UAEM | Universidad Autónoma del Estado de México

Universidad Autónoma del Estado de México

Centro Universitario UAEM Zumpango

Ingeniería en Computación

Graficacion Computacional

Alumno: Diego Argel Navarrete Godines

Profesora: Hazem Álvarez Rodríguez

Fecha: 07 de Octubre del 2024

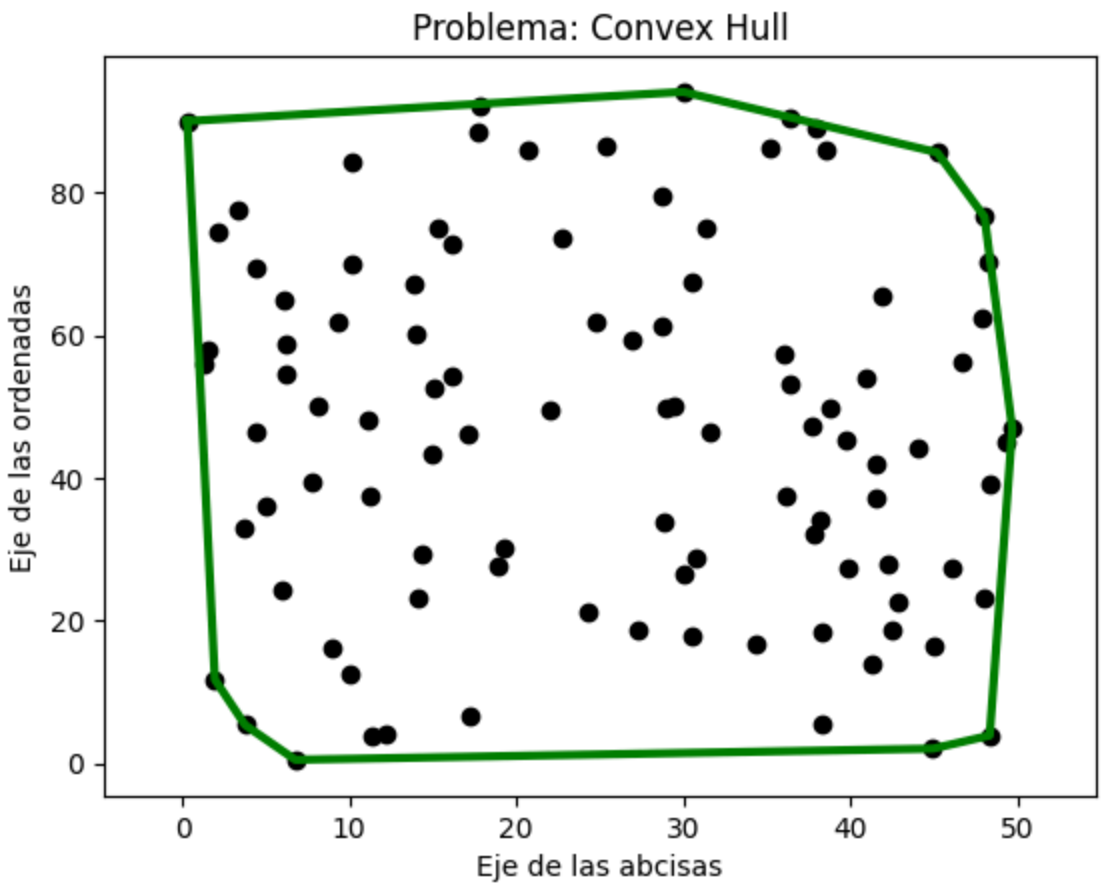
Descripcion: Algoritmo Envolvente Convexo

Este código genera un conjunto de puntos aleatorios en segunda dimension y calcula el "convex hull" (envolvente convexa) de estos puntos de manera que el polígono más pequeño que puede encerrar todos los puntos.

1. Se generan los puntos aleatorios en un rango determinado.
2. Se usa la funcion convex_hull() para calcula la envolvente convexa dividiendo el conjunto en dos partes, superior e inferior.
3. Usando la función turn_right() que utiliza el determinante para mantener la orientación correcta del polígono.
4. La función graph() grafica los puntos y el polígono resultante usando matplotlib.

