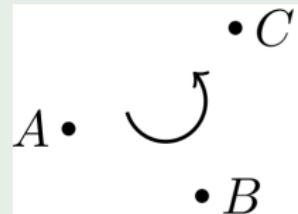


- Del resultado anterior, vemos que el signo de θ siempre es el mismo que el de $\vec{a} \times \vec{b}$ debido a que el seno es una función impar. Entonces, con solo fijarnos en el signo de $\vec{a} \times \vec{b}$ podemos saber la orientación relativa de \vec{b} respecto a \vec{a} .
- Ahora consideraremos el caso en el que tenemos tres puntos A , B y C , y queremos saber si C se encuentra a la izquierda o a la derecha de la línea generada por \overrightarrow{AB} . Basta con hallar el producto $\overrightarrow{AB} \times \overrightarrow{AC} = (\vec{B} - \vec{A}) \times (\vec{C} - \vec{A})$.
- Lo anterior tiene otra interpretación:
 - El producto será positivo si para ir de A a B y luego a C tuvimos que girar a la izquierda o en sentido antihorario.
 - El producto será negativo si para ir de A a B y luego a C tuvimos que girar a la derecha o en sentido horario.
 - El producto será cero si A , B y C son colineales.

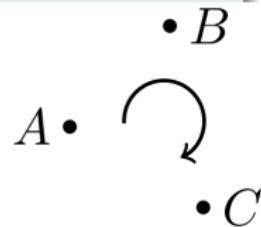
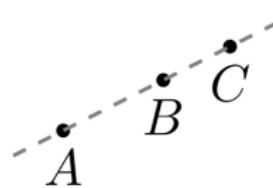
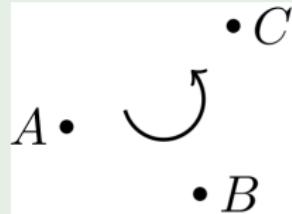


Orientación

Ejemplo

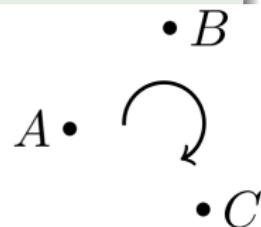
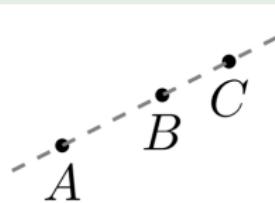
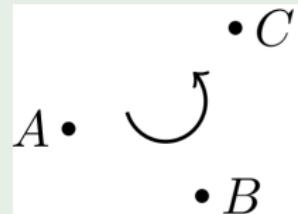


Ejemplo



- En el primer caso hicimos un giro a la izquierda, por lo tanto,
 $\overrightarrow{AB} \times \overrightarrow{AC} > 0$.

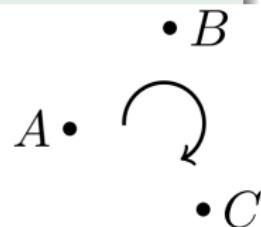
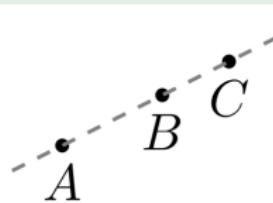
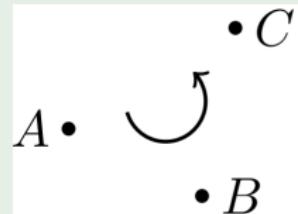
Ejemplo



- En el primer caso hicimos un giro a la izquierda, por lo tanto,
 $\overrightarrow{AB} \times \overrightarrow{AC} > 0$.
- En el segundo caso los tres puntos están alineados, por lo tanto,
 $\overrightarrow{AB} \times \overrightarrow{AC} = 0$.



Ejemplo



- En el primer caso hicimos un giro a la izquierda, por lo tanto,
 $\overrightarrow{AB} \times \overrightarrow{AC} > 0$.
- En el segundo caso los tres puntos están alineados, por lo tanto,
 $\overrightarrow{AB} \times \overrightarrow{AC} = 0$.
- En el tercer caso hicimos un giro a la derecha, por lo tanto,
 $\overrightarrow{AB} \times \overrightarrow{AC} < 0$.



Orientación

Vamos a usar mucho esa función, así que conviene programarla de una vez:

```
double orient(point a, point b, point c){  
    return (b - a).cross(c - a);  
}
```



Orientación

Vamos a usar mucho esa función, así que conviene programarla de una vez:

```
double orient(point a, point b, point c){  
    return (b - a).cross(c - a);  
}
```

- Una propiedad interesante de esa función es que si rotamos cíclicamente los puntos, el valor no cambia, es decir:
 $\text{orient}(A, B, C) = \text{orient}(B, C, A) = \text{orient}(C, A, B)$.



Orientación

Vamos a usar mucho esa función, así que conviene programarla de una vez:

```
double orient(point a, point b, point c){  
    return (b - a).cross(c - a);  
}
```

- Una propiedad interesante de esa función es que si rotamos cíclicamente los puntos, el valor no cambia, es decir:
 $\text{orient}(A, B, C) = \text{orient}(B, C, A) = \text{orient}(C, A, B)$.
- Otra es que si intercambiamos cualesquiera dos puntos, el valor cambia de signo.



Orientación

Vamos a usar mucho esa función, así que conviene programarla de una vez:

```
double orient(point a, point b, point c){  
    return (b - a).cross(c - a);  
}
```

- Una propiedad interesante de esa función es que si rotamos cíclicamente los puntos, el valor no cambia, es decir:
 $\text{orient}(A, B, C) = \text{orient}(B, C, A) = \text{orient}(C, A, B)$.
- Otra es que si intercambiamos cualesquiera dos puntos, el valor cambia de signo.

Ejemplo

Ejercicio: diseña una función que, dado un ángulo un ángulo formado por las líneas \overrightarrow{AB} y \overrightarrow{AC} y un punto P , determine si P se encuentra dentro del ángulo o no.

Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

- Definición
- Implementación
- Propiedades
- Orientación

• Áreas

• Resumen

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos



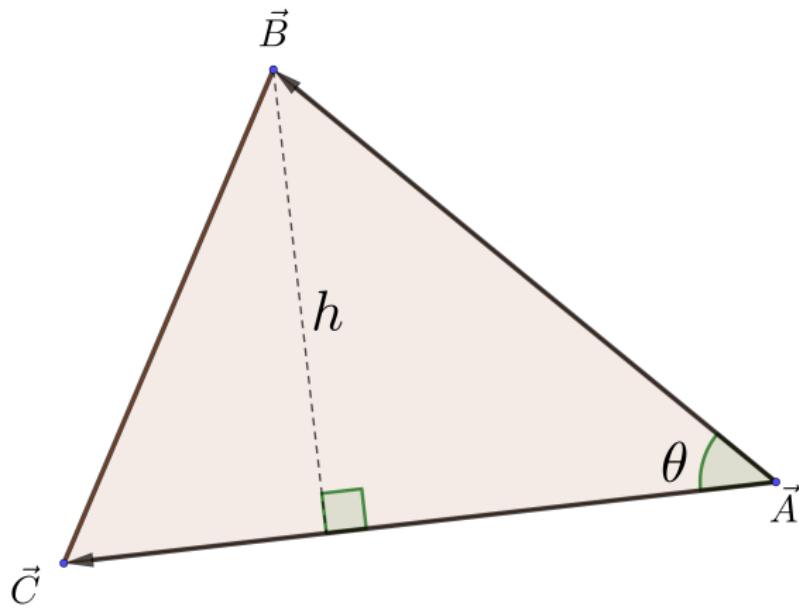
Áreas

Otra aplicación del producto cruz es que nos permite calcular el área de diversas figuras. Comencemos viendo cómo hallar el área de un triángulo dados sus tres vértices A , B y C :



Áreas

Otra aplicación del producto cruz es que nos permite calcular el área de diversas figuras. Comencemos viendo cómo hallar el área de un triángulo dados sus tres vértices A , B y C :



- Primero hallemos la altura del triángulo. Si nos fijamos en el triángulo rectángulo de la derecha, por definición de seno tenemos que $h = \|\overrightarrow{AB}\| \sin \theta$.



Áreas

- Primero hallemos la altura del triángulo. Si nos fijamos en el triángulo rectángulo de la derecha, por definición de seno tenemos que $h = \|\overrightarrow{AB}\| \sin \theta$.
- Ya tenemos la base y la altura, por lo que el área es $[ABC] = \frac{1}{2} \|\overrightarrow{AC}\| h = \|\overrightarrow{AB}\| \|\overrightarrow{AC}\| \sin \theta$.



- Primero hallemos la altura del triángulo. Si nos fijamos en el triángulo rectángulo de la derecha, por definición de seno tenemos que $h = \|\overrightarrow{AB}\| \sin \theta$.
- Ya tenemos la base y la altura, por lo que el área es $[ABC] = \frac{1}{2} \|\overrightarrow{AC}\| h = \|\overrightarrow{AB}\| \|\overrightarrow{AC}\| \sin \theta$.
- Pero por la propiedad del producto cruz, sabemos que $\overrightarrow{AB} \times \overrightarrow{AC} = \|\overrightarrow{AB}\| \|\overrightarrow{AC}\| \sin \theta$, por lo tanto el área es igual a $[ABC] = \frac{1}{2} \overrightarrow{AB} \times \overrightarrow{AC}$.



Áreas

- Primero hallemos la altura del triángulo. Si nos fijamos en el triángulo rectángulo de la derecha, por definición de seno tenemos que

$$h = \|\overrightarrow{AB}\| \sin \theta.$$

- Ya tenemos la base y la altura, por lo que el área es

$$[ABC] = \frac{1}{2} \|\overrightarrow{AC}\| h = \|\overrightarrow{AB}\| \|\overrightarrow{AC}\| \sin \theta.$$

- Pero por la propiedad del producto cruz, sabemos que

$$\overrightarrow{AB} \times \overrightarrow{AC} = \|\overrightarrow{AB}\| \|\overrightarrow{AC}\| \sin \theta, \text{ por lo tanto el área es igual a}$$

$$[ABC] = \frac{1}{2} \overrightarrow{AB} \times \overrightarrow{AC}.$$

- Ok, esa es *casi* el área, pero recordemos que el ángulo θ en el producto cruz es dirigido, pero el que estamos usando aquí es no dirigido, por lo tanto solo tenemos que hallar el valor absoluto de la expresión anterior para garantizar que el área nunca sea negativa:

$$[ABC] = \frac{1}{2} |\overrightarrow{AB} \times \overrightarrow{AC}|.$$



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

- Definición
- Implementación
- Propiedades
- Orientación

● Áreas

● Resumen

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos



Resumen de los productos

- **Producto punto:** $\vec{a} \cdot \vec{b} = a_x b_x + a_y b_y = \|\vec{a}\| \|\vec{b}\| \cos \theta.$
- **Producto cruz:** $\vec{a} \times \vec{b} = a_x b_y - a_y b_x = \|\vec{a}\| \|\vec{b}\| \sin \theta.$



Resumen de los productos

- **Producto punto:** $\vec{a} \cdot \vec{b} = a_x b_x + a_y b_y = \|\vec{a}\| \|\vec{b}\| \cos \theta.$
- **Producto cruz:** $\vec{a} \times \vec{b} = a_x b_y - a_y b_x = \|\vec{a}\| \|\vec{b}\| \sin \theta.$
- El producto punto sirve para obtener el ángulo no dirigido entre dos vectores. No da información sobre la orientación relativa.
- El producto cruz nos da la orientación relativa entre dos vectores, pero no sirve para obtener el ángulo entre dos vectores, porque el rango de la función $\arcsin(x)$ es $[-\frac{\pi}{2}, \frac{\pi}{2}]$.



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

- Definición
- Proyección y reflexión

- Distancia punto-línea
- Ordenar puntos respecto a una línea
- Intersección de líneas

5 Segmentos

6 Polígonos

7 Círculos



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

- Definición
- Proyección y reflexión

- Distancia punto-línea
- Ordenar puntos respecto a una línea
- Intersección de líneas

5 Segmentos

6 Polígonos

7 Círculos



Definición de línea

Definición

Una **Línea** es el conjunto de puntos que se mueven en una dirección determinada y de forma indefinida en sus ambos extremos.



Definición de línea

Definición

Una **Línea** es el conjunto de puntos que se mueven en una dirección determinada y de forma indefinida en sus ambos extremos.

