Tarea GitHub Algoritmos

Juan Pablo Sepulveda Morales

5 de mayo de 2025

Introducción

Este informe presenta un fragmento de código Python obtenido de Internet, utilizado con fines educativos para comprender el funcionamiento de una estructura de datos fundamental: la **cola de prioridad**. El código ha sido modificado para incluir el uso de clases en Python, facilitando así su comprensión desde una perspectiva orientada a objetos.

Concepto Base: Cola de Prioridad

Una cola de prioridad es una estructura de datos abstracta donde cada elemento tiene asociada una prioridad. A diferencia de una cola tradicional (FIFO), en una cola de prioridad el elemento con mayor prioridad es atendido primero.

Implementación en Python

En este ejemplo se implementa una cola de prioridad usando clases en Python y el módulo estándar `heapq`, que ofrece una implementación eficiente de un **heap binario mínimo**.

Listing 1: Implementación de cola de prioridad en Python

```
import heapq

class Tarea:
    def __init__(self, prioridad, descripcion):
        self.prioridad = prioridad
        self.descripcion = descripcion

    def __lt__(self, other):
        return self.prioridad < other.prioridad

    def __repr__(self):
        return f"{self.descripcion} (prioridad: {self.prioridad})"

class GestorDeTareas:
    def __init__(self):
```

```
        self._tareas = []

    def agregar_tarea(self, tarea):
        heapq.heappush(self._tareas, tarea)

    def obtener_tarea_mas_prioritaria(self):
        if self._tareas:
            return heapq.heappop(self._tareas)
        return None

gestor = GestorDeTareas()
gestor.agregar_tarea(Tarea(2, 'Hacer␣compras'))
gestor.agregar_tarea(Tarea(1, 'Estudiar␣Python'))
gestor.agregar_tarea(Tarea(3, 'Llamar␣a␣mam '))

while True:
    tarea = gestor.obtener_tarea_mas_prioritaria()
    if tarea is None:
        break
    print(tarea)
```

Conceptos Utilizados

- **Programación orientada a objetos**: uso de clases y métodos.
- **Comparadores personalizados**: mediante el método especial `__lt__`.
- **Heap binario**: estructura usada internamente por el módulo `heapq`.
- **Eficiencia algorítmica**: el acceso al elemento con mayor prioridad es logarítmico.

Referencia Histórica

La implementación moderna de colas de prioridad se basa en el **heap binario**, una estructura formalizada por **J. W. J. Williams** en 1964 con el algoritmo *Heapsort*:

> J. W. J. Williams, "Algorithm 232: Heapsort", *Communications of the ACM*, 1964.

Orígenes del Concepto

Las colas de prioridad no fueron inventadas por una única persona. Su origen proviene de la evolución de la teoría de estructuras de datos, desarrollada ampliamente en las décadas de **1960 y 1970**. Estas estructuras son fundamentales para algoritmos de búsqueda, planificación y simulación.

Conclusión

Este ejemplo sencillo en Python muestra cómo implementar una cola de prioridad usando clases, combinando teoría de algoritmos con técnicas modernas de programación. Es una excelente forma de aprender estructuras de datos aplicadas a problemas reales.