TDA Lista

[7541/9515] Algoritmos y Programación II

Primer cuatrimestre de 2022

Alumno:	Subero, Andrés
Número de padrón:	109114
Email:	asubero@fi.uba.ar

1. Introducción

El trabajo práctico tiene como finalidad poner en práctica el Tipo de Dato Abstracto lista, junto con dos tipos de ella, que son la pila y la cola. En esta ocasión se utilizaron nodos simplemente enlazados, los cuales permiten usar bloques de memoria dinámica no continuos. Para poder realizar las conexiones entre los distintos bloques, la estructura lista cuenta con un puntero al primer nodo y un puntero al último nodo. La estructura nodo cuenta con un puntero al siguiente nodo y al elemento que contiene.

Con estas estructuras definidas para la lista, se pudo realizar la implementación de pila y cola, haciendo uso de las funciones ya implementadas con algunos casteos. En el caso de la cola, se tiene como “contrato” que el primer elemento en entrar es el primero en salir, entonces para la implementación de encolar se utilizó la función de lista insertar y para desencolar se utilizó la función quitar de posición, usando siempre posición = 0 (primer elemento). En el caso de la pila se tiene como “contrato” que el primero en entrar es el último en salir, entonces para apilar se usó lista insertar y para desapilar se usó lista quitar.

Para confirmar la correcta implementación de los TDAs, se realizaron pruebas unitarias, las cuales simulan la mayor cantidad de casos posibles (en este caso son muy importante los casos bordes) y con cada prueba se asegura que ese caso está cubierto y no fallará en el futuro.Para la realización de las pruebas, se intentó realizar primero la prueba y luego implementar la mínima solución (TDD).

2. Teoría

2.1) TDA pila con nodos simplemente enlazados.

El TDA pila utilizando nodos enlazados fue el implementado en esta ocasión, para ello se hizo uso de la estructura lista. Como se dijo anteriormente, la pila tiene como contrato que el primero en entrar es el último en salir. Para implementarla con nodos simplemente enlazados de manera eficiente, la estructura pila debe tener un puntero a la última posición y un puntero a la primera posición y un entero para guardar la cantidad de elementos.

A la hora de apilar se tiene complejidad O(1) ya que no depende de la cantidad de elementos, siempre se realizan los mismos pasos que son los siguientes:

1. Se reserva memoria para un nuevo nodo, se le asigna el elemento y tiene como siguiente NULL
2. el último nodo pasa a tener como siguiente el nuevo nodo y
3. la pila pasa a tener como nodo final el nuevo nodo, se le suma uno a la cantidad y se retorna el puntero a la pila.

Para desapilar tendríamos complejidad O(n) siendo la cantidad de elementos que tiene la pila y se realiza con los siguientes pasos:

1. Iteramos hasta encontrar el nodo que tenga como siguiente el puntero al último nodo de la pila
2. Se crea un auxiliar que apunte al siguiente del nodo actual (ultimo nodo) y un auxiliar que apunte al elemento que contiene el nodo auxiliar (elemento del último nodo),
3. La pila pasa a tener como último el nodo actual (era el penúltimo) y su siguiente pasa a ser NULL,
4. Se resta uno a la cantidad, se libera el nodo auxiliar y se retorna el elemento apuntado con el auxiliar.

2.2) TDA cola con vector estático circular.

Para implementar el TDA cola un vector estático es necesario que la estructura cola tenga 4 enteros, uno para definir el tamaño del vector (cuántos elementos puede contener en total), un entero para saber cuántos elementos tiene actualmente (cantidad), uno para saber la posición del primer elemento y uno para el último.

Encolar tiene complejidad O(1), ya que solamente se verifica si el vector está lleno, si no lo está se agrega en la posición tope y se le suma uno a la cantidad de elementos. Como es circular, a la hora de encolar se verifica al actualizar la nueva posición de tope (tope+1), si es mayor al tamaño del vector, en este caso se pasa a

la posición 0. Se sabe que hay espacio en esa posición por el entero cantidad que nos indica la cantidad de elementos que se tiene en el vector.

Para desencolar se tiene complejidad O(1) también porque se retorna el elemento y actualiza la posición del primer elemento. Al igual que a la hora de encolar, se tiene que tener en cuenta que si la nueva posición del primer elemento es mayor que el tamaño del vector se pasa a la posición 0.

