DESIGN.pdf ASGN3

Nathan Ong

October 14, 2021

1 Description

This collection of files contains four sorting methods that can be implemented to sort through an array of size *n*. The testing rig used to test these sorting methods is also usable with the following command line options:

- -a: Runs all sorting algorithms
- -e: Runs the heap sort method
- -i: Runs the insertion sort method
- -s: Runs the shell sort method
- -q: Runs the quick sort method
- -r *seed*: Sets the RNG seed to the argument *seed*. The default seed is 13371453.
- -n *size*: Sets the array size to the argument *size*. The default size is 100 elements.
- -p *elements*: Prints out *elements* number of elements from the array. The default number is 100. If the array is smaller than the argument, then the entire array is printed.
- -h: Displays a help message detailing the usage of the program

2 Files Included in the Directory

1. insert.c
   (a) This file contains the source code for the implementation of the insertion sorting algorithm.
2. insert.h
   (a) This file contains the specification of the interface to insert.c.
3. heap.c
   (a) This file contains the source code for the implementation of the heap sorting algorithm.
4. heap.h
   (a) This file contains the specification of the interface to heap.c.
5. quick.c
   (a) This file contains the source code for the implementation of the quick sorting algorithm.
6. quick.h
   (a) This file contains the specification of the interface to quick.c.
7. set.h

(a) This file contains the source code for the implementation and the specification of the interface to the set ADT (abstract data type).

8. stats.c

   (a) This file contains the implementation of the statistics module listed in stats.h.

9. stats.h

   (a) This file contains the specification of the interface to the statistics module in stats.c.

10. shell.c

    (a) This file contains the source code of the implementation of the shell sorting algorithm.

11. shell.h

    (a) This file contains the specification of the interface to shell.c.

12. sorting.c

    (a) This file contains the main function of the interface in addition to any additional functions that are necessary for the implementation of the assignment.

3 Pseudocode and Structure

3.1 insert.c

```
for the length of the array
    equate k to the counter value
    store array element temporarily
    while temp variable is less than A[k-1] and k is less than 0
        swap elements of position k and position k-1
        decrement k
    store element at k temporarily
```

3.2 heap.c

```
build a heap with the first and last values of the array
while branch value is greater than first value
    swap A[first value-1] and A[branch value-1]
    fix the heap
```

3.3 quick.c

```
if low value is less than high value
    get pivot value via partition function
    perform quick sort on the two subarrays

    partition function
for k in the range of the low value, high value
    if A[k-1] is less than A[high value-1]
        increment i counter (originally low value - 1)
        swap A[low value-2] and A[k-1]
swap A[i] and A[high value-1] return i + 1
```

3.4 shell.c

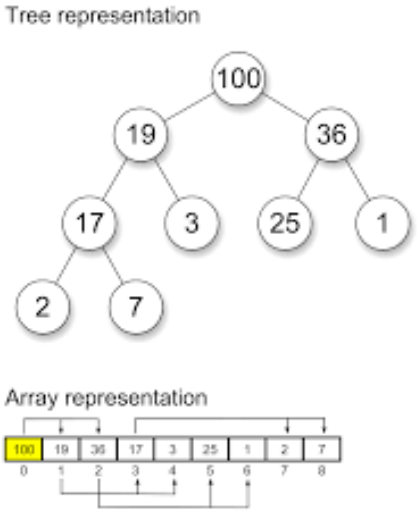
set maximum gap value
while the gap value is less than 1
    make new gap value with formula
        equate k to counter value
        store array element temporarily
        while k is greater than or equal to the gap value and the temp variable is less than A[k - gap]
            equate A[k] to A[k - gap]
            decrement gap value
        store A[k] temporarily
    decrement overall gap value
    when gap reaches 1, perform insertion sort

3.5 sorting.c

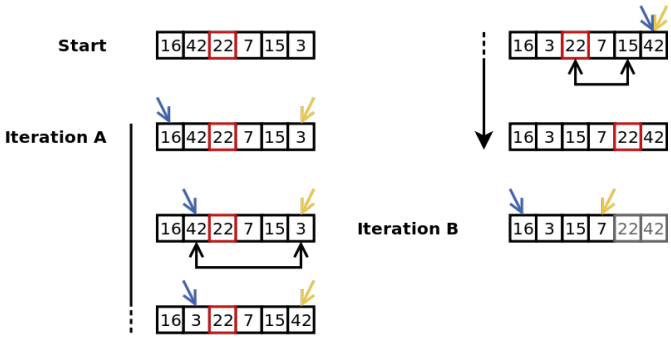
while opt isnt -1
    indice through arguments
    check for required arguments if necessary
    use set to enable functions to be performed
    generate array to be sorted and perform required functions

4 Additional Diagrams

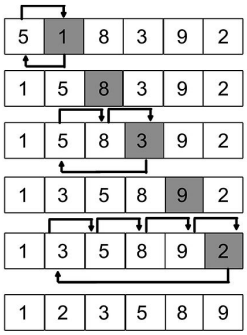
4.1 Heap Sort



4.2 Quick Sort



4.3 Insertion Sort



5 Error Handling

1. Sorts started off as either not running (Shell Sort and Quick Sort) or running but not sorting (Heap Sort and Insertion Sort).
   (a) Solution: Solved by fixing various loop conditionals that were either the opposite of the correct logical operator or were just wrong altogether.
2. The array printer function printed a various amount of the following: excess elements, missing one element, and incorrectly formatting the columns and rows.
   (a) Solution: Going through the array function revealed an incorrect usage of modulus to control the print statements and also incorrectly beginning indicing through the elements beginning at 1 instead of 0.
3. sorting.c would not print the help message as default with not command line arguments.
   (a) Solution: Upon further investigation the default case of the switch statement would not activate. After adjusting the scopes of each switch case, it now successfully prints.
4. With some commands of the library, the variables that were meant to change were not changed when the appropriate command was inputted into the command prompt.
   (a) Solution: Going through the sorting.c code revealed several scope issues, caused by inattentively deleted curly brackets, which caused the scope issues. Brackets were added appropriately and the issue was resolved.

6 Additional Credits

- Sloan's section on October 5th provided the Makefile formatting.
- Images use Creative Commons License CC BY-SA 3.0.

- Pseudocode and code structure based off given python pseudocode from asgn3.doc.
- Eugene's section on October 12th explained the method of using Set and Stats to accomplish the tasks needed for the assignment.