	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: TISTA GARCÍA EDGAR

Asignatura: ESTRUCTURA DE DATOS Y ALGORITMOS I

Grupo: 1

No de Práctica(s): 8) Estructuras de datos lineales: Lista doblemente ligada y doblemente ligada circular.

Integrante(s): MURRIETA VILLEGAS ALFONSO

Semestre: 2018 - 2

Fecha de entrega: 16 DE ABRIL DE 2018

Observaciones:

CALIFICACIÓN: _____

Estructuras de datos lineales: Lista doblemente ligada y doblemente ligada circular.

1] Introducción

Como vimos en la práctica anterior, las LISTAS son uno de los muchos tipos de estructuras de datos (TDA's) que existen, éstas se caracterizan por ser lineales debido a que solamente tienen un predecesor y un solo antecesor, además, las listas también son dinámicas ya que tienen la ventaja de que su tamaño o dimensión no es fijo y se puede definir conforme se requiera.

Sin embargo, dentro de las listas existen distintos tipos, por ejemplo, en la práctica anterior vimos el caso de la lista simple y circular, mientras que en esta práctica se verán las listas doblemente ligada y doblemente circulares, pero, aunque entre éstas existan notables diferencias siguen teniendo las mismas operaciones que son: BUSCAR, INSERTAR Y ELIMINAR.

2] Objetivo de la práctica

- Revisarás las definiciones, características, procedimientos y ejemplos de las estructuras lineales Lista doblemente ligada y Lista doblemente ligada circular, con la finalidad de que comprendas sus estructuras y puedas implementarlas.

3] Desarrollo

3.0] Antecedente

Las listas son un TDA que sobretodo están enfocadas para poder almacenar datos de una manera más óptima, una de sus implementaciones es para sustituir a los arreglos que se caracterizan por ser estáticos y no dinámicos*, pero para que las listas tengan no sean como los arreglos necesitan de un elemento sumamente primordial conocido como NODO, éstos se caracterizan por tener un valor cualquiera conocido como INFO y además tener una referencia la cual se usa para enlazar hacia otros nodos (NEXT). Sin embargo, a diferencia de las listas simples, las listas dobles como dice su nombre se caracterizan por tener una doble referencia, y es que éstas listas tienen la ventaja que dentro de sus nodos no solamente tienen la referencia del sucesor sino también tienen la referencia de su antecesor (Ver Imagen 1).

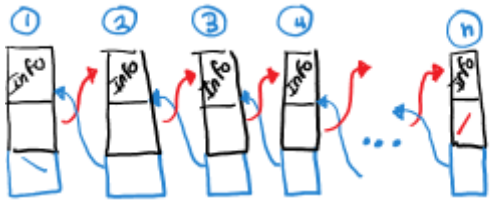

Imagen 1: Diagrama de cómo se enlazan los nodos en una lista doblemente ligada.

3.1] Análisis de la teoría

LISTA DOBLEMENTE LIGADA

Una lista doblemente ligada o también conocida como lista doble está constituida por un conjunto de nodos alineados de manera lineal (uno después de otro) y unidos entre sí mediante dos referencias (NEXT) y (PREV), a diferencia de un arreglo el cual también es un conjunto de nodos alineados de manera lineal, en las listas dobles el orden está determinado mediante dos REFERENCIAS, no por un índice, además de que en la lista el tamaño no es fijo, otro punto relevante es que comparándolo con una lista simple, la lista doble incluye otra referencia más que apunta al antecesor (PREV).

Como se dijo en el apartado de antecedentes, la unidad básica de una lista doble es un elemento o nodo, cada uno de estos elementos es un objeto que contiene la información que se desea almacenar, así como dos referencias (A través del uso de NEXT y PREV), sin embargo, algo importante es que este TDA tanto en el inicio como fin de su estructura sus nodos no apuntan a ninguna parte o dicho de otra forma son NULOS.

Por último, para que realmente se pueda llevar a cabo la naturaleza de este tipo de estructura se necesitan de 3 tipos de funciones u operaciones BUSCAR, INSERTAR Y BORRAR, las cuales serán mencionadas a continuación.

BUSCAR

Una lista doble cuando tiene elementos, puede contener de 1 a n elementos, en tal caso, la referencia al inicio (HEAD) apunta al primer elemento de la lista mientras que el PREV de este se mantiene nulo. D esta forma, es posible recorrer la lista a través de la referencia (NEXT) de cada nodo hasta llegar al que apunta a nulo en NEXT, el cuál será el último elemento.

Sin embargo, al hablar de una lista doble no solamente podemos realizar la búsqueda desde el primer nodo, sino que tenemos la ventaja de empezar desde el último nodo, recordemos que la ventaja de que la lista sea doble es que tenemos referencia tanto de los sucesores como de los antecesores, por lo tanto, en este caso podemos realizar la busque desde ambos extremos.

