Reflexión Individual
Actividad 2.3

Santiago Álvarez Valdivia

A01640172

Programación de Estructuras de Datos y Algoritmos Fundamentales

Tecnológico de Monterrey

Campus Guadalajara

Escuela de Ingeniería y Ciencias

Profesor: Dr. Eduardo Arturo Rodríguez Tello

Viernes, 15 de octubre de 2021

1 Listas Simplemente y Doblemente Ligadas

En informática y ciencia de datos, una lista se define como una colección de elementos donde cada uno de ellos, además de almacenar información, almacena la dirección de los elementos circundantes. Una lista es una estructura lineal de datos, es decir, que cada elemento sólo puede tener un único sucesor y un único predecesor.

Una lista simplemente ligada es una lista dinámica, formada por elementos llamados nodos, que almacenan información, y la dirección del siguiente nodo.

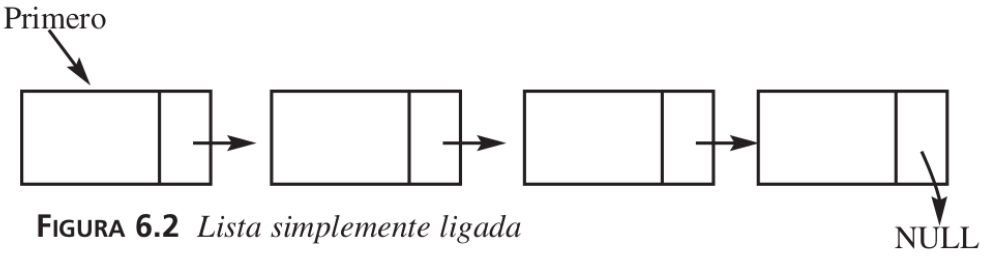


FIGURA 6.2 *Lista simplemente ligada*

Por su parte, una lista doblemente ligada es aquella en la que cada nodo, almacena (además de información) la dirección de su siguiente nodo, y del nodo anterior.

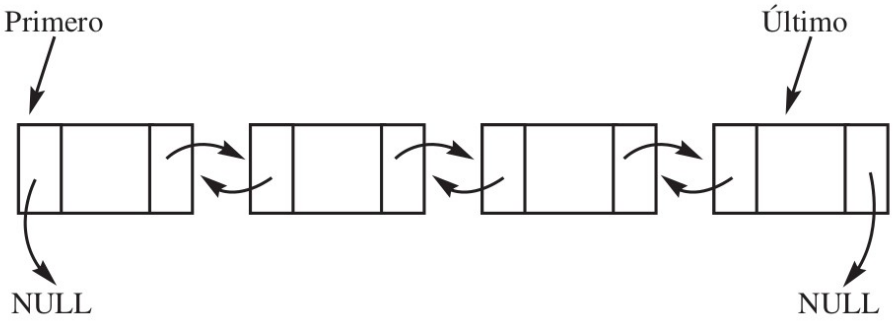


FIGURA 6.14 *Lista doblemente ligada*

Para la realización de este programa, se emplean listas doblemente ligadas, ya que su capacidad de señalar no sólo el dato siguiente, sino también el dato anterior son de utilidad para navegar por la lista.

Esta propiedad es aprovechada en algunas funcionalidades, como por ejemplo en la búsqueda binaria, donde se emplean tanto la dirección al nodo siguiente (`next`), como la dirección del nodo anterior (`prev`).

2 Plantillas

Para la implementación del presente programa, se emplearon las llamadas “plantillas” de C++, que permiten crear clases y funciones capaces de funcionar con cualquier tipo de dato (y no sólo aquel especificado). Los segmentos de código que funcionan a base de plantillas son:

- La clase de lista doblemente ligada (`DoubleLinkedList`)
- La clase nodo (`Node`)
- La clase de pilas (`Stack`)
- Las funciones de ordenamiento y búsqueda (en el archivo `BusquedaOrdenamiento.h`)

Todas las funciones o clases que implementan plantillas se pueden identificar porque antes de su implementación, se observa la siguiente línea.

```
template <class T>
```

3 Algoritmo de Ordenamiento (QuickSort)

En el presente programa, se utiliza el método de ordenamiento *QuickSort* para organizar los registros de incidencia por fecha. El algoritmo, se implementó de manera originalmente recursiva, y después fue ajustado para poder operar iterativamente a base de un `Stack` (Pila), que guarda los datos que emplearía cada llamada recursiva. La razón de este cambio se describe a mayor detalle en la sección 5.

