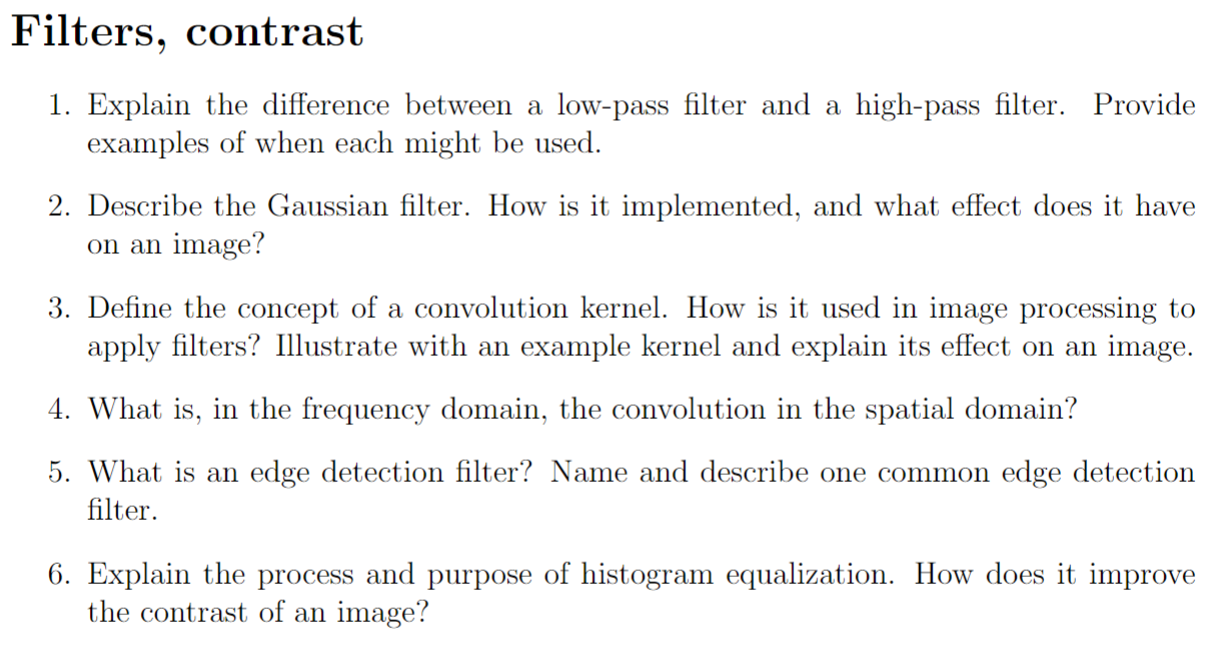
**Resolución de lista de ejercicios Parcial Computación gráfica**



1)

Dado un dominio de frecuencias, un low-pass filter atenúa las frecuencias altas y preserva las frecuencias bajas. Por otro lado, un high-pass filter preserva las frecuencias altas y atenúa las bajas.

Un low-pass filter puede ser utilizado para hacerle un blur a la imagen y eliminar el ruido.

Un high-pass filter para detección de bordes.

2)

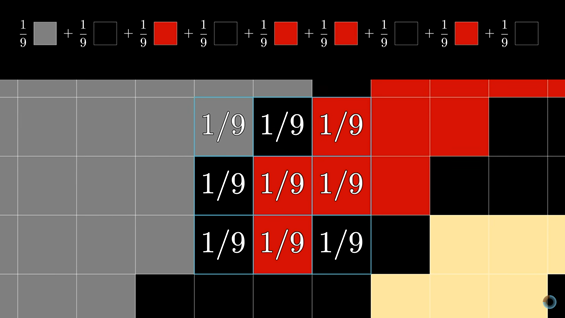
Realiza una ponderación de los píxeles vecinos en base a la distribución gaussiana. Se utiliza para eliminar el ruido gaussiano y es un modelo realista de lente desenfocada. Sigma define la cantidad de desenfoque.

3)

Kernel convolucional: un grid con valores basados en una distribución.

