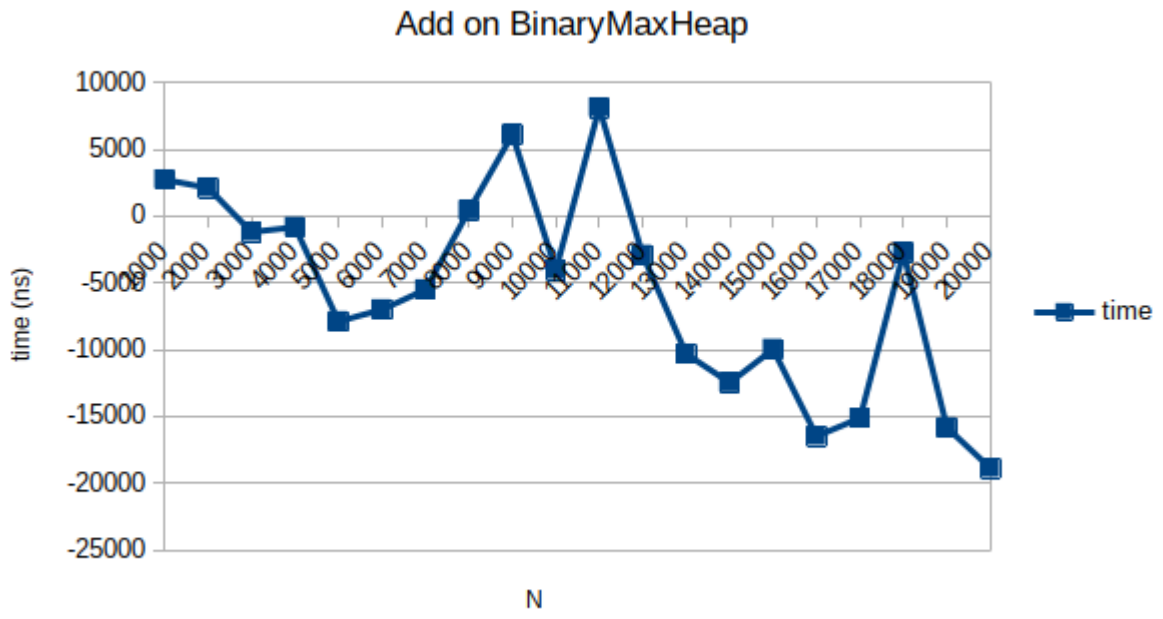
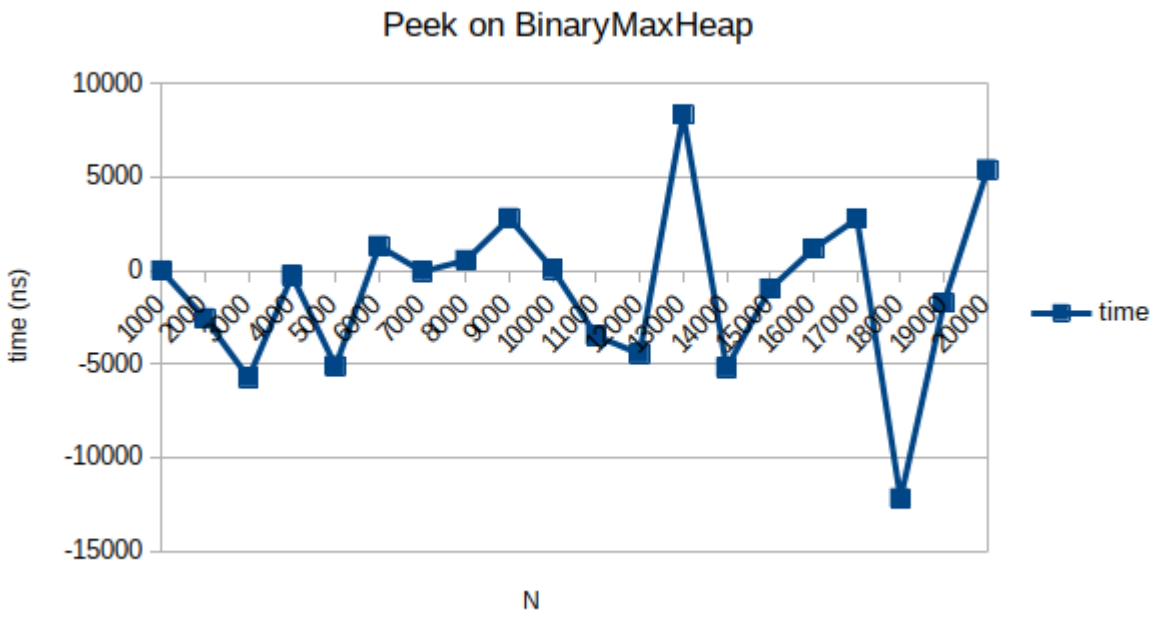
Assignment 10 Analysis Doc