El QuickSort fue empleado por su relativa simplicidad de adaptación para usarse en Stacks, a diferencia de su contraparte *MergeSort*.

4 Algoritmo de Búsqueda

El algoritmo de búsqueda de este programa es el clásico algoritmo de búsqueda binaria. La búsqueda binaria consiste en comparar la mediana de un conjunto ordenado de datos con el dato buscado, para así poder ir descartando cada vuelta la mitad del conjunto de datos.

En este programa se implemento un sistema de dos punteros para encontrar el elemento medio de la lista, independientemente de contadores numéricos (es decir, sin emplear ningún `for` o equivalente).

El proceso empleado se describe a continuación.

Primero, se inicializan ambos punteros, que irán recorriendo la lista en direcciones opuestas. Eventualmente ambos se “encontrarán” es decir, que contendrán la misma referencia. Como ambos punteros se desplazan elemento en elemento, esta referencia es el elemento medio.

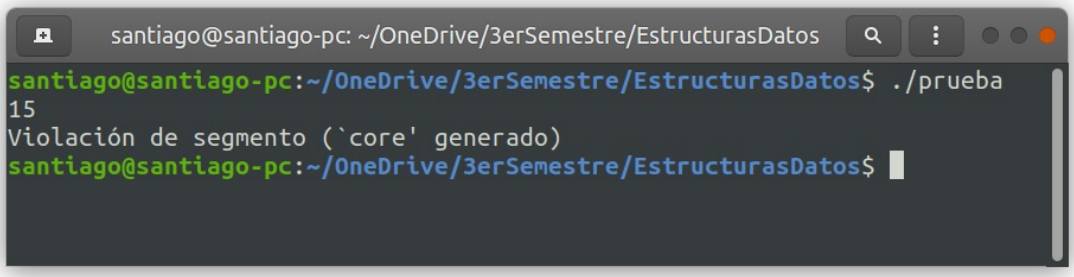
El elemento medio después se compara con el elemento a buscar, lo cual nos indicará en que mitad del arreglo se encuentra lo que buscamos. El proceso se repite con los nuevos límites hasta encontrar el elemento, o bien cerciorarse de que no existe.

5 Empleo de Stacks para iteración

Como ya se describió en la sección 3, este programa utiliza Stacks como un sustituto de la iteración que se empleaba originalmente. La razón de este cambio es evitar fallos debido a las llamadas recursivas.

Al tener una gran cantidad de llamadas recursivas, aparece la posibilidad de un error *Stack-Overflow*, que se da cuando un programa, al ejecutarse, excede la cantidad de memoria de Stack disponible para la operación. Así, el programa se detiene.

A continuación se muestra un error ocasionado por un exceso de llamadas recursivas.



Referencias

[1] Buemo, S. G., Botello, F. O., & Cerpas, J. L. G. (2007). Estructura de datos orientada a objetos. Pearson Educación.

[2] How to Replace Any Recursion with Simple Iteration or Unlimited Iteration with its Own Stack. (2018, 21 septiembre). FreeBasic. Recuperado 14 de octubre de 2021, de https://freebasic.net/forum/viewtopic.php?t=27026

[3] Khalasi, B. (2020, 16 noviembre). QuickSort on Doubly Linked List. CodesDope. Recuperado 14 de octubre de 2021, de https://www.codesdope.com/blog/article/quicksort-on-doubly-linked-list/

[4] Log2Base2. (s. f.). Recursion | Recursion Rules | stack overflow in recursion. Recuperado 14 de octubre de 2021, de https://www.log2base2.com/C/recursion/recursion.html

[5] TutorialsPoint. (s. f.). C++ Templates. Recuperado 14 de octubre de 2021, de https://www.tutorialspoint.com/cplusplus/cpp_templates.htm

[6] Zeil, S. J. (2021, 29 marzo). Converting Recursion to Iteration. Old Dominion University. Recuperado 14 de octubre de 2021, de https://www.cs.odu.edu/%7Ezeil/cs361/latest/Public/recursionConversion/index.html