Para determinar el nuevo valor de un píxel se realiza una ponderación con los píxeles vecinos.

Ejemplo:



Este kernel pondera los pixeles vecinos, asignando dicha ponderacion al pixel central.

4)

5)

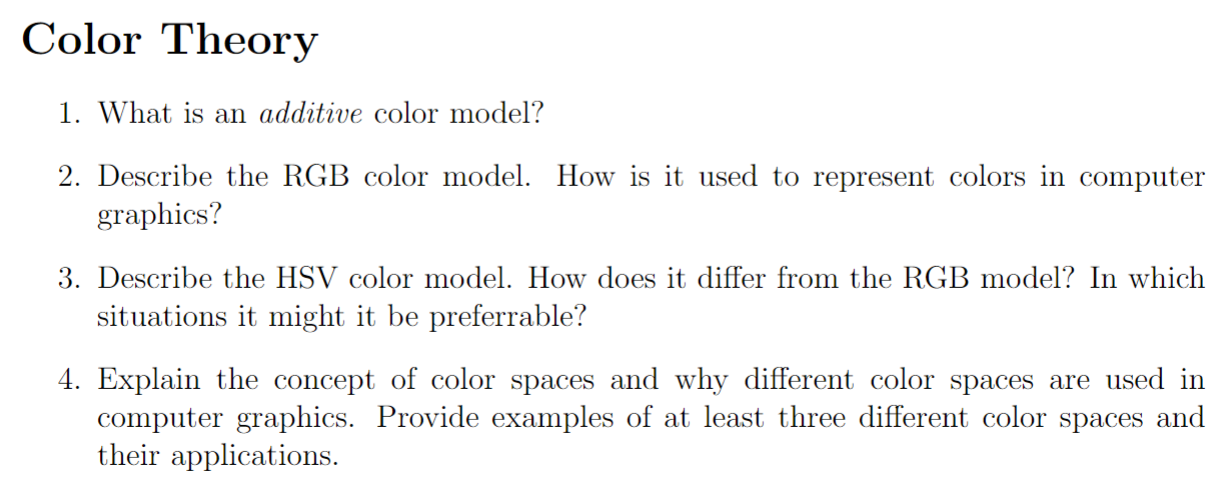
Se utiliza en el procesamiento de imágenes para identificar áreas en una imagen donde hay un cambio significativo en intensidad o color.

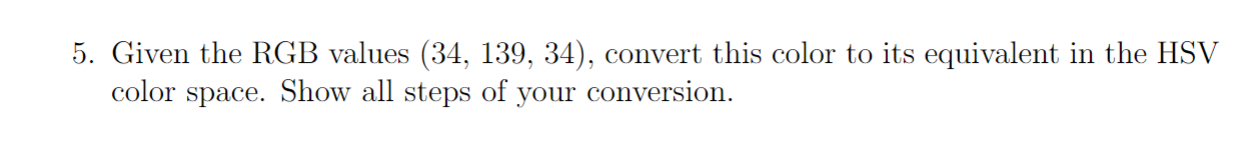
Es un tipo de high-pass filter donde el valor central del kernel es negativo.

Laplacian filter

6)

Aumentar el contraste significa amplificar el rango de valores de los pixeles





1)

Sistema de mezcla de colores basado en la adición de luz de distintos colores.

2)

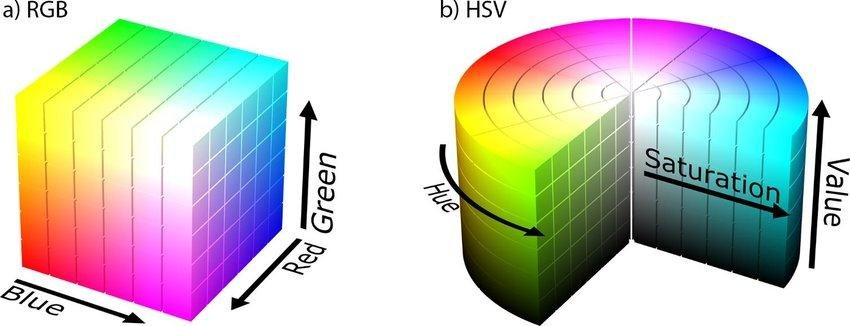
* Es un “sistema de colores aditivo”
* Combinación de diferentes intensidades de luz roja, verde y azul.
* Cada pixel tiene 3 componentes.

