Design Doc

Final 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:

- The implementation of this function requires a nested for loop that iterates over the gap in the gaps file. Within the second loop we will get a tmp var to i value and a temp variable to the array index. Then the next while loop will run with the conduction j greater than or equal to gap and it will also have temp less than array value -gap

    - Within the loop i'll set the array index value of jk to the value of j -gaps
    - Then i will subtract j-gap and set it to j
    - Theni wills et the array j value to temp

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. This function will have a left and right value that are set to 2*left and left +1. Then a if condition will have a right value equal to last and a right is greater than to a left and will return right.
    - 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. It will have a while loop that ks mother less than or equal to last and will divide it by 2.
            - Within the loop an if will check if the mother is less than great. Then it will swap the values and set mother equal to great.
            - Else it will return true.
    - 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. This function will have a for leaf in range from last to first -1. Within the loop I will switch the first and leaf.

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.
- For this function I will have a comparator function that compares two index values in an array.
- Then I will have a batcher_sort function that will use a if statement to check if the length of the array is 0. If so then it will break
- Then I will have a nested while loop. They will have the first odiction of P greater than 0 and d greater than 0 for the inner loop.

Sorting.c

- For the sorting .c file I have a cases that present each flag and they will have designated action attached to them
    - Then I will have an if conduction that will toggle a true or false for each of the flags. Once the flags are true and then they will fund a for loop to run the function for a number of iterations and use a print statement to run at each other iteration.

Set.c

- My set.c file is the same file I used during last quarter's class. I used my own code and modified it to meet the assignment's needs.

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.