Existen varias formas de representar una línea matemáticamente, las tres más comunes son:

- Punto-pendiente: $y = mx + b$.
- Forma general: $ax + by = c$.
- Forma paramétrica o vectorial: $\vec{r} = \vec{a} + t\vec{v}$, donde $\vec{r} = (x, y)$ es el vector de posición de toda la línea.



Definición de línea

Definición

Una **Línea** es el conjunto de puntos que se mueven en una dirección determinada y de forma indefinida en sus ambos extremos.

Existen varias formas de representar una línea matemáticamente, las tres más comunes son:

- Punto-pendiente: $y = mx + b$.
- Forma general: $ax + by = c$.
- Forma paramétrica o vectorial: $\vec{r} = \vec{a} + t\vec{v}$, donde $\vec{r} = (x, y)$ es el vector de posición de toda la línea.

Por comodidad y por la facilidad de uso que ofrece, usaremos la tercer forma de aquí en adelante.



Definición de línea

- Supongamos que la línea pasa por un punto fijo \vec{a} y un vector paralelo a ella es \vec{v} . Vemos que la tarea de \vec{a} es trasladar la línea y la tarea de \vec{v} es darle la dirección a la línea.



Definición de línea

- Supongamos que la línea pasa por un punto fijo \vec{a} y un vector paralelo a ella es \vec{v} . Vemos que la tarea de \vec{a} es trasladar la línea y la tarea de \vec{v} es darle la dirección a la línea.
- Sea \vec{r} el vector de posición de cada uno de los puntos de la línea, es decir, su flecha barrerá toda la línea.



Definición de línea

- Supongamos que la línea pasa por un punto fijo \vec{a} y un vector paralelo a ella es \vec{v} . Vemos que la tarea de \vec{a} es trasladar la línea y la tarea de \vec{v} es darle la dirección a la línea.
- Sea \vec{r} el vector de posición de cada uno de los puntos de la línea, es decir, su flecha barrerá toda la línea.
- Como solo podemos movernos en la dirección de \vec{v} , cualquier múltiplo escalar de \vec{v} estará sobre la línea.

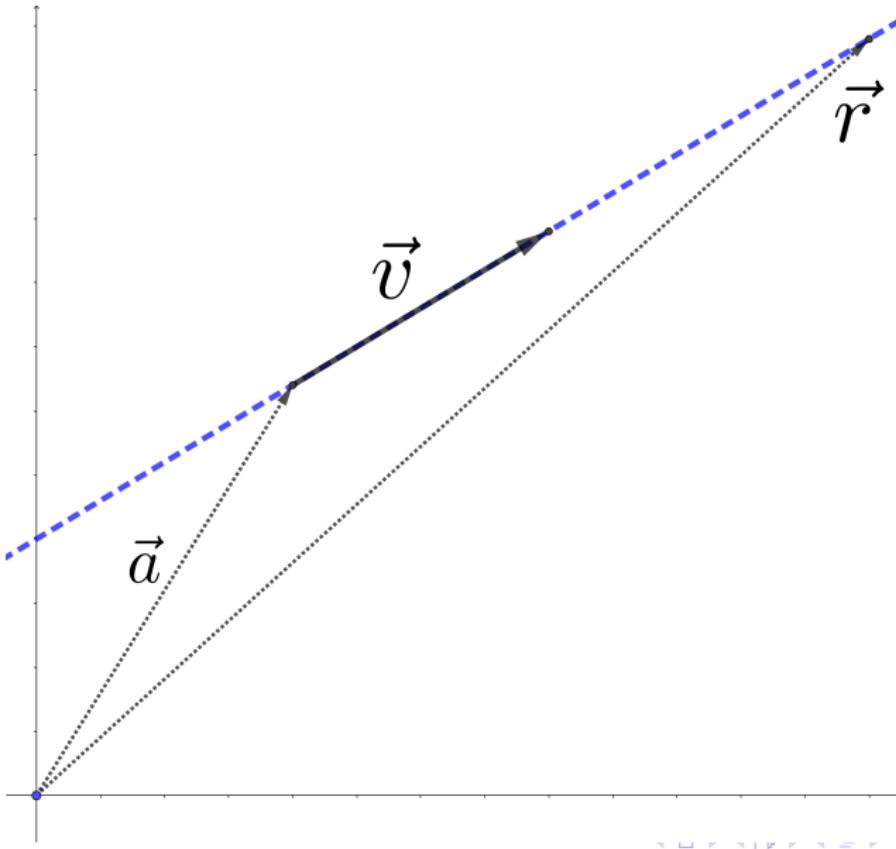


Definición de línea

- Supongamos que la línea pasa por un punto fijo \vec{a} y un vector paralelo a ella es \vec{v} . Vemos que la tarea de \vec{a} es trasladar la línea y la tarea de \vec{v} es darle la dirección a la línea.
- Sea \vec{r} el vector de posición de cada uno de los puntos de la línea, es decir, su flecha barrerá toda la línea.
- Como solo podemos movernos en la dirección de \vec{v} , cualquier múltiplo escalar de \vec{v} estará sobre la línea.
- Entonces, $\vec{r} = \vec{a} + t\vec{v}$, para toda $t \in \mathbb{R}$.



Definición de línea



Definición de línea

- Ahora supongamos que en lugar de un punto y un vector paralelo, nos dan dos puntos distintos A y B y queremos hallar la ecuación de la línea que pasa por ellos.



Definición de línea

- Ahora supongamos que en lugar de un punto y un vector paralelo, nos dan dos puntos distintos A y B y queremos hallar la ecuación de la línea que pasa por ellos.
- Un vector paralelo es simplemente el vector de desplazamiento \overrightarrow{AB} , es decir, $\vec{v} = \vec{B} - \vec{A}$.



Definición de línea

- Ahora supongamos que en lugar de un punto y un vector paralelo, nos dan dos puntos distintos A y B y queremos hallar la ecuación de la línea que pasa por ellos.
- Un vector paralelo es simplemente el vector de desplazamiento \overrightarrow{AB} , es decir, $\vec{v} = \vec{B} - \vec{A}$.
- Podemos tomar tanto a \vec{A} como a \vec{B} como puntos fijos.

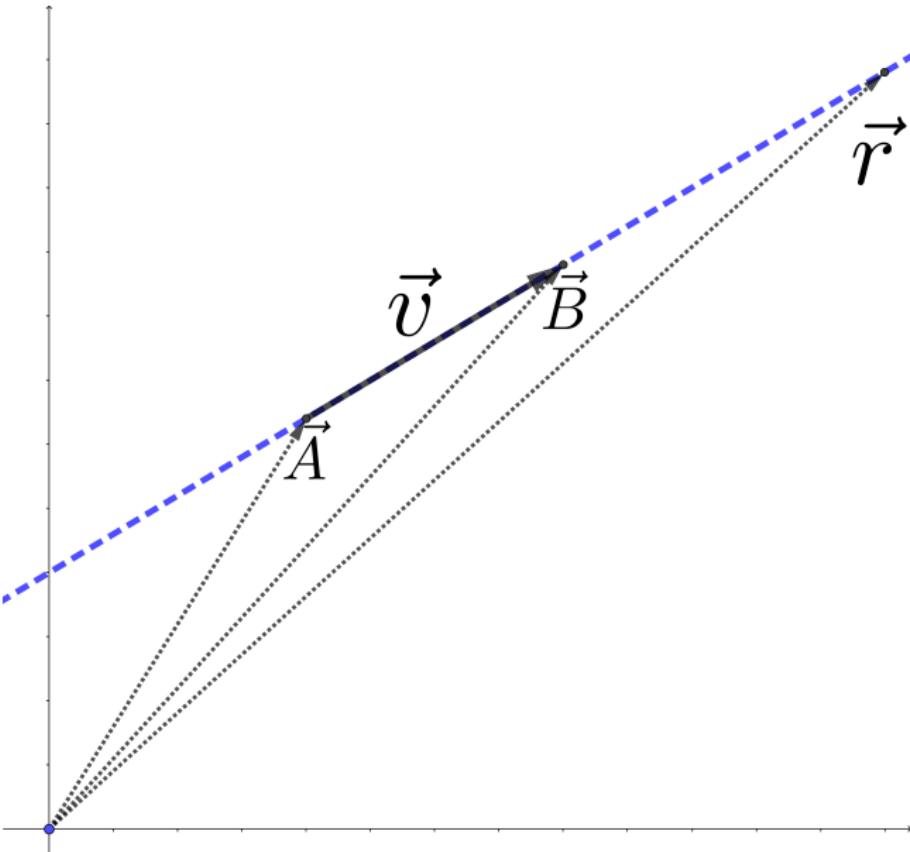


Definición de línea

- Ahora supongamos que en lugar de un punto y un vector paralelo, nos dan dos puntos distintos A y B y queremos hallar la ecuación de la línea que pasa por ellos.
- Un vector paralelo es simplemente el vector de desplazamiento \overrightarrow{AB} , es decir, $\vec{v} = \vec{B} - \vec{A}$.
- Podemos tomar tanto a \vec{A} como a \vec{B} como puntos fijos.
- Por lo tanto, la ecuación de la línea queda como $\vec{r} = \vec{A} + t(\vec{B} - \vec{A})$.



Definición de línea



Definición de línea

- Por último, es fácil ver que una ecuación de la forma $y = mx + b$ es equivalente a $\vec{r} = (0, b) + t(1, m)$.
- De forma similar, la ecuación $ax + by = c$ es equivalente a $\vec{r} = (0, \frac{c}{b}) + t(-b, a)$ si $b \neq 0$.



Contenido

1 Conceptos básicos

- Distancia punto-línea
- Ordenar puntos respecto a una línea
- Intersección de líneas

2 Producto punto

3 Producto cruz

4 Líneas

- Definición
- Proyección y reflexión

5 Segmentos

6 Polígonos

7 Círculos



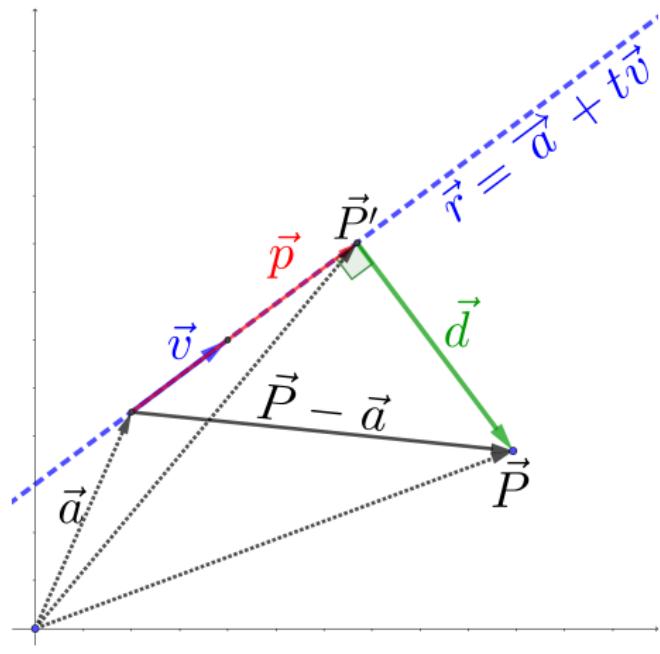
Proyección

Si tenemos un punto P y una línea $\vec{r} = \vec{a} + t\vec{v}$, hallar la proyección de P sobre dicha línea es fácil, pues ya sabemos cómo proyectar un vector sobre otro, solo hay que hacer unos pequeños ajustes:



Proyección

Si tenemos un punto P y una línea $\vec{r} = \vec{a} + t\vec{v}$, hallar la proyección de P sobre dicha línea es fácil, pues ya sabemos cómo proyectar un vector sobre otro, solo hay que hacer unos pequeños ajustes:



Proyección

- Sea P' la proyección de P sobre la línea. Vemos que realmente lo que hay que proyectar es el vector de desplazamiento \overrightarrow{aP} sobre el vector \vec{v} , pero eso es justamente $\vec{p} = \text{proj}_{\vec{v}}(\overrightarrow{aP})$.



Proyección

- Sea P' la proyección de P sobre la línea. Vemos que realmente lo que hay que proyectar es el vector de desplazamiento \overrightarrow{aP} sobre el vector \vec{v} , pero eso es justamente $\vec{p} = \text{proy}_{\hat{v}}(\overrightarrow{aP})$.
- Finalmente, para obtener \vec{P}' solo hay que sumar el punto \vec{a} con el desplazamiento \vec{p} , o sea: $\vec{P}' = \vec{a} + \text{proy}_{\hat{v}}(\overrightarrow{aP})$.



