Algorithms – CMSC-37000

Instructor: László Babai Ryerson 164 e-mail: laci@cs.etc

Recall that a sorting algorithm is said to be *comparison based* if the only information available to the algorithm about the input is the number of items (real numbers) to be sorted and answers to queries of the form "$A[i] \le A[j]$?" where $A[i]$ is item number i.

In class we proved that to sort a list of n items, every comparison-based sorting algorithm requires $\ge \log_2(n!) \sim n \log_2 n$ comparisons. Preprocessing the input by asking some of the comparisons in advance can in principle reduce the number of remaining comparisons needed.

For instance, when performing comparison-based sorting of data prearranged in a heap, certain queries do not need to be asked: no item needs to be compared with its ancestors or descendants in the tree since heap-ordering already answers those comparisons.

The following exercise shows that preprocessing the data by arranging them in a heap is not much help.

Problem: sorting a heap. Suppose we have n data (real numbers) arranged in a heap, and the heap is implemented as an array (as in class). Prove that in order to sort the data by comparisons, we still need to make asymptotically at least $n \log_2 n$ comparisons.

WARNINGS.

1. This problem is NOT about the Heapsort algorithm. We are allowed to compare any pair of data by naming their addresses in the array. You need to show that the savings from having the data in a heap is negligible compared to the total cost (number of comparisons).

2. You need to prove more than an $\Omega(n \log_2 n)$ lower bound: you need to prove that the number of comparisons is greater than $(1 - \epsilon)n \log_2 n$ for all $\epsilon > 0$ and all sufficiently large n.

Last updated 1-22-2014