Readings Sections 8.1,9.1

Self test Ex 8.1

Lecture 23

The Problem of Sorting

How fast can we sort?

Existence of algorithms that run in worst-case time $O(n \log n)$ confirm that sorting can be done in $O(n \log n)$ but does not rule out existence of better algorithms.

We know how to analyze the worst-case complexity of algorithms worst case complexity of *problems* involves extra work.

For problem P, $C(P)$ = best (minimum) worst case running time of any algorithm that solves P.

Upper bound on $C(P)$: give an algorithm and analyze its runtime. E.g. sorting is $O(n \log n)$

Lower bound on $C(P)$: have to prove *every* algorithm requires a certain amount of time. In practice, analyze for a "class" of algorithms.

Comparison Algorithms

- compare one element to another
- use a comparison tree

Ex. binary search on sorted $A[1 \ldots 3], x$
return index of x (or 0 if not found)

- need a leaf for every possible output in the decision tree
- height of the tree is a bound on the worst-case complexity

Information Theoretic Lower Bounds

Every binary tree with height h has $\leq 2^h$ leaves.
$\Rightarrow$ every binary tree with L leaves has height $\geq \lceil \log_2 L \rceil$.

Every comparison tree that solves a problem P has a leaf for every possible output. Every comparison tree for P has height $\geq \lceil \log_2 m \rceil$ where m is # of outputs.