# CS 255, Spring 2014, SJSU

## Quicksort

Fernando Lobo



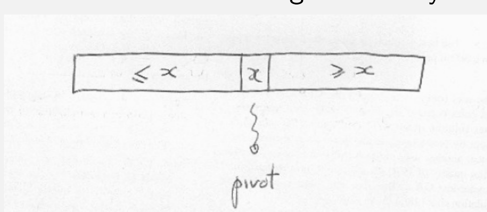
## Quicksort

- Invented by Hoare in 1962.
- It's a D&C algorithm.
- Very practical. Most library sorting algorithms are based on it.
- Worst case running time is $\Theta(n^2)$ but randomized version has an expected running time of $\Theta(n \lg n)$.



## Divide Step

- Partition the array into 2 subarrays around an element $x$ called the *pivot* so that the elements on the left subarray are $\leq x \leq$ than the elements on the right subarray.






## Conquer and Combine steps

- Conquer step: recursively sort the 2 subarrays.
- Combine step: do nothing.

## Quicksort vs. MergeSort

- MergeSort does all the work in the combine step.
- QuickSort does all the work in the divide step.
- QuickSort sorts in place, MergeSort does not.



## Pseudocode

$\textsc{Quicksort}(A, p, r)$
  **if** $p < r$
    $q = \textsc{Partition}(A, p, r)$
    $\textsc{Quicksort}(A, p, q-1)$
    $\textsc{Quicksort}(A, q+1, r)$

Initial call: $\textsc{Quicksort}(A, 1, n)$



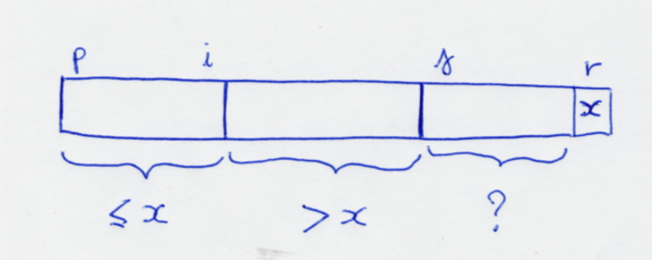
## Pseudocode for partition

$\textsc{Partition}(A, p, r)$
  $x = A[r]$    // pivot
  $i = p - 1$
  **for** $j = p$ **to** $r - 1$
    **if** $A[j] \leq x$
      $i = i + 1$
      exchange $A[i]$ with $A[j]$
  exchange $A[i+1]$ with $A[r]$
  **return** $i + 1$



## Partition

- During its execution, divides the array into 4 (possible empty) regions.



## Analysis: worst-case

- Worst-case: array is sorted or reverse sorted.

$$
\begin{aligned}
T(n) &= T(n-1) + T(1) + \Theta(n) \\
&= T(n-1) + \Theta(n) \\
&= \Theta(n^2)
\end{aligned}
$$



## Analysis: best-case

- For intuition only.
- Best-case: even split.

$$
\begin{aligned}
T(n) &= 2T(n/2) + \Theta(n) \\
&= \Theta(n \lg n)
\end{aligned}
$$

- Same as MergeSort.



## Analysis

- Suppose split is always $n/10$, $9n/10$.

$$T(n) = T(n/10) + T(9n/10) + \Theta(n)$$

- Cannot apply Master method.
- Can draw recursion tree to get intuition and then prove by substitution.
- You will get $T(n) = \Theta(n \lg n)$
- $T(n) = \Theta(n \lg n)$ whenever split has constant proportionality.



## Randomized Quicksort

- Introduce randomization in order to obtain good expected performance.
- Idea: randomize the input array.
- Simpler idea: choose the pivot randomly.
- Performance is not affected by bad inputs.

## Pseudocode

RANDOMIZED-PARTITION$(A, p, r)$

$i = \text{RANDOM}(p, r)$ // random int drawn unif. from $[p \mathinner{..} r]$
exchange $A[r]$ with $A[i]$
**return** PARTITION$(A, p, r)$

RANDOMIZED-QUICKSORT$(A, p, r)$

**if** $p < r$
  $q = \text{RANDOMIZED-PARTITION}(A, p, r)$
  RANDOMIZED-QUICKSORT$(A, p, q - 1)$
  RANDOMIZED-QUICKSORT$(A, q + 1, r)$



## Analysis of Randomized Quicksort

- Worst-case is still $T(n) = \Theta(n^2)$
- But since we are using randomization, worst case is very unlikely.
- Note the difference between standard (non-randomized) version.
  - sorted or almost sorted input can occur quite often.
  - with randomization those cases have near zero probability to occur.



