

Actividad 5.1

MergeSort

Algoritmos y Estructuras de Datos

Tema 5: ordenación

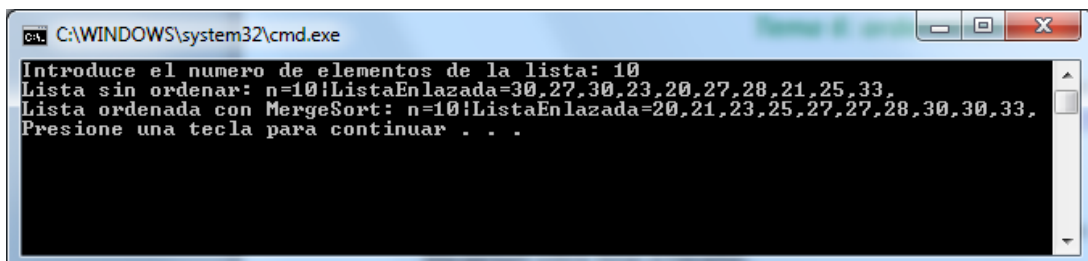
Añade en la actividad 4.4 resuelta un método “ordenar()” (sin parámetros y sin valor de retorno), que ordene la lista enlazada actual mediante el algoritmo MergeSort.

Ten en cuenta que:

- Como el método “ordenar()” no tiene parámetros ni devuelve nada, necesitamos un método privado recursivo para ordenar una lista por MergeSort. Ese método privado recibirá el puntero a una lista enlazada y la ordenará.
- Estamos ordenando una lista enlazada, no un vector
- El espacio adicional a emplear debe ser $O(1)$ para mezclar.
- La complejidad temporal de ordenar debe ser $O(n \lg n)$. La complejidad temporal de dividir la lista en dos partes puede ser $O(n)$. Tanto la complejidad espacial como temporal son requisitos imprescindibles del programa.
- Una vez ordenada, la lista debe seguir siendo doblemente enlazada y circular
- Hay que aprovechar todo lo que ya está hecho de la clase ListaEnlazada. De hecho, no hay que tocar ningún método excepto ordenar. No trabajes con nodos directamente, haz como si no supieras que existen.

Pruébalo con un main que haga lo siguiente:

1. Pregunta al usuario cuál será el tamaño de la lista
2. Genera una lista de ese tamaño, con números aleatorios entre 20 y 40.
3. Imprime la lista
4. Ordena la lista llamando al método ordenar()
5. Imprime la lista



```
C:\WINDOWS\system32\cmd.exe
Introduce el numero de elementos de la lista: 10
Lista sin ordenar: n=10!ListaEnlazada=30,27,30,23,20,27,28,21,25,33,
Lista ordenada con MergeSort: n=10!ListaEnlazada=20,21,23,25,27,27,28,30,30,33,
Presione una tecla para continuar . . .
```