3)

Hue, Saturation, Value

Basado en la percepcion de color del ser humano y no en una mezcla de luces.

Permite un control mas intuitivo.



Situaciones donde HSV es Preferible:

Edición de Imágenes: HSV es más intuitivo para tareas como ajustar el tono (matiz), la intensidad (saturación) y el brillo (valor) de los colores, facilitando la corrección de color y la creación de efectos visuales.

4)

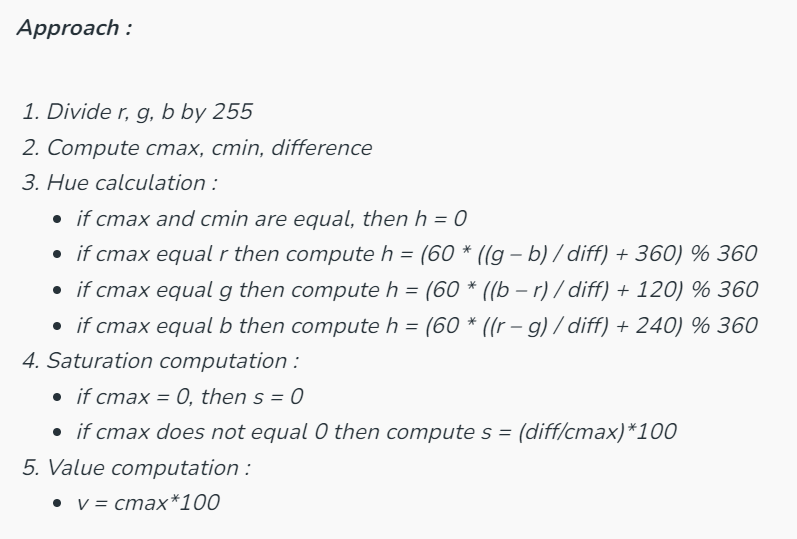
Un espacio de color es un modelo matemático que describe **la manera en que los colores pueden representarse como combinaciones de números**, típicamente en tres o más dimensiones. Los espacios de color proporcionan un marco para describir colores en términos específicos y consistentes, lo que es esencial para la reproducción precisa de los colores en diferentes dispositivos y contextos.

5)

