Algoritmo de Floyd's Tortoise

1st Juan Esteban Pinzon
Hugo Alejandro Latorre
José David Ramírez Beltrán
Amir Rodriguez Mejia
Fundación Universitaria Konrad Lorenz Bogotá, Colombia

***Abstract*—Este código Python implementa el "Algoritmo de Floyd" para encontrar números duplicados en una secuencia numérica. Utiliza dos punteros para detectar un ciclo en la secuencia y, posteriormente, encuentra el número duplicado y sus ubicaciones. Esta implementación se proporciona como una API web mediante FastAPI, permitiendo a los usuarios enviar una lista de números y recibir el número duplicado y sus índices en respuesta. El algoritmo es útil en la gestión de datos para identificar duplicados de manera eficiente en conjuntos numéricos extensos.**

***Index Terms*—Optimización, recursos, lógica, complejidad, análisis**

I. Introducción

Algoritmo de la Tortuga y la Liebre" para encontrar números duplicados en una secuencia de números. Este algoritmo es especialmente útil para identificar de manera eficiente números duplicados en una lista, incluso cuando la secuencia es grande y no se conoce de antemano el número duplicado ni su posición en la lista. La implementación utiliza la biblioteca FastAPI para crear una API web que permite a los usuarios enviar una lista de números y, como respuesta, proporciona el número duplicado y los índices en los que se encuentra. El algoritmo se basa en el concepto de detección de ciclos en una secuencia y se divide en dos fases: detección y búsqueda del número duplicado. La capacidad de encontrar duplicados en una lista es esencial en diversas aplicaciones, como la gestión de datos y la detección de errores en conjuntos de información. Este código proporciona una solución efectiva para tal propósito, ofreciendo una forma rápida y eficiente de encontrar duplicados en una secuencia de números.

Claro, aquí tienes el análisis del código en formato LaTeX:
"'latex

II. Análisis del Código

A. Definición de la API FastAPI

El código comienza con la importación de las bibliotecas necesarias, incluyendo `FastAPI`, `BaseModel` y `List`. Luego, se crea una instancia de la aplicación FastAPI llamada `app`, que servirá como el punto de entrada para la API.

B. Modelo de Datos `NumeroInput`

Se define un modelo de datos llamado `NumeroInput`, que hereda de `BaseModel`. Este modelo tiene un atributo llamado `numeros`, que es una lista de enteros. Se utiliza para validar los datos de entrada en la ruta `/floyd`.

C. Ruta Raíz `/`

Se define una ruta raíz `/` con el método `GET` que devuelve un mensaje de bienvenida cuando se accede al punto de entrada de la aplicación. Esta ruta no requiere datos de entrada y simplemente proporciona información sobre el servicio.

D. Ruta `/floyd` para Encontrar Números Duplicados

La ruta `/floyd` se define con el método `POST` y se utiliza para encontrar números duplicados en una secuencia de números. Aquí se implementa el "Algoritmo de Floyd" o el "Algoritmo de la Tortuga y la Liebre."

- Se extraen los números de entrada de la solicitud y se inicializan dos punteros, la "tortuga" y la "liebre," al primer número de la secuencia.

- Se ejecuta un bucle para avanzar los punteros a través de la secuencia. La "tortuga" avanza un paso a la vez, mientras que la "liebre" avanza dos pasos a la vez. El bucle continúa hasta que ambos punteros se encuentran en un número duplicado.

- Después de encontrar el punto de encuentro, se inicia una fase de búsqueda. Se utilizan dos punteros adicionales para encontrar el número duplicado. Uno de los punteros comienza desde el inicio de la secuencia y el otro desde el punto de encuentro. Ambos avanzan un paso a la vez hasta que se vuelven a encontrar. El punto de encuentro en esta fase es el número duplicado.

- Se recorre la secuencia original para encontrar todos los índices donde se encuentra el número duplicado.

E. Respuesta

Se crea un diccionario `resultado` que contiene el número duplicado y una lista de índices donde se encuentra. Esta información se envía como una respuesta en formato JSON.

En resumen, el código implementa una API web que permite a los usuarios encontrar números duplicados en una secuencia numérica utilizando el Algoritmo de Floyd. Esta API es útil en situaciones en las que se necesita detectar duplicados en datos numéricos y proporciona una respuesta clara y estructurada con el número duplicado y sus ubicaciones. "'

Puedes copiar y pegar este análisis en tu documento LaTeX según sea necesario. Asegúrate de ajustar cualquier formato adicional o diseño de acuerdo a tus necesidades específicas.

main.py

```python
from fastapi import FastAPI
from pydantic import BaseModel
from typing import List

app = FastAPI()

1 usage
class NumeroInput(BaseModel):
    numeros: List[int]

@app.get("/")
def root():
    return {
        "Servicio": "Estructuras de datos"
    }

@app.post("/floyd")
def encontrar_duplicado(input_data: NumeroInput):
    numeros = input_data.numeros

    tortuga = numeros[0]
    liebre = numeros[0]
```

Fig. 1. Definición de la API FastAPI

```python
    while True:
        tortuga = numeros[tortuga]
        liebre = numeros[numeros[liebre]]
        if tortuga == liebre:
            break

    # Fase de búsqueda
    puntero1 = numeros[0]
    puntero2 = tortuga
    while puntero1 != puntero2:
        puntero1 = numeros[puntero1]
        puntero2 = numeros[puntero2]

    # Encontrar el número que se repite y sus índices
    numero_repetido = puntero1
    indices_repetidos = []
    for i, num in enumerate(numeros):
        if num == numero_repetido:
            indices_repetidos.append(i)
```

Fig. 2. Se realiza el ciclo.

III. Conclusiones

1) **E**ste código demuestra una implementación eficiente del ”Algoritmo de Floyd” para encontrar números duplicados en una secuencia numérica. La eficiencia radica en la complejidad temporal del algoritmo, que es lineal (O(n)), donde ”n” representa la longitud de la secuencia de números. Esta característica lo hace particularmente adecuado para manejar grandes conjuntos de datos de manera rápida y efectiva.
   Además, al exponer su funcionalidad a través de una API web con FastAPI, el código proporciona una interfaz amigable para los usuarios, lo que facilita su integración en una variedad de aplicaciones y servicios. Esto amplía su utilidad y accesibilidad en tareas de detección de duplicados en datos numéricos, lo que, en última instancia, contribuye a la eficiencia y productividad en el procesamiento de datos a gran escala.