**Merge Sort Algorithm Analysis**

*Tyler Weiss*

*CSC 382 Data Structures and Algorithms*

*University of Advancing Technology*

Merge Sort Algorithm:

Merge sort is a divide and conquer algorithm. It takes a data series and splits it in half over and over again until there is a collection of singular nodes. In groups of two these nodes are compared and sorted giving you many small collections of sorted lists. This process continues until all lists are merged and the data is sorted.

Program Guide:

File: MergeSort.h

Description:

A recursive implementation of the merge sort algorithm using a custom doubly linked list.

Functions:

void mergeSortDLL(DLL\* dll)

This function takes a pointer to a doubly linked list and calls the mergeSort function. The merge sort function returns a pointer to the head node of a sorted list. After merge sort has completed acting upon the list the head and tail of the list passed in will be set to the new appropriate nodes.

Node\* mergeSort(Node\* head)

The head node of a doubly linked list is passed in. If the node is the only node in the list or the list is empty the head node is returned. If not the list is split in half using the split function and a recursive call on of mergeSort is called on the two lists. Following the recursive call the merge function is called to sort the list passed to mergeSort and the head node of the merge list is returned.

Node\* split(Node\* head)

This function splits a doubly linked list into two and return the a pointer to the head of the second list.

Node\* merge(Node\* first, Node\* second)

This function compares the nodes of the first and second doubly linked list that is passed in to the function. Comparison is sequential and effectively merges to lists into a single sorted list. The head of the new sorted list is returned.

File: main.cpp

Description:

The main function is the dashboard for a user to created and sorted randomly generated doubly linked lists. Initially the user is prompted to choose the size of the list to create. Then the user is prompted to choose int, double or char to designate the datatype of the node values within the list. A list is created and printed to screen. Then that same list is sorted and printed to screen. The time taken to sort the list is recorded, using the high\_resolution\_clock from the chrono library, and is printed to the screen in microseconds. At this point the user is asked whether or not to the program should run again. If the ‘y’ is entered the program loops through a second time and if ‘n’ is entered the program terminates.

Analysis**:**

Using the “big O cheat sheet” as a reference it was determined that O(nlog(n)) is the worst case scenario measurement representing the runtime of merge algorithm. Since merge sort will do the same amount of operations whether a list is sorted or not the any scenario run on the doubly linked list using this algorithm will yield a similar result. The excel spread sheet included contains the data collected from trials using integer lists of 10, 100 and 1000 nodes. The data sets are completely random and out of order when created. Five runs were performed on each list and the average was taken. Using the averages of the merge sort algorithm a graph was created and evaluated. The graph uses a log base 10 scale. If the data sets yield a nlog(n) trend then the graph will be linear. The trend line of the raw data is linear and supports the conclusion that the algorithm indeed produces a time complexity of O(nlog(n)).