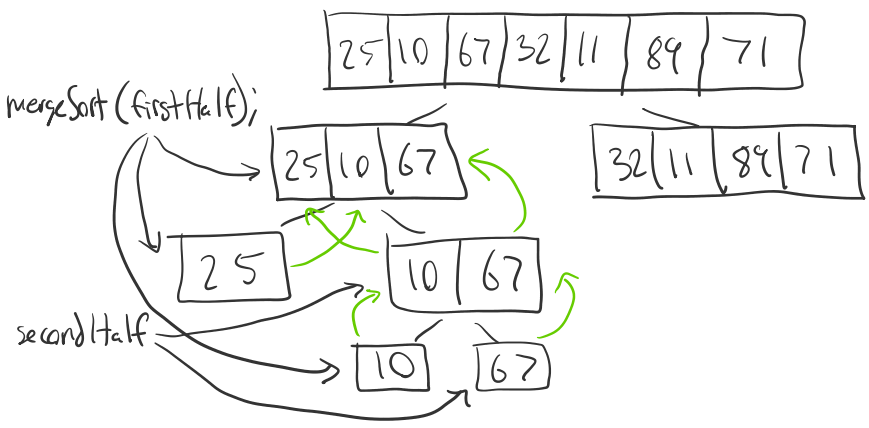
Merge Sort Analysis

Friday, October 6, 2023

Brooks Walsh

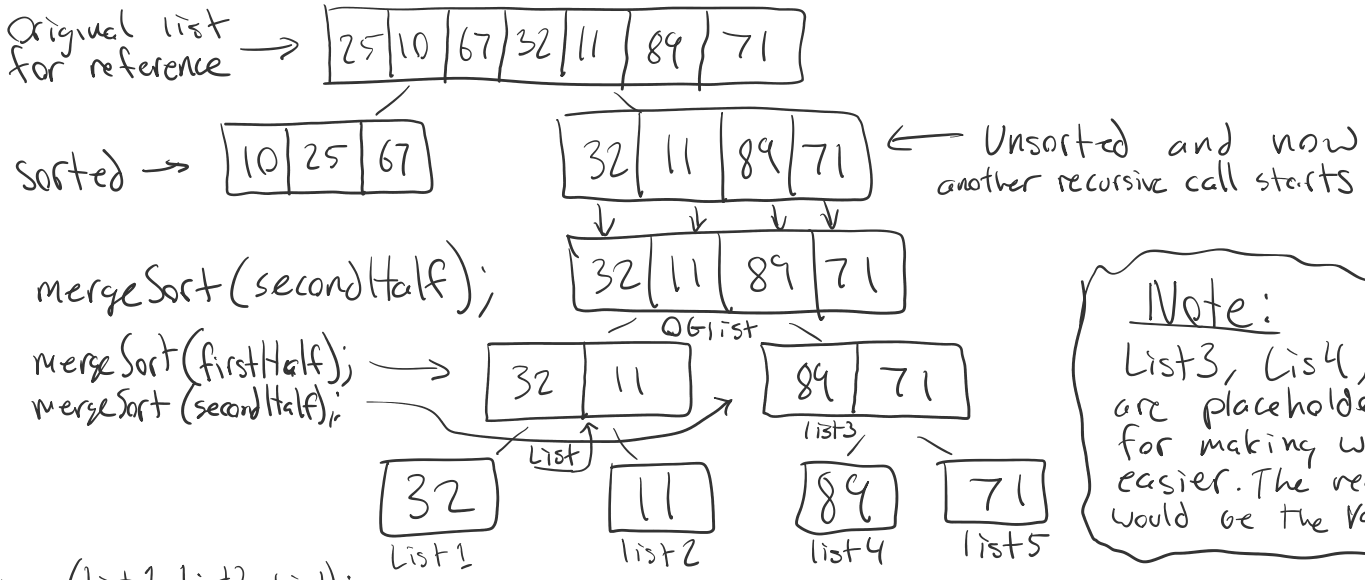
Initial Array: `int[] array = {25, 10, 67, 32, 11, 89, 71};`

Initial Call of MergeSort method: `mergeSort(array);`



Note: As this process happens twice nearly the exact same... I went through the first half quickly and went into more detail below

List1: [25] List2: [10, 67]. Compare the "head" of each list to merge them into one sorted list. Each "head" is tracked with an incrementing index.



Note: List3, List4, etc. are placeholders for making writing easier. The real names would be the var. names

`merge(list1, list2, list);`

32 < 11 so list is modified as 11 | 32 list6

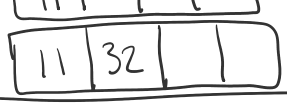
`merge(list4, list5, list3);`

89 < 71 so list3 is modified as 71 | 89 list7

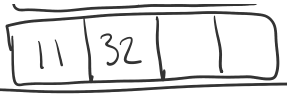
`merge(list6, list7, OGlist);`

11 < 71 so

32 < 71 so



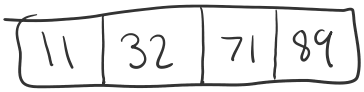
32 < 71 so



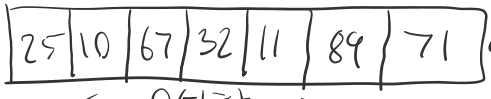
End of Page 1 (Ignore Overlap)

list6 is now empty so final while loop conditions apply and all elements of list7 are added to OGlist.

OGlist



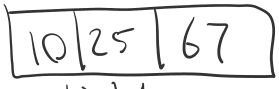
Original list for reference →



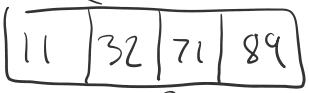
← still unsorted

OGlist

sorted →



list1

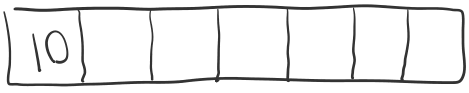


list2

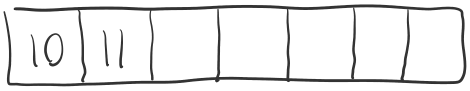
← sorted

merge(list1, list2, OGlist);

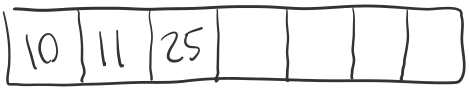
10 < 11 so → Inc. Left index



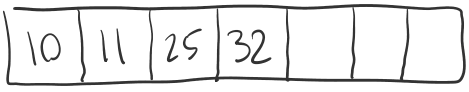
25 > 11 so → Inc. right index



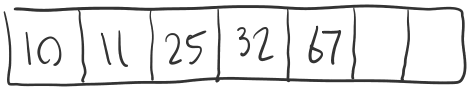
25 < 32 so → Inc. Left index



67 > 32 so → Inc right index



67 < 71 so → Inc. Left index

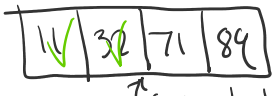


Note: The blank squares represent unsorted elements. In reality, there are elements in those positions, but they are now irrelevant given our broken-down and sorted sub-lists that are merging.

Sub-Note: OGlist index is incrementing each time a sorted element is added back

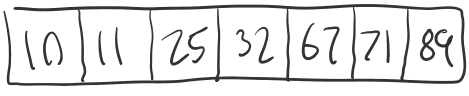
While loop condition fails because left index ≥ list1.length

Now, copies the final elements from whichever list's index is not at that lists length. In this case, list2's elements are moved



move 71 and 89 →

↑ current left index



fully sorted list!