2. Write a brief description of why this worked

Heap sort is a sorting algorithm that starts by constructing a max heap from the given input data. The max heap ensures that the maximum element is always at the root. The algorithm repeatedly removes the maximum element from the heap and restructures it. As a result, the elements are removed in decreasing order. To obtain a sorted list in ascending order, the order of removal is reversed.

The time complexity of heap sort is O(n log n), where n represents the number of elements. This is because the heapify operation, responsible for maintaining the heap property, takes O(log n) time and is performed n times (once for each element). The most resource-intensive sections of the code are the heapifyUp and heapifyDown methods, which ensure the heap property is preserved.

3. Do a Big-O analysis of your code, what is the Big-O of the sort and why (e.g., which lines of code are the most expensive)

   The extract operation in heap sort has a constant time complexity, but the heapifyDown operation it invokes can take up to O(log n) time in the worst case. This is because heapifyDown may require moving an element from the top to the bottom of the heap. Since this process is performed for each of the n elements, the total time complexity for this step becomes O(n log n).

   In addition, the Collections.reverse method, which is used to reverse the order of removals and obtain a sorted list in ascending order, has a linear time complexity of O(n), where n is the number of elements.

   Therefore, the overall time complexity of the sort method is O(n log n + n) = O(n log n), as the dominant term is O(n log n) for larger values of n.

   The lines of code with the highest time complexity are those within the heapifyDown method, which is called by the extract method. The heapifyDown method is responsible for maintaining the heap property and can potentially involve moving an element a distance proportional to the height of the heap, which is logarithmic (log n) in the worst case.