Proyección

- Sea P' la proyección de P sobre la línea. Vemos que realmente lo que hay que proyectar es el vector de desplazamiento \vec{aP} sobre el vector \vec{v} , pero eso es justamente $\vec{p} = \text{proj}_{\hat{v}}(\vec{aP})$.
- Finalmente, para obtener \vec{P}' solo hay que sumar el punto \vec{a} con el desplazamiento \vec{p} , o sea: $\vec{P}' = \vec{a} + \text{proj}_{\hat{v}}(\vec{aP})$.
- Usando la función `proj()` que ya teníamos, programar esto es muy sencillo:

```
point proj_line(point a, point v, point p){  
    return a + proj(p - a, v);  
}
```



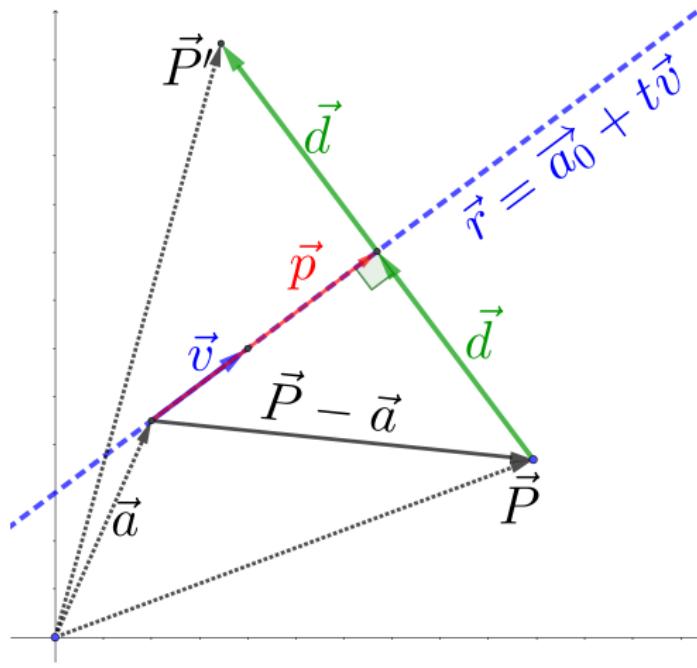
Reflexión

Ahora supongamos que la línea es un espejo y queremos hallar un punto P' tal que esté del otro lado y tenga la misma distancia a la línea que el punto original. Entonces diremos que P' es la reflexión de P sobre esta línea:



Reflexión

Ahora supongamos que la línea es un espejo y queremos hallar un punto P' tal que esté del otro lado y tenga la misma distancia a la línea que el punto original. Entonces diremos que P' es la reflexión de P sobre esta línea:



Reflexión

- Casi es el mismo dibujo que el que usamos en la proyección, solo invertimos el sentido de \vec{d} y lo dibujamos dos veces para ver en dónde quedará \vec{P}' . Del dibujo vemos que $\vec{d} = \vec{p} - \overrightarrow{aP}$.



Reflexión

- Casi es el mismo dibujo que el que usamos en la proyección, solo invertimos el sentido de \vec{d} y lo dibujamos dos veces para ver en dónde quedará \vec{P}' . Del dibujo vemos que $\vec{d} = \vec{p} - \overrightarrow{aP}$.
- Ya tenemos \vec{p} , y del dibujo vemos que $\vec{P}' = \vec{a} + \vec{p} + \vec{d}$, entonces $\vec{P}' = 2\vec{a} - \vec{P} + 2\text{proy}_{\hat{v}}(\overrightarrow{aP})$.



Reflexión

- Casi es el mismo dibujo que el que usamos en la proyección, solo invertimos el sentido de \vec{d} y lo dibujamos dos veces para ver en dónde quedará \vec{P}' . Del dibujo vemos que $\vec{d} = \vec{p} - \vec{a}\vec{P}$.
- Ya tenemos \vec{p} , y del dibujo vemos que $\vec{P}' = \vec{a} + \vec{p} + \vec{d}$, entonces $\vec{P}' = 2\vec{a} - \vec{P} + 2\text{proj}_{\vec{v}}(\vec{a}\vec{P})$.
- De nuevo, programar esto es muy sencillo:

```
point reflection_line(point a, point v, point p){  
    return a*2 - p + proj(p - a, v)*2;  
}
```



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

- Definición
- Proyección y reflexión

- Distancia punto-línea
- Ordenar puntos respecto a una línea
- Intersección de líneas

5 Segmentos

6 Polígonos

7 Círculos



Distancia punto-línea

- Ahora queremos hallar la mínima distancia de un punto P a la recta $\vec{r} = \vec{a} + t\vec{v}$. Realmente ya lo tenemos, pues de los dos dibujos anteriores, es simplemente la longitud del vector \vec{d} , el cual era $\vec{d} = \text{proj}_{\hat{v}}(\overrightarrow{aP}) - \overrightarrow{aP}$.



Distancia punto-línea

- Ahora queremos hallar la mínima distancia de un punto P a la recta $\vec{r} = \vec{a} + t\vec{v}$. Realmente ya lo tenemos, pues de los dos dibujos anteriores, es simplemente la longitud del vector \vec{d} , el cual era $\vec{d} = \text{proj}_{\hat{v}}(\vec{aP}) - \vec{aP}$. Entonces, el código queda así:

```
double distance_point_line(point a, point v, point p){  
    return (proj(p - a, v) - (p - a)).length();  
}
```



Distancia punto-línea

- Pero resulta que hay una manera más fácil: si nos fijamos en el triángulo rectángulo que se forma y si θ es el ángulo entre el vector \vec{v} y el vector \overrightarrow{aP} , por definición de seno tenemos que $\|\vec{d}\| = \|\overrightarrow{aP}\| \sin \theta$.



Distancia punto-línea

- Pero resulta que hay una manera más fácil: si nos fijamos en el triángulo rectángulo que se forma y si θ es el ángulo entre el vector \vec{v} y el vector \overrightarrow{aP} , por definición de seno tenemos que $\|\vec{d}\| = \|\overrightarrow{aP}\| \sin \theta$.
- Y si combinamos la propiedad del producto cruz, es decir,
$$\vec{v} \times \overrightarrow{aP} = \|\vec{v}\| \|\overrightarrow{aP}\| \sin \theta,$$
 entonces $\|\vec{d}\| = \frac{\vec{v} \times \overrightarrow{aP}}{\|\vec{v}\|}.$



Distancia punto-línea

- Pero resulta que hay una manera más fácil: si nos fijamos en el triángulo rectángulo que se forma y si θ es el ángulo entre el vector \vec{v} y el vector \overrightarrow{aP} , por definición de seno tenemos que $\|\vec{d}\| = \|\overrightarrow{aP}\| \sin \theta$.
- Y si combinamos la propiedad del producto cruz, es decir,
$$\vec{v} \times \overrightarrow{aP} = \|\vec{v}\| \|\overrightarrow{aP}\| \sin \theta,$$
 entonces $\|\vec{d}\| = \frac{\vec{v} \times \overrightarrow{aP}}{\|\vec{v}\|}.$
- La expresión anterior puede ser negativa si \vec{P} está del lado derecho de la línea, por lo que se le conoce como la **distancia signada**. Su valor absoluto siempre será correcto, así que si solo queremos la distancia, simplemente sacamos valor absoluto.



Distancia punto-línea

- Pero resulta que hay una manera más fácil: si nos fijamos en el triángulo rectángulo que se forma y si θ es el ángulo entre el vector \vec{v} y el vector \vec{aP} , por definición de seno tenemos que $\|\vec{d}\| = \|\vec{aP}\| \sin \theta$.
- Y si combinamos la propiedad del producto cruz, es decir,
$$\vec{v} \times \vec{aP} = \|\vec{v}\| \|\vec{aP}\| \sin \theta,$$
 entonces $\|\vec{d}\| = \frac{\vec{v} \times \vec{aP}}{\|\vec{v}\|}.$
- La expresión anterior puede ser negativa si \vec{P} está del lado derecho de la línea, por lo que se le conoce como la **distancia signada**. Su valor absoluto siempre será correcto, así que si solo queremos la distancia, simplemente sacamos valor absoluto.

```
double distance_point_line(point a, point v, point p){  
    return abs(v.cross(p - a)) / v.length();  
}  
double signed_distance_point_line(point a, point v, point p){  
    return v.cross(p - a) / v.length();  
}
```

Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

- Definición

- Proyección y reflexión

- Distancia punto-línea
- Ordenar puntos respecto a una línea
- Intersección de líneas

5 Segmentos

6 Polígonos

7 Círculos



Ordenar puntos respecto a una línea

- Dada una línea $\vec{r} = \vec{a} + t\vec{v}$ y varios puntos que **pertenecen** a ella, vamos a determinar en qué orden aparecen, es decir, los vamos a ordenar tal como vayan apareciendo sobre la línea, siguiendo la dirección de \vec{v} .



Ordenar puntos respecto a una línea

- Dada una línea $\vec{r} = \vec{a} + t\vec{v}$ y varios puntos que **pertenecen** a ella, vamos a determinar en qué orden aparecen, es decir, los vamos a ordenar tal como vayan apareciendo sobre la línea, siguiendo la dirección de \vec{v} .
- Para esto nos sirve el valor de t asociado a cada punto. Supongamos que existe alguna t_i tal que $\vec{p}_i = \vec{a} + t_i\vec{v}$, entonces $t_i = \frac{(\vec{p}_i - \vec{a}) \cdot \vec{v}}{\vec{v} \cdot \vec{v}}$.



Ordenar puntos respecto a una línea

- Dada una línea $\vec{r} = \vec{a} + t\vec{v}$ y varios puntos que **pertenecen** a ella, vamos a determinar en qué orden aparecen, es decir, los vamos a ordenar tal como vayan apareciendo sobre la línea, siguiendo la dirección de \vec{v} .
- Para esto nos sirve el valor de t asociado a cada punto. Supongamos que existe alguna t_i tal que $\vec{p}_i = \vec{a} + t_i\vec{v}$, entonces $t_i = \frac{(\vec{p}_i - \vec{a}) \cdot \vec{v}}{\vec{v} \cdot \vec{v}}$.
- Es decir, un punto p_i aparecerá antes que p_j si y solo si $t_i < t_j$.

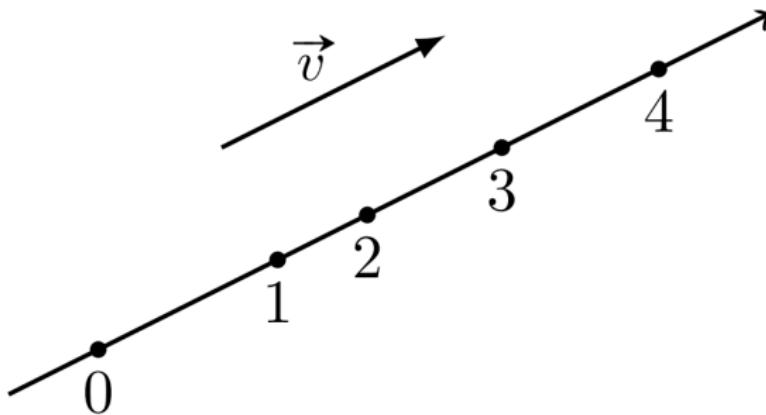


Ordenar puntos respecto a una línea

- Dada una línea $\vec{r} = \vec{a} + t\vec{v}$ y varios puntos que **pertenecen** a ella, vamos a determinar en qué orden aparecen, es decir, los vamos a ordenar tal como vayan apareciendo sobre la línea, siguiendo la dirección de \vec{v} .
- Para esto nos sirve el valor de t asociado a cada punto. Supongamos que existe alguna t_i tal que $\vec{p}_i = \vec{a} + t_i\vec{v}$, entonces $t_i = \frac{(\vec{p}_i - \vec{a}) \cdot \vec{v}}{\vec{v} \cdot \vec{v}}$.
- Es decir, un punto p_i aparecerá antes que p_j si y solo si $t_i < t_j$.
- Pero resulta que para efectos de comparación, solo necesitamos el valor de $\vec{p}_i \cdot \vec{v}$, pues tanto $-\vec{a} \cdot \vec{v}$ como $\vec{v} \cdot \vec{v}$ son constantes y no afectan el orden.



