Design Doc

Initial Design

Objective: The objective of this assignment is to get familiarized with sorting algorithms such as Quicksort, Heap sort, Shell sort, and Batcher Sort. The implementation of these sorts should be done with time complexity in mind and having an efficient program.

Files Included:

- ☐ • batcher.c implements Batcher Sort.
- ☐ • batcher.h specifies the interface to batcher.c.
- ☐ • shell.c implements Shell Sort.
- ☐ • shell.h specifies the interface to shell.c.
- ☐ • gaps.h provides a gap sequence to be used by Shell sort.
- ☐ • heap.c implements Heap Sort.
- ☐ • heap.h specifies the interface to heap.c.
- ☐ • quick.c implements recursive Quicksort.
- ☐ • quick.h specifies the interface to quick.c.
- ☐ • set.c implements bit-wise Set operations.
- ☐ • set.h specifies the interface to set.c.
- ☐ • stats.c implements the statistics module.
- ☐ • stats.h specifies the interface to the statistics module.
- ☐ • sorting.c contains main() and may contain any other functions necessary to complete the assignment.
- ☐ Makefile
- ☐ README.md
- ☐ DESIGN.pdf
- ☐ WRITEUP.pdf

Quick Sort:

- Quick sort file will include the standard library. The function will take in three arguments of the type stats and uint32_t. The implementation of this sorting method will consist of having two functions. The partition function will take in arguments to place elements less than the pivot into the left side of the array and elements greater than or equal to the pivot to the right. It will also return the index at which the partition occurs. The second function is the quick_sorter function. This function will have an off that checks the high elements and the low elements and passes them to partition and uses recursion to call partition to arrange the elements in the array.

Shell Sort:

- Shell sort file will include the standard library. This file will only consist of one function that will have a nested for loop that iterates through the gaps in the array and using a

while loop will check if the high or low value is less than or greater than or equal to the gap.

Heap Sort:

- I have not yet understood the full functionality of this sorting algorithm but to my current knowledge, I will have to implement the functions of max_child and fix_heap and alo build_heap and heap_sort.
    - Max_child
        - This function will take in arguments to calculate the right and left values of the heap.
    - Fix_heap
        - This function is used to check if the mother and father values of the heap are less than the last and not found. This function will also use the max_child function inside it.
    - Build_heap
        - This function will build the heap by using a for loop that will call fix_heap to set it right.
    - Heap_sort
        - This function will iterate through the heap and call on fix_heap after every iteration.

Batcher Sort

- Batches sort will require two functions. These are called the comparator function and the batcher_sort function. The sorting file is to create a sorting network also known as a comparator network. For the first function I will have an if condition that compares the index values. And sets them equal to each other. Still in the works of figuring out this function.

- First create gaps.h
- Then run gaps.h.py and put the output to gaps.h file.