Assignment 3 DESIGN.pdf
Yash Sharma

Description of Program:

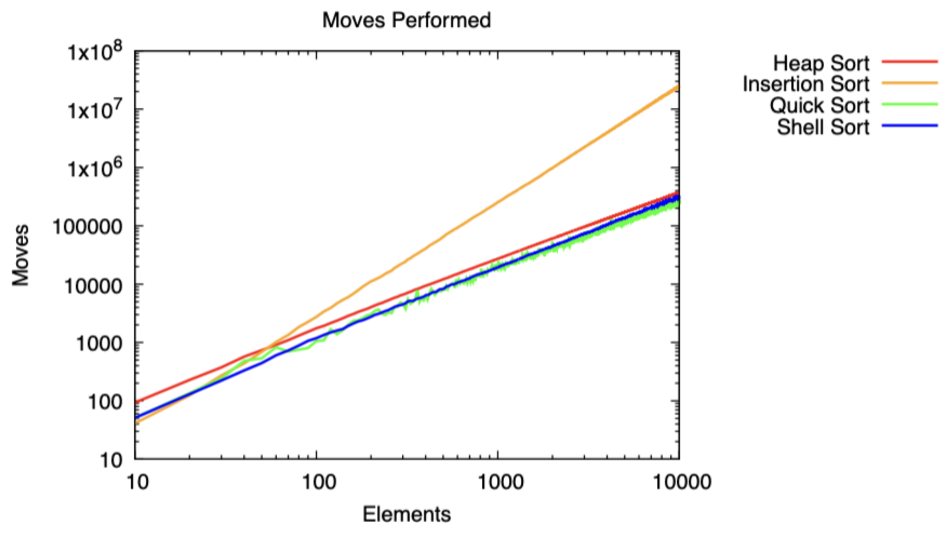
This assignment requires us to implement sorting algorithms like Insertion Sort, Batcher Sort, Heap Sort, and recursive Quicksort for the purpose of us getting fully familiarized with each sorting algorithm. Through this assignment, the program will also sort data using comparisons based on the number of elements needed to be sorted. For example, if the set of elements to be sorted is small, a counting sort is rather to be implemented as it would count the number of occurrences of each element in an array. In addition, by developing this program we will get a feel for computational complexity.

Files to be included in directory "asgn2":

1. batcher.c:
    ○ This file contains the implementations of Batcher Sort.
2. batcher.h:
    ○ This file contains the function interface to batcher.c
3. insert.c:
    ○ This file contains the implementation of Insertion Sort.
4. insert.h:
    ○ This file contains the function interface to insert.c
5. heap.c:
    ○ This file contains the implementation of the heap.c file.
6. heap.h:
    ○ This file contains the interface of the heap.c file.
7. insert.c:
    ○ This file contains the implementation of Insertion Sort.
8. quick.c:
    ○ This file contains the implementation of Quicksort.
9. quick.h:
    ○ This file specifies the interface to quick.c
10. set.h:
    ○ This file implements and specifies the interface for the set ADT.
11. stats.c:

  - This file contains the implementation of the statistics module.

12. stats.h:
    - This file specifies the interface to the statistics module.
13. sorting.c:
    - This file contains the implementation of the main function. It also contains any other functions necessary to complete the assignment.
14. Makefile:
    - A file that formats program into clang-format and compiles it into program executables with make/make all from Makefile
    - Additionally, make clean from Makefile must remove compiler-generated files (such as the executables)
15. README.md:
    - This file describes how to use the program and MakeFile. It will also list and explain any command-line options the program accepts.
16. DESIGN.pdf:
    - This describes the design for the program thoroughly with pseudocode and descriptions.
17. WRITEUP.pdf:
    - This describes the program thoroughly with graphs displaying the integrated values for each number of partitions. The graphs will be generated using gnuplot and other UNIX tools. It will also include an analysis of the graphs and any lessons learned from floating-point numbers.



Pseudocode / Structure:

batcher.c

Include header files

A function for bit length that counts the number of bits being processed

A function for comparator that takes in stats, the array list, and two integers (x,y)

If the array of x is greater than that of y, then swap the two arrays of x and y with each other

Create a function for batcher sorts which take in stats, an array list for parameters, and length

If the length of the array is 0, return the array

Initialize the variables n, t, q, r, and d

Create a while loop that checks if p is greater than 0 and sets variables equal to a parameter

Create another while that if d is greater than 0, it should check in a for loop if the value = r

If the value is equal, then compare values for stats, the array, the iterator and d

insert.c

Include header files

Create an insertion sort function that takes in an Array A in increasing order

that checks if A is in the list

Make a for loop that checks whether i is in a range between 1 and the length of the array

Set one variable equal to i

Set a temp variable equal to an array with the index inside

Create a while loop that checks if a variable is greater than 0 and if the index of A is less than temp

If so, then subtract 1 from the index of j in the array

Subtract and equal the variable to 1

Set the array of j equal to the movement of stats and temp

Return array of temp variable

<u>Heap.c</u>

Include header files

Create a max child function that uses an array to sort through first and last names

Initialize variables for left and right

Set an if statement so that if the last is greater than or equal to the right and the Array of right is greater than left, then return right

Return left

Create a function to fix heap under an array taking in first and last names from a list

Initialize variables for found, mother, and great

Set a while loop that checks for the arrays if they are greater than mother and not found

Set an if statement that checks for the array index of mother and great

Else return found equal to true

Create a function to build heap using parameters stats, array, first and last from a list

Set a for loop that iterates through father and divides last by 2, and runs until father is greater than first - 1 and subtracts from father until it gets there

Call fix heap function

Create a function to run heap sort that calls the heap and fixes it within the same function

Set variables first and last

Call build heap function for parameters stats, array, first, and last

Set a for loop to check whether leaf is equal to last and runs until leaf is greater than first, subtracting 1 from leaf on each iteration

Swap the values of the two arrays we are comparing

Call the fix heap function

Quick.c

Include header files

Set a function partition that contains parameters: stats, array, low, and high

Initialize variables

Set a for loop to check whether j = low and to run it till j is less than high, incrementing by 1 each time

Compare the two arrays i and j under the for loop

Add to the iterator if i and j condition is met

Swap the arrays

Return the iterator

Create a function for quick sorter containing the same parameters as partition

```
If low is less than high,
    Call the partition function and set it equal to a variable p
    Call the quick sorter function and set the parameters to check p - 1 and p + 1
Return statement
```

```
Create a quick sort function that takes in the parameters: stats, array, and length
    Call the quick sorter function
```

Stats.c

Include header files

Create a stats structure that takes in the array of uint32_ts to sort as the first parameter and the length of the array as the second parameter

Sorting.c

Include header files

Create a usage function to list all the usage cases and options a user can select while running ./sorting

Initialize a set and include all 4 sorts in the list

Create a main function that initializes all the variables and creates an empty set

Create a while loop that uses get opt and switches for each case within the usage function

At the end of the cases, create a default switch case that sends the user to the usage case if reached

Include a statement for malloc with initializing the array

Create an if statement that checks if the sort is a member in the set list and if so, then call that sort function within the for loop

Print out the results for each sort

Repeat process for the remaining 3 sorts

Return 0

Credit:

- I attended Eugene's section on 1/21/21, which helped give me general guidance on how to approach this lab, as well as sketch out how the program runs.
- I used the pseudocode from the asgn3 documentation
- I also used the asgn3 documentation and reviewed the pseudocode provided by Elmer in discord.