Ordenar puntos respecto a una línea



```
void sort_along_line(point a, point v, vector<point> & points){  
    sort(points.begin(), points.end(), [](point u, point w){  
        return u.dot(v) < w.dot(v);  
    });  
}
```

Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

- Definición
- Proyección y reflexión

- Distancia punto-línea
- Ordenar puntos respecto a una línea
- Intersección de líneas**

5 Segmentos

6 Polígonos

7 Círculos



Intersección de dos líneas

- Sean $\vec{r}_1 = \vec{a}_1 + t\vec{v}_1$ y $\vec{r}_2 = \vec{a}_2 + u\vec{v}_2$ dos líneas. Queremos saber si se intersectan, y si lo hacen, en qué punto.



Intersección de dos líneas

- Sean $\vec{r}_1 = \vec{a}_1 + t\vec{v}_1$ y $\vec{r}_2 = \vec{a}_2 + u\vec{v}_2$ dos líneas. Queremos saber si se intersectan, y si lo hacen, en qué punto.
- Vamos a tratar de igualar ambas expresiones para hallar dicho punto:

$$\vec{a}_1 + t\vec{v}_1 = \vec{a}_2 + u\vec{v}_2$$

$$\vec{a}_1 \times \vec{v}_2 + t\vec{v}_1 \times \vec{v}_2 = \vec{a}_2 \times \vec{v}_2 + \cancel{u\vec{v}_2 \times \vec{v}_2}$$

$$t\vec{v}_1 \times \vec{v}_2 = (\vec{a}_2 - \vec{a}_1) \times \vec{v}_2$$

$$t = \frac{(\vec{a}_2 - \vec{a}_1) \times \vec{v}_2}{\vec{v}_1 \times \vec{v}_2}$$



Intersección de dos líneas

- Sean $\vec{r}_1 = \vec{a}_1 + t\vec{v}_1$ y $\vec{r}_2 = \vec{a}_2 + u\vec{v}_2$ dos líneas. Queremos saber si se intersectan, y si lo hacen, en qué punto.
- Vamos a tratar de igualar ambas expresiones para hallar dicho punto:

$$\begin{aligned}\vec{a}_1 + t\vec{v}_1 &= \vec{a}_2 + u\vec{v}_2 \\ \vec{a}_1 \times \vec{v}_2 + t\vec{v}_1 \times \vec{v}_2 &= \vec{a}_2 \times \vec{v}_2 + \cancel{u\vec{v}_2} \times \cancel{\vec{v}_2} \\ t\vec{v}_1 \times \vec{v}_2 &= (\vec{a}_2 - \vec{a}_1) \times \vec{v}_2 \\ t &= \frac{(\vec{a}_2 - \vec{a}_1) \times \vec{v}_2}{\vec{v}_1 \times \vec{v}_2}\end{aligned}$$

- Ya que tenemos el valor de t , lo sustituimos en \vec{r}_1 para hallar el punto de intersección.



Intersección de dos líneas

- Sean $\vec{r}_1 = \vec{a}_1 + t\vec{v}_1$ y $\vec{r}_2 = \vec{a}_2 + u\vec{v}_2$ dos líneas. Queremos saber si se intersectan, y si lo hacen, en qué punto.
- Vamos a tratar de igualar ambas expresiones para hallar dicho punto:

$$\begin{aligned}\vec{a}_1 + t\vec{v}_1 &= \vec{a}_2 + u\vec{v}_2 \\ \vec{a}_1 \times \vec{v}_2 + t\vec{v}_1 \times \vec{v}_2 &= \vec{a}_2 \times \vec{v}_2 + \cancel{u\vec{v}_2} \times \cancel{\vec{v}_2} \\ t\vec{v}_1 \times \vec{v}_2 &= (\vec{a}_2 - \vec{a}_1) \times \vec{v}_2 \\ t &= \frac{(\vec{a}_2 - \vec{a}_1) \times \vec{v}_2}{\vec{v}_1 \times \vec{v}_2}\end{aligned}$$

- Ya que tenemos el valor de t , lo sustituimos en \vec{r}_1 para hallar el punto de intersección.



Intersección de dos líneas

- Sin embargo, si $\vec{v_1} \times \vec{v_2} = 0$, habría división entre cero.
Geométricamente significa que las dos líneas son paralelas, entonces puede que nunca se intersecten o se intersecten infinitamente (son la misma línea).



Intersección de dos líneas

- Sin embargo, si $\vec{v}_1 \times \vec{v}_2 = 0$, habría división entre cero. Geométricamente significa que las dos líneas son paralelas, entonces puede que nunca se intersecten o se intersecten infinitamente (son la misma línea).
 - Es fácil ver que si $\overrightarrow{a_1 a_2} \times \vec{v}_1 = 0$ las dos líneas son en realidad la misma, por lo que en este caso hay infinitos puntos de intersección.



Intersección de dos líneas

- Sin embargo, si $\vec{v}_1 \times \vec{v}_2 = 0$, habría división entre cero. Geométricamente significa que las dos líneas son paralelas, entonces puede que nunca se intersecten o se intersecten infinitamente (son la misma línea).
 - Es fácil ver que si $\overrightarrow{a_1 a_2} \times \vec{v}_1 = 0$ las dos líneas son en realidad la misma, por lo que en este caso hay infinitos puntos de intersección.
 - En caso de que $\overrightarrow{a_1 a_2} \times \vec{v}_1 \neq 0$, las dos líneas son paralelas y nunca se intersectan.



Intersección de dos líneas

Programemos dos funciones: una que solo nos diga cuántas intersecciones hay entre dos líneas, y otra que nos diga el punto de intersección asumiendo que existe.



Intersección de dos líneas

Programemos dos funciones: una que solo nos diga cuántas intersecciones hay entre dos líneas, y otra que nos diga el punto de intersección asumiendo que existe.

```
int intersectLinesInfo(point a1, point v1, point a2, point v2){  
    double det = v1.cross(v2);  
    if(det == 0){  
        if((a2 - a1).cross(v1) == 0){  
            return -1; //infinity points  
        }else{  
            return 0; //no points  
        }  
    }else{  
        return 1; //single point  
    }  
}  
  
point intersectLines(point a1, point v1, point a2, point v2){  
    return a1 + v1 * ((a2 - a1).cross(v2) / v1.cross(v2));  
}
```

Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

- Definición
- Punto pertenece a segmento
- Intersección línea-segmento
- Intersección de segmentos

6 Polígonos

7 Círculos



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

- Definición

- Punto pertenece a segmento
- Intersección línea-segmento
- Intersección de segmentos

6 Polígonos

7 Círculos



Definición de segmento

Definición

Sean \vec{a} y \vec{b} dos puntos. Definimos al **segmento** de \vec{a} a \vec{b} como $\vec{r} = \vec{a} + t(\vec{b} - \vec{a})$, donde $0 \leq t \leq 1$. También podemos escribirlo como \overrightarrow{ab} .



Definición de segmento

Definición

Sean \vec{a} y \vec{b} dos puntos. Definimos al **segmento** de \vec{a} a \vec{b} como $\vec{r} = \vec{a} + t(\vec{b} - \vec{a})$, donde $0 \leq t \leq 1$. También podemos escribirlo como \overrightarrow{ab} .

Básicamente un segmento es el vector de desplazamiento del punto \vec{a} al punto \vec{b} , o bien, la recta que pasa por \vec{a} y \vec{b} limitada desde $t = 0$ hasta $t = 1$.



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

- Definición
- Punto pertenece a segmento
- Intersección línea-segmento
- Intersección de segmentos

6 Polígonos

7 Círculos



Punto pertenece a segmento

Sea \overrightarrow{ab} un segmento y p un punto. Para checar si p pertenece al segmento, se tienen que cumplir dos condiciones:



Punto pertenece a segmento

Sea \vec{ab} un segmento y \vec{p} un punto. Para checar si p pertenece al segmento, se tienen que cumplir dos condiciones:

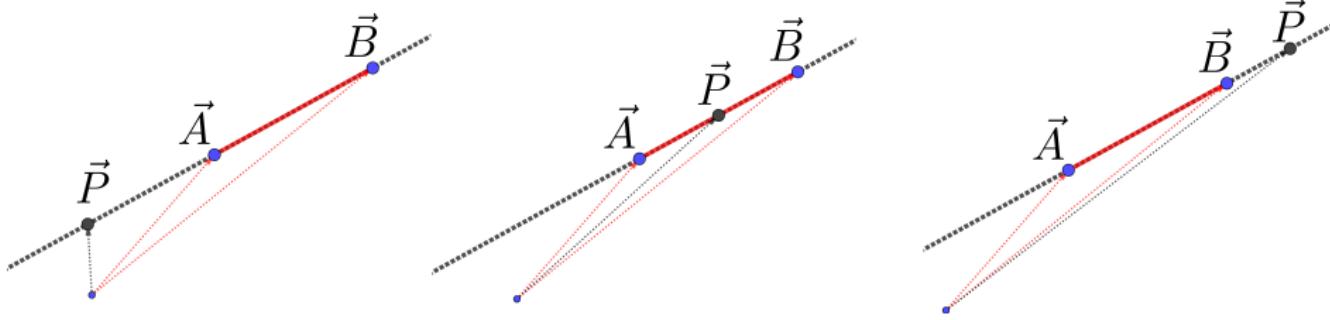
- Que el punto pertenezca a la recta que pasa por \vec{a} y \vec{b} . Ya vimos cómo saber eso: $\vec{ab} \times \vec{ap} = 0$.



Punto pertenece a segmento

Sea \vec{ab} un segmento y \vec{p} un punto. Para checar si p pertenece al segmento, se tienen que cumplir dos condiciones:

- Que el punto pertenezca a la recta que pasa por \vec{a} y \vec{b} . Ya vimos cómo saber eso: $\vec{ab} \times \vec{ap} = 0$.
- Que el punto esté justamente entre \vec{a} y \vec{b} . Para checar esto vamos a usar el producto punto: de la figura vemos que si \vec{p} está a la izquierda de \vec{a} o a la derecha de \vec{b} , entonces $\vec{pa} \cdot \vec{pb} > 0$. Por lo tanto, la condición es $\vec{pa} \cdot \vec{pb} \leq 0$.



Punto pertenece a segmento

Por lo tanto, la función queda como:

```
bool pointInSegment(point a, point b, point p){  
    return (b - a).cross(p - a) == 0 && (a - p).dot(b - p) <= 0;  
}
```



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

- Definición
- Punto pertenece a segmento
- **Intersección línea-segmento**
- Intersección de segmentos

6 Polígonos

7 Círculos



Intersección línea-segmento

Sea $\vec{r} = \vec{a} + t\vec{v}$ una línea y \overrightarrow{cd} un segmento. Para saber si se intersectan, hay varios casos:



Intersección línea-segmento

Sea $\vec{r} = \vec{a} + t\vec{v}$ una línea y \overrightarrow{cd} un segmento. Para saber si se intersectan, hay varios casos:

- Si ambos son paralelos, es decir, $\vec{v} \times \overrightarrow{cd} = 0$:



Intersección línea-segmento

Sea $\vec{r} = \vec{a} + t\vec{v}$ una línea y \overrightarrow{cd} un segmento. Para saber si se intersectan, hay varios casos:

- Si ambos son paralelos, es decir, $\vec{v} \times \overrightarrow{cd} = 0$:
 - El segmento está contenido totalmente en la recta, es decir, $\overrightarrow{ac} \times \vec{v} = 0$. Aquí hay infinitos puntos de intersección.



Intersección línea-segmento

Sea $\vec{r} = \vec{a} + t\vec{v}$ una línea y \overrightarrow{cd} un segmento. Para saber si se intersectan, hay varios casos:

- Si ambos son paralelos, es decir, $\vec{v} \times \overrightarrow{cd} = 0$:
 - El segmento está contenido totalmente en la recta, es decir, $\overrightarrow{ac} \times \vec{v} = 0$. Aquí hay infinitos puntos de intersección.
 - El segmento no está contenido en la recta, es decir, $\overrightarrow{ac} \times \vec{v} \neq 0$. Aquí no hay puntos de intersección.