2.3) TDA lista con nodos enlazados.

Para implementar el TDA lista con nodos simplemente enlazados, la estructura lista debe tener un puntero al primer nodo y uno al último, un entero para saber la cantidad de elementos que hay en la lista.

Para insertar un elemento en la última posición se realizan los siguiente pasos(Complejidad O(1) ya que no depende de la cantidad de elementos, siempre se realizan los mismos pasos):

1. Se reserva memoria para un nuevo nodo, se le asigna el elemento y tiene como siguiente NULL
2. el último nodo pasa a tener como siguiente el nuevo nodo y
3. La lista pasa a tener como nodo final el nuevo nodo, se le suma uno a la cantidad y se retorna el puntero a la lista.

Si se quiere insertar en una posición n. (Complejidad O(n) siendo n la posición buscada, ya que se tiene que iterar hasta la posición anterior a n)

1. Se reserva memoria para el nuevo nodo y se le asigna el elemento
2. se itera hasta llegar al nodo con posición n-1 (nodo actual)
3. el nuevo nodo tiene como siguiente al siguiente del nodo actual y nodo actual tiene como siguiente al nuevo nodo.
4. se le suma uno a la cantidad.

Para eliminar un elemento de la última posición (Complejidad O(n), siendo n la cantidad de elementos que tiene la lista, el tamaño del problema es la posición buscada que en este caso es la anterior a la última.

1. Iteramos hasta encontrar el nodo que tenga como siguiente el puntero al último nodo
2. Se crea un auxiliar que apunte al siguiente del nodo actual (ultimo nodo) y un auxiliar que apunte al elemento que contiene el nodo auxiliar (elemento del último nodo),
3. La pila pasa a tener como último el nodo actual (era el penúltimo) y su siguiente pasa a ser NULL,
4. Se resta uno a la cantidad, se libera el nodo auxiliar y se retorna el elemento apuntado con el auxiliar.

Para eliminar un elemento en una posición n (Complejidad O(n) siendo n la posición buscada, ya que se tiene que iterar hasta la posición anterior a n):

1. Iteramos hasta que el nodo actual esté en la posición n-1.
2. Se crea un auxiliar que apunta al siguiente del nodo actual (nodo a eliminar).
3. El nodo actual ahora tendrá como siguiente al siguiente del nodo auxiliar.
4. se apunta al elemento del nodo auxiliar con un puntero void auxiliar
5. Se resta uno a la cantidad
6. Se libera el nodo auxiliar y se retorna el elemento.

