Nombre: Josue Salvador Cano Martinez ***Matrícula:*** A00829022

Programación de estructuras de datos y algoritmos fundamentales

Módulo: Estructura de datos lineales

Actividad 2.3: Actividad integral estructura de datos lineales

Investigación y reflexión individual

Fecha de entrega: 11 de octubre de 2020

Objetivo

Realizar una investigación y reflexión en forma individual de la importancia y eficiencia del uso de las listas doblemente ligadas en una situación problema de esta naturaleza.

Parte 1: Investigación

Listas doblemente ligadas

Una lista doblemente ligada contiene un puntero adicional (a diferencia de las listas enlazadas), normalmente denominado puntero anterior, junto con el siguiente puntero y los datos que contiene la lista (figura 1.0).

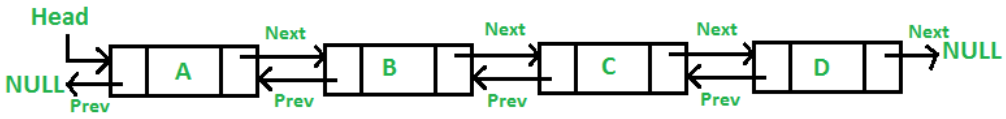

Figura 1.0: representación gráfica de una lista doblemente ligada.

En la implementación de programación, dicho puntero adicional deberá ser creado en la clase en que se define el ‘nodo’:

```
// Nodo de una lista doblemente ligada
struct Node {
    int data;
    struct Node* next; // Puntero al nodo siguiente
```

```
    struct Node* prev; // Puntero al nodo previo
};
```

Ventajas sobre las listas enlazadas

1. Una lista doblemente ligada puede iterarse tanto hacia adelante como hacia atrás, la lista enlazada sólo puede hacerlo hacia adelante.
2. La operación de eliminación en una lista doblemente ligada es más eficiente si se proporciona un puntero al nodo que será eliminado.
3. Es posible insertar de forma más rápida un nuevo nodo en la lista doblemente ligada; en la lista enlazada para eliminar un nodo se necesita el puntero al nodo anterior y para obtenerlo muchas veces se necesita recorrer la lista; dicho nodo anterior es posible de obtener utilizando el puntero previo en una lista doblemente enlazada.

Desventajas sobre las listas enlazadas

1. Cada nodo de la lista doblemente enlazada requiere espacio adicional para un puntero anterior.
2. Todas las operaciones requieren un puntero adicional previo para mantenerse. Por ejemplo, en la inserción se necesitan modificar los punteros anteriores junto con los punteros siguientes.

(GeeksforGeeks, 2020)

Complejidad computacional

Operación	***Caso promedio***	***Peor caso***
Acceder	O(n)	O(n)
Buscar	O(n)	O(n)
Insertar (al inicio o final)	O(1)	O(1)
Eliminar (al inicio o final)	O(1)	O(1)

(Rowell, s.f.)

GeeksforGeeks. (04 de 07 de 2020). *GeeksforGeeks*. Recuperado el 10 de 10 de 2020, de https://www.geeksforgeeks.org/doubly-linked-list/

Rowell, E. (s.f.). *Big-O Cheat Sheet*. Recuperado el 10 de 10 de 2020, de https://www.bigocheatsheet.com/

Parte 2: Reflexión

Después de haber estudiado la parte teórica referente a las estructuras de datos lineales y de haber realizado prácticas para el entendimiento de su aplicación, alcanzo a concluir que para la solución de esta situación problema (y para las soluciones de otras más que resultan similares) realmente el uso de listas doblemente enlazadas no genera una ventaja considerable sobre el uso de listas enlazadas. Como bien se sabe, una de las principales ventajas de las listas doblemente enlazadas es la posibilidad de iterar las listas hacia atrás y hacia adelante gracias al par de punteros que cada nodo tiene, permitiendo ahorrar tiempo en algunas acciones que se efectúan sobre la estructura de datos, tal es el caso de la 'eliminación'. La principal desventaja de estas listas doblemente enlazadas es el uso de mayor memoria debido a la necesidad de implementar dos punteros en lugar de uno (caso de la lista enlazada), sin embargo, para este tipo de cuestiones prácticas, el tiempo resulta ser más importante que el espacio. Para explicar de una forma más sencilla la principal ventaja que se tiene de una lista doblemente enlazada contra una lista enlazada imaginemos que la primera de ellas es un auto que tiene reversa mientras que la segunda de ellas es un auto que no tiene reversa; los conductores estacionaron ambos autos fuera de la casa 107 mientras permanecían en una reunión, dos horas después se tuvieron que trasladar a la casa 100 (tenían que retroceder); para el auto que sí tenía reversa (lista doblemente enlazada) fue muy rápido al sólo tener que moverse 7 casas (posiciones hacia atrás), sin embargo, para el auto sin reversa (lista enlazada), fue necesario dar la vuelta a toda la manzana para poder llegar de nuevo al inicio de la calle y así posicionarse fuera de la casa 100, teniendo que invertir una mayor cantidad de tiempo en conseguirlo.