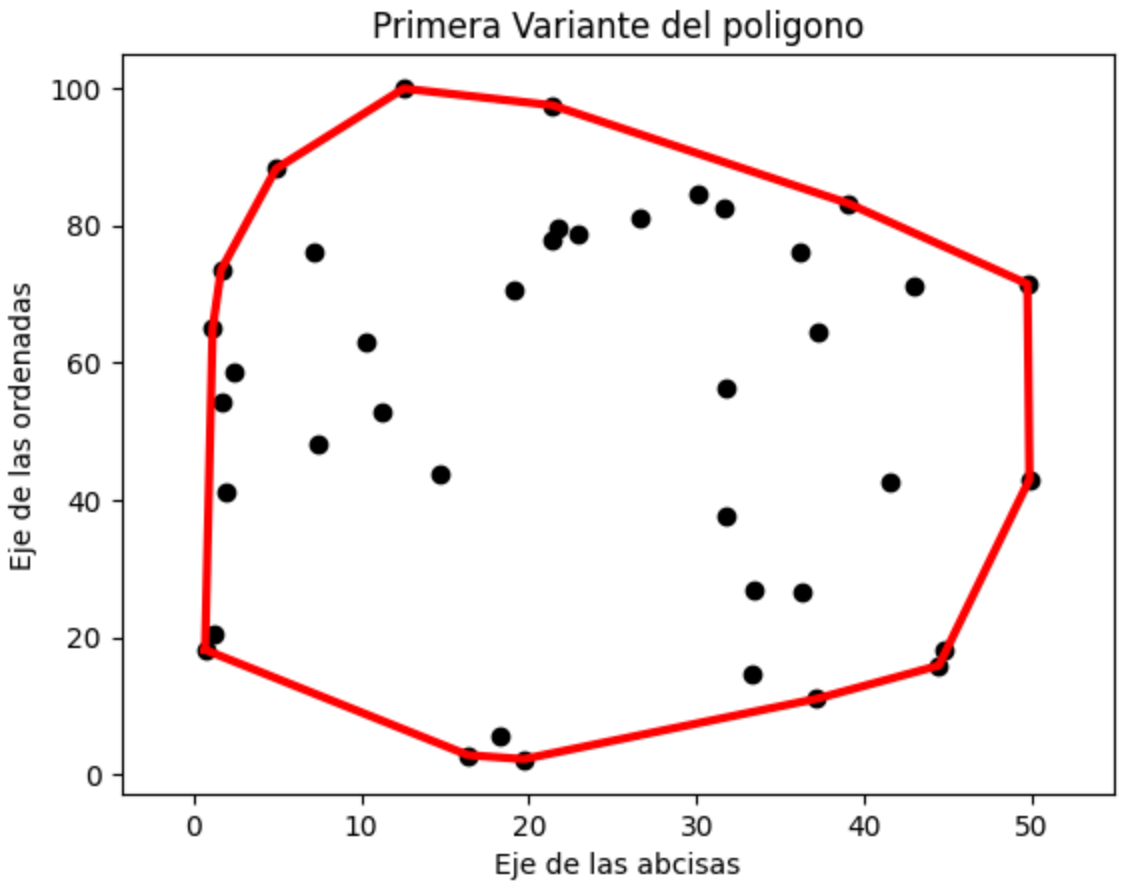
In [3]:
```
import random as rand
import matplotlib.pyplot as plt

# ------------------- FUNCIONES ------------------#
def turn_right():
    array = [coord_points[0], coord_points[1]]
    for i in range(2, len(coord_points)):
        array.append(coord_points[i])
        while len(array) > 2 and np.linalg.det([array[-3], array[-2], array[-1]])>0:
            array.pop(-2)
    return array

def convex_hull():
    coord_points.sort()
    l_upper = turn_right()
    coord_points.reverse()
```

```
    l_lower = turn_right()
    l = l_upper + l_lower
    return l

def graph(convex_pol, coord_points):
    # Acomodando listas adecuadas para graficar en matplotlib
    x_points = [i[0] for i in coord_points]
    y_points = [i[1] for i in coord_points]

    x_polygon = [i[0] for i in convex_pol]
    y_polygon = [i[1] for i in convex_pol]

    # Definiendo limites extremos de la grafica
    x_lim_der = max(x_points) + 5
    y_lim_sup = max(y_points) + 5
    x_lim_izq = min(x_points) - 5
    y_lim_inf = min(y_points) - 5

    plt.xlim(x_lim_izq, x_lim_der)
    plt.ylim(y_lim_inf, y_lim_sup)

    # Graficación
    plt.title('Problema: Convex Hull')
    plt.xlabel('Eje de las abcisas')
    plt.ylabel('Eje de las ordenadas')
    plt.plot(x_points, y_points, 'ko')
    plt.plot(x_polygon, y_polygon, 'g-', linewidth=3.0)
    plt.show()

# Generar los puntos
num_points = 100
coord_points = []

for i in range(num_points):
    coord_points.append([rand.uniform(0, 50), rand.uniform(0, 100), 1.0])

# Obtener el convex hull
convex_pol = convex_hull()

# Graficar el resultado
graph(convex_pol, coord_points)
```



