CSSE 230 Day 25

Quicksort

Sorting Lower Bound

Reminders/Announcements

- **Exam 2 tomorrow evening**
  - Topics were listed in Day 24 slides
  - Some WA9 problems are good reinforcement of exam material.
  - Nothing from that assignment will explicitly ne on the exam.
  - Written part: 100 points
  - Computer part: 44 points (plus some extra credit)
- **WA9 due Thursday**
- **Scrabble Milestone 2 due Friday**
  - 5 days since Milestone 1, 4 days until Milestone 2
  - You want to have everything except your computer player working for Milestone 2
- **Agenda:**
  - Questions about exam, Scrabble, anything else?
  - Quicksort improvements
  - Lower Bound for Sorting Algorithms
  - Radix Sort

Recap

- Best, worst, average time for Quicksort
- What causes the worst case?

Q1

Improvements to QuickSort

- Avoid the worst case
  - Select pivot from the middle
  - Randomly select pivot
  - Median of 3 pivot selection.
  - Median of k pivot selection
- "Switch over" to a simpler sorting method (insertion) when the subarray size gets small

  Weiss's code does Nedian of 3 and switchover to insertion sort at 10.
  - Linked from schedule page

Other Sorting Demos

- http://maven.smith.edu/~thiebaut/java/sort/demo.html
- http://www.cs.ubc.ca/~harrison/Java/sorting-demo.html

A Lower-Bound on Sorting

- We can't do much better than what we already know how to do.

What's the best best case?

- Lower bound for best case?
- A particular algorithm that achieves this?

What's the best worst case?

- Want a function f(N)
  such that the **worst case running time**
  for all sorting algorithms is Ω(f(N))
- How do we get a handle on
  "all sorting algorithms"?

Tricky!

What are "all sorting algorithms"?

- We can't list all sorting algorithms and analyze all of them
  - Why not?
- But we can find a **uniform representation** of any sorting algorithm that is based on **comparing** elements of the array to each other

This "uniform representation" idea is exploited in a big way in Theory of Computation, e.g., to demonstrate the unsolvability of the "Halting Problem"

First of all…

- The problem of sorting N elements is at least as hard as determining their ordering
  - e.g., determining that $a_3 < a_4 < a_1 < a_5 < a_2$
- So any lower bound on all "order-determination" algorithms is also a lower bound on "all sorting algorithms"

Q2

Sort Decision Trees

- Let A be any **comparison-based algorithm** for sorting an array of distinct elements
- Note: sorting is asymptotically equivalent to determining the correct order of the originals
- We can draw an EBT that corresponds to the comparisons that will be used by A to sort an array of N elements
  - This is called a **sort decision tree**
  - Just a pen-and-paper concept, not actually a data structure
  - Different algorithms will have different trees

Q3-5

So what?

- Minimum number of external nodes in a sort decision tree? (As a function of N)
- Is this number dependent on the algorithm?
- What's the height of the shortest EBT with that many external nodes?

$$\lceil \log N! \rceil \approx N \log N - 1.44N = \Omega(N \log N)$$

No comparison-based sorting algorithm, known or not yet discovered, can **ever** do better than this!

Can we do better than *N* log *N*?

- Ω(N log N) is the best we can do if we compare items
- Can we sort without comparing items?

Yes, we can! We can sort if we avoid comparing items

Q6

- O(N) sort: Bucket sort
  - Works if possible values come from limited range
  - Example: Exam grades histogram
- A variation: Radix sort

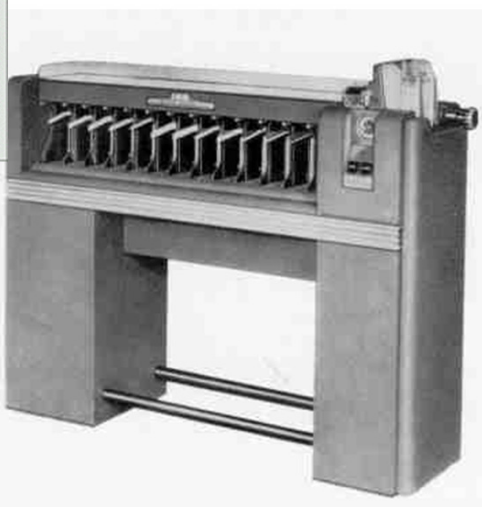
Q7-10

Radix sort

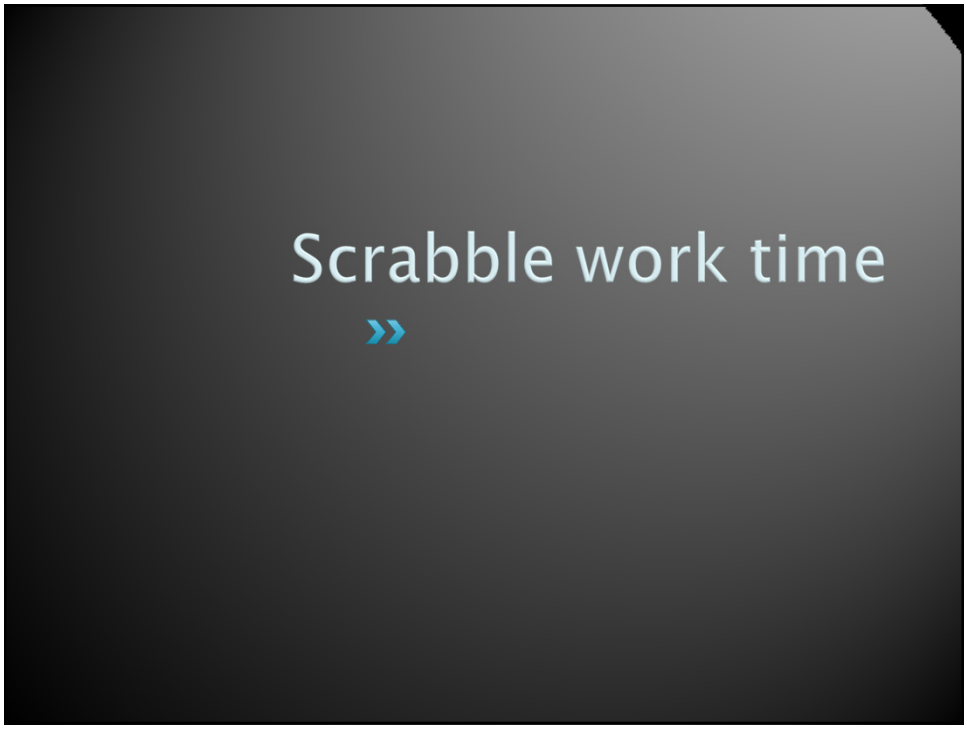
- A picture is worth 10^3 words, but an animation is worth 2^{10} pictures, so we will look at one.
- http://www.cs.auckland.ac.nz/software/AlgAnim/radixsort.html

Radix sort example: card sorter

Used an appropriate combo of mechanical, digital, and human effort to get the job done.



Type 82 Electric Punched Card Sorting Machine


Scrabble work time