Extra Credit Assignment
CS206
Fall 2015

Here are a few challenging problems in case you want to get better at implementation of data structures.

For each problem hand in the class code, the driver program code, and writeup describing your results.

Problem 1. Implement a circular double linked list from scratch. Your class should implement the SimplifiedList2 interface, and it should use your own Node class that is doubly linked. It is up to you to decide what fields to have in your CircularLinkedList class. In addition to the methods defined in SimplifiedList2, create a splice method, that takes a list and allows you to splice an arbitrary sized sublist into the current list at any point in the list.

For instance if I have the list x:

1,2,3,4

and I want to splice the list y into x from the 3rd item to the 10th item of y at index 2 of x:

y: 10,9,8,7,6,5,4,3,2,1,2,3,4,5

x.splice(y,2,10,2) would yield:

x: 1,2,8,7,6,5,4,3,2,1,3,4

and either:

y: 10,9,2,3,4,5

or

y: 10,9,8,7,6,5,4,3,2,1,2,3,4,5

depending on how you decide to implement splice.

Once you have implemented the class, write a driver class that creates 2 CircularLinkedList<Number>, One of <Float> and one of <Integer>. Each list should be populated with 10 random numbers, and then one should be spliced into the other at a random position. Your driver should first create each list, then print each list, then splice one list into the other, then print each list again.

Problem 2: Implement a Queue with a circular array instead of a linked list. You will need to write a grow method for when the size goes beyond your initial capacity. Write testing code to verify that your queue works correctly and resizes correctly.

Problem 3: Implement Mergesort. Create a class called Merge with one public static method called List<E> sort(List<E> sortMe). You can have as many private static helper methods as you want. The sort() method should be a recursive method that performs Merge Sort. You should write a driver for your merge sort algorithm to test that it works correctly in a variety of circumstances.