In [5]:
```python
import random as rand
import matplotlib.pyplot as plt

# ------------------- FUNCIONES -------------------#
def turn_right():
    array = [coord_points[0], coord_points[1]]
    for i in range(2, len(coord_points)):
        array.append(coord_points[i])
        while len(array) > 2 and np.linalg.det([array[-3], array[-2], array[-1]])>0:
            array.pop(-2)
    return array

def convex_hull():
    coord_points.sort()
    l_upper = turn_right()
    coord_points.reverse()
    l_lower = turn_right()
    l = l_upper + l_lower
    return l

def graph(convex_pol, coord_points):
    # Acomodando listas adecuadas para graficar en matplotlib
    x_points = [i[0] for i in coord_points]
    y_points = [i[1] for i in coord_points]

    x_polygon = [i[0] for i in convex_pol]
    y_polygon = [i[1] for i in convex_pol]

    # Definiendo limites extremos de la grafica
    x_lim_der = max(x_points) + 5
    y_lim_sup = max(y_points) + 5
    x_lim_izq = min(x_points) - 5
```

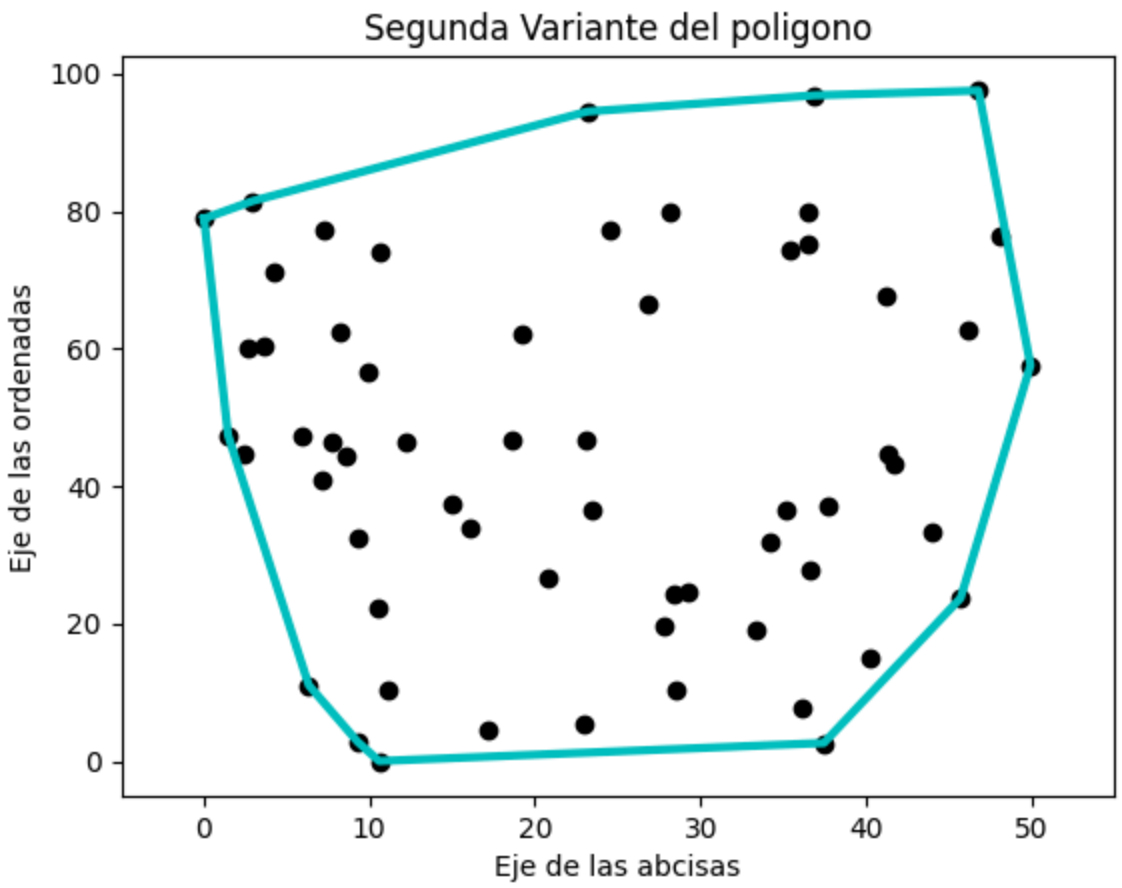
```
    y_lim_inf = min(y_points) - 5

    plt.xlim(x_lim_izq, x_lim_der)
    plt.ylim(y_lim_inf, y_lim_sup)

    # Graficación
    plt.title('Primera Variante del poligono')
    plt.xlabel('Eje de las abcisas')
    plt.ylabel('Eje de las ordenadas')
    plt.plot(x_points, y_points, 'ko')
    plt.plot(x_polygon, y_polygon, 'r-', linewidth=3.0)
    plt.show()

# Generar los puntos
num_points = 40
coord_points = []

for i in range(num_points):
    coord_points.append([rand.uniform(0, 50), rand.uniform(0, 100), 1.0])

# Obtener el convex hull
convex_pol = convex_hull()

# Graficar el resultado
graph(convex_pol, coord_points)
```



