

DESIGN - Assignment 3

Brian Nguyen

October 14, 2021

1 Description

This program will contain a library of files with functions that implements sorting algorithms in C using the methods: Insertion, Shell, Heap, and Quick sort. It also contains a test harness that prints out an n-element array along with statistics (moves and comparisons) about the sorting method.

2 Files

1. insert.c

- This source file contains the code that sorts an array using Insertion Sort.

2. heap.c

- This source file contains the code that sorts an array using Heap Sort.

3. quick.c

- This source file contains the code that sorts an array using Quick Sort.

4. shell.c

- This source file contains the code that sorts an array using Shell Sort.

5. sorting.c

- This source file contains the main() function that return/print the sorts in action and its statistics.

6. stats.c

- This source file contains the code that returns the moves and comparisons the sorts make.

7. insert.h

- This header file contains the initialized function for Insertion Sort.

8. heap.h

- This header file contains the initialized function for Heap Sort.

9. quick.h

- This header file contains the initialized function for Quick Sort.

10. shell.h

- This header file contains the initialized function for Shell Sort.

11. stats.h

- This header file contains the initialized functions and variables for stats.c.

12. set.h

- This header file contains the initialized functions for Sets used for sorting.c and stats.c.

13. Makefile

- This make file contains the code that builds and compiles the sorting library program to be run. It also cleans all compiler generated files and formats the code to be submitted.

14. README.md

- This markdown file describes the program, how to build it, how to run it, and also lists and explains all the command-line options that the sorting program accepts. It also documents any false positives given by scan-build.

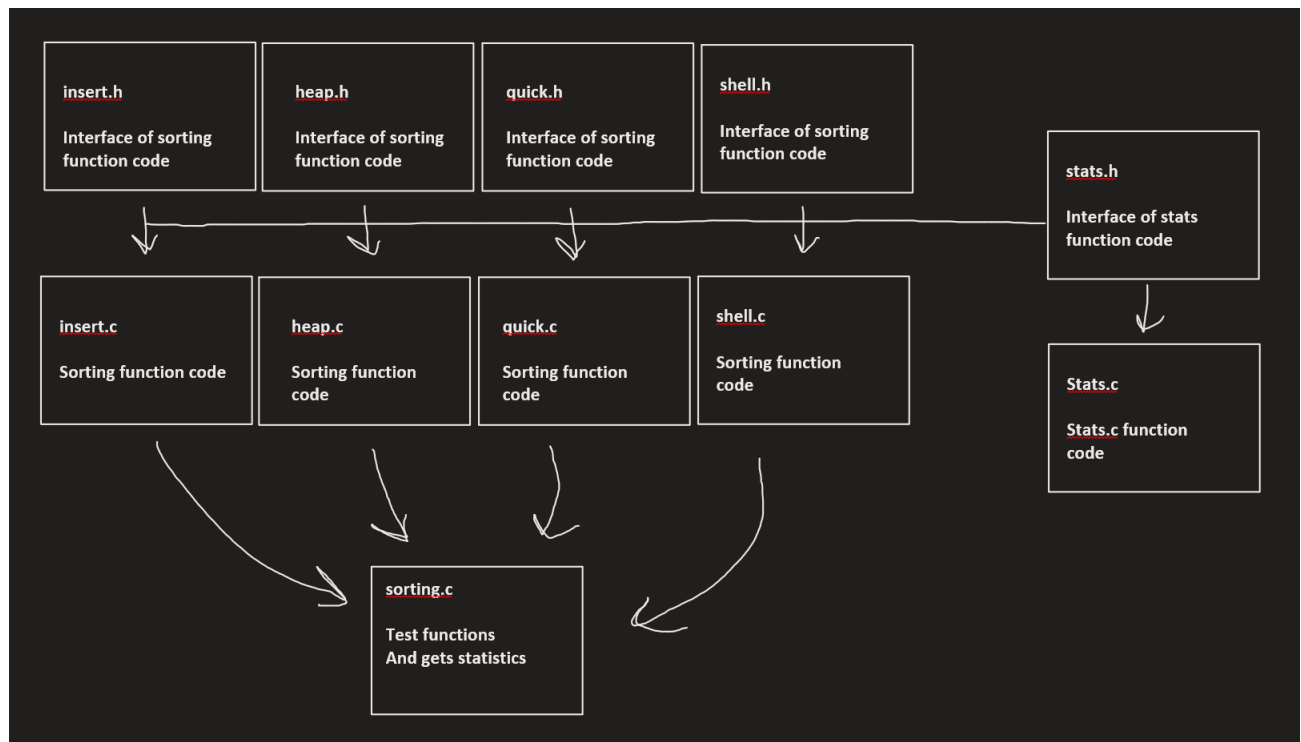
15. DESIGN.pdf

- This pdf is the manual that explains the program, files included, layout or structure, and pseudo-code of the sorting library.

16. WRITEUP.pdf

- This pdf is a scientific writeup made up of graphs and explanations by the gnuplot tool which shows the moves and elements of each sort algorithm and its performance.

3 Structure



4 Pseudo-code

All pseudo code is based off asgn3.pdf python code

4.1 Insertion Sort

define insertion_sort

```
for loop using i from 0 to array length
set variable current index element to i
set temp to i index element array (move value to a temp var to store)
while loop comparing values
    current index array element = previous index array element
    decrement j
set current index array element to temp (move it back to correct spot)
```

4.2 Heap Sort

define gaps

```
for loop using i from max gap length to 0 (Decrement)
```

"yield" gap to use

define shell_sort

```
for loop using gap from 0 to array length
for loop using i from gap to array length
set current index element to i
set temp to i indexed element array (move value to a temp var to store)
while loop comparing values
set current index element array
    current index array element = gap diff index array element (move it back to correct spot)
    decrement j by gap
set current index array element to temp
```

4.3 Shell Sort

define max_child

```
set left child to 2 times first
set right child to left plus 1
if right is bigger
    return right
return left
```

def fix_heap

```
set found to false
set mother to first
set great to max child of mother and last
while mother isnt last/2 and not found
    if mother element is less than great element
        swap elements
        set mother to great
        set great to max child of mother and last
    else make found true
```

def build_heap

```
for loop using father from last/2 to first - 1 (0 index, decrement)
    fix heap with father and last
```

def heap_sort

```
set first to 1
```

```

set last to length of array
do build heap with first and last
for loop using leaf from last to first
    swap leaf and first
    fix heap with first and leaf - 1

```

4.4 Quick Sort

```

def partition
    set low checker to lo - low checker
    for loop using high checker from lo to high
        if current element of array is less than hi element of array
            increment low checker
            swap elements
    return low checker + 1

def quick_sorter (recursive helper function for quick sort)
    if low less than hi
        do partition of lo and high
        quick_sorter of lo and p - 1
        quick_sorter of p + 1 and high

def quick_sort
    quick_sorter of 1 and array length

```

4.5 Sorting

```

use sets
enum, the sets to sort names (for convenience)

main()
    set a random seed with srand
    generate random numbers and put in array (30-bits)
    make another array with same numbers (to be used to "unsort" the sorted array)
    make an empty set
    use getopt
    use switch cases with command line args
    detect which is being used and assign using sets then break
    use if statements based on set chosen to execute sort
        do sort

```

print out stats and array
replace sorted array with unsorted (in case other functions are sorting too)
free memory

5 Credits

1. I used the asgn3.pdf from Professor Long for explanations and pseudocode.
2. I watched the Lab Section recording from Eugene held on 10/12.
3. I watched the Lab Section recording from Christian held on 10/15.