Instituto Politécnico Nacional

Escuela Superior de Cómputo

Unidad académica: Análisis y Diseño de Algoritmos

Actividad:

“Algoritmos de la jarra de agua”

Equipo:

Solares Velasco Arturo Misael

Solis Lugo Mayra

Grupo: 3CV1

Profesor: García Floriano Andrés

Fecha:

21 de Mayo de 2024

Reporte de Algoritmos en `capcha.py` y `ecuaciones.py`

Algoritmo en `capcha.py`

1. Generación de Coordenadas de un Triángulo Equilátero:

- **Función:** `generate_triangle_coordinates()`
- **Descripción:** Devuelve los vértices de un triángulo equilátero cuyos vértices están en (0,0)(0, 0)(0,0), (1,0)(1, 0)(1,0) y (0.5,3/2)(0.5, \sqrt{3}/2)(0.5,3/2).

código de la functión:

```python
def generate_triangle_coordinates():
    # Vértices del triángulo equilátero
    V1 = (0, 0)
    V2 = (1, 0)
    V3 = (0.5, math.sqrt(3) / 2)
    return V1, V2, V3
```

2. Cálculo del Punto Medio:

- **Función:** `midpoint(P, V)`
- **Descripción:** Calcula el punto medio entre dos puntos PPP y VVV.

Código de la función:

```python
def midpoint(P, V):
    return ((P[0] + V[0]) / 2, (P[1] + V[1]) / 2)
```

3. Generación de Puntos para el CAPTCHA:

- **Función:** `generate_captcha_points(iterations=100)`
- **Descripción:**
    - Genera los vértices del triángulo equilátero.
    - Inicializa un punto aleatorio PPP.
    - En cada iteración, selecciona aleatoriamente uno de los vértices del triángulo y calcula el punto medio entre el punto actual PPP y el vértice seleccionado.
    - Agrega el nuevo punto a la lista de puntos.

Código de la función:

```python
def generate_captcha_points(iterations=100):
    V1, V2, V3 = generate_triangle_coordinates()
    vertices = [V1, V2, V3]

    # Punto inicial aleatorio
    P = (random.uniform(0, 1), random.uniform(0, 1))
    points = [P]

    for _ in range(iterations):
        V = random.choice(vertices)
        P = midpoint(P, V)
        points.append(P)

    return points
```

4. Guardado de Puntos en un Archivo:

- **Función:** `save_points_to_file(points, filename="captcha_points.txt")`
- **Descripción:** Guarda la lista de puntos generados en un archivo de texto.

Código de la función:

```python
def save_points_to_file(points, filename="captcha_points.
txt"):
    with open(filename, "w") as file:
        for point in points:
            file.write(f"{point[0]} {point[1]}\n")
```

5. Graficación de Puntos:

- **Función:** `plot_points(filename="captcha_points.txt")`
- **Descripción:**
    - Lee los puntos desde un archivo.
    - Extrae las coordenadas xxx e yyy.
    - Grafica los puntos utilizando `matplotlib`.

```python
def plot_points(filename="captcha_points.txt"):
    points = []
    with open(filename, "r") as file:
        for line in file:
            x, y = map(float, line.split())
            points.append((x, y))

    x_vals = [p[0] for p in points]
    y_vals = [p[1] for p in points]

    plt.scatter(x_vals, y_vals, s=1)
    plt.title("MODELO CAPTCHA")
    plt.xlabel("X")
    plt.ylabel("Y")
    plt.show()
```

Secuencia de Ejecución:

- Genera 1000 puntos usando `generate_captcha_points`.
- Guarda los puntos en un archivo.
- Lee los puntos del archivo y los grafica.

Algoritmo en `ecuaciones.py`

1. Verificación de Solución Exacta:

- **Función:** `es_solucion(A, x, b, tol=1e-6)`
- **Descripción:** Comprueba si A⋅x≈bA \cdot x \approx bA⋅x≈b con una tolerancia especificada.

Código de la función

```python
def es_solucion(A, x, b, tol=1e-6):
    return np.allclose(np.dot(A, x), b, atol=tol)
```

2. Algoritmo de Las Vegas para Resolver Sistemas de Ecuaciones:

- **Función:** `las_vegas_resolver(A, b, max_iteraciones=100)`
- **Descripción:**
    - Inicializa la mejor solución y el mejor error.
    - En cada iteración, genera un vector aleatorio xxx con valores enteros entre -5 y 5.
    - Calcula el error entre A⋅xA \cdot xA⋅x y bbb.

- Actualiza la mejor solución si el error actual es menor que el mejor error registrado.
- Si se encuentra una solución exacta, se retorna inmediatamente.
- Si no se encuentra una solución exacta después de las iteraciones, se retorna la mejor aproximación.

