Actividad 5.3

QuickSort avanzado

Algoritmos y Estructuras de Datos

Tema 5: ordenación

Mejora la actividad 5.2, reduciendo las constantes ocultas del tiempo que tarda QuickSort.

Primero mejoraremos la elección del pivote, pero manteniendo la complejidad temporal en O(1). En una lista enlazada simple la única opción sería tomar como pivote el primer elemento de la lista. Como nuestra lista es doblemente enlazada y circular, recuerda que acceder al último elemento es también O(1), por lo tanto tenemos más opciones para elegir un mejor pivote y que siga siendo O(1). Una opción es hacer la media entre el primero y el último. De este modo sigue siendo O(1) y además funciona bien para listas ordenadas o casi ordenadas.

De nuevo, deberemos diseñar el bucle que realiza el reparto de tal modo que nos aseguremos de que ninguna de las dos sublistas quede vacía. Como hemos elegido que el pivote sea la media entre el primero y el último, pueden ocurrir dos situaciones:

- Que el primero y el último sean valores diferentes. En cuyo caso no hay problema pues cada uno irá a una sublista diferente, y por lo tanto ninguna de ellas quedará vacía.
- Que ambos sean iguales. Entonces basta con que cada uno lo llevemos a la sublista más pequeña en el momento en que procesemos cada uno. Si la lista sólo tuviera dos elementos y los dos fueran iguales, y procesamos la lista desde el primer elemento hasta el último, el primero lo llevaríamos a la primera sublista y el segundo (el último) lo llevaríamos a la más pequeña, que ahora sería la segunda.
- Por lo tanto la solución que funciona en todos los casos es llevar cada elemento que sea igual que el pivote a la sublista más pequeña en ese momento. Te puedes asegurar de que funciona realizando un main “temporal” que cree una lista enlazada de dos elementos iguales, la ordene y la imprima.

Juntar las dos sublistas al final del algoritmo debe ahora tardar un tiempo de O(1). Como nuestra clase lista es doblemente enlazada y circular, debemos:

- Conectar doblemente el último elemento de la primera sublista con el primer elemento de la segunda sublista.
- Conectar doblemente el primer elemento de la primera sublista con el último elemento de la segunda sublista.
- Debemos configurar la lista resultado para que apunte al primer elemento de la primera sublista y ajustar su tamaño a la suma de los tamaños de las dos sublistas.

- Debemos configurar las dos sublistas para que ya no apunten a los nodos y para que su tamaño sea 0. Una vez hecho esto, las dos sublistas se podrán destruir sin que sus respectivos nodos (que ahora ya los tenemos en la lista resultado) se destruyan. En caso de que no lo hiciéramos así, el destructor de ambas listas destruiría todos los nodos.

El main comparará el tiempo que tardamos en ordenar la misma lista enlazada de tamaño introducido por teclado y componentes aleatorios con nuestros MergeSort y QuickSort avanzado. Fíjate en que, cuando la lista es pequeña, QuickSort puede incluso tardar más que MergeSort. Sin embargo, cuando la lista es grande, QuickSort tarda visiblemente menos que MergeSort. Esto se debe a que, aunque ambos tienen la misma complejidad temporal, en el caso medio QuickSort tiene constantes ocultas bastante menores que el MergeSort.