RGB to HSV

<https://colordesigner.io/convert/rgbtohsv>

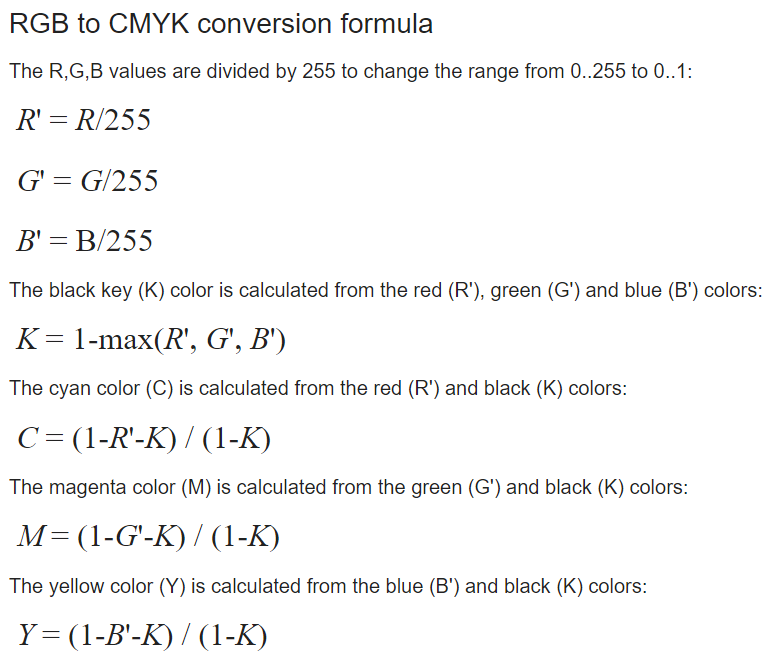
GeeksforGeeks



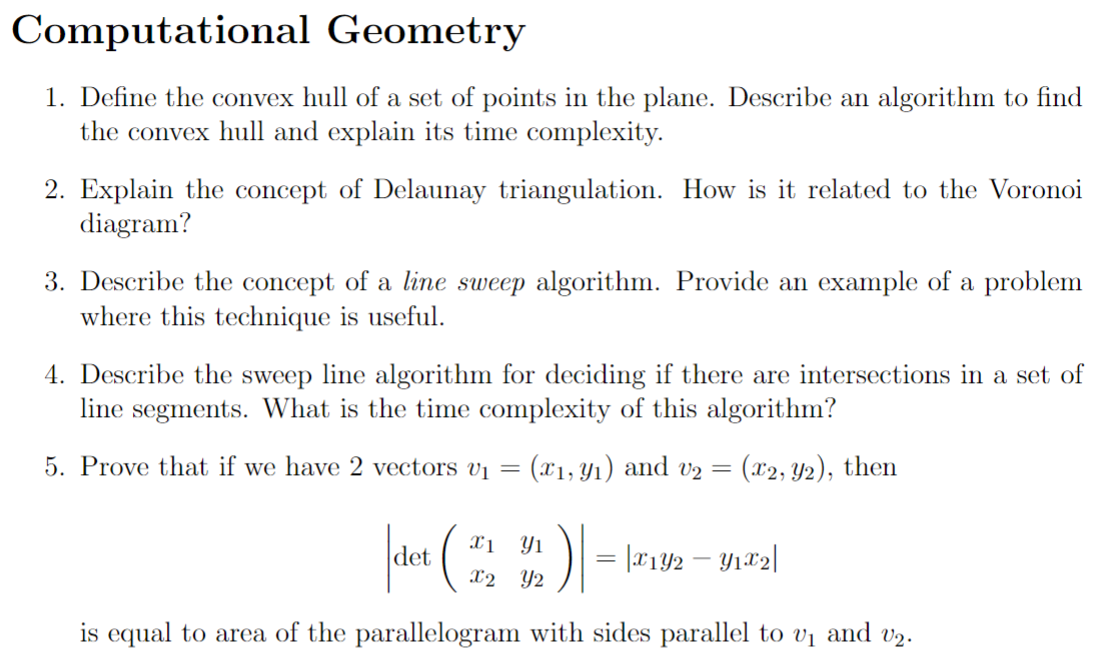
(120.0, 75.54, 54.51)

RGB to CMYK

<https://colordesigner.io/convert/rgbtocmyk>



(76, 0, 76, 45)



1)

Polígono convexo más pequeño tal que contenga a todos los puntos del conjunto.

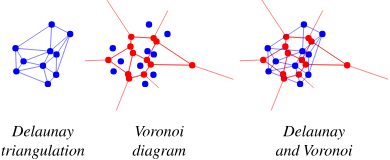
Graham scan

* Encontramos un punto inicial p0 con el menor valor valor y
* Ordenamos los puntos en base al ángulo polar con respecto a p0.
* De forma iterativa, agregamos los puntos en un stack. Los puntos son agregados solo si hacen un giro en sentido antihorario con respecto a los dos puntos anteriores del stack. Si hacen un giro en sentido horario se retira del stack.
* El resultado final se encontrará en el stack.

2)

La triangulación de Delaunay: triangulación de un polígono tal que los circun-círculos de cada triángulo no contiene ningún otro punto del conjunto.

El diagrama de Voronoi corresponde al grafo dual de la triangulación de Delaunay.



