# **Colin Powers**

# **Algorithms Merge Sort; Project 1**

## Executive Summary

The program I constructed was a simple merge sort for linked lists in C++. Merge sort is excellent and fast sorting. It is especially great for a linked list. What I did was create a small program that took in 1000 nodes then broke the node into 2 pieces and in my second experiment, it was broken into 3 pieces.   
  
I personally thought a 2 split rather than a 3 split would run faster and be better because there is less work the program has to do. When the program breaks it into 3 parts it has to sort through the same numbers, but 3 times, rather than the same numbers 2 times.

## Experiment

I wrote the program in C++ and just first created a linked list class.  
  
struct node{

int data;

node \*next; };

node\* NewNode(int d)

{

struct node \*temp = new node;

temp->data = d;

temp->next = NULL;

}  
  
First I split the linked lists into 2, then later I split them into 3. Below is the code for the 3 split:  
void LinkedList::merge(Node \*\*head) {

Node \*first = new Node(NULL);

Node \*second = new Node(NULL);

Node \*third = new Node(NULL);

first = \*head;

third = \*head;

if (first == NULL || first->next == NULL) {

return;

}

else {

while (first->next != NULL) {

first = first->next;

if (first->next != NULL) {

third = third->next;

first = first->next;

}

}

second = third->next;

third->next = NULL;

first = \*head;

}

merge(&first);

merge(&second);

\*head = MergeLinkedLists(first, second);

}

This code will split the linked list into 3 pieces then recursively sort the lists.  
  
Once the list is broken into 3 parts then recursively sorted, it will call the next function MergeLinkedList to recompile the list back into one single list. In Int Main() A new linked List is initialized and populated with the 1000 integers needed for the nodes. Once this is completed the program displays all the raw data.

## Results

As a way to make sure everything was completed and ran properly, I ran the program multiple times. My first 2 split ran in under 5.073 seconds every single time. When I ran the 3 split it ran at 6.034 seconds.

## Conclusion

As I thought in my opening summary, the 3 split did take more time, but if we’re counting in seconds, not by much. Merge sort shows itself to be incredibly reliable and fast in this, even when pressured with more work, it still pumps out great results.

## Appendix 1

The raw data results show that with the 2 split, the exact run time of the merge sort linked list was 5.073.  
The raw data results show that with 3 way split, the exact run time of the merge sort linked list was 6.034.