Intersección línea-segmento

Sea $\vec{r} = \vec{a} + t\vec{v}$ una línea y \overrightarrow{cd} un segmento. Para saber si se intersectan, hay varios casos:

- Si ambos son paralelos, es decir, $\vec{v} \times \overrightarrow{cd} = 0$:
 - El segmento está contenido totalmente en la recta, es decir, $\overrightarrow{ac} \times \vec{v} = 0$. Aquí hay infinitos puntos de intersección.
 - El segmento no está contenido en la recta, es decir, $\overrightarrow{ac} \times \vec{v} \neq 0$. Aquí no hay puntos de intersección.
- Si no son paralelos, es decir, $\vec{v} \times \overrightarrow{cd} \neq 0$, basta ver que \vec{c} esté de un lado de la recta y \vec{d} del otro. O sea, que $\vec{v} \times \overrightarrow{ac}$ y $\vec{v} \times \overrightarrow{ad}$ no tengan el mismo signo.



Intersección línea-segmento

La implementación queda como:

```
int sgn(double x){  
    if(x > 0) return 1;  
    if(x < 0) return -1;  
    return 0;  
}  
  
int intersectLineSegmentInfo(point a, point v, point c, point d){  
    point v2 = d - c;  
    ld det = v.cross(v2);  
    if(det == 0){  
        if((c - a).cross(v) == 0){  
            return -1; //infinity points  
        }else{  
            return 0; //no point  
        }  
    }else{  
        return sgn(v.cross(c - a)) != sgn(v.cross(d - a));  
    }  
}
```



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

- Definición
- Punto pertenece a segmento
- Intersección línea-segmento
- **Intersección de segmentos**

6 Polígonos

7 Círculos



Intersección de segmentos

Sean \overrightarrow{ab} y \overrightarrow{cd} dos segmentos. Para saber si se intersectan hay varios casos:



Intersección de segmentos

Sean \vec{ab} y \vec{cd} dos segmentos. Para saber si se intersectan hay varios casos:

- Si el segmento \vec{cd} está completamente del mismo lado que \vec{ab} , es decir, $\vec{ab} \times \vec{ac}$ y $\vec{ab} \times \vec{ad}$ tienen el mismo signo y no es cero, entonces no hay intersección.



Intersección de segmentos

Sean \overrightarrow{ab} y \overrightarrow{cd} dos segmentos. Para saber si se intersectan hay varios casos:

- Si el segmento \overrightarrow{cd} está completamente del mismo lado que \overrightarrow{ab} , es decir, $\overrightarrow{ab} \times \overrightarrow{ac}$ y $\overrightarrow{ab} \times \overrightarrow{ad}$ tienen el mismo signo y no es cero, entonces no hay intersección.
- Si los dos segmentos son paralelos, hay que revisar si algún segmento contiene un extremo del otro segmento. Si esto ocurre, hay infinitos puntos, si no, no se intersectan.



Intersección de segmentos

Sean \vec{ab} y \vec{cd} dos segmentos. Para saber si se intersectan hay varios casos:

- Si el segmento \vec{cd} está completamente del mismo lado que \vec{ab} , es decir, $\vec{ab} \times \vec{ac}$ y $\vec{ab} \times \vec{ad}$ tienen el mismo signo y no es cero, entonces no hay intersección.
- Si los dos segmentos son paralelos, hay que revisar si algún segmento contiene un extremo del otro segmento. Si esto ocurre, hay infinitos puntos, si no, no se intersectan.
- Si el punto \vec{c} está de un lado del segmento \vec{ab} y el punto \vec{d} está del otro lado, basta con ver lo opuesto: es decir, que el punto \vec{a} esté de un lado del segmento \vec{cd} y el punto \vec{b} esté del otro lado, o sea, que $\vec{cd} \times \vec{ca}$ y $\vec{cd} \times \vec{cb}$ no tengan el mismo signo.



Intersección de segmentos

La implementación queda como:

```
int intersectSegmentsInfo(point a, point b, point c, point d){  
    point v1 = b - a, v2 = d - c;  
    int t = sgn(v1.cross(c - a)), u = sgn(v1.cross(d - a));  
    if(t == u){  
        if(t == 0){  
            if(pointInSegment(a, b, c) || pointInSegment(a, b, d) ||  
                → pointInSegment(c, d, a) || pointInSegment(c, d, b)){  
                return -1; //infinity points  
            }else{  
                return 0; //no point  
            }  
        }else{  
            return 0; //no point  
        }  
    }else{  
        return sgn(v2.cross(a - c)) != sgn(v2.cross(b - c));  
    }  
}
```

Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

- Definición
- Características
- Clasificación
- Representación
- Perímetro
- Área
- Punto en polígono
- Convex Hull

7 Círculos



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

- Definición
- Características
- Clasificación
- Representación
- Perímetro
- Área
- Punto en polígono
- Convex Hull

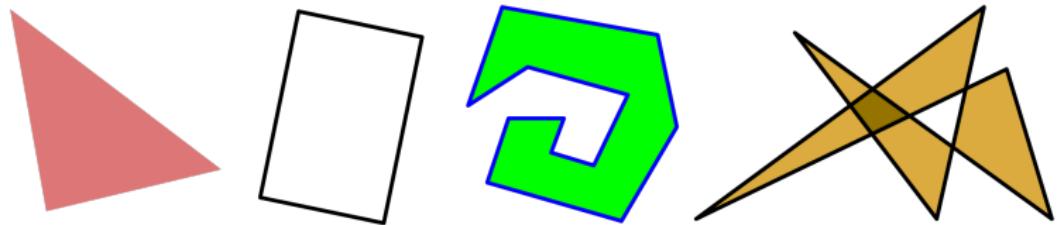
7 Círculos



Definición de polígono

Definición

Un **polígono** es una figura geométrica plana compuesta por una secuencia finita de segmentos de recta consecutivos que encierran una región del plano. Los segmentos se llaman **lados** o **aristas** y los puntos en los que se intersectan son los **vértices**.



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

- Definición
- Características
- Clasificación
- Representación
- Perímetro
- Área
- Punto en polígono
- Convex Hull

7 Círculos



Características de los polígonos

- Todo polígono tiene igual su número de vértices que de lados.



Características de los polígonos

- Todo polígono tiene igual su número de vértices que de lados.
- Cada par de aristas consecutivas forma dos ángulos: el **ángulo interior** y el **ángulo exterior**.



Características de los polígonos

- Todo polígono tiene igual su número de vértices que de lados.
- Cada par de aristas consecutivas forma dos ángulos: el **ángulo interior** y el **ángulo exterior**.
- La suma de los ángulos internos de un polígono de n lados es $180^\circ(n - 2)$.



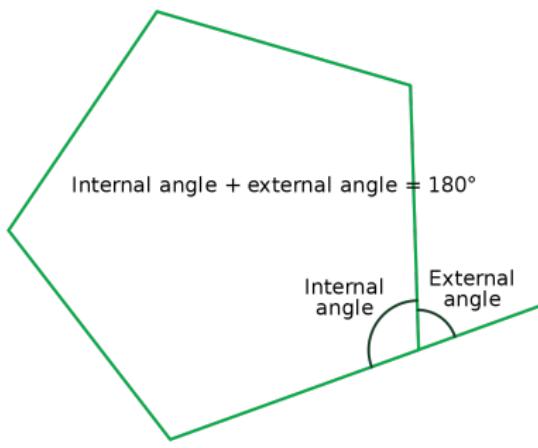
Características de los polígonos

- Todo polígono tiene igual su número de vértices que de lados.
- Cada par de aristas consecutivas forma dos ángulos: el **ángulo interior** y el **ángulo exterior**.
- La suma de los ángulos internos de un polígono de n lados es $180^\circ(n - 2)$.
- Podemos ver a los ángulos exteriores como los ángulos que hay que “girar” para trazar el polígono.



Características de los polígonos

- Todo polígono tiene igual su número de vértices que de lados.
- Cada par de aristas consecutivas forma dos ángulos: el **ángulo interior** y el **ángulo exterior**.
- La suma de los ángulos internos de un polígono de n lados es $180^\circ(n - 2)$.
- Podemos ver a los ángulos exteriores como los ángulos que hay que “girar” para trazar el polígono.



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

- Definición
- Características
- Clasificación
- Representación
- Perímetro
- Área
- Punto en polígono
- Convex Hull

7 Círculos



Clasificación de polígonos

Los polígonos se pueden clasificar de varias maneras de acuerdo a sus propiedades:

- Por su número de lados: triángulo, cuadrilátero, pentágono, ...



Clasificación de polígonos

Los polígonos se pueden clasificar de varias maneras de acuerdo a sus propiedades:

- Por su número de lados: triángulo, cuadrilátero, pentágono, ...
- **Simple:** decimos que un polígono es simple si ningún par de aristas consecutivas se corta. Es decir, su frontera solo tiene un contorno.



Clasificación de polígonos

Los polígonos se pueden clasificar de varias maneras de acuerdo a sus propiedades:

- Por su número de lados: triángulo, cuadrilátero, pentágono, ...
- **Simple**: decimos que un polígono es simple si ningún par de aristas consecutivas se corta. Es decir, su frontera solo tiene un contorno.
- **Convexo**: un polígono es convexo si cualquier segmento que une dos puntos dentro del polígono, cae totalmente dentro del polígono. En caso contrario, decimos que es **cóncavo**. Todos los polígonos simples y con ángulos internos menores a 180° son convexos.



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

- Definición
- Características
- Clasificación
- Representación**
- Perímetro
- Área
- Punto en polígono
- Convex Hull

7 Círculos



Representación de un polígono

- Para representar un polígono de n lados en cualquier lenguaje de programación, simplemente almacenamos en un arreglo los puntos de los vértices en orden, comenzando desde un vértice arbitrario.
- Hay dos formas de hacer lo anterior, podemos guardar dichos puntos en sentido antihorario o en sentido horario.



Representación de un polígono

- Para representar un polígono de n lados en cualquier lenguaje de programación, simplemente almacenamos en un arreglo los puntos de los vértices en orden, comenzando desde un vértice arbitrario.
- Hay dos formas de hacer lo anterior, podemos guardar dichos puntos en sentido antihorario o en sentido horario.
- Si los vértices del polígono son P_0, P_1, \dots, P_{n-1} , podemos considerar que los subíndices se interpretan módulo n para resaltar la naturaleza cíclica del polígono, es decir, $P_n = P_0, P_{n+1} = P_1, \dots$



Representación de un polígono

- Para representar un polígono de n lados en cualquier lenguaje de programación, simplemente almacenamos en un arreglo los puntos de los vértices en orden, comenzando desde un vértice arbitrario.
- Hay dos formas de hacer lo anterior, podemos guardar dichos puntos en sentido antihorario o en sentido horario.
- Si los vértices del polígono son P_0, P_1, \dots, P_{n-1} , podemos considerar que los subíndices se interpretan módulo n para resaltar la naturaleza cíclica del polígono, es decir, $P_n = P_0, P_{n+1} = P_1, \dots$
- Por lo tanto, en C++ simplemente usaremos `vector<point>` o `point[]`.



Representación de un polígono

- Para representar un polígono de n lados en cualquier lenguaje de programación, simplemente almacenamos en un arreglo los puntos de los vértices en orden, comenzando desde un vértice arbitrario.
- Hay dos formas de hacer lo anterior, podemos guardar dichos puntos en sentido antihorario o en sentido horario.
- Si los vértices del polígono son P_0, P_1, \dots, P_{n-1} , podemos considerar que los subíndices se interpretan módulo n para resaltar la naturaleza cíclica del polígono, es decir, $P_n = P_0, P_{n+1} = P_1, \dots$
- Por lo tanto, en C++ simplemente usaremos `vector<point>` o `point[]`.
- Los lados del polígono podemos verlos como los vectores de desplazamiento entre vértices consecutivos, es decir, $\overrightarrow{P_0P_1}, \overrightarrow{P_1P_2}, \dots, \overrightarrow{P_{n-1}P_0}$.



Verificando si un polígono es convexo

Problema: dado un polígono de n vértices, determinar si es convexo o no.
Los vértices pueden estar en orden horario o antihorario.



Verificando si un polígono es convexo

Problema: dado un polígono de n vértices, determinar si es convexo o no. Los vértices pueden estar en orden horario o antihorario.

- Imaginemos que caminamos sobre los lados del polígono. En el momento en que lleguemos a un vértice hay que realizar un giro para seguir avanzando por el siguiente lado.



Verificando si un polígono es convexo

Problema: dado un polígono de n vértices, determinar si es convexo o no. Los vértices pueden estar en orden horario o antihorario.

