Solution 1

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

package main

// Best: O(nlog(n)) time | O(1) space
// Average: O(nlog(n)) time | O(1) space
// Worst: O(nlog(n)) time | O(1) space
func HeapSort(array []int) []int {
  buildMaxHeap(array)
  for endIndex := len(array) - 1; endIndex >= 1; endIndex-- {
    swap(0, endIndex, array)
    siftDown(0, endIndex-1, array)
  }
  return array
}

func buildMaxHeap(array []int) {
  first := (len(array) - 2) / 2
  for currentIndex := first + 1; currentIndex >= 0; currentIndex-- {
    siftDown(currentIndex, len(array)-1, array)
  }
}

func siftDown(currentIndex int, endIndex int, heap []int) {
  childOneIndex := currentIndex*2 + 1
  for childOneIndex <= endIndex {
    childTwoIndex := -1
    if currentIndex*2+2 <= endIndex {
      childTwoIndex = currentIndex*2 + 2
    }
    indexToSwap := childOneIndex
    if childTwoIndex > -1 && heap[childTwoIndex] > heap[childOneIndex] {
      indexToSwap = childTwoIndex
    }
    if heap[indexToSwap] > heap[currentIndex] {
      swap(currentIndex, indexToSwap, heap)
      currentIndex = indexToSwap
      childOneIndex = currentIndex*2 + 1
    } else {
      return
    }
  }
}

func swap(i, j int, array []int) {
  array[i], array[j] = array[j], array[i]
}

```