Auto con reversa (lista doblemente enlazada)

Auto sin reversa (lista enlazada)

Una vez entendida la principal ventaja que se puede obtener de cada caso es posible decir que para esta situación problema (tal y como fue planteada y desarrollada) la aplicación de una lista doblemente enlazada no da un ventaja considerable, las complejidades computacionales que se hubiesen requerido para los procesos de cada una de las funciones hubiese resultado igual, sin embargo, el hecho de haber creado la estructura de datos como una lista doblemente enlazada permite cimentar una infraestructura que resulta mayormente viable para futuros casos que puedan llegar a implementarse, por ejemplo, imaginando que es una situación problema que se implementará en la vida real el hecho de haberla construido a partir de listas doblemente enlazadas posibilita que en un futuro se pueda sacar un mejor provecho de la misma si es que se llegasen a agregar funciones tales como eliminar, o procesos que se miren beneficiados en cuestión de tiempo al requerir retroceder al momento de efectuar el instanciamiento, entonces resulta conveniente haber optado por esta alternativa desde un inicio. Para entender de una mejor manera la forma en que fue desarrollada la codificación a continuación se expone un pseudocódigo de cada función implementada, mismo que permite exponer la indiferencia, en este caso, de optar por una lista enlazada o una lista doblemente enlazada.

Descripción de algoritmos implementados:

CreateNewIp: el string que contiene la ip es llenado con ceros (de ser necesario), a fin de tener longitudes uniformes de 3 elementos.

- ✓ Se recibe un string que contiene la ip.
- ✓ Se encuentra la posición de los dos puntos ‘:’
- ✓ Se encuentra el ínice final sumando 4 al valor de la posición donde fue encontrado ‘:’
- ✓ Se extrae todo el valor con substr, desde 0 hasta la posición final previamente encontrada.
- ✓ Se crea una ip tomando los primeros 12 dígitos
- ✓ Se busca la posición del punto (###**.**###...)
- ✓ Se hace un substr desde la posición 0 hasta la posición obtenida en el paso anterior (aquí se obtiene el primer bloque numérico de la ip).
- ✓ Si el tamaño del bloque obtenido es menor que tres entonces se agregan los ceros necesarios para que la longitud sea tres.
- ✓ A un nuevo string se le adjunta el bloque formado (para crear el nuevo ip).
- ✓ Si aún no se llega al bloque de los dos puntos ‘:’ se agrega un punto ‘,’
- ✓ Cuando ya se llenan todos los primeros 4 bloques de número, se agregan los dos puntos ‘:’ y la parte final de la ip (el últmo bloque).

CreateNewLine: se cambia el formato de: “mes, hora, día, ip” a formato: “ip, mes, día, hora”; esto con el fin de que la ip quede al inicio de cada registro y se facilite el proceso de hacer el ordenamiento.

- ✓ Se extrae la ip (a partir de la posición 16 hasta que encuentra el espacio en blanco).
- ✓ Se elimina la ip (porque ya se extrajo) y se agrega al inicio del string.

readFile: se lee el fichero y se adjunta cada registro a la lista doblemente enlazada.

- ✓ Se abre el archivo.
- ✓ Mientras no se acaben los registros se crean nuevas líneas con la clase hecha y se adjuntan a la lista enlazada.

revertLine: se intercambia la posición de la ip al índice posterior a ‘hora’

- ✓ Se tienen los registros donde la ip se encuentra al inicio (debido a haber creado las líneas con la clase previamente descrita), por lo tanto, se determina la posición donde esta termina para así poder extraerla.
- ✓ Se eliminan los ceros que se le adjuntaron a la ip.
- ✓ Se elimina la ip del inicio y se adjunta en la posición 16 (después del mes, día y hora).

main: se desarrolla la aplicación que gestionará el uso de las funciones previamente creadas.

- ✓ Se crea la lista doblemente enlazada.
- ✓ Se crea y se abre un archivo que contendrá las ip ordenadas (para poder imprimir los datos de salida).
- ✓ Se cargan los datos del fichero a la lista doblemente enlazada.
- ✓ Se ordenan ascendentemente los registros con sort()
- ✓ Se piden los datos de entrada para realizar la búsqueda
- ✓ Se obtienen los registros correspondientes al intervalo que agrupan los datos de entrada.
- ✓ Se imprimen los registros.

Como es posible percatarse, no fue necesario en alguna parte de la implementación algorítmica el uso de ‘regresar’ que ofrece la lista doblemente enlazada al incluir el segundo puntero, sin embargo, como se mencionó, puede resultar útil para futuras implementaciones/modificaciones.