3)

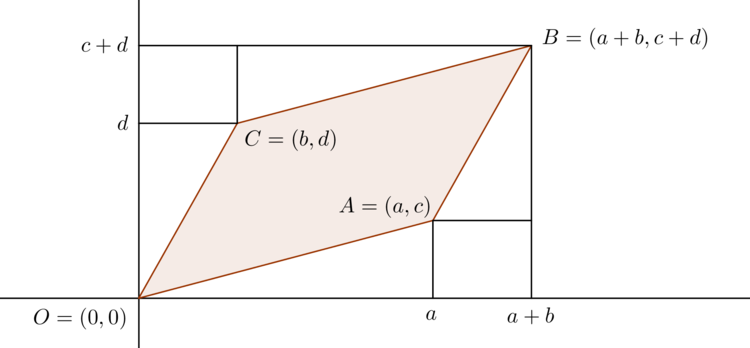
Idea general: mover una línea a través del plano e ir procesando eventos a medida que la línea pase por ellos.

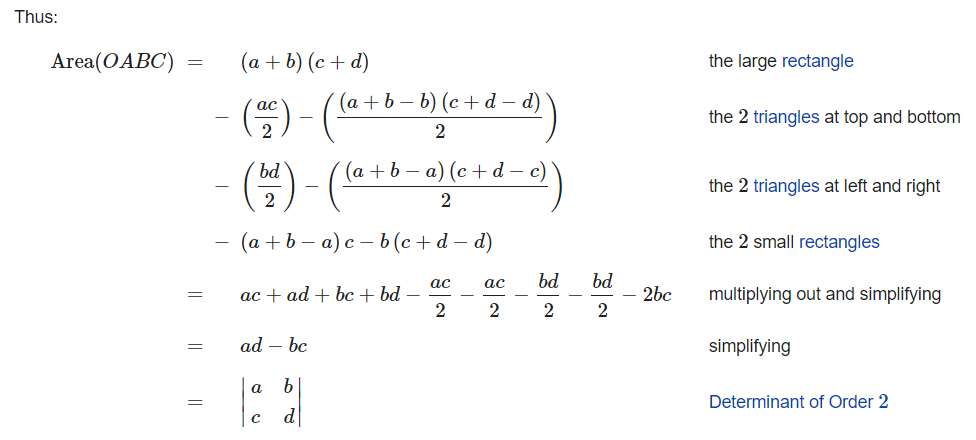
Este paradigma provee un algoritmo eficiente para encontrar de un conjunto de puntos el par más cercano.

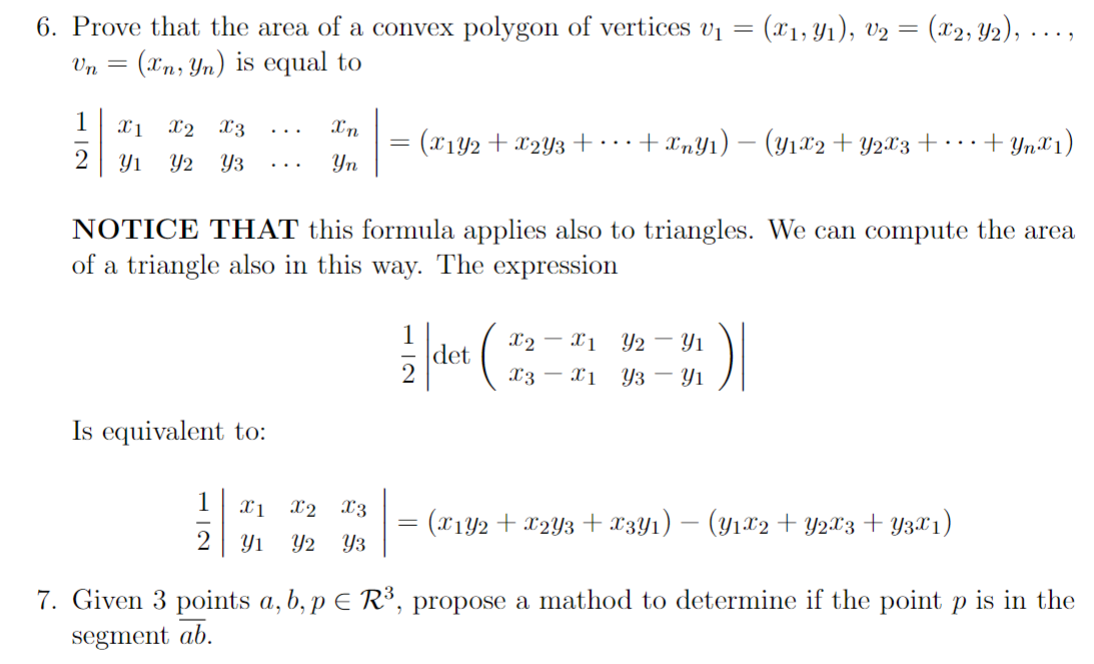
<https://www.geeksforgeeks.org/closest-pair-of-points-using-sweep-line-algorithm/>

