27 DE JULIO DE 2020

TRABAJO PRÁCTICO N°2

ALGORITMOS Y PROGRAMACIÓN II – CURSO BUCHWALD

GRUPO 23:
Di Nardo Chiara A. - 105295
Martin Álvaro -
Corrector: Matías Cano

Empezamos el trabajo desde la base brindada por el curso. De esta manera, pensamos la clínica como una estructura de datos donde cada primitiva sea una funcionalidad de la clínica y cada una de estas es independiente entre sí. Esta misma está compuesta por otras estructuras internas donde se guardan los datos necesarios para su funcionalidad (doctores, pacientes y especialidades).

Al iniciar el programa se guarda en memoria la lista de doctores y de pacientes. En el caso de los doctores, utilizamos un Tda ABB para guardar los datos de cada uno en orden alfabético, siendo así para facilitar la función de la clínica *pedir informe* y cumplir con la complejidad temporal pedida.

Por otra parte, cada especialidad fue insertada en un Hash con una estructura (Sala) asociada, la cual representa la sala donde se atiende esa especialidad. La elección del Hash fue específicamente por su complejidad a la hora de obtener el dato asociado a una clave porque esto optimiza el tiempo, en el momento que el comando ingresado al programa sea *pedir turno* o *atender siguiente paciente*, a comparación de otros Tdas. Por último, los pacientes fueron guardados en un Hash con sus respectivos datos por el mismo motivo que las especialidades.

Como se mencionó anteriormente, los datos de doctores y pacientes fueron guardados en sus respectivos Tdas. Ambos con su nombre como clave y de valor, una estructura de datos que representa una Persona de la clínica. Esta estructura está compuesta por un Hash donde se guardan los datos necesarios de la persona para el funcionamiento del programa y para que conseguir los datos de la misma (sea paciente o doctor) no empeore la complejidad de las funciones.

Por otro lado, en el caso de la estructura Sala, se eligió conformarla por dos Tdas que representan las listas de espera de los pacientes dependiendo de la urgencia. La lista de espera de los turnos de urgencia fue representada por una Cola para poder encolar y desencolar pacientes en O(1) y, la de los turnos regulares, fue representada por un Heap. En este caso, la prioridad para encolar y desencolar pacientes fue dependiendo su año de ingreso a la clínica.

Complejidad de funciones de la clínica

- *Pedir turno*: en el caso de que el turno sea urgente, el programa buscará y devolverá del Hash de especialidades la Sala donde deberá atenderse el paciente. Luego, encolará el paciente en la Cola de la lista de espera de urgentes, logrando así que todas las operaciones sean O(1) y formando el resultado pedido por consigna. Luego, si la urgencia del turno es regular, se comienza de la misma manera que el otro caso, pero a la hora de encolar el paciente en su respectiva lista de espera se lo hace en la de turnos regulares. Como es necesario “ordenar” el Heap

para saber en qué posición se encontrará el paciente según su fecha de ingreso, la complejidad será O(log n) por las propiedades del Heap.

- *Atender siguiente paciente*: se busca y obtiene del ABB la estructura Persona asociada al doctor ingresado según el comando (O(log d), siendo d la cantidad total de doctores ingresados en la clínica). Para obtener la especialidad donde trabaja este, se obtiene de su ficha (Hash) el nombre de esta y se busca la Sala donde atender. En el caso de que haya pacientes urgentes, se atiende primero a ellos (se los desencola de la cola O(1)). De esta manera, la complejidad final sería O(log d). Por otra parte, si la lista de espera de urgentes está vacía y haya pacientes esperando en la lista de regulares, se desencola un paciente del Heap (O(log n) siendo n la cantidad de pacientes en espera para turnos regulares) y esto da el resultado final de **O(log d + log n))**.

- *Informe doctores*: para que este comando sea posible, modificamos el iterador interno del ABB implementado anteriormente para que se pueda iterar por rangos. De esta manera, si se pide mostrar la información de todos los doctores la complejidad sería **O(d)**, siendo d la cantidad total de doctores. En el caso de que se quiera la información de un cierto rango de doctores, la complejidad sería **O(log d)**. Para obtener la información de cada médico, fue necesario acceder a la ficha de la estructura Persona y obtener el valor de la clave *especialidad* y *pacientes atendidos*. Como se mencionó anteriormente, esto fue implementado con un Hash para obtener la mejor complejidad temporal posible y no empeorar con esta la complejidad final.