- Imaginemos que caminamos sobre los lados del polígono. En el momento en que lleguemos a un vértice hay que realizar un giro para seguir avanzando por el siguiente lado.
- En un polígono convexo, dichos giros siempre son todos hacia un mismo lado. Si el polígono nos lo dan en sentido antihorario, los giros son a la izquierda. Si nos lo dan en sentido horario, los giros son a la derecha.



Verificando si un polígono es convexo

Problema: dado un polígono de n vértices, determinar si es convexo o no. Los vértices pueden estar en orden horario o antihorario.

- Imaginemos que caminamos sobre los lados del polígono. En el momento en que lleguemos a un vértice hay que realizar un giro para seguir avanzando por el siguiente lado.
- En un polígono convexo, dichos giros siempre son todos hacia un mismo lado. Si el polígono nos lo dan en sentido antihorario, los giros son a la izquierda. Si nos lo dan en sentido horario, los giros son a la derecha.
- Y también resulta que en un polígono cóncavo siempre habrá al menos un giro a la izquierda y otro a la derecha.



Verificando si un polígono es convexo

Problema: dado un polígono de n vértices, determinar si es convexo o no. Los vértices pueden estar en orden horario o antihorario.

- Imaginemos que caminamos sobre los lados del polígono. En el momento en que lleguemos a un vértice hay que realizar un giro para seguir avanzando por el siguiente lado.
- En un polígono convexo, dichos giros siempre son todos hacia un mismo lado. Si el polígono nos lo dan en sentido antihorario, los giros son a la izquierda. Si nos lo dan en sentido horario, los giros son a la derecha.
- Y también resulta que en un polígono cóncavo siempre habrá al menos un giro a la izquierda y otro a la derecha.
- Por lo tanto, solo hay que verificar que todos los productos cruz entre lados consecutivos tengan el mismo signo.



Verificando si un polígono es convexo

```
bool isConvex(vector<point> P){  
    int n = P.size();  
    bool hasPos = false, hasNeg = false;  
    for(int i = 0; i < n; ++i){  
        point first = P[(i+1)%n] - P[i];  
        point second = P[(i+2)%n] - P[(i+1)%n];  
        double sign = first.cross(second);  
        if(sign > 0) hasPos = true;  
        if(sign < 0) hasNeg = true;  
    }  
    return !(hasPos && hasNeg);  
}
```



Verificando si un polígono es convexo

```
bool isConvex(vector<point> P){  
    int n = P.size();  
    bool hasPos = false, hasNeg = false;  
    for(int i = 0; i < n; ++i){  
        point first = P[(i+1)%n] - P[i];  
        point second = P[(i+2)%n] - P[(i+1)%n];  
        double sign = first.cross(second);  
        if(sign > 0) hasPos = true;  
        if(sign < 0) hasNeg = true;  
    }  
    return !(hasPos && hasNeg);  
}
```

Pregunta: ¿qué pasa si tres vértices consecutivos en el polígono están alineados?



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

- Definición
- Características
- Clasificación
- Representación

• Perímetro

- Área
- Punto en polígono
- Convex Hull

7 Círculos



Perímetro de un polígono

Hallar el perímetro de un polígono es fácil, solo hay que sumar las longitudes de todas las aristas:



Perímetro de un polígono

Hallar el perímetro de un polígono es fácil, solo hay que sumar las longitudes de todas las aristas:

```
double perimeter(vector<point> P){  
    int n = P.size();  
    double sum = 0;  
    for(int i = 0; i < n; ++i){  
        sum += (P[(i+1)%n] - P[i]).length();  
    }  
    return sum;  
}
```



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

- Definición
- Características
- Clasificación
- Representación
- Perímetro
- Área**
- Punto en polígono
- Convex Hull

7 Círculos



Área de un polígono

Hallar el área de un polígono también es fácil de implementar, pero la idea está muy interesante.



Área de un polígono

Hallar el área de un polígono también es fácil de implementar, pero la idea está muy interesante.

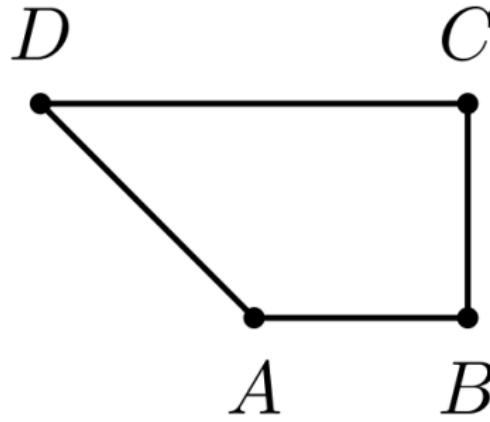
- Ya sabemos hallar el área de cualquier triángulo. Entonces podríamos dividir nuestro polígono en triángulos y sumar todas sus áreas, sin embargo hallar dicha triangulación no es sencillo.



Área de un polígono

Hallar el área de un polígono también es fácil de implementar, pero la idea está muy interesante.

- Ya sabemos hallar el área de cualquier triángulo. Entonces podríamos dividir nuestro polígono en triángulos y sumar todas sus áreas, sin embargo hallar dicha triangulación no es sencillo.
- En lugar de eso, vamos a sumar y restar áreas de triángulos de una forma más astuta. Por ejemplo, consideremos este cuadrilátero:



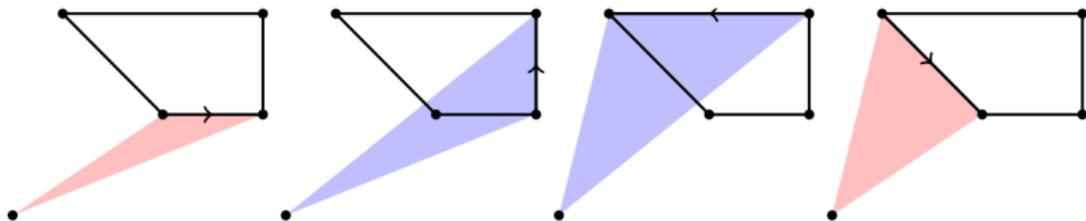
Área de un polígono

- Sea O el origen y consideremos los vértices A, B, C, D en orden. Para cada par de puntos consecutivos P_i y P_{i+1} , sumaremos el **área signada** del triángulo $\triangle OP_iP_{i+1}$, es decir, $\frac{1}{2}\overrightarrow{P_i} \times \overrightarrow{P_{i+1}}$, a la suma total. Haciendo esto, al final tendremos el área total del polígono. Veamos por qué:



Área de un polígono

- Sea O el origen y consideremos los vértices A, B, C, D en orden. Para cada par de puntos consecutivos P_i y P_{i+1} , sumaremos el **área signada** del triángulo $\triangle OP_iP_{i+1}$, es decir, $\frac{1}{2}\overrightarrow{P_i} \times \overrightarrow{P_{i+1}}$, a la suma total. Haciendo esto, al final tendremos el área total del polígono. Veamos por qué:

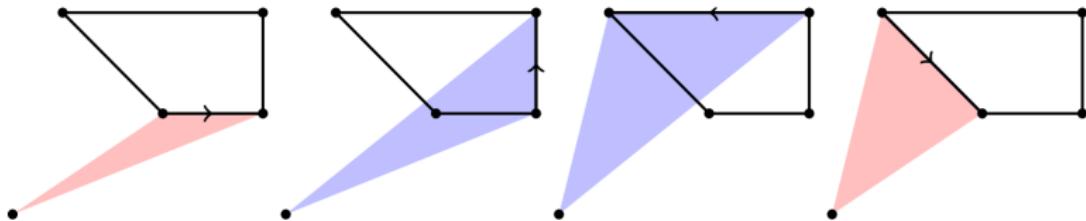


- Los triángulos rojos son los que tienen área negativa y los azules tienen área positiva.



Área de un polígono

- Sea O el origen y consideremos los vértices A, B, C, D en orden. Para cada par de puntos consecutivos P_i y P_{i+1} , sumaremos el **área signada** del triángulo $\triangle OP_iP_{i+1}$, es decir, $\frac{1}{2}\overrightarrow{P_i} \times \overrightarrow{P_{i+1}}$, a la suma total. Haciendo esto, al final tendremos el área total del polígono. Veamos por qué:



- Los triángulos rojos son los que tienen área negativa y los azules tienen área positiva.
- Este método funciona para cualquier tipo de polígono, no importa si es convexo o cóncavo.



Área de un polígono

```
double area(vector<point> P){  
    int n = P.size();  
    double sum = 0;  
    for(int i = 0; i < n; ++i){  
        sum += P[i].cross(P[(i+1)%n]);  
    }  
    return abs(sum / 2);  
}
```



Área de un polígono

```
double area(vector<point> P){  
    int n = P.size();  
    double sum = 0;  
    for(int i = 0; i < n; ++i){  
        sum += P[i].cross(P[(i+1)%n]);  
    }  
    return abs(sum / 2);  
}
```

Preguntas:

- ¿Qué pasa si usamos otro punto distinto del origen como referencia?



Área de un polígono

```
double area(vector<point> P){  
    int n = P.size();  
    double sum = 0;  
    for(int i = 0; i < n; ++i){  
        sum += P[i].cross(P[(i+1)%n]);  
    }  
    return abs(sum / 2);  
}
```

Preguntas:

- ¿Qué pasa si usamos otro punto distinto del origen como referencia?
- ¿Por qué hay que tomar el valor absoluto de la suma final?



Área de un polígono

```
double area(vector<point> P){  
    int n = P.size();  
    double sum = 0;  
    for(int i = 0; i < n; ++i){  
        sum += P[i].cross(P[(i+1)%n]);  
    }  
    return abs(sum / 2);  
}
```

Preguntas:

- ¿Qué pasa si usamos otro punto distinto del origen como referencia?
- ¿Por qué hay que tomar el valor absoluto de la suma final?
- Si todos los puntos del polígono tienen coordenadas enteras, ¿qué se puede decir sobre el área?.



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

- Definición
- Características
- Clasificación
- Representación
- Perímetro
- Área
- Punto en polígono**
- Convex Hull

7 Círculos



Punto en polígono

Dado un polígono P_0, P_1, \dots, P_{n-1} y un punto \vec{p} , determinar si \vec{p} está dentro, en el perímetro o fuera del polígono.

- El problema se simplifica si el polígono es convexo. Simplemente tenemos que sumar las áreas de todos los triángulos que se forman al unir \vec{p} con cada vértice del polígono. Si dicha suma es igual al área del polígono, \vec{p} está dentro, de lo contrario está \vec{p} fuera.



Punto en polígono

Dado un polígono P_0, P_1, \dots, P_{n-1} y un punto \vec{p} , determinar si \vec{p} está dentro, en el perímetro o fuera del polígono.

- El problema se simplifica si el polígono es convexo. Simplemente tenemos que sumar las áreas de todos los triángulos que se forman al unir \vec{p} con cada vértice del polígono. Si dicha suma es igual al área del polígono, \vec{p} está dentro, de lo contrario está \vec{p} fuera.
- Pero este método, conocido como *ray casting*, funciona para cualquier polígono:



Punto en polígono

Dado un polígono P_0, P_1, \dots, P_{n-1} y un punto \vec{p} , determinar si \vec{p} está dentro, en el perímetro o fuera del polígono.

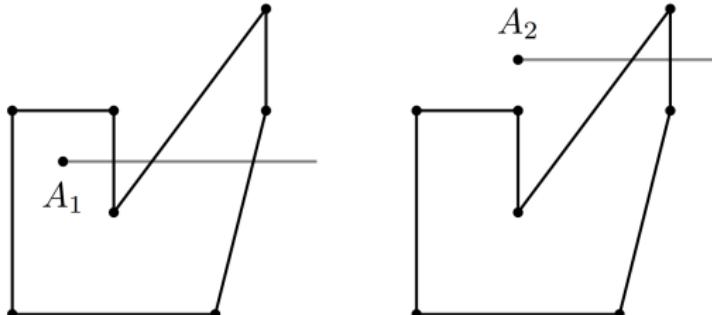
- El problema se simplifica si el polígono es convexo. Simplemente tenemos que sumar las áreas de todos los triángulos que se forman al unir \vec{p} con cada vértice del polígono. Si dicha suma es igual al área del polígono, \vec{p} está dentro, de lo contrario está \vec{p} fuera.
- Pero este método, conocido como *ray casting*, funciona para cualquier polígono:
 - Vamos a trazar un rayo imaginario desde \vec{p} que se extienda al infinito y contar cuántas veces intersecta las aristas del polígono. Si este número es impar, \vec{p} está dentro, si es par, \vec{p} está fuera.