Para poder buscar dentro de esta lista se debe buscar el primer elemento que coincida con la llave K (La llave k como menciona el manual de prácticas también lo podemos encontrar como un nodo temporal que sirve como comparador) dentro de una lista, a través de una búsqueda lineal simple, regresando un apuntador a dicho elemento si éste se encuentra en la lista o nulo en caso contrario.

NOTA: En caso de que la lista se encuentre vacía, la referencia al inicio de la misma (HEAD) apunta a nulo, de este modo tenemos que en una lista vacía no es posible buscar elementos puesto no existen elementos a buscar.

BORRAR

En el caso de la función eliminar lo primero que se realiza es la búsqueda del elemento a eliminar obviamente para esto se utiliza la función anterior la cual se encarga de encontrar el nodo a eliminar. Una

vez que se ha encontrado el elemento, se deben mover las referencias de la estructura de tal manera de que el antecesor del nodo a eliminar apunte al sucesor del mismo y el predecesor del nodo sucesor apunte al predecesor del nodo (PREV).

NOTA: En el caso de que la lista doble se encuentre vacía la operación de borrado realmente no se puede hacer debido a que no existen elementos.

INSERTAR

La función insertar se puede dar en dos casos, el primero es cuando la lista doble se encuentra vacía y el segundo caso es cuando la lista tiene elementos. Cuando la lista doble se encuentra vacía, se inserta un nuevo elemento en una lista y la referencia de este está al inicio de la lista (HEAD).
Pero cuando se inserta un nuevo elemento en una lista doble con elementos, la referencia del nuevo nodo (NEXT) apunta al mismo nodo al que apunta el inicio de la lista (HEAD), la referencia anterior (PREV) del nodo siguiente (NEXT) del inicio de la lista apunta al nuevo nodo, y HEAD también apunta al nuevo nodo.

Dicho de otra forma, lo que se necesita para insertar un nodo dentro de una lista que ya tiene previamente elementos, primero es saber el nodo que se encuentra en la posición donde se quiere insertar, posteriormente referenciar el nodo creado al nodo que se encuentra en la posición a insertar y a su vez referenciar también el sucesor del elemento pues recordemos que en este tipo de lista los nodos tienen referencias tanto del anterior como del siguiente elemento.

LISTA DOBLEMENTE LIGADA CIRCULAR

La lista doblemente ligada circular o también conocida como Lista doble circular, a diferencia de la lista doble, ésta es una lista doble “modificada”, ya que, a diferencia de la lista doble, está tanto en el apuntador de su primer elemento como el apuntador de su último elemento están referenciados, en el caso del primer elemento apunta mediante el PREV al último elemento, mientras que el último elemento apunta o referencia al primer elemento mediante el NEXT.

Por último, al igual que las listas dobles, para que realmente se pueda llevar a cabo la naturaleza de este tipo de estructura se necesitan de 3 tipos de funciones u operaciones que son: BUSCAR, INSERTAR Y BORRAR, las cuales serán mencionadas a continuación.

NOTA: Una de las mayores diferencias entre la lista doble y la lista doblemente circular es que la lista doble circular considera o tiene una variable más que es la **dimensión** o **tamaño** de lista. Es se debe a su naturaleza ya que en las listas circulares si no se considerara la dimensión de estas, al momento de buscar o determinar posiciones no sería posible.

BUSCAR

La búsqueda de un elemento dentro de las listas circulares dobles se hace a través de la comparación de nodos mediante la implementación de un nodo temporal también conocido como llave K a través de una búsqueda lineal simple, regresando un apuntador a dicho elemento si éste se encuentra en la lista o nulo

en caso contrario (Cuando la lista se encuentra vacía realmente no se puede buscar ningún elemento debido a que no existen).

Una lista circular doble cuando tiene elementos para realizar la búsqueda de un elemento simplemente podemos ir recorriendo o comparando el nodo temporal con los nodos de la lista desde el primer elemento o desde el último elemento y además podemos hacer la busque hacia adelante o hacia atrás puesto al tener las referencias tanto de los sucesores como de los antecesores tenemos la disponibilidad de poder realizar la búsqueda en cualquier sentido de la lista.

Otro punto importante es que para no realizar la búsqueda infinitamente lo que se debe considerar al momento de la búsqueda es el tamaño o dimensión de la lista.

INSERTAR

En el caso de la función u operación insertar debido al tipo de estructura se pueden tener distintos casos de uso. En el caso de que la lista se encuentre vacía es posible insertar elementos, en este caso se inserta un nuevo elemento en una lista circular vacía la referencia al inicio de la lista (HEAD) apunta al nodo insertado, mientras que la referencia al sucesor (NEXT) y el predecesor (PREV) del nodo apunta a sí mismo.

Sin embargo, en el caso de que la lista tenga elementos, el sucesor del nodo creado (NEXT) apunta al nodo sucesor (Dependiendo de la posición en la que se quiera insertar, esto irá cambiando), mientras que (PREV) apunta al nodo antecesor. Cuando insertamos un nodo recordemos que pueden existir distintos casos como son: La inserción al inicio de la lista, al final de la lista y en cualquier parte de la lista, dependiendo de la posición es como irá cambiando la manera en que se tenga que ir haciendo la búsqueda y posteriormente la inserción.