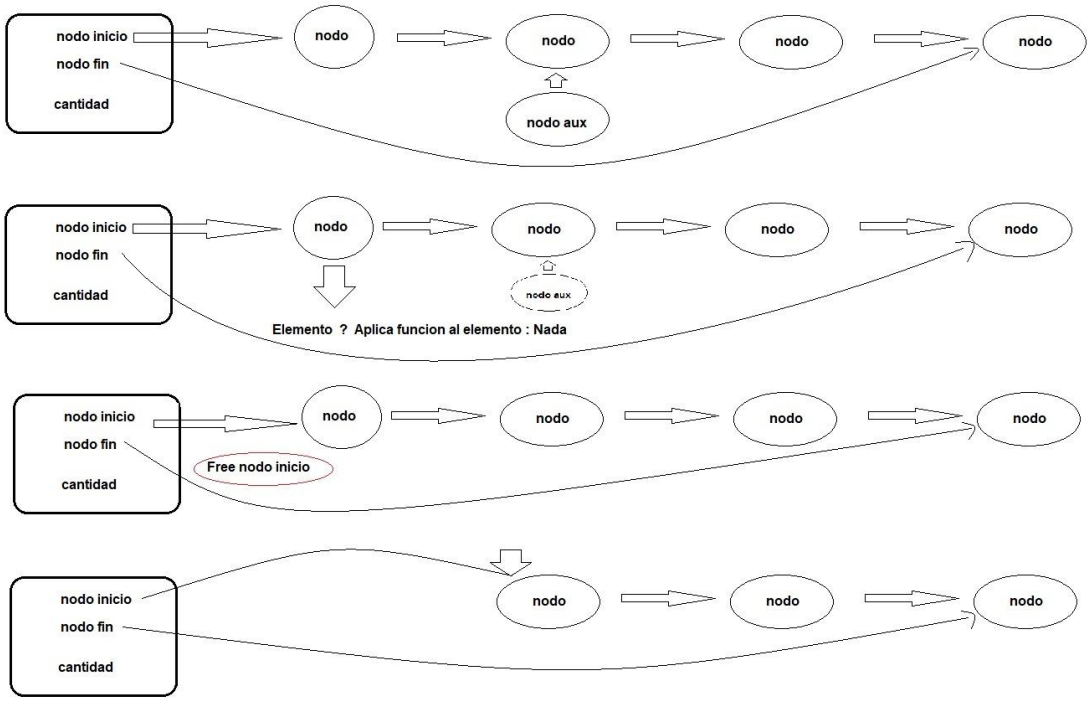
3. Detalles de implementación

3.1. TDA LISTA

En esta sección se explicará la implementación del TDA lista con nodos simplemente enlazados. Parte de la implementación fue explicada anteriormente así que se hará énfasis en funciones y detalles no explicados.

- **Lista destruir todo**

Esta función se encarga de iterar los nodos de la lista, aplicar una función al elemento (en el caso de que haya elemento) y liberar el espacio reservado por el nodo. Finalmente se libera el espacio reservado por la lista.

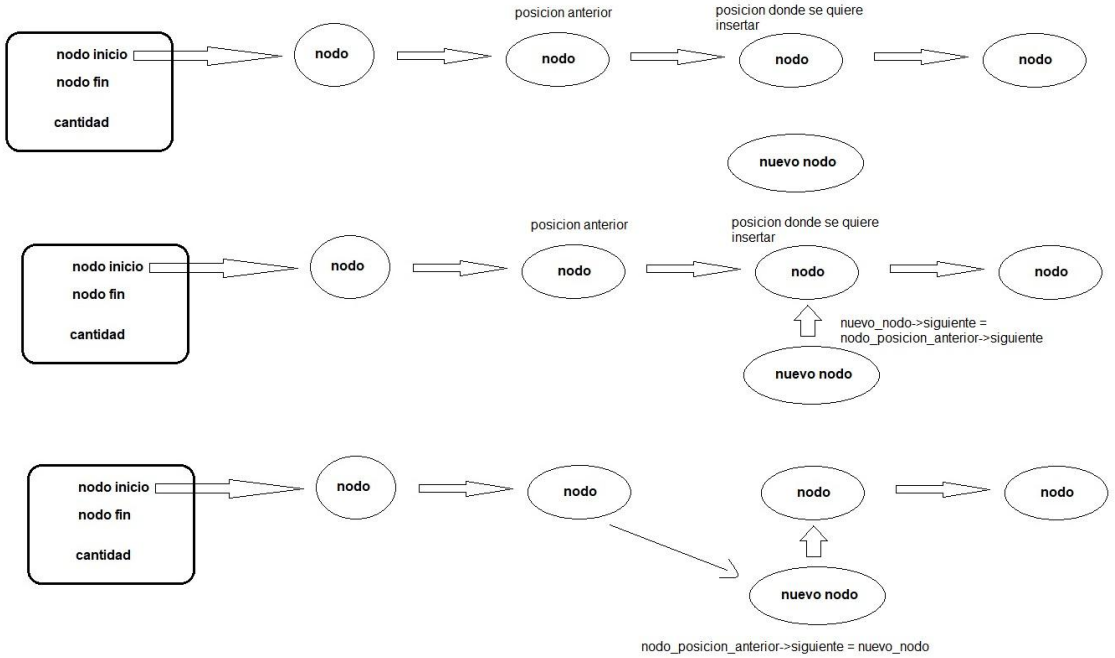


Este proceso se repite hasta que nodo inicio sea igual a NULL.