In [7]:

```
import random as rand
import matplotlib.pyplot as plt

# ------------------- FUNCIONES -------------------#
def turn_right():
    array = [coord_points[0], coord_points[1]]
```

```
    for i in range(2, len(coord_points)):
        array.append(coord_points[i])
        while len(array) > 2 and np.linalg.det([array[-3], array[-2], array[-1]])>0:
            array.pop(-2)
    return array

def convex_hull():
    coord_points.sort()
    l_upper = turn_right()
    coord_points.reverse()
    l_lower = turn_right()
    l = l_upper + l_lower
    return l

def graph(convex_pol, coord_points):
    # Acomodando listas adecuadas para graficar en matplotlib
    x_points = [i[0] for i in coord_points]
    y_points = [i[1] for i in coord_points]

    x_polygon = [i[0] for i in convex_pol]
    y_polygon = [i[1] for i in convex_pol]

    # Definiendo limites extremos de la grafica
    x_lim_der = max(x_points) + 5
    y_lim_sup = max(y_points) + 5
    x_lim_izq = min(x_points) - 5
    y_lim_inf = min(y_points) - 5

    plt.xlim(x_lim_izq, x_lim_der)
    plt.ylim(y_lim_inf, y_lim_sup)

    # Graficación
    plt.title('Segunda Variante del poligono')
    plt.xlabel('Eje de las abcisas')
    plt.ylabel('Eje de las ordenadas')
    plt.plot(x_points, y_points, 'ko')
    plt.plot(x_polygon, y_polygon, 'c-', linewidth=3.0)
    plt.show()

# Generar los puntos
num_points = 60
coord_points = []

for i in range(num_points):
    coord_points.append([rand.uniform(0, 50), rand.uniform(0, 100), 1.0])

# Obtener el convex hull
convex_pol = convex_hull()

# Graficar el resultado
graph(convex_pol, coord_points)
```



In [9]:

```
import random as rand
import matplotlib.pyplot as plt

# ------------------- FUNCIONES -------------------#
def turn_right():
    array = [coord_points[0], coord_points[1]]
    for i in range(2, len(coord_points)):
        array.append(coord_points[i])
        while len(array) > 2 and np.linalg.det([array[-3], array[-2], array[-1]])>0:
            array.pop(-2)
    return array

def convex_hull():
    coord_points.sort()
    l_upper = turn_right()
    coord_points.reverse()
    l_lower = turn_right()
    l = l_upper + l_lower
    return l

def graph(convex_pol, coord_points):
    # Acomodando listas adecuadas para graficar en matplotlib
    x_points = [i[0] for i in coord_points]
    y_points = [i[1] for i in coord_points]

    x_polygon = [i[0] for i in convex_pol]
    y_polygon = [i[1] for i in convex_pol]

    # Definiendo limites extremos de la grafica
    x_lim_der = max(x_points) + 5
    y_lim_sup = max(y_points) + 5
    x_lim_izq = min(x_points) - 5
```

```
    y_lim_inf = min(y_points) - 5

    plt.xlim(x_lim_izq, x_lim_der)
    plt.ylim(y_lim_inf, y_lim_sup)

    # Graficación
    plt.title('Tercera variante del poligono')
    plt.xlabel('Eje de las abcisas')
    plt.ylabel('Eje de las ordenadas')
    plt.plot(x_points, y_points, 'ko')
    plt.plot(x_polygon, y_polygon, 'm-', linewidth=3.0)
    plt.show()

# Generar los puntos
num_points = 80
coord_points = []

for i in range(num_points):
    coord_points.append([rand.uniform(0, 50), rand.uniform(0, 100), 1.0])

# Obtener el convex hull
convex_pol = convex_hull()

# Graficar el resultado
graph(convex_pol, coord_points)
```