Jay Offerdahl
EECS 560
Lab 7 Report

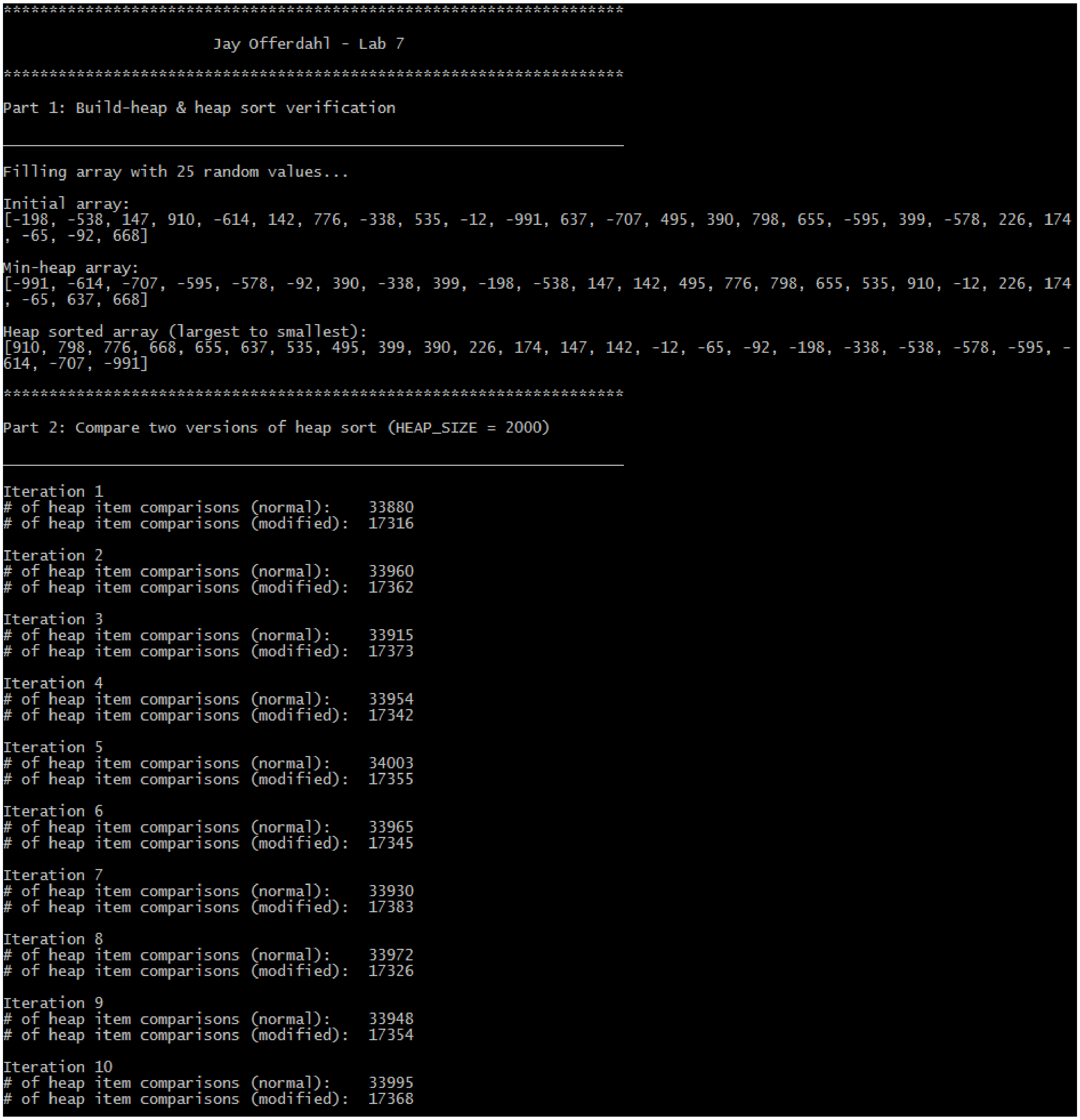
Lab 7 Report

Having never coded a heap before, it was interesting to see how simple the data structure was. The properties, along with the fact that it's array based makes it quite easy to program. In a general discussion about what I learned with this lab, I found that the modified version is especially useful with larger and larger heap sizes. As shown in the output from my program, the modified version had almost half the comparisons than the regular version for a list size of 2000. I also tested this with a smaller list size, say, 100. Here's the output:

```
Iteration 1
# of heap item comparisons (normal):    840
# of heap item comparisons (modified):  438

Iteration 2
# of heap item comparisons (normal):    856
# of heap item comparisons (modified):  454

Iteration 3
# of heap item comparisons (normal):    854
# of heap item comparisons (modified):  441

Iteration 4
# of heap item comparisons (normal):    867
# of heap item comparisons (modified):  445

Iteration 5
# of heap item comparisons (normal):    857
# of heap item comparisons (modified):  435

Iteration 6
# of heap item comparisons (normal):    846
# of heap item comparisons (modified):  436

Iteration 7
# of heap item comparisons (normal):    841
# of heap item comparisons (modified):  437

Iteration 8
# of heap item comparisons (normal):    860
# of heap item comparisons (modified):  439

Iteration 9
# of heap item comparisons (normal):    854
# of heap item comparisons (modified):  441

Iteration 10
# of heap item comparisons (normal):    846
# of heap item comparisons (modified):  444
```

Jay Offerdahl
EECS 560
Lab 7 Report

We can clearly see even with smaller list sizes, the number of comparisons is significantly smaller using the modified version of the program. The reason this works is because even though we might be pushing a value down the tree when we don't need to, we only have to push it up once from the bottom, so it takes significantly less comparisons. The only case where it would take more comparisons (or at least closer to the same number of comparisons), would be if we pushed the value down the whole tree, and almost all the way back up. While this case exists, it's not common, which is why this newer method is better in terms of heap-item comparisons. Note: the heap sort prints in descending order because I implemented a min heap. However, it's still easy to see that the sort performs correctly.



