CS 0445 Spring 2022
Recitation Exercise 8

Introduction:

In our discussion of MergeSort, we noted that MergeSort does not sort in place and requires a temporary array for the merge process. In the author's mergeSort() code we noted that a single temp array is created in a non-recursive "stub" method and that single temp array is passed in to all of the recursive calls – being reused over and over for the various merge calls. Let's see what difference it will make if we instead make a temp array within the merge() method – to be used only within that single call. Note that this would cause many arrays to be allocated as the program runs, but we can size them to be only as big as needed – so we don't have to make each array size N.

Details:

File TextMerge.java contains the author's original version of mergeSort() plus a second version which I added, called mergeSortB(). Look at the code for mergeSortB() to see how it differs from the original mergeSort(). In particular, note that rather than creating a single temp array prior to the initial call of the recursive method, it creates a temp array within the mergeB() method, thus requiring a distinct temp array to be created each time the mergeB() method is called. Note also that the array created in mergeB() is only as big as the current call requires – which is (last – first + 1).

To compare these two versions of MergeSort, you should write a main program, CS445Rec8.java, which will do the following:

1) Ask the user to enter an array size and input it
2) Create two arrays of Integer equal in length to the value input by the user
3) Fill the arrays with random integers
   Note: These three steps are the same as those in Exercise 7
4) Time a single call to mergeSort() and a single call to mergeSortB(), using System.nanoTime() before and after and using the difference of the calls as the runtime.
5) Print out the results for each sort.

Once you have your program complete and working, run it with various array sizes to see the differences in the run-times. Try the following array sizes: 500000, 1000000, 2000000, 4000000, 8000000. Try a couple of runs for each array size and consider your results.

Toward the end of the recitation period, share your impressions and conclusions with your classmates. Is one version clearly superior to the other? Is one better after the array reaches a certain size? Explain your results and reasoning to the rest of the class.