# Análisis de complejidad temporal y espacial de los algoritmos editar recordatorio y editar tarea

Análisis para editar recordatorio

Manageable reminder = hashTable.search(key); - Esta línea realiza una búsqueda en la tabla hash. Supongamos que la búsqueda toma 100 operaciones en el peor de los casos (una por cada elemento en la tabla hash).

if ( reminder != null ){...} - Esta línea realiza una comparación, que cuenta como una operación.

Dentro del bloque if, se realizan varias operaciones:

Reminder tmp = (Reminder) reminder; - Esto cuenta como una operación.

userActionStack.push(new UserAction("EditReminder", key, new String[]{tmp.getTitle(), tmp.getDescription(), String.valueOf(tmp.getDate())})); - Aquí se crea un nuevo objeto de tipo useraction y se pushea a la pila. Esto podría contar como 5 operaciónes por las asignaciones que se hacen en el constructor de useraction y el agregamiento a la pila.

tmp.setTitle(title);, tmp.setDescription(description);, SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");, tmp.setDate(dateFormat.parse(date)); - Estas son asignaciones a los métodos setters, por lo que sería un total de 4 operaciones.

Por lo tanto, el total de operaciones sería: 100 (búsqueda) + 1 (comparación) + 1 (asignación) + 5 (pushear a la pila) + 4 (asignaciones de los setters) = 111 operaciones.

Por lo tanto, la complejidad temporal del método editReminder sería O(111) en este caso específico. Sin embargo, para expresarlo en notación Big O la complejidad temporal sería O(n) que es el número de elementos de la tabla hash.

Complejidad Espacial Total = Entrada + Auxiliar + Salida

Guiándome principalmente por las entradas que son de tipo String, que se constituyen de char, y como en clase vimos que los elementos de tipo char ocupan 16 bits, la complejidad espacial depende de las entradas pues no hay un ciclo for. Por lo tanto la complejidad seria de O(n)

Análisis para editar tarea

Manageable task = hashTable.search(key); - Esta línea realiza una búsqueda en la tabla hash. Supongamos que la búsqueda toma 100 operaciones en el peor de los casos (una por cada elemento en la tabla hash).

if ( task != null ){...} - Esta línea realiza una comparación, que cuenta como una operación.

Dentro del bloque if, se realizan varias operaciones:

if ( task instanceof PriorityTask ){...} - Esta línea realiza una comparación, que cuenta como una operación.

Dentro del bloque if, se realizan varias operaciones:

PriorityTask tmp = (PriorityTask) task; - Esta es una asignación, que cuenta como una operación.

userActionStack.push(new UserAction("EditTask", key, new String[]{tmp.getTitle(), tmp.getDescription(), String.valueOf(tmp.getDate()), String.valueOf(tmp.getPriorityLevel())})); - Aquí se crea un nuevo objeto y se empuja a la pila. Supongamos que esto toma el numero de operaciones de los setters más la pusheada que serían 7 acciones.

tmp.setTitle(title);, tmp.setDescription(description);, SimpleDateFormat dateFormat = new SimpleDateFormat("dd/MM/yyyy");, tmp.setDate(dateFormat.parse(date));, tmp.setPriorityLevel(priorityLevel); - Estas son asignaciones y llamadas a setters. cada una toma 1 operación, para un total de 5 operaciones.

En el bloque else, se realizan varias operaciones similares a las del bloque if, por lo que también contaremos 16 operaciones para este bloque.

Por lo tanto, el total de operaciones sería: 100 (búsqueda) + 1 (comparación) + 1 (comparación) + 16 (operaciones en el bloque if) + 16 (operaciones en el bloque else) = 134 operaciones.

Por lo tanto, la complejidad temporal del método editTask sería O(134) en este caso específico. Sin embargo, para expresarlo en notación Big O la complejidad temporal sería O(n) que es el número de elementos de la tabla hash.

En cuanto a la complejidad espacial, muy similar al anterior caso: Complejidad Espacial Total = Entrada + Auxiliar + Salida

Guiándome principalmente por las entradas que son de tipo String, que se constituyen de char, y como en clase vimos que los elementos de tipo char ocupan 16 bits, la complejidad espacial depende de las entradas pues no hay un ciclo for. Por lo tanto la complejidad seria de O(n)