

Accelerated Heapsort

August 16, 2007

Goal

Implement the accelerated heapsort algorithm as described in section 4.8.6 in the text.

Details

This project has four main goals:

1. It must correctly implement the accelerated heapsort algorithm
2. It must work without modification on any comparable data type
NB: Some data types might need some help
3. It must work with an arbitrary amount of data
4. Compare the algorithm to the conventional heapsort

Data types

The code must work on any data type without modification. However, you may wrap your data type in a suitable object if necessary (e.g., to provide $<$, $=$, etc., or `isLessThan()`, `isEqualTo()`, ...).

Algorithm comparison

We know from the text and in-class discussion that the conventional heapsort requires $2n\lceil\lg(n+1)\rceil$ comparisons in the worst case, and accelerated heapsort requires $n\lceil\lg(n+1)\rceil + \Theta(n\lg\lg(n+1))$ comparisons in the worst case.

You must generate several test cases (at least ten), with between 10 and 100,000 elements (one case must have 10, one must have 100,000). Run the test cases through the conventional and accelerated heapsorts. Report how well the algorithms fare against each other, and how closely they match their theoretical counts.

What to turn in

Turn in an electronic copy of your code, and the results of your comparisons.