```
*********************************************************************

                       Jay Offerdahl - Lab 7

*********************************************************************

Part 1: Build-heap & heap sort verification

_____________________________________________________________________

Filling array with 25 random values...

Initial array:
[-198, -538, 147, 910, -614, 142, 776, -338, 535, -12, -991, 637, -707, 495, 390, 798, 655, -595, 399, -578, 226, 174
, -65, -92, 668]

Min-heap array:
[-991, -614, -707, -595, -578, -92, 390, -338, 399, -198, -538, 147, 142, 495, 776, 798, 655, 535, 910, -12, 226, 174
, -65, 637, 668]

Heap sorted array (largest to smallest):
[910, 798, 776, 668, 655, 637, 535, 495, 399, 390, 226, 174, 147, 142, -12, -65, -92, -198, -338, -538, -578, -595, -
614, -707, -991]

*********************************************************************

Part 2: Compare two versions of heap sort (HEAP_SIZE = 2000)

_____________________________________________________________________

Iteration 1
# of heap item comparisons (normal):    33880
# of heap item comparisons (modified):  17316

Iteration 2
# of heap item comparisons (normal):    33960
# of heap item comparisons (modified):  17362

Iteration 3
# of heap item comparisons (normal):    33915
# of heap item comparisons (modified):  17373

Iteration 4
# of heap item comparisons (normal):    33954
# of heap item comparisons (modified):  17342

Iteration 5
# of heap item comparisons (normal):    34003
# of heap item comparisons (modified):  17355

Iteration 6
# of heap item comparisons (normal):    33965
# of heap item comparisons (modified):  17345

Iteration 7
# of heap item comparisons (normal):    33930
# of heap item comparisons (modified):  17383

Iteration 8
# of heap item comparisons (normal):    33972
# of heap item comparisons (modified):  17326

Iteration 9
# of heap item comparisons (normal):    33948
# of heap item comparisons (modified):  17354

Iteration 10
# of heap item comparisons (normal):    33995
# of heap item comparisons (modified):  17368
```