Punto en polígono

Dado un polígono P_0, P_1, \dots, P_{n-1} y un punto \vec{p} , determinar si \vec{p} está dentro, en el perímetro o fuera del polígono.

- El problema se simplifica si el polígono es convexo. Simplemente tenemos que sumar las áreas de todos los triángulos que se forman al unir \vec{p} con cada vértice del polígono. Si dicha suma es igual al área del polígono, \vec{p} está dentro, de lo contrario está \vec{p} fuera.
- Pero este método, conocido como *ray casting*, funciona para cualquier polígono:
 - Vamos a trazar un rayo imaginario desde \vec{p} que se extienda al infinito y contar cuántas veces intersecta las aristas del polígono. Si este número es impar, \vec{p} está dentro, si es par, \vec{p} está fuera.



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

- Definición
- Características
- Clasificación
- Representación
- Perímetro
- Área
- Punto en polígono
- Convex Hull

7 Círculos



Convex Hull

Definición

Sea S un conjunto de puntos. Definimos al **convex hull** o a la **envolvente convexa** de S como el mínimo subconjunto convexo de S que contiene a todos los puntos de S .



Definición

Sea S un conjunto de puntos. Definimos al **convex hull** o a la **envolvente convexa** de S como el mínimo subconjunto convexo de S que contiene a todos los puntos de S .

Una forma intuitiva de comprender la definición es imaginando que en cada punto de S colocamos un clavo, y después estiramos una banda de goma de tal forma que abarque todos los puntos. Cuando la banda de goma trate de recuperar su forma original, adquirirá la forma del convex hull.

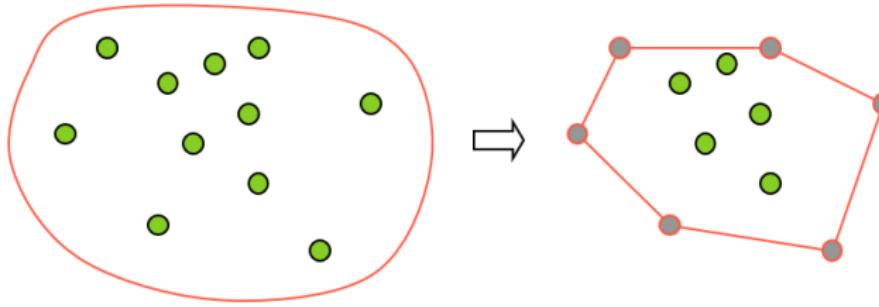


Convex Hull

Definición

Sea S un conjunto de puntos. Definimos al **convex hull** o a la **envolvente convexa** de S como el mínimo subconjunto convexo de S que contiene a todos los puntos de S .

Una forma intuitiva de comprender la definición es imaginando que en cada punto de S colocamos un clavo, y después estiramos una banda de goma de tal forma que abarque todos los puntos. Cuando la banda de goma trate de recuperar su forma original, adquirirá la forma del convex hull.



Convex Hull

Para hallar el convex hull de un conjunto de n puntos con complejidad $O(n \log n)$, usaremos un algoritmo llamado **Andrew's monotone chain**:



Convex Hull

Para hallar el convex hull de un conjunto de n puntos con complejidad $O(n \log n)$, usaremos un algoritmo llamado **Andrew's monotone chain**:

- Ordenamos los puntos de S lexicográficamente: primero por su coordenada en x , y en caso de empate, por su coordenada en y . De esta manera quedan ordenados como si una línea vertical infinita barriera todo el plano de izquierda a derecha.



Convex Hull

Para hallar el convex hull de un conjunto de n puntos con complejidad $O(n \log n)$, usaremos un algoritmo llamado **Andrew's monotone chain**:

- Ordenamos los puntos de S lexicográficamente: primero por su coordenada en x , y en caso de empate, por su coordenada en y . De esta manera quedan ordenados como si una línea vertical infinita barriera todo el plano de izquierda a derecha.
- Sea L la parte de abajo del convex hull y U la parte de arriba. Al principio están vacías.



Convex Hull

Para hallar el convex hull de un conjunto de n puntos con complejidad $O(n \log n)$, usaremos un algoritmo llamado **Andrew's monotone chain**:

- Ordenamos los puntos de S lexicográficamente: primero por su coordenada en x , y en caso de empate, por su coordenada en y . De esta manera quedan ordenados como si una línea vertical infinita barriera todo el plano de izquierda a derecha.
- Sea L la parte de abajo del convex hull y U la parte de arriba. Al principio están vacías.
- Para construir L iteramos por todos los puntos de S . Sea S_i el punto actual. Mientras la arista que forma el último punto de L y S_i esté a la derecha o sea colineal con la última arista de L , quitamos el último punto de L . Al final, insertamos S_i a L .



Convex Hull

Para hallar el convex hull de un conjunto de n puntos con complejidad $O(n \log n)$, usaremos un algoritmo llamado **Andrew's monotone chain**:

- Ordenamos los puntos de S lexicográficamente: primero por su coordenada en x , y en caso de empate, por su coordenada en y . De esta manera quedan ordenados como si una línea vertical infinita barriera todo el plano de izquierda a derecha.
- Sea L la parte de abajo del convex hull y U la parte de arriba. Al principio están vacías.
- Para construir L iteramos por todos los puntos de S . Sea S_i el punto actual. Mientras la arista que forma el último punto de L y S_i esté a la derecha o sea colineal con la última arista de L , quitamos el último punto de L . Al final, insertamos S_i a L .
- La construcción de U es casi idéntica, solo que aquí iteramos los puntos de S en reversa.



Convex Hull

Para hallar el convex hull de un conjunto de n puntos con complejidad $O(n \log n)$, usaremos un algoritmo llamado **Andrew's monotone chain**:

- Ordenamos los puntos de S lexicográficamente: primero por su coordenada en x , y en caso de empate, por su coordenada en y . De esta manera quedan ordenados como si una línea vertical infinita barriera todo el plano de izquierda a derecha.
- Sea L la parte de abajo del convex hull y U la parte de arriba. Al principio están vacías.
- Para construir L iteramos por todos los puntos de S . Sea S_i el punto actual. Mientras la arista que forma el último punto de L y S_i esté a la derecha o sea colineal con la última arista de L , quitamos el último punto de L . Al final, insertamos S_i a L .
- La construcción de U es casi idéntica, solo que aquí iteramos los puntos de S en reversa.
- Quitamos el último punto tanto de L como de U , y la respuesta será la unión de L con U . Dicha unión será un polígono convexo en orden antihorario.



Convex Hull

```
vector<point> convexHull(vector<point> P){  
    vector<point> L, U;  
    sort(P.begin(), P.end());  
    P.erase(unique(P.begin(), P.end()), P.end());  
    int n = P.size();  
    if(n <= 2) return P;  
  
    for(int i = 0; i < n; ++i){  
        while(L.size() >= 2 && (L[L.size()-1] - L[L.size()-2]).cross(P[i] - L[L.size()-1]) <= 0){  
            L.pop_back();  
        }  
        L.push_back(P[i]);  
    }  
  
    for(int i = n-1; i >= 0; --i){  
        while(U.size() >= 2 && (U[U.size()-1] - U[U.size()-2]).cross(P[i] - U[U.size()-1]) <= 0){  
            U.pop_back();  
        }  
        U.push_back(P[i]);  
    }  
  
    L.pop_back(), U.pop_back();  
    L.insert(L.end(), U.begin(), U.end());  
    return L;  
}
```



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos

- Definición
- Círculo a través de tres puntos
- Distancia punto-círculo
- Tangentes desde punto exterior
- Intersección círculo-línea



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos

- Definición

- Círculo a través de tres puntos
- Distancia punto-círculo
- Tangentes desde punto exterior
- Intersección círculo-línea



Definición de círculo

Definición

Un **círculo** es el conjunto de puntos que están a la misma distancia de otro punto llamado **centro**. A la distancia común le llamamos **radio**.



Definición de círculo

Definición

Un **círculo** es el conjunto de puntos que están a la misma distancia de otro punto llamado **centro**. A la distancia común le llamamos **radio**.

- Usando geometría analítica, si el centro es el punto (x_0, y_0) y el radio es r , la ecuación del círculo es $(x - x_0)^2 + (y - y_0)^2 = r^2$.



Definición de círculo

Definición

Un **círculo** es el conjunto de puntos que están a la misma distancia de otro punto llamado **centro**. A la distancia común le llamamos **radio**.

- Usando geometría analítica, si el centro es el punto (x_0, y_0) y el radio es r , la ecuación del círculo es $(x - x_0)^2 + (y - y_0)^2 = r^2$.
- De forma paramétrica, es $\vec{r} = (x_0 + r \cos \theta, y_0 + r \sin \theta)$, para $0 \leq \theta \leq 2\pi$.



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos

- Definición
- Círculo a través de tres puntos
- Distancia punto-círculo
- Tangentes desde punto exterior
- Intersección círculo-línea



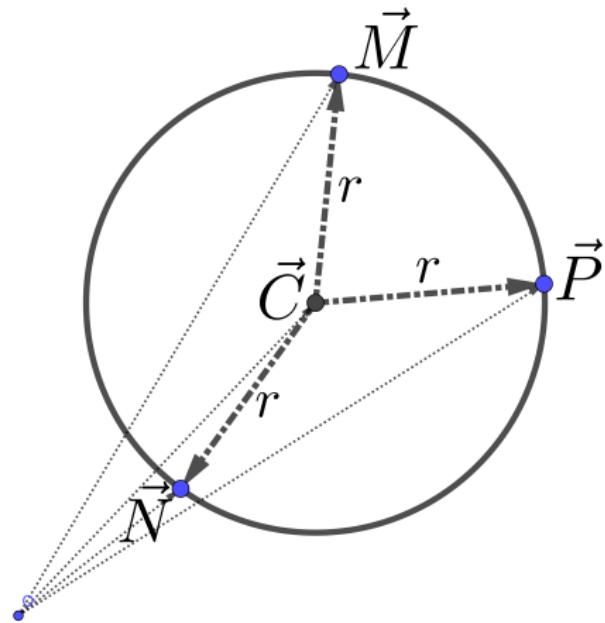
Círculo a través de tres puntos

Si tenemos tres puntos no alineados \vec{M} , \vec{N} y \vec{P} , existe un único círculo que pasa por los tres.



Círculo a través de tres puntos

Si tenemos tres puntos no alineados \vec{M} , \vec{N} y \vec{P} , existe un único círculo que pasa por los tres.



Se puede demostrar que las mediatrices del triángulo $\triangle MNP$ concurren en el centro del círculo. Entonces, para hallarlo:



Se puede demostrar que las mediatrices del triángulo $\triangle MNP$ concurren en el centro del círculo. Entonces, para hallarlo:

- El punto medio de \overrightarrow{MN} y un vector perpendicular a \overrightarrow{MN} definen una mediatriz.



Se puede demostrar que las mediatrices del triángulo $\triangle MNP$ concurren en el centro del círculo. Entonces, para hallarlo:

- El punto medio de \overrightarrow{MN} y un vector perpendicular a \overrightarrow{MN} definen una mediatriz.
- El punto medio de \overrightarrow{MP} y un vector perpendicular a \overrightarrow{MP} definen otra mediatriz.



Se puede demostrar que las mediatrices del triángulo $\triangle MNP$ concurren en el centro del círculo. Entonces, para hallarlo:

- El punto medio de \overrightarrow{MN} y un vector perpendicular a \overrightarrow{MN} definen una mediatriz.
- El punto medio de \overrightarrow{MP} y un vector perpendicular a \overrightarrow{MP} definen otra mediatriz.
- La intersección de esas dos líneas es el centro.



Se puede demostrar que las mediatrices del triángulo $\triangle MNP$ concurren en el centro del círculo. Entonces, para hallarlo:

- El punto medio de \overrightarrow{MN} y un vector perpendicular a \overrightarrow{MN} definen una mediatriz.
- El punto medio de \overrightarrow{MP} y un vector perpendicular a \overrightarrow{MP} definen otra mediatriz.
- La intersección de esas dos líneas es el centro.
- Y para hallar el radio r , tomamos la distancia de cualquiera de los puntos $\vec{M}, \vec{N}, \vec{P}$ al centro.



