Assignment 3

Sorting: Putting your affairs in order

DESIGN.pdf

Reuben T. Chavez

January 27, 2022

Files in asgn2

Brief Description of the following files are as followed:

- **sorting.c**: Program that contains main() and when called run specfied sorting algorithm
- **batcher.c**: Implements the Batcher Sort algorithm
- **batcher.h**: Header File to specify the Batcher Sort algorithm
- **insert.c**: Implements the Insertion Sort algorithm
- **insert.h**: Header File to specify the Insertion Sort algorithm
- **heap.c**: Implements the Heap Sort algorithm
- **heap.h**: Header File to specify the Heap Sort algorithm
- **quick.c**: Implements the Quick Sort algorithm
- **quick.h**: Header File to specify the Quick Sort algorithm
- **set.c**: File that keep track of all the command-line options
- **set.h**: Header File to specify the interface for the set ADT
- **stats.c**: Implements the Statistics module
- **stats.h**: Header File to specify the Statistic module
- **Makefile**: File that stores commands to complies the sorting.c be executable programs , format the all the .c and .h files in clang format, and removes any unnecessary binary files before the program, through the command make clean
- **README.md**: Marks Down File that briefly describes how the program can be run and built on the terminal, which command-line options the program accepts, and error handling on the program
- **DESIGN.pdf**: A .pdf file that covers the purpose the program, has a lay out and structure of the program, with clear description/explanation on how each part works

- **WRITEUP.pdf**: A .pdf file that contains illustrations of the graphs produced through the program to analyze the how the sorting algorithms work, the conditions when the work at their best/worst, graphs noting the various run times for different arrays along with analysis and conclusion on the graphs

Pseudeocode

```
#Libaries
import stats
import sets
import batcher
import heap
import insert
import quick
import math
import random
import time

#define {heap, batcher,insert, quick} algorithm;

# Psuedocode for Sorting

## Usage ##
function usage is
    input: executable
    output: void

    print(
    "Synopsis of Sorting\n"

    "Usage of Sorting\n"

    "Options for Sorting\n"
    )
function print array is
    input: array , array length number of elements
    output: nothing

    if the array length < number of elements
       print items in array in increments of five up to number of
           elements
    else
        print entire array in increments of five

function populate array is
    input: the seed, array length, array
    output: nothing
```

```
    insert seed in srandom

    for the size of n do
         create random number
         bit mask to keep 30 bits
         insert number into array
    return

function main is
    input : argument count argc and argument vector argv
    output: zero to exit program

    initialize set s to be empty
    initialize seed to 13371453
    initialize size to 100
    initialize elements to 100
    while getting commands from command line do
        switch command:
            case all:
                set s is equal to set algorithm
            case heap:
                insert heap into set s
            case batcher :
                insert batcher into set s
            case insert:
                the algorithm insert is in the set s
            case quick:
                insert quick into set
            case seed:
                create random number staring at seed
            case size:
                create size of the array
            case elements:

                number of element to print out the array

                number of element to print out the array

            default help:
                prints usage and ends program

    create pointer A

    initialize Stat st
```

```
initialize stat compare to 0
initialize stat moves to 0

for item in the algorithm set do
    if item is in set s:
        switch item:

            case heap:
               set pointer A to empty array
               populate array A
               call the heap algorithm with stat st array A size n
               prints the the algorithm number of elements number
                   of moves and number of compares
               prints the sorted array A
               reset stat
               remove all unnecessary bits

            case batcher:
               set pointer A to empty array
               populate array A
               call the heap algorithm with stat st array A size n
               prints the the algorithm number of elements number
                   of moves and number of compares
               prints the sorted array A
               reset stat
               remove all unnecessary bits

            case insert:
               set pointer A to empty array
               populate array A
               call the heap algorithm with stat st array A size n
               prints the the algorithm number of elements number
                   of moves and number of compares
               prints the sorted array A
               reset stat
               remove all unnecessary bits

            case quick:
               set pointer A to empty array
               populate array A
               call the heap algorithm with stat st array A size n
               prints the the algorithm number of elements number
                   of moves and number of compares
               prints the sorted array A
               reset stat
```

```
                remove all unnecessary bits

    return 0
```

The Libraries located at the top are the header files and standard libraries that are called into sorting.c to: create a random number generator, keep track of the terminal commands when calling the algorithms, keep track of the total number of moves and comparisons when calling the sorting algorithms, and calls over the other sorting algorithms.
The first function, named usage, only prints sorting.c 's synopsis, usage, and options. It only takes the input of the executable file since it prints it out when the function while it returns nothing because it's a void function. The synopsis is that sorting.c is a program that calls multiple sorting algorithms. The usage of sorting.c is to tell which characters can be called on the command line. While the options for sorting.c give more details about the parameter and the utility of each flag.
The print array function is a simple void function that only prints the number of elements within the array, depending on the input. When the user inputs the number of elements to be greater than the array size, then the entirety of the array is printed. If that is not the case, the array is printed normally, where the array is printed out to the number of elements.
The populate array function is used to create a random array of unsorted numbers so that the sorting algorithm can organize it. It takes the input of seed and array length to create an array of said length. The function doesn't return anything since functions in c can't return arrays, so thats' the reason why it also takes an array as input. This function uses the input seed to get the same sequence of numbers whenever random is called so that the array will always have the same integers.
The main function is used to keep track of which sorting algorithms to call, then creates an array to be sorted by all the called sorting algorithms. For the inputs, the main function takes in the argument count and argument vector to get the inputs from the command lines when sorting.c is called. It outputs zero to signify the program. The main function starts by initializing an empty set to determine if either all the functions are being called, one or more functions are being called, or none of the functions are being called. The seed, array size, and elements are initialized as 13371453,100, and 100, respectively. The seed is the deterministic starting location for the random number generator. The array size is the initial array size, and the element is an integer that tells how many elements from the array should be printed out. The first while loop and switch case is

used to add onto the set s and change the seed, size and, element variable, respectively, to the command line inputs. The first five cases change the contents of the empty set s. Depending on which case it adds on to the set s to contain either one or more items from the enum ALGORITHMS, HEAP, BATCHER, INSERT, QUICK. In the case seed, the srand() is used to create a random number and takes the user input to create a random number using the strtoul command. In the case size, the array size variable is changed to the user's input using the strtol. In the default case, the program calls usage and then ends the program. After initializing the array pointer and stat struct, the following for loop is used to iterate through the set algorithm. The if statement that follows is used as a reference to make sure only the algorithms being called from the command line are being used called. Each case follows the same procedure when calling the algorithm following the switch statement. The array pointer is initialized to a calloc with the specified array size to be filled with zeros. After populating the array pointer with the specified seed, the sorting algorithm is called. Then what's printed is the algorithm name, number of moves, and comparisons while running—followed by the printed order sequence of elements within the array with the print array. The stat struct is reset, and A is deallocated. After repeating the same process for any of the other called cases, the program is terminated.