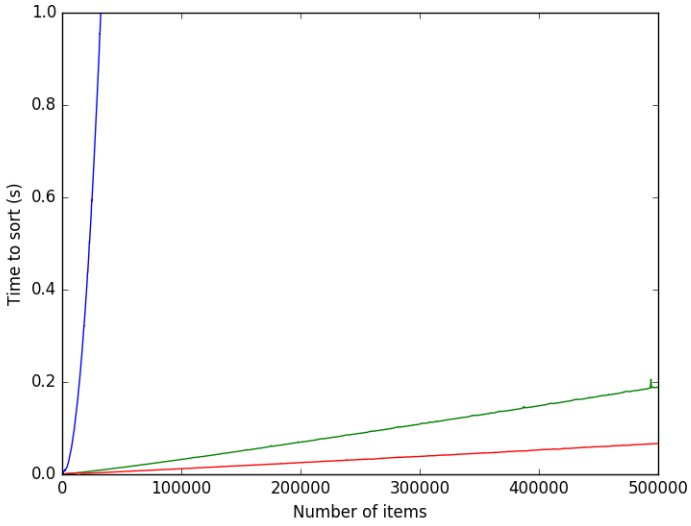
Colin Smith and Emma Steinman
CS 271 Project 3
September 26, 2017

1. In a min-heap, the children of a given node will be larger than the node. Thus, the largest element in the entire heap will be located at a leaf node. However, it will not necessarily be the last leaf node.

2. Yes, if the array is sorted in ascending order. In a sorted array, for $j > i$, $A[j] > A[i]$. In a heap, the indices of the children are greater than the index of the parent. This implies in a sorted array, that the values of the children are greater than the value of the parent, which is the defining feature of a min-heap.

3. See code.

4. Graph of Array size vs Sort Time
   Blue is Insertion Sort
   Green is Heap Sort
   Red is Merge Sort



5. The asymptotic time complexity of heap sort on an array that is already in sorted order is $O(nlogn)$. This is because the algorithm will go through every item in the array ($\Theta(n)$) and call heapify which has time complexity of $O(logn)$.
   The time complexity of heap sort on an array in reverse order will still be $O(nlogn)$ because it still has to go through every element in the array($\Theta(n)$) and call heapify ($O(logn)$).
   The best case asymptotic time occurs when every item is equal and is $\Theta(n)$. This is because the algorithm will always go through every element $\Theta(n)$ but every call to heapify which will be $\Theta(1)$ because it will only have one iteration as no item will ever be greater than its children.