Recursive Sorting: Mergesort

CS 221 Programming Assignment

*“Simplicity is the ultimate sophistication.”*  
– Leonardo da Vinci

# Objectives

* Implement a recursive mergesort algorithm in Java.

# Tasks

Complete the Sort static utility class, using a recursive implementation of the mergesort sorting algorithm.

1. Complete the Sort class without modifying the public sort methods or the method signatures of the private mergesort methods. No new public methods should be added to the Sort class.
   * One version of the algorithm requires that elements in the given list be of a class type that implements the Comparable interface. Comparisons between elements are made with the objects' own compareTo method. Your Sort class will be tested with lists containing Integer objects, which implement Comparable.
   * The other version of the algorithm requires a Comparator object along with the list, to compare elements using the Comparator's compare method. The given test class defines a Comparator internally that it uses to compare Integers.
   * Your implementation must work with an instance of the provided WrappedDLL class, or for extra credit, an instance of your IUDoubleLinkedList class.
     + Your mergesort implementations should run in O(n log(n)) time. Do not write or call code that would make them worse than that. Therefore, you should avoid all indexed list methods, including the indexed ListIterator constructor.
     + For extra credit, you can replace all references to WrappedDLL with your IUDoubleLinkedList class. Again, do not write or call code that would make your algorithm run in worse than O(n log(n)) time. If you have list methods that are O(n) when they should be O(1), for example, it will negatively impact the performance of sorting.
2. Use the provided SortTester class to test your implementation. If you wish, you may add additional tests, but the given tests will be expected to pass as written.

# Mergesort Algorithm

1. If the list has more than one element, move the first half of the original list elements into a left list and the remaining elements into a right list.
2. Recursively mergesort both halves.
3. Compare the first element of each sorted half and move the smaller of the two elements to the end of the original list. Continue until all elements have been returned to the original list.

# Files

You will need the following files:

* [Sort.java](https://raw.githubusercontent.com/BoiseState/CS221-Public/master/assignments/SearchAndSort/files/Sort.java)
* [WrappedDLL.java](https://raw.githubusercontent.com/BoiseState/CS221-Public/master/assignments/SearchAndSort/files/WrappedDLL.java)
* [IndexedUnsortedList.java](https://raw.githubusercontent.com/BoiseState/CS221-Public/master/assignments/DataStructures/files/IndexedUnsortedList.java)
* [SortTester.java](https://raw.githubusercontent.com/BoiseState/CS221-Public/master/assignments/SearchAndSort/files/SortTester.java)

All of these files can be found in this [zip file](https://cs.boisestate.edu/~mvail/221/assignments/homeworks/SearchAndSort/files/Mergesort.zip).

# Extra Credit

For extra credit, do all of the following:

* Include your fully functional IUDoubleLinkedList and any necessary supporting files in your submission.
* Update SortTester to use IUDoubleLinkedList wherever WrappedDLL was being used.
* Update the newList() method in Sort to return a new IUDoubleLinkedList rather than a WrappedDLL.

# Grading

Points will be awarded according to the following breakdown:

| **Tasks** | **Points** |
| --- | --- |
| Required algorithm - sorts correctly | 20 |
| Design - uses recursion correctly and efficiently | 10 |
| Extra Credit: Works with (and internally uses) your IUDoubleLinkedList | up to 5 |

# Required Files

You should submit your copies of the following files:

* Sort.java (updated)
* SortTester.java
* IUDoubleLinkedList.java (if attempting extra credit) or WrappedDLL.java (if not attempting extra credit)
* IndexedUnsortedList.java *and any other files required to compile and run your program*

# Submission

Submit all files from the same directory. Do not include any unnecessary files.

Use the submission command given on your section's class web page from the directory containing your files.