Assignment 8: Heap Sort Comparison of Heap Sort and BST Sort

This assignment is very similar to the previous one. This time you have to program a heap, and use it for sorting. As we learned in class, heaps are designed specifically to find the minimum (or maximum) of a data set in guaranteed O(log n), since the heap is always complete (and therefore, its height is always O(log n)). Heapsort works exactly like the BST sort you programmed last week, except that you build and destroy a heap, not a BST. We also learned that heapsort should be faster than BST sort. Since you now have both sorts at hand, you have to compare and see if this assumption is true.

The assignment has 2 parts:

1. program a heap data structure to perform heapsort. Important: the heap must be implemented as an array, NOT a linked data structure. We will learn in class on Monday how that works. If you want to prepare: the textbook will tell you.
2. Compare the performance between heap and BST.

Remark: there is no graphical output involved this time, no visualization. It is just calling two different sorting algorithms on data sets of different size, and compare their runtimes.

Details

About part I:

There's not too much to say, except: do not even attempt to NOT use an array, i.e. you MUST use an ARRAY.

About part II:

A straightforward comparison would be: Generate n random numbers, sort them using BST sort, sort the same array of numbers using heap sort, get the ratio of runtime and print it. Do the same with 2n, 3n, 4n, … 20n random numbers. **For such a comparison, we need to find a number n that results in quantifiable and reasonable runtimes:** if n is too low, your timing will be very imprecise (usually telling you that your algorithm needed 0ms).

Hence, the first task in part II is to find an appropriate number n:

- Find a number n of random numbers, for which heap sort needs about 1 second to sort. Take this n as a start value for your comparison program (which then generates runtime measuring for n, 2n, 3n, .. 20n).
- This means, your main program starts with a method "findAppropriateNumber", which tries to find a good number n of data elements to process heap sort on, such that it takes 1s on your computer. Btw., you can compare the speed of your computer with others looking at this number.
- Then, your main program goes through a for loop, calling heap and BST sort with n, 2n, ... 20n data elements.
- Measure the time each time, print it.

Remark: If you didn't finish last week's assignment, finish this one for heap sort only. The time comparison would then just be a runtime report for heap sort.