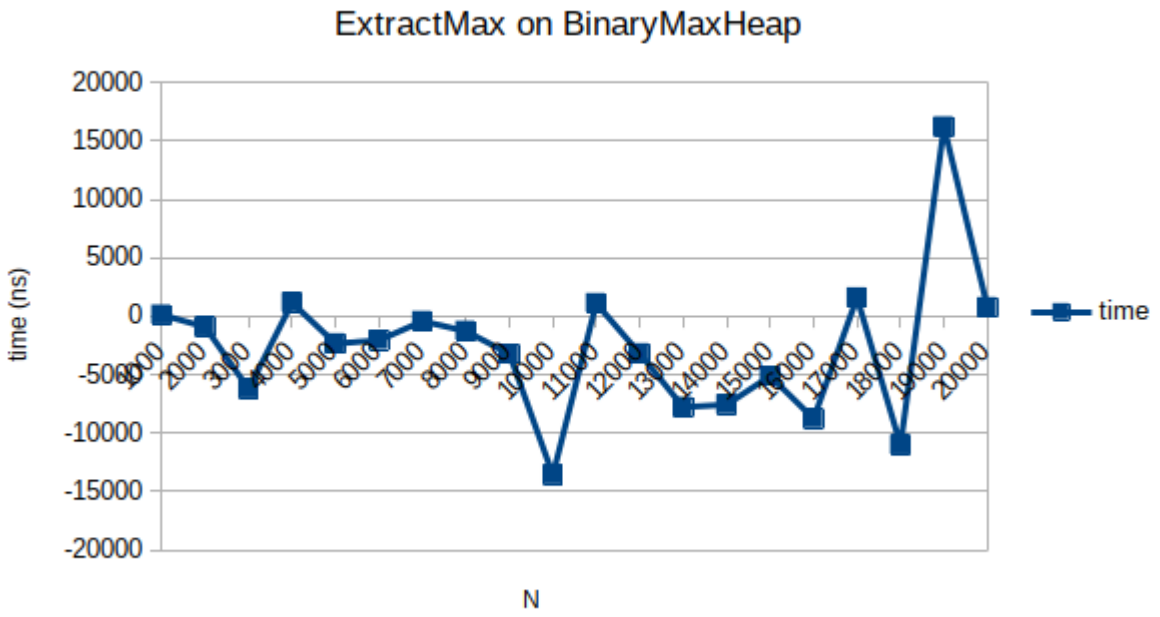
I (Chase Yates) and Liam Astin worked on this assignment. I was the driver and he was the navigator. I may or may not work with him next time, to no fault of his own.

Question 1

In this experiment I created large lists of size N - N ranging from 1,000 to 20,000 - turning those lists into a BinaryMaxHeap. I then ran over 10,000 iterations the add, peek, and extractMax methods respectively. The following were the results (any negative times are due likely to fluctuations in the nanoTime method and other code, as well as external factors)







Question 2

For the running times of add, the chart does support the average case of O(1) as expected, with wild fluctuations, staying around a small value.

The same applies to the rest of the functions, however the extractMax method can take up to O(logN) time, but our running times seem to be so small as to indicate that or O(1) time, somewhat unexpectedly. (Perhaps I just wrote a fast method for it).

Question 3

In `findKLargestHeap`, k number of largest items extracted from the heap, whereas N is the size of the heap.

