Colleen Lemak
Professor Bowers
CPSC 223
19 October 2021

Write-Up Homework 4

ArraySeq Merge Sort

EmptySeqMergeSort : Tests if the empty function returns true if the array sequence is empty. Calls merge sort to make sure seq is still empty after.

OneElemMergeSort: Makes sure that before and after calling merge sort, after inserting one element, the element is in the correct location and the seq has the correct size.

TwoElemInOrderMergeSort: Inserts two elements in order and checks that they are in the correct positions and have the same values before and after calling merge sort.

TwoElemReverseMergeSort: Inserts two elements in reverse order and checks that they are in the correct positions before merge sort, and are sorted as expected after calling merge sort.

ThreeElemMergeSortCases: Makes three sequences in differing orders, and makes sure they were correctly sorted after calling merge sort.

LinkedSeq Merge Sort

EmptySeqMergeSort : Tests if the empty function returns true if the linked sequence is empty. Calls merge sort to make sure seq is still empty after.

OneElemMergeSort: Makes sure that before and after calling merge sort, after inserting one element, the element is in the correct location and the seq has the correct size.

TwoElemInOrderMergeSort: Inserts two elements in order and checks that they are in the correct positions and have the same values before and after calling merge sort.

TwoElemReverseMergeSort: Inserts two elements in reverse order and checks that they are in the correct positions before merge sort, and are sorted as expected after calling merge sort.

ThreeElemMergeSortCases: Makes three sequences in differing orders, and makes sure they were correctly sorted after calling merge sort.

ArraySeq Quick Sort

EmptySeqQuickSort: Tests if the empty function returns true if the array sequence is empty. Calls quick sort to make sure seq is still empty after.

OneElemQuickSort: Makes sure that before and after calling quick sort, after inserting one element, the element is in the correct location and the seq has the correct size.

TwoElemInOrderQuickSort: Inserts two elements in order and checks that they are in the correct positions and have the same values before and after calling quick sort.

TwoElemReverseQuickSort: Inserts two elements in reverse order and checks that they are in the correct positions before quick sort, and are sorted as expected after calling quick sort.

ThreeElemQuickSortCases: Makes three sequences in differing orders, and makes sure they were correctly sorted after calling quick sort.

LinkedSeq Quick Sort

EmptySeqQuickSort: Tests if the empty function returns true if the linked sequence is empty. Calls quick sort to make sure seq is still empty after.

OneElemQuickSort: Makes sure that before and after calling quick sort, after inserting one element, the element is in the correct location and the seq has the correct size.

TwoElemInOrderQuickSort: Inserts two elements in order and checks that they are in the correct positions and have the same values before and after calling quick sort.

TwoElemReverseQuickSort: Inserts two elements in reverse order and checks that they are in the correct positions before quick sort, and are sorted as expected after calling quick sort.

ThreeElemQuickSortCases: Makes three sequences in differing orders, and makes sure they were correctly sorted after calling quick sort.

4-Element Tests

(ArraySeq) FourElemMergeSortCases: Makes four sequences in differering orders based on the type of cases needed to consider for ArraySeq merge sort, and makes sure they were correctly sorted after calling merge sort.

(LinkedSeq) FourElemMergeSortCase: Makes four sequences in differering orders based on the type of cases needed to consider for LinkedSeq merge sort, and makes sure they were correctly sorted after calling merge sort.

(ArraySeq) FourElemQuickSortCases: Makes four sequences in differering orders based on the type of cases needed to consider for ArraySeq quick sort, and makes sure they were correctly sorted after calling quick sort.

(LinkedSeq) FourElemQuickSortCases: Makes four sequences in differering orders based on the type of cases needed to consider for LinkedSeq quick sort, and makes sure they were correctly sorted after calling quick sort.

4-Element Merge and Quick Sort Tests

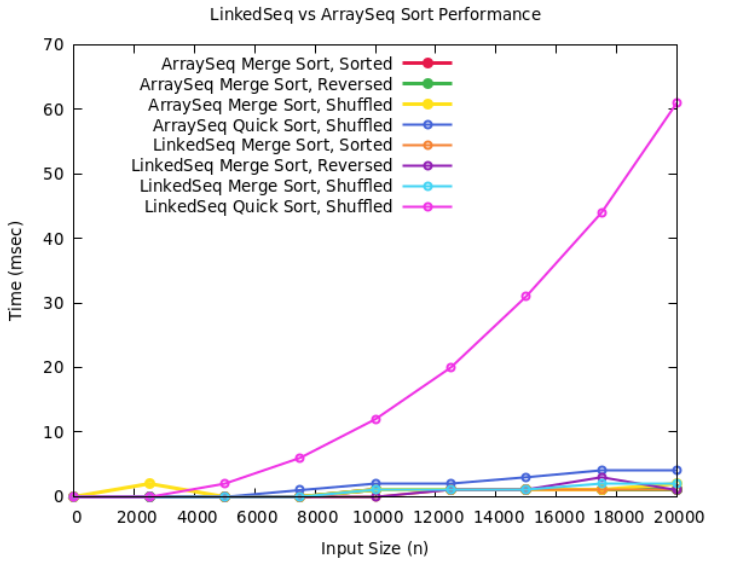
(ArraySeq) FunctionalityMergeSortCases: Tests the functionality of a sequence made by inserting, erasing, copying, and accessing values. Calls merge sort and checks if it is correctly sorting the modified sequence.

(LinkedSeq) FunctionalityMergeSortCases: Tests the functionality of a sequence made by inserting, erasing, copying, and accessing values. Calls merge sort and checks if it is correctly sorting the modified sequence.

(ArraySeq) FunctionalityQuickSortCases: Tests the functionality of a sequence made by inserting, erasing, copying, and accessing values. Calls quick sort and checks if it is correctly sorting the modified sequence.

(LinkedSeq) FunctionalityQuickSortCases: Tests the functionality of a sequence made by inserting, erasing, copying, and accessing values. Calls quick sort and checks if it is correctly sorting the modified sequence.

Performance Tests



In these performance tests, the most expensive, the pink outline for LinkedSeq Quick Sort, Shuffled, makes sense. Quick sort is usually efficient unless the list is randomly ordered because it uses a pivot to compare to all values, so there are more moving parts in this situation

compared to one where everything is sorted or even in reverse order. There will be more moving nodes in the shuffled LinkedSeq quick sort than any other test because of this added time.

Quick and merge sort on LinkedSeq were tricky, but after drawing it out and spending a lot of time on it, I got them passing tests. I had to consider different cases and what could possibly occur. In my tests, I inserted values that would include the variations specific for that type of sorting, which accounted for the majority of “atypical” cases we might encounter.