

Actividad 4.4

Lista doblemente enlazada y circular

Algoritmos y Estructuras de Datos

Tema 4: listas, pilas y colas

Modificar la actividad 4.3 para que la lista ahora sea doblemente enlazada y circular. Para el usuario de la clase no habrá ninguna diferencia de utilización en comparación con la clase de la actividad 4.3, por lo cual las cabeceras de los métodos públicos (nombre, número y tipo de parámetros, retornos) serán idénticas a los de la actividad 4.3.

En concreto, realizaremos las siguientes modificaciones:

- Ahora la estructura `Nodo` es diferente. Tiene un puntero tanto al anterior nodo como al siguiente.
- En la clase `ListaEnlazada` vamos a añadir dos atributos más: `posicionUltimoNodoAccedido` y `punteroUltimoNodoAccedido`. Representan la posición y el puntero del último nodo que se devolvió la última vez que se llamó a `getNode()`. Los pondremos a `-1` y a `NULL` respectivamente en el constructor. Igualmente, cada vez que llamemos a `eliminar()` o a `insertar()` también los pondremos a `-1` y a `NULL`, para evitar complicaciones (esto podría mejorarse mucho).
- Si el método `getNode()` tiene que devolver un nodo que es el inmediatamente siguiente o anterior al último nodo accedido (o incluso es el mismo nodo al que se accedió la última vez), usará los dos atributos descritos para acceder a él de forma inmediata, en $O(1)$, en vez de empezar por el principio. De este modo sucesivas llamadas a `getNode()` con elementos en posiciones sucesivas (como por ejemplo en `imprimirListaEnlazada()`, o en `concatenar()`, o en `buscar()`) tendrán una complejidad temporal de $O(1)$ cada una de las llamadas, en vez de $O(n)$. De este modo, por ejemplo, `imprimirListaEnlazada()` ahora será $O(n)$ en vez de $O(n^2)$, sin necesidad de haber tocado nada en el código interno de `imprimirListaEnlazada()`.
- En caso de que el nodo a buscar no esté pegado al último accedido, el método `getNode()` buscará “hacia delante” o “hacia atrás” según la posición del nodo cuyo puntero queramos obtener. Ej: si la lista tiene 100 elementos, y queremos acceder al nodo 30, iremos iterando “hacia delante” (nodo 0, nodo 1, nodo 2... así hasta el nodo 30). Sin embargo, si quisiéramos acceder al nodo 60, iremos iterando “hacia atrás” (nodo 0, nodo 99, nodo 98, nodo 97... hasta nodo 60), pues de esta manera pasamos por menos nodos que yendo hacia delante.
- Un método `concatenar()`, con la misma cabecera y características que en la actividad 4.2. Mejorará mucho su complejidad temporal debido a las mejoras en `getNode()`, descritas anteriormente.

- Un método “buscar()”, con la misma cabecera y características que en la actividad 4.2. Igualmente, su complejidad temporal permanecerá en $O(n)$ debido a las mejoras en `getNode()`, descritas anteriormente.

Para probarlo, amplía el main de la actividad 4.3 para probar también la concatenación y la búsqueda, tal cual como aparece en la siguiente prueba de caja negra:

```

C:\WINDOWS\system32\cmd.exe
Nueva ListaEnlazada creada:
n=0|ListaEnlazada=vacia
Inserto 10 con la lista vacia:
n=1|ListaEnlazada=10
Inserto 20 y 21 al final:
n=2|ListaEnlazada=10,20
n=3|ListaEnlazada=10,20,21
Inserto 30 al principio:
n=4|ListaEnlazada=30,10,20,21
Inserto 40 en la posicion 2:
n=5|ListaEnlazada=30,10,40,20,21
Elimino el primer elemento:
n=4|ListaEnlazada=10,40,20,21
Elimino el ultimo elemento:
n=3|ListaEnlazada=10,40,20
Elimino el elemento del medio:
n=2|ListaEnlazada=10,20
Elimino el 20 y el 10 para dejar la lista vacia:
n=1|ListaEnlazada=10
n=0|ListaEnlazada=vacia
Vuelvo a repetir las inserciones del principio:
Inserto 10 con la lista vacia:
n=1|ListaEnlazada=10
Inserto 20 y 21 al final:
n=2|ListaEnlazada=10,20
n=3|ListaEnlazada=10,20,21
Inserto 30 al principio:
n=4|ListaEnlazada=30,10,20,21
Inserto 40 en la posicion 2:
n=5|ListaEnlazada=30,10,40,20,21
Insertamos 50 y 60 al final:
n=7|ListaEnlazada=30,10,40,20,21,50,60
Elemento 0 es 30
Elemento 4 es 21
Elemento 2 es 40
Elemento 6 es 60
Cambio elementos 0, 4, 2 y 6 por sus inversos. El resultado es:
n=7|ListaEnlazada=-30,10,-40,20,-21,50,-60
Elemento -30 encontrado en posicion 0
Elemento 20 encontrado en posicion 3
Elemento -60 encontrado en posicion 6
Elemento 100 encontrado en posicion -1
Nueva lista2:
n=6|ListaEnlazada=500,501,502,503,504,505
Concatenamos la nueva lista a la vieja:
n=13|ListaEnlazada=-30,10,-40,20,-21,50,-60,500,501,502,503,504,505
Presione una tecla para continuar . . . ■

```