Solution 1

```
// Copyright © 2020 AlgoExpert, LLC. All rights reserved.

// Best: O(nlog(n)) time | O(1) space
// Average: O(nlog(n)) time | O(1) space
// Worst: O(nlog(n)) time | O(1) space
function heapSort(array) {
  buildMaxHeap(array);
  for (let endIdx = array.length - 1; endIdx > 0; endIdx--) {
    swap(0, endIdx, array);
    siftDown(0, endIdx - 1, array);
  }
  return array;
}

function buildMaxHeap(array) {
  const firstParentIdx = Math.floor((array.length - 2) / 2);
  for (let currentIdx = firstParentIdx; currentIdx >= 0; currentIdx--) {
    siftDown(currentIdx, array.length - 1, array);
  }
}

function siftDown(currentIdx, endIdx, heap) {
  let childOneIdx = currentIdx * 2 + 1;
  while (childOneIdx <= endIdx) {
    const childTwoIdx = currentIdx * 2 + 2 <= endIdx ? currentIdx * 2 + 2 : -1;
    let idxToSwap;
    if (childTwoIdx !== -1 && heap[childTwoIdx] > heap[childOneIdx]) {
      idxToSwap = childTwoIdx;
    } else {
      idxToSwap = childOneIdx;
    }
    if (heap[idxToSwap] > heap[currentIdx]) {
      swap(currentIdx, idxToSwap, heap);
      currentIdx = idxToSwap;
      childOneIdx = currentIdx * 2 + 1;
    } else {
      return;
    }
  }
}

function swap(i, j, array) {
  const temp = array[j];
  array[j] = array[i];
  array[i] = temp;
}

exports.heapSort = heapSort;

```