John Carrabino
carrabij@oregonstate.edu
July 31st, 2016

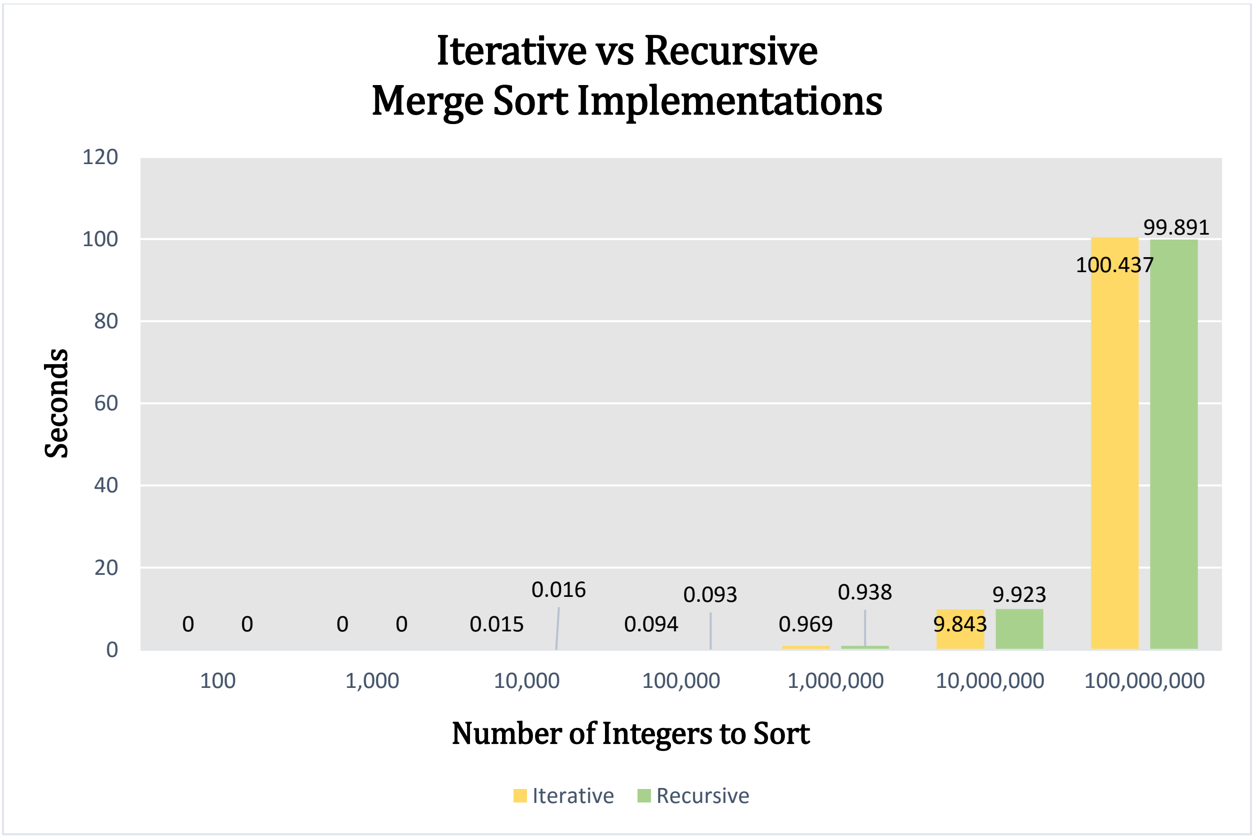
Lab H

TESTING:

For my driver program the user is prompted with to initialize a file with the amount of random integers of their choosing. The program then initializes the file, uploads the file contents into the arraya of an iterative merge sort object and a recursive merge sort object. After all of the integers from the file have been stored into their respective arrays the program then performs a benchmark test using the <ctime> library, keeping track of the start and stop times for the merge sort runtime of each object using the clock() function. The total runtime is then calculated by subtracting the start time from the stop time, then dividing the result by the CLOCKS_PER_SECOND variable form the <ctime> library in order to return the runtime of each function in seconds.

Test Case	Input	Test Function	Observed Outcomes
Sort a file with 100 random integers	File Name: "test.txt" Amount of Integers: 100	itMergeSort(); recMergeSort();	Iterative: 0.000 Recursive: 0.000
Sort a file with 1,000 random integers	File Name: "test.txt" Amount of Integers: 1,000	itMergeSort(); recMergeSort();	Iterative: 0.000 Recursive: 0.000
Sort a file with 10,000 random integers	File Name: "test.txt" Amount of Integers: 10,000	itMergeSort(); recMergeSort();	Iterative: 0.015 Recursive: 0.016
Sort a file with 100,000 random integers	File Name: "test.txt" Amount of Integers: 100,000	itMergeSort(); recMergeSort();	Iterative: 0.094 Recursive:0.093
Sort a file with 1,000,000 random integers	File Name: "test.txt" Amount of Integers: 1,000,000	itMergeSort(); recMergeSort();	Iterative: 0.969 Recursive: 0.938
Sort a file with 10,000,000 random integers	File Name: "test.txt" Amount of Integers: 10,000,000	itMergeSort(); recMergeSort();	Iterative: 9.843 Recursive: 9.923

Sort a file with 100,000,000 random integers	File Name: “test.txt” Amount of Integers: 100,000,000	itMergeSort(); recMergeSort();	Iterative: 100.437 Recursive: 99.891



As is clearly demonstrated from the graph above there really is not much of a difference between the runtime execution of recursive and iterative implementations of the merge-sort algorithm. The results shown above were not all of the tests that I conducted, but they follow the general trend of what was observed. I believe the runtime of both merge sort functions were similar because both implementations of the merge sort function have a Big O notation of $O(n \log_2 n)$ which is reflected in the relatively slow increase in execution time until the amount of digits being sorted were raised by several orders of magnitude. I also found that when I ran individual tests without using valgrind the functions executed much more quickly, which indicates how much memory is required to have valgrind on during run time.

Iterative and Recursive Merge Sort functions were from the following source: http://www.geeksforgeeks.org/iterative-merge-sort/