Lecture Notes for I500
Fall 2009

Fundamental Computer Concepts of Informatics

by Chad Burrus (Instructor: Predrag Radivojac)

Last Time: BST & Quicksort
Today: Quicksort and order statistics

Quicksort Idea: pick an element (pivot point) and sort based on that element

\[
\begin{array}{c|c|c}
< a & > a & a \\
\end{array}
\]

\[\Rightarrow\] Simple partitioning scheme – move from left to right
\[\Rightarrow\] Worst case – pivot belongs on one of the ends \(\Rightarrow O(n^2)\)

\[
\begin{array}{c}
< a \\
\end{array}
\]

\[
\begin{array}{c|c}
 > a & \text{shaded} \\
\end{array}
\]

\[\Rightarrow\] Best case – pivot belongs in the middle \(\Rightarrow O(n^2)\)

\[
\begin{array}{c|c|c}
< a & > a & a \\
\end{array}
\]

Fixed partitioning
\[\Rightarrow\] Assume we always have \(\frac{1}{10}\)th of the data in partition 1 and \(\frac{9}{10}\)’s in partition 2

\[
\begin{array}{c|c}
\frac{1}{10} & \text{shaded} \\
\text{9/10} & \text{unshaded} \\
\end{array}
\]
What's the recurrence tree look like for fixed partitioning?

Depth of the tree at the shallowest point is $\log_{10} n$ -- solve for $i$

$$\frac{n}{10^i} = 1$$

$$i = \log_{10} n$$

Depth of the tree at the deepest point is $cn \log_{10} \frac{n}{9}$

$$\frac{n}{(10/9)^i} = 1$$

$$i = \log_{10} n \Rightarrow \text{max depth of the tree}$$

Therefore, quicksort is $\Theta(n \log n)$ with fixed partitioning, which is good.

Problem: What happens if array A is already sorted? (Quicksort is slow for sorted arrays.)

$\Theta(n^2)$ vs. $\Theta(n)$ for insertion sort

Random partitioning - pick random pivot point, move to last element, then sort

One way to handle “tough” arrays is to randomly pick pivots

Check if array is already sorted? Maybe, but doesn’t really occur often enough to justify the cost.

Question: Can we actually guarantee $n \log n$ worst case?

Idea: Find the median (middle point by value) in linear time

1, 2, 4, 3, 5 $\rightarrow$ 3 is the median

Not easy!

Going away, but we’ll be back.
Order statistics – interested in finding the \( k \)th smallest element in an array.

\( \Rightarrow \) Problem: Given a sequence \((a_1, a_2, \ldots, a_n)\), and an integer \( k \) find the \( k \)th smallest element

- Solution #1: Sort array and return \( k \)th element
  - Obviously takes \( \theta(n \log n) \) time, which is too much
- Solution #1.5: Find \( k \)th smallest \( k \) times—discarded before even talked about
  - Approaches \( \theta(n^2) \)
- Solution #2: Quicksort-type like approach

Claim: such an algorithm works in \( \theta(n) \) time

Restatement:

1. Find 3\(^{rd}\) best → sort 1 .. \( j \) and find 3\(^{rd}\) best
2. Find 11\(^{th}\) best → sort \( j \) .. \( n \) and find 1\(^{st}\) best

Remember quicksort recurrence – best case

\[ T(n) = 2T\left(\frac{n}{2}\right) + cn \]

The recurrence is similar here.

\[ T(n) = T\left(\frac{n}{2}\right) + cn \]

\[ a = 1, b = 2, f(n) = n, f(b) = 2 \Rightarrow a < f(b) \Rightarrow \theta(n) \text{ expected time} \]

Question: Can we really find the median (the element with \( k = \frac{n}{2} \)) in linear time?

Possible, but not trivial

Solution #3:
- Process
  - Split array into segments of length 5 (5 is special number, not all of them will work)
For each segment, find the local median and recursively find the median of those medians \( M \).

- Split array \( A \) into \( A_{<M}, A_M, A_{>M} \) and recursively search the appropriate partition.

**Analysis of Solution #3**

- How many elements of \( A \) are \( \geq M? \)
  - Initially we have \( \frac{n}{5} \) local medians.
  - Half of local medians \( \left( \frac{n}{10} \right) \) elements are \( \geq M \), since \( M \) is the median of them.
  - For those local medians, there are at least two more elements that each of them brings along that are \( \geq M \).

For local medians \( \geq M \), there are at least two more of these group of 5 that are \( \geq M \).

- Therefore, \( A \) contains at least \( \frac{3n}{10} \) elements \( \geq M \) and there are at most \( \frac{7n}{10} \) elements \( \leq M \).
- Things are beginning to look like fixed positioning.

**Recurrence for Solution #3**

\[
T(n) \leq \begin{cases} 
\frac{cn}{5} & \text{to find local medians and partition} \\
T\left(\frac{n}{5}\right) & \text{to find } M \\
T\left(\frac{7n}{10}\right) & \text{to proceed in appropriate partition}
\end{cases}
\]

- Claim: \( T(n) = \theta(n) \rightarrow \) Homework to verify this! (Solve by substitution.)

**Back to Quicksort**

- \( cn \) to find median
- \( dn \) to partition
- \( 2 \ast T\left(\frac{n}{2}\right) \) to sort partitions
- Recurrence

\[
T(n) = 2 \ast T\left(\frac{n}{2}\right) + en
\]

\[
T(n) = \theta(n \lg n)
\]

- This guarantees quicksort will work in \( \theta(n \lg n) \), though with a bigger constant than before.
- Quicksort is the fastest!
Partition sorting
  - Other sorting algorithms similar to quicksort
  - Some might work faster than quicksort, but not by much.

Sorts make good examples of how to analyze algorithms because they’re easy to understand.

Quicksort is the last topic for the mid-term

What’s Next?
  - Formal languages
  - Finite automata
  - Dynamic and linear programming
  - Graph algorithms

Mid-Term
  - Study!
  - Seventy-five minutes isn’t much time
  - Read problems carefully!
  - Don’t explain in detail.
  - If you don’t know an answer, move on and solve others. Come back to the ones you don’t know later.
  - Mostly from lectures, but homework and book information is fair game.
  - Goal is learning to solve problems
  - No Matlab
    - Might be some small code, but can be pseudocode
  - Extra office hours this week
    - Wednesday, probably from 10 - 12