a. The time complexity of heapSort() function:

The first loop builds the max heap using heapify() function. So, the time complexity of first loop is O(n).

The second loop runs as a size of array which is n times, it swaps the two elements of the array and then call the heapify() function. Heapify() function traverse the height of the tree which runs logn times. So, the overall complexity of this loop is O(nlogn).

Total time complexity for this function is O(n) + O(nlogn) = O(nlogn).

b. The space complexity of heapSort() function:

The heapify() function didn't use any extra space or any dynamic memory. So, the space complexity of this function is O(1).

The heapSort() function also constant space complexity because it swap the elements without using extra space and call the heapify() function.

So, the space complexity of heapSort() function is O(1).

c. Min-heap mostly used to sort elements in descending order. If it is used to sort elements in ascending order, it change the efficiency of the algorithm but the complexity would be same. So, using min-heap to sort elements in ascending order is less efficient than using max heap because min-heap need to reverse every time and set the minimum elements to the root node.