- **Inserción y eliminación**

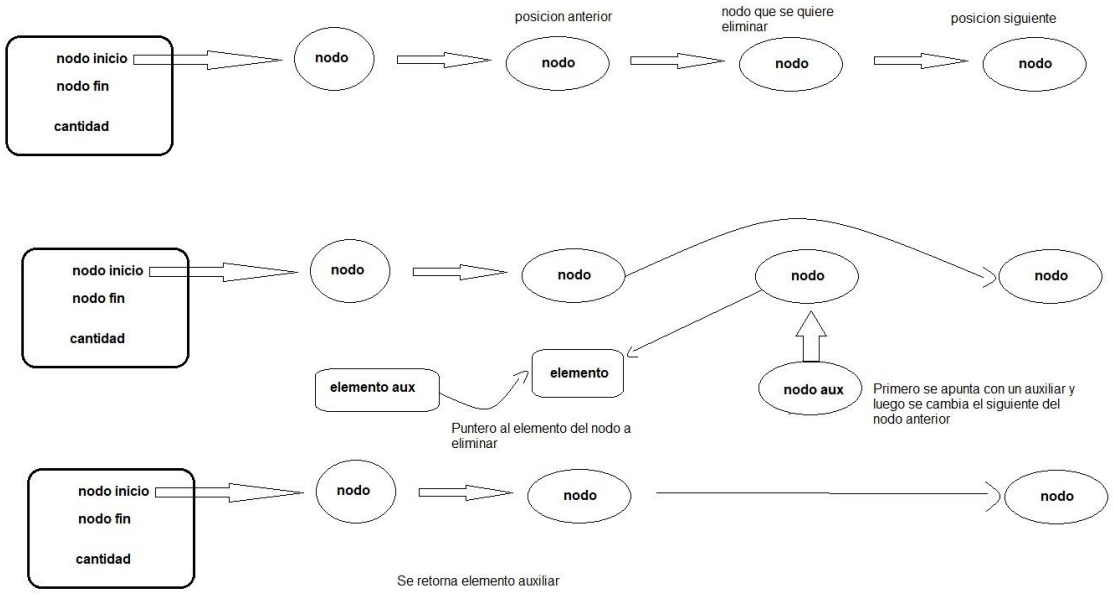
Para realizar estas operaciones, se usa una función auxiliar llamada buscar nodo, no está dentro de las funciones públicas. Se encarga de iterar la lista hasta llegar al nodo en la posición dada. Si se quiere retornar el elemento solo se busca el nodo en la posición dada. Si se quiere eliminar o insertar se busca la posición anterior para poder guardar en auxiliares los punteros necesarios (como se explicó en la parte teórica).

Diagrama de insertar:



En el caso de la inserción se reserva memoria con malloc para el nuevo nodo y se le asigna el puntero al nuevo elemento. Si falla malloc se retorna NULL en lugar del puntero a la lista. Si la posición es 0 es un caso particular ya que no existe una posición anterior, el siguiente del nuevo nodo será lista->nodo inicio y luego lista->nodo inicio será el nuevo nodo.

Diagrama de eliminación



En el caso de la eliminación sólo se libera la memoria reservada por el nodo. Al igual que en la inserción, la eliminación en la posición cero es particular y no se realiza como en el diagrama, se apunta al primer nodo con un auxiliar y también a su elemento con un auxiliar. El segundo nodo pasa a ser el primer nodo de la lista, se libera el nodo apuntado con el auxiliar y se retorna el elemento.

3.2. TDA Pila y Cola

Para estos casos particulares de la lista se usaron las estructuras y funciones implementadas para la lista como se mencionó en la parte teorica. Para ello se casteo el puntero a la lista como si fuese un puntero a pila o cola respectivamente.

En general, en las funciones donde se tiene que retornar un puntero a pila o cola se casteaba a la hora de retornar el puntero a lista a (pila_t *) o (cola_t*)

dependiendo del caso. Y a la hora de usar las funciones que reciben un puntero a lista como parámetro, se casteaba el puntero a (lista_t*) de modo de que la funcion tomara el puntero que se estaba pasando como si fuese un puntero a lista.

Para las operaciones con pila y cola, se hizo uso de las funciones implementadas para la lista.

Como se mencionó anteriormente, en el caso de la cola, se tiene como “contrato” que el primer elemento en entrar es el primero en salir, entonces para la implementación de encolar se utilizó la función de lista insertar y para desencolar se utilizó la función quitar de posición, usando siempre posición = 0 (primer elemento). En el caso de la pila se tiene como “contrato” que el primero en entrar es el último en salir, entonces para apilar se usó lista insertar y para desapilar se usó lista quitar.