**CLC 5 Sorting Algorithms**

Bubble Sort

1. *Is the sort stable?*

Bubble sort is stable because an adjacent pair of identical elements will never be swapped because they are identical, or close enough so that they would be considered in order right from the start.

1. *Is the sort easily adaptable to singly linked lists?*

Bubble sort is good for singly linked list because of the way the data is referenced. It is easy to start at the head node and then iterate through the rest, swapping as needing and assigning new pointers.

1. *Is the sort easily adaptable to doubly linked lists?*

Bubble sort is good with doubly linked lists because the lowest values bubble up near the front of the dataset, which makes it easy to set the new previous pointers.

1. *What is the worst case distribution for the sort?*

Worst case for a Bubble sort is when the data is in backwards order, meaning that every single item will have to be checked and then subsequently moved into place.

1. *What is the best case distribution for the sort?*

Best case for a Bubble Sort is that the data is already nearly sorted so that the algorithm does not have to move many values around.

1. *Draw a decision tree for the sort for the data set [a b c d].*

Check A < B

If Yes

If No

Swap

A and B

Check B < C

Check C < D

If No

If Yes

If No

If Yes

Swap

B and C

Swap

C and D

If (Yes) every time

Close Sort

Else

Start from Beginning

Merge Sort

1. *Is the sort stable?*

Merge Sort is stable because even though the elements are broken up and sorted according to size, two nearly identical elements are going to end up back in the same order as they were originally. This is because of the way Merge Sort breaks the array into smaller halves.

1. *Is the sort easily adaptable to singly linked lists?*

Merge Sort is generally preferred when sorting list. It works well with the pointers and slow random accessed performance of a linked list. Given that nodes are only ever swapped one location at each step the pointer to next is easy to maintain.

1. *Is the sort easily adaptable to doubly linked lists?*

Merge Sort can work well for a doubly linked list because the pointers for next and previous are never shifted randomly. The data set is only ever swapped on location at a time, which makes it a reliable manner of maintaining a doubly linked list. There are no large jumps which would make the pointer inaccurate or impossible to determine the next or previous.

1. *What is the worst case distribution for the sort?*

Merge Sort has no worst case distribution because of the way it is implemented. It breaks down the entire dataset regardless of its distribution and then sorts it regardless of the way it is inputted.

1. *What is the best case distribution for the sort?*

Merge Sort has no best case distribution because of the way it is implemented. It breaks down the entire dataset regardless of its distribution and then sorts it regardless of the way it is inputted.

1. *Draw a decision tree for the sort for the data set [a b c d].*

Start with A, B, C, D

Break into Subsections

Subsection A, B

Subsection C, D

A

B

C

D

Compare A and B to determine which is smaller

Compare C and D to determine which is smaller

If A < B

Merge to create A, B

If B < A

Merge to create B, A

If C < D

Merge to create C, D

If D < C

Merge to create D, C

Compare AB and CD to determine which is smaller

If A < B < C < D Marge to create A/B/C/D

If A < C < B < D Marge to create A/C/B/D

If C < A < B < D Marge to create C/A/B/D

If C < A < D < B Marge to create C/A/D/B

I will leave out the other numerous possibilities that can occur based upon the relative size of A, B, C, and D.