

Laboratorio 9 Estructura de datos y algoritmos

Integrantes:

Simón Peña Alarcón - 202512907- s.penaa2@uniandes.edu.co

Juan Sebastian Chacón Ochoa-202513196- j.chacono@uniandes.edu.co

Manuel Santiago Figueroa-202511697- m.figueroag@uniandes.edu.co

- a) ¿Qué estructura de datos subyacente se utiliza comúnmente para implementar una cola de prioridad en Python?
- b) ¿Cuál es la diferencia entre una cola FIFO tradicional y una cola de prioridad?
- c) ¿Qué módulo proporciona Python para trabajar fácilmente con colas de prioridad?
- d) ¿Qué ventajas tiene el uso de una cola de prioridad sobre una lista ordenada manualmente?
- e) Si dos elementos tienen la misma prioridad, ¿cómo decide la cola cuál atender primero?
- f) ¿Qué situaciones del mundo real se pueden modelar con colas de prioridad? Mencione al menos dos.
- g) En un sistema de atención médica, ¿cómo se puede usar una cola de prioridad para organizar a los pacientes?
- h) ¿Cómo afectaría al rendimiento usar una lista simple en lugar de una estructura especializada como heap_pq para manejar prioridades?
- i) ¿Qué complejidad temporal tiene la inserción y eliminación/atención en una cola de prioridad basada en heap?

Desarrollo:

- a. La estructura que se usa para implementar una cola de prioridad es el heap, más específicamente un heap binario. Este tipo de estructura permite organizar los elementos de manera que siempre sea fácil acceder al que tiene la mejor prioridad.
- b. La diferencia es que en una cola que es first in, first out, los elementos se atienden en el mismo orden en el que llegan, sin importar su importancia. En cambio, en una cola de prioridad, el elemento que se atiende primero es el que tiene mayor prioridad, aunque no

haya llegado primero. Entonces la cola de prioridad ordena por importancia, no por tiempo de llegada.

c. Python nos da la posibilidad de usar el heap, que permite trabajar con colas de prioridad usando funciones ya listas para insertar, sacar y mantener la estructura como un heap. heap utiliza listas internas y maneja la propiedad del heap automáticamente.

d. La ventaja principal es que una cola de prioridad mantiene el orden por prioridad de forma automática. Si uno usa una lista normal y quiere mantenerla ordenada manualmente, toca ordenar o buscar la posición correcta cada vez que se inserta algo, lo cual es más lento y complicado. Con el heap, las operaciones de inserción y eliminación ya están optimizadas para mantener la prioridad sin tener que reordenarlo todo.

e. Si dos elementos tienen la misma prioridad, la cola normalmente los atiende en el orden en el que fueron insertados. Esto se conoce como estabilidad y asegura que no se mezclen cuando la prioridad es igual, aunque dependiendo de la implementación exacta puede variar un poco.

f. Las colas de prioridad se usan en situaciones donde no solo importa el orden de llegada, sino la importancia de cada elemento. Un ejemplo claro es en una sala de urgencias: los pacientes son atendidos según la gravedad de su condición así uno que llegó después puede pasar primero si su estado es más crítico.

g. En un hospital, cada paciente puede recibir un valor de prioridad según su estado. Por ejemplo, alguien con un paro cardíaco tiene prioridad más alta que alguien con dolor de cabeza. La cola de prioridad organiza a los pacientes de forma que los casos más graves pasen primero, y los demás esperan. Esto permite atender de forma más justa en términos de urgencia, no solo por orden de llegada.

h. Si se usa una lista simple cada vez que se inserta un elemento habría que recorrer la lista completa o reordenarla, lo cual es más lento. Esto puede llevar tiempo $O(n)$ o más si se ordena. En cambio el heap está hecho para que mantener la prioridad cueste menos tiempo entonces que el rendimiento es mucho mejor cuando hay muchos elementos.

i. En una cola de prioridad basada en heap, tanto insertar como remover el elemento con mejor prioridad cuesta $O(\log n)$ en tiempo. Ya que el heap es un árbol casi completo y solo se necesita recorrer su altura para reorganizarlo.