Tecnológico de Monterrey

Programación de estructuras de datos y algoritmos fundamentales (Gpo 607)

Eduardo Arturo Rodríguez Tello

Act 2.3 - Actividad Integral estructura de datos lineales (Evidencia Competencia)

Iván Alberto Romero Wells | A00833623
Mariano Barberi Garza | A01571226

Mariano Barberi Garza A01571226

Reflexiona por qué en la solución de este reto es preferible emplear una Doubly Linked List y no una Linked List.
En la solución es mejor usar una doubly linked list encima de una linkedlist, porque al momento en el que la doubly linked list te deja tener una conexión tanto al nodo anterior como al siguiente abre posibilidades para crear algoritmos de búsqueda que se lleven a cabo más rápido.

Analiza la complejidad computacional de las operaciones básicas de la estructura de datos empleada en tu implementación (inserción, borrado, búsqueda) y cómo esto impacta en el desempeño de tu solución.
Inserción: o(1)
Borrado: o(n)
Búsqueda: o(n)
La complejidad de cada operación impacta la solución ya que como algunas operaciones como la de búsqueda se realizan tantas veces, si es una complejidad alta el programa tardará más en desarrollarse.

Iván Alberto Romero Wells A00833623

Reflexiona por qué en la solución de este reto es preferible emplear una Doubly Linked List y no una Linked List.

Permite agilizar procedimientos para recorrer nuestra lista, de tal modo que podamos analizar los registros desde el inicio al final y viceversa. Así mismo, nos permite insertar registros al inicio de la lista, al contrario de una lista ligada de un sentido.

Analiza la complejidad computacional de las operaciones básicas de la estructura de datos empleada en tu implementación (inserción, borrado, búsqueda) y cómo esto impacta en el desempeño de tu solución.

Inserción:
Si se utiliza las funciones de inserción de inicio o fin, se tiene una complejidad de O(1). Pero si se desea insertar en un índice dado en medio de la lista, la complejidad sería de O(n), porque se ocupan recorrer la cantidad de nodos hasta llegar al índice dado.

Borrado:
O(n) dado que se ocupan recorrer los nodos hasta el índice o valor dado, para poder acceder al nodo en cuestión y operar con él.

Búsqueda:
O(n) A pesar de utilizar la búsqueda binaria para la búsqueda de nuestros rangos, se hacen n/2 operaciones para acceder eventualmente a las mitades en cuestión, para que de este modo, se hagan solamente O(logn) comparaciones. Por el contrario, en el caso de

utilizar la búsqueda secuencial, tendríamos la misma complejidad O(n), pero se requieren hacer O(n) operaciones y O(n) comparaciones, con lo cuál, termina resultando mejor utilizando la búsqueda binaria, a pesar de no dar los mismos resultados que tradicionalmente da, en una estructura indexada, como un arreglo o vector.