Se puede demostrar que las mediatrices del triángulo $\triangle MNP$ concurren en el centro del círculo. Entonces, para hallarlo:

- El punto medio de \overrightarrow{MN} y un vector perpendicular a \overrightarrow{MN} definen una mediatriz.
- El punto medio de \overrightarrow{MP} y un vector perpendicular a \overrightarrow{MP} definen otra mediatriz.
- La intersección de esas dos líneas es el centro.
- Y para hallar el radio r , tomamos la distancia de cualquiera de los puntos $\vec{M}, \vec{N}, \vec{P}$ al centro.

```
pair<point, double> getCircle(point m, point n, point p){  
    point c = intersectLines((n + m) / 2, (n - m).perp(), (p + m) / 2, (p  
    ↪ - m).perp());  
    double r = (c - m).length();  
    return {c, r};  
}
```



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos

- Definición
- Círculo a través de tres puntos
- **Distancia punto-círculo**
- Tangentes desde punto exterior
- Intersección círculo-línea



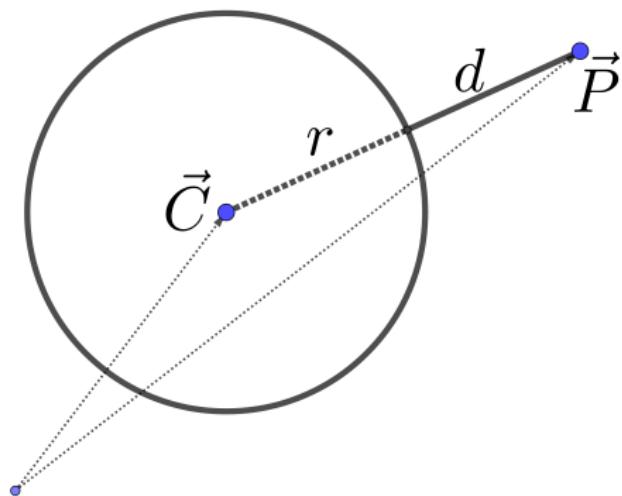
Distancia punto-círculo

Si tenemos un círculo con centro en \vec{C} y radio r , y un punto \vec{P} afuera del círculo, su mínima distancia al círculo es igual a la distancia al centro menos el radio:



Distancia punto-círculo

Si tenemos un círculo con centro en \vec{C} y radio r , y un punto \vec{P} afuera del círculo, su mínima distancia al círculo es igual a la distancia al centro menos el radio:



$$d = \|\overrightarrow{CP}\| - r$$



Distancia punto-círculo

Lo anterior también nos sirve para checar si \vec{P} está dentro, fuera o en el perímetro del círculo:



Distancia punto-círculo

Lo anterior también nos sirve para checar si \vec{P} está dentro, fuera o en el perímetro del círculo:

- Si $d > 0$, el punto está afuera.



Distancia punto-círculo

Lo anterior también nos sirve para checar si \vec{P} está dentro, fuera o en el perímetro del círculo:

- Si $d > 0$, el punto está afuera.
- Si $d = 0$, el punto está sobre el perímetro del círculo.



Distancia punto-círculo

Lo anterior también nos sirve para checar si \vec{P} está dentro, fuera o en el perímetro del círculo:

- Si $d > 0$, el punto está afuera.
- Si $d = 0$, el punto está sobre el perímetro del círculo.
- Si $d < 0$, el punto está dentro.



Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

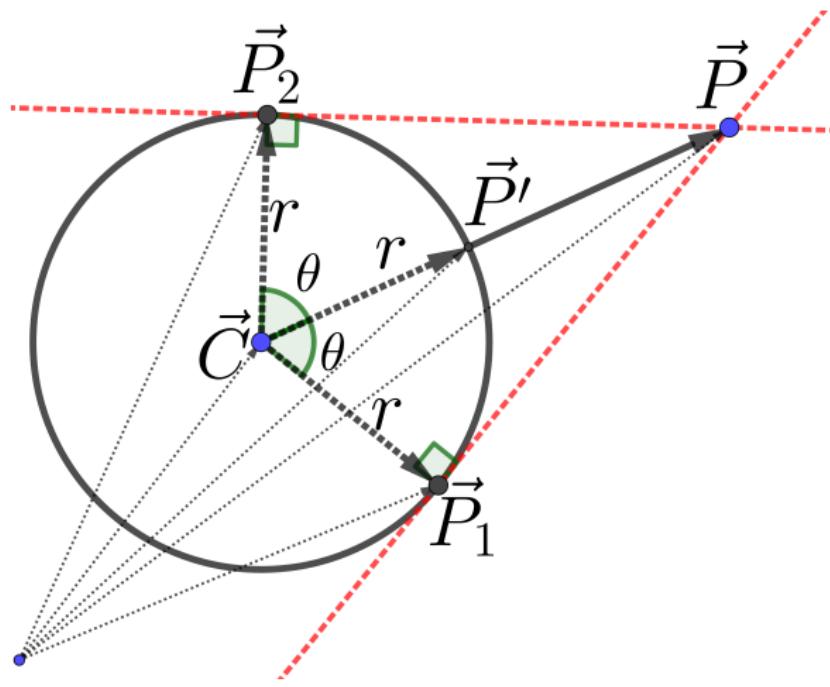
7 Círculos

- Definición
- Círculo a través de tres puntos
- Distancia punto-círculo
- **Tangentes desde punto exterior**
- Intersección círculo-línea



Tangentes desde punto exterior

Para hallar los puntos de tangencia desde un punto exterior \vec{P} al círculo, hacemos lo siguiente:



Tangentes desde punto exterior

- Primero vamos a hallar el vector $\overrightarrow{CP'}$, el cual lleva la misma dirección y sentido que \overrightarrow{CP} pero mide r , entonces $\overrightarrow{CP'} = r\hat{C}P$.



Tangentes desde punto exterior

- Primero vamos a hallar el vector $\overrightarrow{CP'}$, el cual lleva la misma dirección y sentido que \overrightarrow{CP} pero mide r , entonces $\overrightarrow{CP'} = r\hat{C}P$.
- Hallemos el ángulo θ . En la figura se forman dos triángulos rectángulos congruentes, por lo tanto, $\cos \theta = \frac{r}{\|\overrightarrow{CP}\|}$.



Tangentes desde punto exterior

- Primero vamos a hallar el vector $\overrightarrow{CP'}$, el cual lleva la misma dirección y sentido que \overrightarrow{CP} pero mide r , entonces $\overrightarrow{CP'} = r\overrightarrow{CP}$.
- Hallemos el ángulo θ . En la figura se forman dos triángulos rectángulos congruentes, por lo tanto, $\cos \theta = \frac{r}{\|\overrightarrow{CP}\|}$.
- Finalmente, vemos que si rotamos el vector $\overrightarrow{CP'}$ tanto a la izquierda como a la derecha en un ángulo de θ alrededor del centro C , obtenemos los puntos deseados: \vec{P}_1 y \vec{P}_2 .



Tangentes desde punto exterior

- Primero vamos a hallar el vector $\overrightarrow{CP'}$, el cual lleva la misma dirección y sentido que \overrightarrow{CP} pero mide r , entonces $\overrightarrow{CP'} = r\hat{C}P$.
- Hallemos el ángulo θ . En la figura se forman dos triángulos rectángulos congruentes, por lo tanto, $\cos \theta = \frac{r}{\|\overrightarrow{CP}\|}$.
- Finalmente, vemos que si rotamos el vector $\overrightarrow{CP'}$ tanto a la izquierda como a la derecha en un ángulo de θ alrededor del centro \vec{C} , obtenemos los puntos deseados: \vec{P}_1 y \vec{P}_2 .

En código, esto se ve así:

```
pair<point, point> pointsOfTangency(point c, double r, point p){  
    point v = (p - c).unit() * r;  
    double cos_theta = r / (p - c).length();  
    double theta = acos(max(-1.0, min(1.0, cos_theta)));  
    return {c + v.rotate(-theta), c + v.rotate(theta)};  
}
```

Contenido

1 Conceptos básicos

2 Producto punto

3 Producto cruz

4 Líneas

5 Segmentos

6 Polígonos

7 Círculos

- Definición
- Círculo a través de tres puntos
- Distancia punto-círculo
- Tangentes desde punto exterior
- Intersección círculo-línea



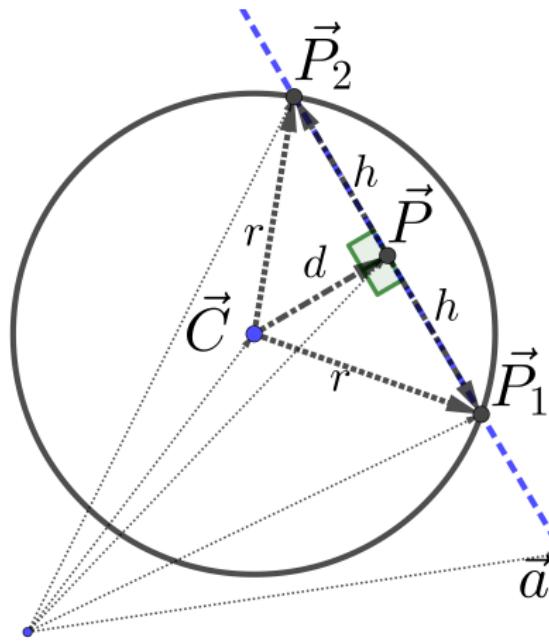
Intersección círculo-línea

Supongamos que queremos intersectar un círculo con centro \vec{C} y radio r con una línea $\vec{r} = \vec{a} + t\vec{v}$. Hay tres casos posibles: que tengan cero, una o dos intersecciones. En la siguiente figura analizaremos el caso de dos intersecciones en \vec{P}_1 y \vec{P}_2 :



Intersección círculo-línea

Supongamos que queremos intersectar un círculo con centro \vec{C} y radio r con una línea $\vec{r} = \vec{a} + t\vec{v}$. Hay tres casos posibles: que tengan cero, una o dos intersecciones. En la siguiente figura analizaremos el caso de dos intersecciones en \vec{P}_1 y \vec{P}_2 :



Intersección círculo-línea

- Primero haremos el punto \vec{P} , que es justamente la proyección de \vec{C} en la recta. El valor de d es la longitud del vector \overrightarrow{CP} , es decir,
$$d = \|\overrightarrow{CP}\|.$$



Intersección círculo-línea

- Primero haremos el punto \vec{P} , que es justamente la proyección de \vec{C} en la recta. El valor de d es la longitud del vector \overrightarrow{CP} , es decir,
$$d = \|\overrightarrow{CP}\|.$$
- Luego, podemos hallar h con teorema de Pitágoras: $h = \sqrt{r^2 - d^2}$.



Intersección círculo-línea

- Primero hallemos el punto \vec{P} , que es justamente la proyección de \vec{C} en la recta. El valor de d es la longitud del vector \overrightarrow{CP} , es decir,
$$d = \|\overrightarrow{CP}\|.$$
- Luego, podemos hallar h con teorema de Pitágoras: $h = \sqrt{r^2 - d^2}$.
- Por último, notemos que desde el punto \vec{P} nos podemos mover h unidades hacia abajo o hacia arriba en la dirección de \vec{v} para obtener los puntos deseados \vec{P}_1 y \vec{P}_2 .



Intersección círculo-línea

- Primero hallemos el punto \vec{P} , que es justamente la proyección de \vec{C} en la recta. El valor de d es la longitud del vector \overrightarrow{CP} , es decir, $d = \|\overrightarrow{CP}\|$.
- Luego, podemos hallar h con teorema de Pitágoras: $h = \sqrt{r^2 - d^2}$.
- Por último, notemos que desde el punto \vec{P} nos podemos mover h unidades hacia abajo o hacia arriba en la dirección de \vec{v} para obtener los puntos deseados \vec{P}_1 y \vec{P}_2 .

```
vector<point> intersectLineCircle(point a, point v, point c, double r){  
    point p = proj_line(a, v, c);  
    double d = (p - c).length();  
    double h = r*r - d*d;  
    if(h == 0) return {p}; //line tangent to circle  
    else if(h < 0) return {};//no intersection  
    else{  
        point u = v.unit() * sqrt(h);  
        return {p - u, p + u}; //two points of intersection (chord)  
    }  
}
```