## Analysis of expected running time

- Analysis assumes elements in array are distinct.
- Let $T(n)$ = random variable for the running time assuming that random numbers are independent.
- All splits are equally likely.
- For $k = 0, 1, \ldots, n - 1$, let

$$X_k = \begin{cases} 1 & \text{if partition generated a } (k, n-k-1) \text{ split} \\ 0 & \text{otherwise} \end{cases}$$



## Analysis of expected running time

- We have $n$ random variables.
- For a given execution of RANDOMIZED-PARTITION all $X_k$'s will be 0, except one, which corresponds to the split that actually occurs.

$$\begin{aligned} \mathrm{E}[X_k] &= 0 \cdot Prob\{X_k = 0\} + 1 \cdot Prob\{X_k = 1\} \\ &= Prob\{X_k = 1\} \\ &= 1/n \end{aligned}$$

## Analysis of expected running time

Recurrence for $T(n)$:

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

- $n$ cases. Only one occurs.



## Analysis of expected running time

- Let's convert the cases into a summation.

$$T(n) = \sum_{k=0}^{n-1} X_k \cdot (T(k) + T(n-k-1) + \Theta(n))$$

- $X_k$ is only going to be 1 in one of the cases, all others are 0.
- The summation is equivalent to the case-based recurrence.



## Analysis of expected running time

$$\begin{aligned} \mathrm{E}[T(n)] &= E\Big[\sum_{K=0}^{n-1} X_k \cdot (T(k) + T(n-k-1) + \Theta(n))\Big] \\ &= \sum_{k=0}^{n-1} \mathrm{E}\Big[X_k \cdot (T(k) + T(n-k-1) + \Theta(n))\Big] \\ &= \sum_{k=0}^{n-1} \mathrm{E}\big[X_k\big] \cdot \mathrm{E}\big[T(k) + T(n-k-1) + \Theta(n)\big] \end{aligned}$$

- First step because of linearity of expectation.
- Second step because the $X_k$'s are independent from other $X_k$'s in further recursive calls.



## Analysis of expected running time

$$\begin{aligned} &= \frac{1}{n}\sum_{k=0}^{n-1} \mathrm{E}[T(k)] + \frac{1}{n}\sum_{k=0}^{n-1} \mathrm{E}[T(n-k-1)] + \frac{1}{n}\sum_{k=0}^{n-1} \mathrm{E}[\Theta(n)] \\ &= \frac{2}{n}\sum_{k=0}^{n-1} \Big(\mathrm{E}[T(k)]\Big) + \Theta(n) \end{aligned}$$

We absorb $\mathrm{E}[T(0)]$ and $\mathrm{E}[T(1)]$ into the $\Theta(n)$ term (will see why in a moment.)

$$\mathrm{E}[T(n)] = \frac{2}{n}\sum_{k=2}^{n-1} \Big(\mathrm{E}[T(k)]\Big) + \Theta(n)$$

## Analysis of expected running time

- We're going to try to prove that

$$E[T(n)] \leq c \cdot n \lg n$$

for some constant $c > 0$ and for sufficiently large $n$.

- This is why we removed $T(0)$ and $T(1)$ from the recurrence. ($\lg 0$ and $\lg 1$ would give us a problem!)

- Will use the following fact:

$$\sum_{k=2}^{n-1} k \lg k \leq \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2$$



## Proof using the substitution method

$$\begin{aligned}
E[T(n)] &= \frac{2}{n} \sum_{k=2}^{n-1} \Big( E[T(k)] \Big) + \Theta(n) \\
&\leq \frac{2}{n} \sum_{k=2}^{n-1} c\ k \lg k + \Theta(n) \\
&\leq \frac{2c}{n} \left( \frac{1}{2} n^2 \lg n - \frac{1}{8} n^2 \right) + \Theta(n) \\
&= c\ n \lg n - \left( \frac{cn}{4} - \Theta(n) \right) \\
&\leq c\ n \lg n \qquad \text{as long as } \frac{cn}{4} - \Theta(n) \geq 0
\end{aligned}$$

Can choose $c$ big enough so that $cn/4$ dominates the $\Theta(n)$ term $\Longrightarrow$ expected running time of RANDOMIZED-QUICKSORT is $O(n \lg n)$