4)

5)







6)

7)

Primero determinar si

min(a.x, b.x) <= p.x <= max(a.x, b.x) and.

Verificar si los segmentos ap y pb son colineares.

Si estas 2 condiciones se cumplen, el punto p está en el segmento ab

class Point:

def \_\_init\_\_(self, x, y):

self.x = x

self.y = y

def orientation(p : Point, q : Point, r : Point):

val = (q.y - p.y) \* (r.x - q.x) - (q.x - p.x) \* (r.y - q.y)

if val == 0:

return 0 # collinear

elif val > 0:

return 1 # clockwise

else:

return 2 # counterclockwise

def inSegment(a : Point, b : Point, p : Point):

return (

min(a.x, b.x) <= p.x and

p.x <= max(a.x, b.x) and

min(a.y, b.y) <= p.y and

p.y <= max(a.y, b.y)

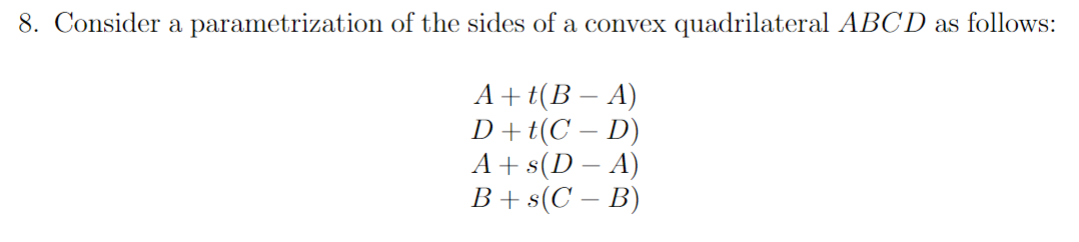
) and orientation(a, p, b) == 0

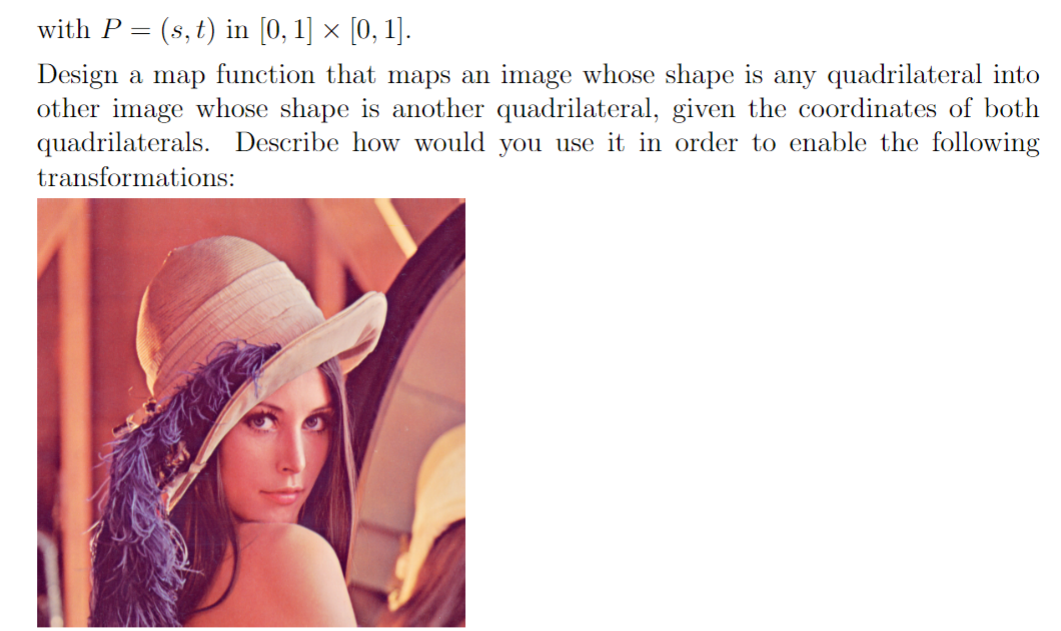
a = Point(-1, -1)

