LECTURE 8
Analyzing Quick Sort

Sofya Raskhodnikova and Adam Smith

S. Raskhodnikova and A. Smith. Based on notes by E. Demaine and C. Leiserson
Reminder: QuickSort

Quicksort an $n$-element array:

1. **Divide:** Partition the array around a *pivot* $x$
such that elements on left are $\leq x$ and elements on right are $\geq x$

2. **Conquer:** Recursively sort the two

3. **Combine:** Nothing!

**Key:** *Linear-time partitioning subroutine.*
PARTITION($A, p, r$)

- Gets array $A$ and bounds $p$, $r$

- Basic idea
  - Use first element $A[p]$ of array as pivot
  - Arrange elements in array so that:
    
    \[
    \begin{array}{c|c|c}
    \leq x & x & \geq x \\
    \hline
    p & & q
    \end{array}
    \]
  - Sort left and right pieces recursively
Pseudocode for quicksort

\textbf{QUICKSORT}(A, p, r)

\begin{itemize}
  \item Sort $A[p .. q]$ in place.
  \item If $p < r$ then $q \leftarrow \text{PARTITION}(A, p, r)$
  \item QUICKSORT($A, p, q-1)$
  \item QUICKSORT($A, q+1, r)$
\end{itemize}

If $p \geq r$ do nothing

\textbf{Initial call:} QUICKSORT($A, 1, n$)
Review Question

• How long does Quicksort run on a sorted array, e.g. \( [1,2,\ldots,n] \)?
  – Write down a recurrence
  – Guess a solution.

(Answers in the next few slides)
Analysis of quicksort

• Assume all input elements are distinct.
• In practice, there are better partitioning algorithms for when duplicate input elements may exist.
• Let $T(n) =$ worst-case running time on an array of $n$ elements.
Worst-case of quicksort

- Input sorted or reverse sorted.
- Partition around min or max element.
- One side of partition always has no elements.

\[
T(n) = T(0) + T(n-1) + \Theta(n) \\
= \Theta(1) + T(n-1) + \Theta(n) \\
= T(n-1) + \Theta(n) \\
= \Theta(n^2) \quad \text{(arithmetic series, as with insertion sort)}
\]
Worst-case recursion tree

\[ T(n) = T(0) + T(n-1) + cn \]
Worst-case recursion tree

\[ T(n) = T(0) + T(n-1) + cn \]
Worst-case recursion tree

\[ T(n) = T(0) + T(n-1) + cn \]
Worst-case recursion tree

\[ T(n) = T(0) + T(n-1) + cn \]
Worst-case recursion tree

\[ T(n) = T(0) + T(n-1) + cn \]
Worst-case recursion tree

\[ T(n) = T(0) + T(n-1) + cn \]

\[ \Theta\left( \sum_{k=1}^{n} k \right) = \Theta(n^2) \]
Worst-case recursion tree

\[ T(n) = T(0) + T(n-1) + cn \]

\[ T(n) = \Theta(n) + \Theta(n^2) \]

\[ h = n \]

S. Raskhodnikova and A. Smith. Based on notes by E. Demaine and C. Leiserson
Best-case analysis

(For intuition only!)

If we’re lucky, PARTITION splits the array evenly:

\[ T(n) = 2T(n/2) + \Theta(n) = \Theta(n \lg n) \text{ (same as merge sort)} \]
An “OK” case analysis

A split (or pivot) is “OK” if the smaller piece has at least \( \left\lfloor \frac{n}{4} \right\rfloor \) elements.

What if the split is always OK, i.e. \( \frac{1}{4} : \frac{3}{4} \) ?

\[
T(n) = T\left(\frac{1}{4} n\right) + T\left(\frac{3}{4} n\right) + \Theta(n)
\]

What is the solution to this recurrence?

\[
T(n) = \Theta(n \lg n) \quad \text{(homework)}
\]
Average Split?

• If input is sorted, every split is bad

• What if input is in random order?
  – Quality of split depends on rank of first element
    \[ \text{rank}(x) = \# \{ i : A[i] < x \} \]
  – \( E[ \text{size of smaller subarray} ] \)
    \[ = E[ \text{random number between 0 and } n/2 ] \]
    \[ = n / 4 \]

• Problem: inputs are not typically random
Randomized Quicksort

**BIG IDEA:**
Partition around a *random* element.

- Randomness in the input is unreliable
- Rely instead on random number generator
- We’ll see many places where this helps
Randomized Quicksort

**Big Idea:** Partition around a *random* element.

- Running time is independent of the input order.
- No assumptions need to be made about the input distribution.
- No specific input elicits the worst-case behavior.
- The worst case is determined only by the output of a random-number generator.
Pseudocode for quicksort

\[
\text{RQUICKSORT}(A, p, r) \\
\quad \triangleright \text{Sort } A[p \ldots r] \text{ in place.}
\]

\[
\text{if } p < r \\
\quad \text{then } t \leftarrow \text{RANDOM}(p, r) \\
\quad \text{Exchange } A[p] \leftrightarrow A[t] \\
\quad q \leftarrow \text{PARTITION}(A, p, r) \\
\quad \text{RQUICKSORT}(A, p, q-1) \\
\quad \text{RQUICKSORT}(A, q+1, r)
\]

**Initial call:** \text{RQUICKSORT}(A, 1, n)
Randomized quicksort analysis

Let $T(n)$ = the random variable for the running time of randomized quicksort on an input of size $n$, assuming random numbers are independent.

For $k = 0, 1, \ldots, n-1$, define the indicator random variable

$$X_k = \begin{cases} 1 & \text{if PARTITION generates a } k : n-k-1 \text{ split,} \\ 0 & \text{otherwise.} \end{cases}$$

$$E[X_k] = \Pr\{X_k = 1\} = 1/n,$$ since all splits are equally likely, assuming elements are distinct.
Analysis (continued)

\[
T(n) = \begin{cases} 
T(0) + T(n-1) + \Theta(n) & \text{if } 0 : n-1 \text{ split}, \\
T(1) + T(n-2) + \Theta(n) & \text{if } 1 : n-2 \text{ split}, \\
\vdots & \\
T(n-1) + T(0) + \Theta(n) & \text{if } n-1 : 0 \text{ split}, 
\end{cases}
\]

\[
= \sum_{k=0}^{n-1} X_k (T(k) + T(n - k - 1) + \Theta(n))
\]
Analysis continued in Lecture 9