BORRAR

Para eliminar un nodo en una lista circular doble con elementos, primero se debe usar la operación BUSCAR para poder encontrar el elemento a eliminar, una vez encontrado el nodo dentro de la lista, se deben mover las referencias de la estructura de tal manera de que el antecesor del nodo a eliminar apunte al sucesor del mismo(NEXT) y a su vez la referencia del sucesor que apunte al antecesor (PREV). Dicho lo anterior, para poder elementos es necesario saber la ubicación del nodo a eliminar, por lo tanto, primero se debe realizar una búsqueda del elemento dentro de la lista.

NOTA: Como es obvio en una lista circular vacía no es posible eliminar, debido a que esta estructura no contiene elementos.

3.2] Relación entre los ejercicios y el tema

Las listas son un tipo de estructura sumamente importante, a lo largo de esta práctica-examen y mediante la abstracción y análisis de distintos problemas se empleó 2 principales estructuras (Listas simples y listas circulares) para poder resolver o corregir los distintos problemas.

La relación de los ejercicios que se realizaron a lo largo de esta práctica con respecto al tema de estructura de datos lineales fue sobretodo como iban cambiando algunas de las funciones principales de las listas, sobretodo, porque cuando hablamos de listas dobles podemos notar ventajas como el poder regresar y avanzar en el recorrido de la lista a través del uso de NEXT y PREV.

Por otro lado, en el ejercicio 6 lo que podemos ver es que realmente la ventaja de lista doble circular radica en como al recorrer la lista creada podemos realizar las comparaciones de nodos ya sea utilizando PREV para ir de forma inversa o usando NEXT para buscar de la forma en la que siempre hemos hecho, también debemos destacar como podemos realizar cualquiera de estas dos, tanto desde el inicio de la lista como desde el final de la lista.

4] Conclusiones

Esta práctica fue el primer acercamiento a uno de los conceptos más relevantes de las estructuras de datos, a lo largo de esta práctica tanto de manera teórica como a través de distintos ejercicios, nos hemos acercado a dos principales TDA´s como son la lista doble y la lista doble circular, las cuales principalmente se caracterizan por tener referencias en ambas direcciones de los nodos.

Las listas dobles además de tener ventajas tanto al eliminar y agregar elementos con respecto a las pilas y colas, también presentan ventajas considerables con respecto a otras listas, sobretodo porque la manera en que se manejan estas estructuras resulta realmente conveniente debido a que, por ejemplo, en la búsqueda no solamente dependemos del NEXT además tenemos la disponibilidad de usar la función inversa que es el PREV. Otro aspecto relevante fue como para poder emplear de manera apropiada esta estructura usamos 3 principales operaciones que son BUSCAR, INSERTAR y ELIMINAR donde sobretodo el uso adecuado del paso por referencia para no cometer errores.

Por otro lado, podemos notar la implementación tanto de la lista doble como de la lista circular doble en casos muy comunes de la vida cotidiana, por ejemplo, una lista de reproducción de videos de Youtube reproduce videos consecutivamente (De forma lineal), además de que tenemos la posibilidad de regresar al video anterior y también tenemos la opción de avanzar al siguiente video, sin embargo, cuando llegamos al último video tenemos la disponibilidad de regresar al primer video, de forma implícita lo que podemos notar es el uso de una lista circular doble de videos pues todos los videos están referenciados tanto al anterior como el sucesor.
También podemos destacar que en el caso de las listas dobles las podemos ver sobre todo en los manejadores webs o también conocidos como gestores de correo donde cada mensaje o correo están enlazados a través de referencias, sin embargo, el primer elemento solamente apunta al siguiente ya que su antecesor es nulo, además de que el último elemento no tiene un sucesor ya que este último correo es precisamente el último de la lista.

Por último, y como opinión personal, la forma en la que se realizó la presente práctica durante la clase realmente me pareció buena, sobretodo porque a veces los problemas podemos resolverlos de forma grupal o en parejas (Como fue el caso), y es que al final uno como ser humano siempre va a necesitar ayuda de otras personas independientemente del ambiente o tema en el que este. Tal vez en la práctica me pude exaltar un poco debido a que mi compañero no había entendido del todo el ejercicio 6, sin embargo, pese pude verme exaltado quiero aclarar que nunca me ví agresivo, realmente entre nosotros (Es mi amigo) nos llevamos comúnmente así, y en cierta forma es lo que me agrada de trabajar con amigos. De cualquier forma y para concluir, en verdad me gustó trabajar de esta forma, salvo la parte de tener que escribir el código en el papel ya que en nuestro caso lo hicimos de prisa debido a que lo comprobamos primero en la computadora.

5] Referencias

1) Introduction to Algorithms. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein, McGraw-Hill.

2) The Algorithm Design Manual. Steven S. Skiena, Springer.

3) Curso de Programación C / C++. Javier Ceballos, tercera edición. USA, Ra – Ma. Educación 2007

4) El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.