b = Point(2, 2)

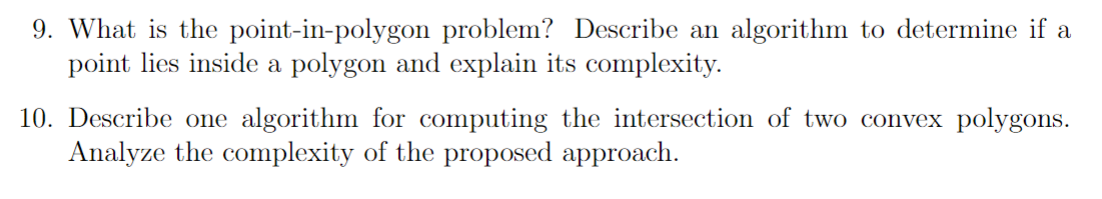
p = Point(0.5, 0.5)

print(inSegment(a, b, p))









9)

El problema pregunta si un punto dado se encuentra dentro, fuera o sobre el contorno de un polígono.

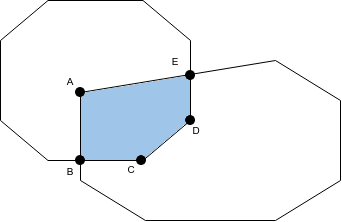
* Ray casting algorithm: cuenta la cantidad de intersecciones del polígono y un rayo imaginario con origen en el punto. En base a ello toma una decisión.

<https://en.wikipedia.org/wiki/Point_in_polygon>

10)

* Determinar los puntos de intersección del polígono

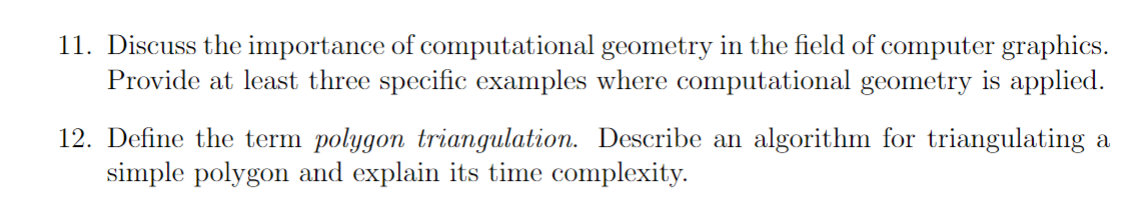
<https://autowarefoundation.gitlab.io/autoware.auto/AutowareAuto/convex-polygon-intersection-2d.html>



Complejidad: O(PQ)

P: number of vertices of the first intersection

Q: number of vertices of the second intersection



11)

Geometry simplification para la comprensión de objetos geométricos.

Detección de colisiones para simulación de físicas realistas.

Ray tracing, se utilizan algoritmos de geometría computacional para trazar las trayectorias de los rayos de luz cuando interactúan con los objetos en una escena, produciendo luces y sombras muy realistas.

12)

Triangulación del polígono:

Subdivisión del polígono en triángulos.