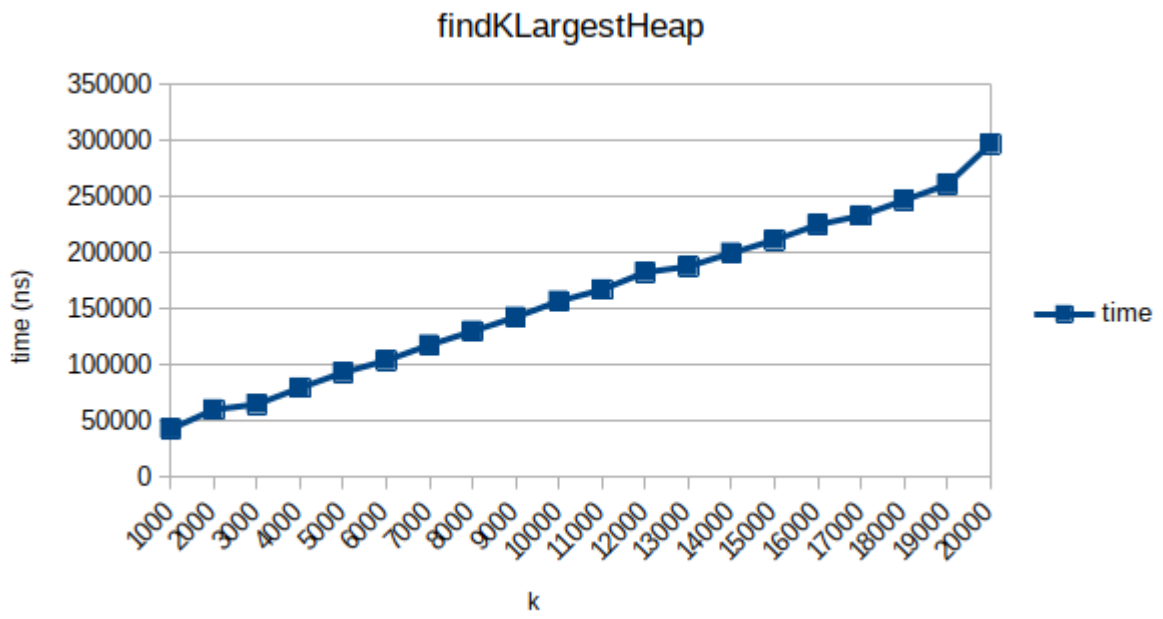
The behavior of this method is O(N+KlogN)

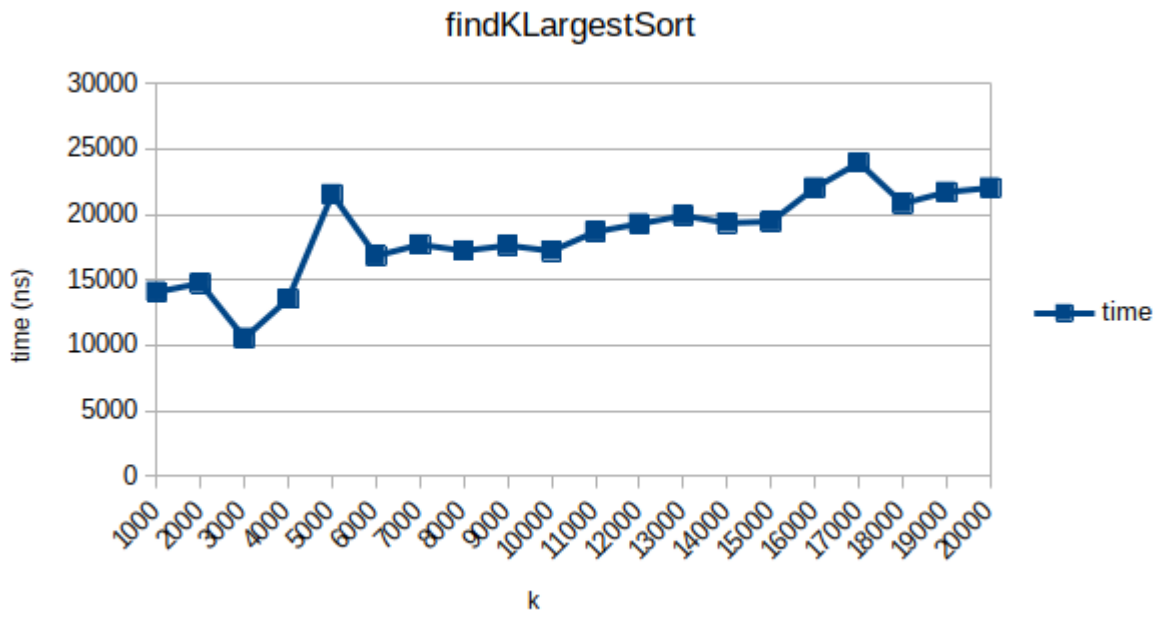
For small values of k, k may be considered constant, and thus the running time would be O(N)

For values of k close to N, k may be considered as N, and thus the running time is closer to O(NlogN)

Question 4

In this experiment I used lists of size 20,000 with 1,000 loops over running the findKLargest Heap and Sort methods. My k ranged from 1,000 to 20,000. Here were the results:





Question 5

With a constant N, the normal O(N + klogN) time for these methods becomes O(k). This is clearly shown in the first graph using Heap Sort, and somewhat similarly (although more varying) in the second graph of Java's Sort method. The running time is as expected.