Algorithms and Data Structures

Chapter 7

Catherine Durso

cdurso@cs.du.edu

Chapter 7

Quicksort is a comparison sort that is not stable and has $\Theta\left(n^2\right)$ worst case running time. However, Randomized Quicksort has $O\left(n \lg n\right)$ expected running time. It is optimized to have very good performance on average compared to known comparison sorts with $\Theta\left(n \lg n\right)$ expected running times.

Comparison Sorting

A comparison sort is a sort that uses only the $\leq$ operator among keys to sort the data. Comparison sorts have a theoretical $\Omega(n \lg n)$ bound on running time. Roughly, this follows because a comparison sort of n items must be able to return any of the $n!$ possible permutations of the input order. There must be $n!$ paths through the comparisons. Since each comparison results in two branches, there must be at least $\lg(n!)$comparisons.

Sorts that take advantage of features of the data can be $\Theta(n)$. For example, if the values are known to be n positive integers bounded by Mn for some finite M, they can be sorted in linear time by reading them into a direct address table, then reading them out in order.

Partitioning

The basic component of Quicksort is a routine, PARTITION. For a value x in the array $A[p, r]$,PARTITION rearranges A so that for some q, $x = A[q]$, $s \in \{p, ..q - 1\} \rightarrow A[s] \leq x$, and $s \in \{q + 1, ..r\} \rightarrow A[s] > x$.The routine returns q.

The PARTITION routine works by maintaining two iterators, i, and j, with values in $A[p, i]$ all less than or equal to x, and values in $A[i + 1, j]$ all greater than x.

PARTITION

PARTITION (A, p, r)

1. $x = A[r]$
2. $i = p - 1$
3. for $j = p$ to $r - 1$
4.   if $A[j] \leq x$
5. $i = i + 1$
6.   exchange $A[i]$ with $A[j]$
7. exchange $A[i + 1]$ with $A[r]$
8. return $i + 1$

QUICKSORT

QUICKSORT (A, p, r)

1. if $p < r$
2. $\quad q =$ PARTITION(A, p, r)
3. $\quad$ QUICKSORT $(A, p, q-1)$
4. $\quad$ QUICKSORT $(A, q+1, r)$

Performance

If, miraculously, every partition value set in the recursive calls is the median of its array, QUICKSORT will have the recurrence
$T(n) \leq 2T\left(\frac{n}{2}\right) + cn$, so $\Theta\left(n \lg n\right)$ running time.

If, miraculously, every partition value set in the recursive calls is the minimum of its array, QUICKSORT will have the recurrence
$T(n) \leq T(n-1) + cn$, so $\Theta\left(n^2\right)$ running time.

Randomizing

RANDOMIZED-PARTITION (A, p, r)

1. $i =$ RANDOM (p, r)
2. exchange $A[r]$ with $A[i]$
3. return PARTITION (A, p, r)

Randomized Quicksort

RANDOMIZED-QUICKSORT (A, p, r)

1. if $p < r$
2. $q =$ RANDOMIZED-PARTITION(A, p, r)
3. RANDOMIZED-QUICKSORT $(A, p, q - 1)$
4. RANDOMIZED-QUICKSORT $(A, q + 1, r)$

Expected Running Time

Note that the running time of all the RANDOMIZED-PARTITION calls together gives a $\Theta-$bound on the running time of RANDOMIZED-QUICKSORT.

The expected number of total comparisons in the calls to RANDOMIZED-PARTITION gives a $\Theta-$bound on the expected running time of RANDOMIZED-PARTITION.

Indicator RVs

To simplify notation, denote the elements of A in sorted order by $z_1 \leq z_2 \leq ..z_n$. Define the indicator random variable $X_{ij} =$
$I\{z_i$ is compared to z_j in a particular execution of RANDOMIZED-QUIC

Observe that comparison takes place only between a pivot and a non-pivot. Values are used as pivots at most once. If z_k with $z_i < z_k < z_j$ is chosen as a pivot before z_i and before z_j, then z_i and z_j will not be compared in that execution of RANDOMIZED-QUICKSORT.

$E[X_{ij}]$

$Pr(z_i \text{ compared to } z_j)$ is equal to the probability that the first pivot selected from $\{z_i, z_{i+1}..z_j\}$ is z_i or z_j. This is $\frac{2}{j-i+1}$.

The expected total number of comparisons is
$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1} = \sum_{i=1}^{n-1} \sum_{k=2}^{n-i+1} \frac{2}{k}$
$\leq \sum_{i=1}^{n-1} \sum_{k=2}^{n} \frac{2}{k} \leq 2 \sum_{i=1}^{n} \ln(n) = O(n \lg n)$

Conclude that the expected running time of RANDOMIZED-QUICKSORT is $O(n \lg n)$.