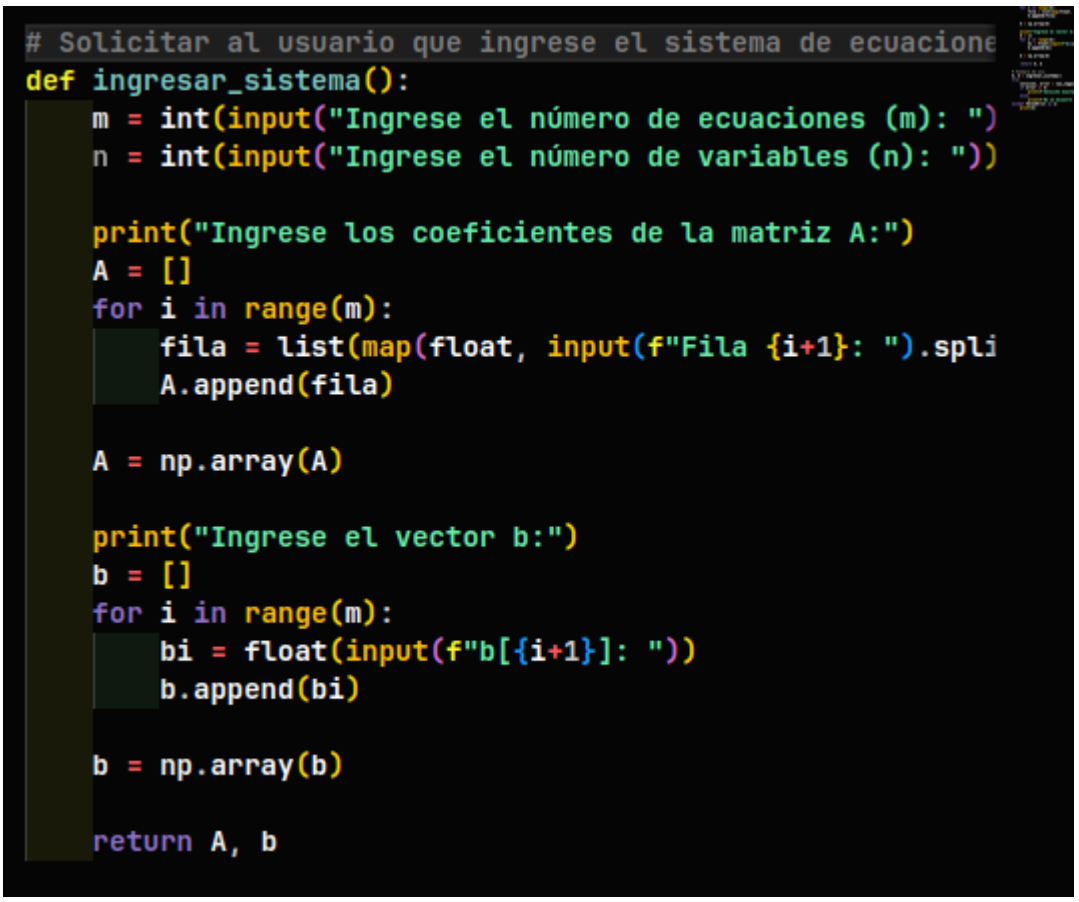
Código de la función:

```python
def las_vegas_resolver(A, b, max_iteraciones=100):
    m, n = A.shape
    mejor_x = None
    mejor_error = float('inf')

    for _ in range(max_iteraciones):
        x = np.random.randint(-5, 6, n)  # Generar números
        error = np.linalg.norm(np.dot(A, x) - b)

        if error < mejor_error:
            mejor_error = error
            mejor_x = x

        if es_solucion(A, x, b):
            return x, error

    # Si no se encuentra una solución exacta, retornar la
    return mejor_x, mejor_error
```

3. Ingreso de Sistema de Ecuaciones:

- **Función:** `ingresar_sistema()`
- **Descripción:**
  - Solicita al usuario ingresar el número de ecuaciones y variables.
  - Recoge los coeficientes de la matriz AAA y el vector bbb.

Código de la función:



```python
# Solicitar al usuario que ingrese el sistema de ecuacione
def ingresar_sistema():
    m = int(input("Ingrese el número de ecuaciones (m): ")
    n = int(input("Ingrese el número de variables (n): "))

    print("Ingrese los coeficientes de la matriz A:")
    A = []
    for i in range(m):
        fila = list(map(float, input(f"Fila {i+1}: ").spli
        A.append(fila)

    A = np.array(A)

    print("Ingrese el vector b:")
    b = []
    for i in range(m):
        bi = float(input(f"b[{i+1}]: "))
        b.append(bi)

    b = np.array(b)

    return A, b
```

Secuencia de Ejecución:

- El usuario ingresa el sistema de ecuaciones.
- Intenta resolver el sistema usando el algoritmo de Las Vegas.
    - Si se encuentra una solución exacta, se imprime.
    - Si no, se imprime la mejor aproximación y el error asociado.

Resumen

- `capcha.py` genera puntos que forman un patrón al calcular puntos medios sucesivos en un triángulo equilátero y los grafica.
- `ecuaciones.py` intenta resolver sistemas de ecuaciones lineales utilizando un método aleatorio (Las Vegas), que busca una solución exacta o la mejor aproximación posible.