CIS 313

week of Nov 19

ninth week of the term
lower bounds

• $O(n \log n)$ seems like a common time bound for sorting
• there is a reason for this
• look at the *comparison-based* model
• access to data items are only through comparisons of two items
  • is $a \leq b$?
• the more common sorts are comparison based: merge-sort, heap-sort, quick-sort, bubble-sort, etc.
• counting sort is $O(n)$ times (sort of), but it is not comparison-based
  • it uses data item as array index
comparison-based sorts require $n/\lg n$ time

- the $\Omega(n \cdot \lg n)$ lower bound is on the number of comparisons required to sort $n$ items
- use the *decision tree* model
- a decision tree is a full binary tree
  - internal node represents a comparison between two items
  - left/right branches indicate yes/no outcomes
  - external nodes are the outcomes
- any comparison based sorting algorithm on $n$ elements corresponds to a (BIG!!) decision tree
- height of the tree is the worst case number of comparisons
decision tree example

This tree has $3! = 6$ external nodes, one for each possible outcome of sorting 3 elements.
observations about decision trees

• any decision tree for sorting $n$ items must have at least $n!$ external nodes (outcomes): $\#\text{outcomes} \geq n!$

• a decision tree of (internal) height $h$ has at most $2^h$ external nodes
  • height $h$ means $h+1$ comparisons

• combining: $n! \leq \#\text{outcomes} \leq 2^h$

• take logs of both sides: $\log n! \leq \log 2^h = h$

• Stirling’s Approx: $\log n! = \log \sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n \cdot (1 + \Theta\left(\frac{1}{n}\right)) = \Theta(n \log n)$

• or simpler: $h \leq \log n! \leq \log n^n = n \log n$
aside: avoid Stirling’s Approx

• $n! = n(n - 1)(n - 2) \cdots 2 \cdot 1 \geq n(n - 1) \cdots \left(\frac{n}{2} + 1\right)\left(\frac{n}{2}\right) \geq \left(\frac{n}{2}\right)^n$

• so from previous page...

• $h = \lg 2^h \geq \lg n! \geq \lg \left(\frac{n}{2}\right)^n = \left(\frac{n}{2}\right)(\lg n - 1)$

• conclude: $h = \Omega(n \lg n)$
lower bound: conclusion

- theorem 8.1: Any comparison sort algorithm requires $\Omega(n \cdot \log n)$ comparisons in the worst case.
- this general technique is called an *information theoretic* lower bound.
- general idea: find number of outcomes, take logarithm.
- application: merge 2 sorted lists of $n$ elements, requires $n$ comparisons.
exercise 8.1-4 from text

8.1-4
Suppose that you are given a sequence of $n$ elements to sort. The input sequence consists of $n/k$ subsequences, each containing $k$ elements. The elements in a given subsequence are all smaller than the elements in the succeeding subsequence and larger than the elements in the preceding subsequence. Thus, all that is needed to sort the whole sequence of length $n$ is to sort the $k$ elements in each of the $n/k$ subsequences. Show an $\Omega(n \lg k)$ lower bound on the number of comparisons needed to solve this variant of the sorting problem. (Hint: It is not rigorous to simply combine the lower bounds for the individual subsequences.)
linear time sorts

• Counting sort, Radix sort

• for Counting sort, we sort n elements, each in the range 0 to k (k fixed)
  • sometimes k=n
  • use element as array index