Design and Analysis of Algorithms I

QuickSort

Choosing a Good Pivot

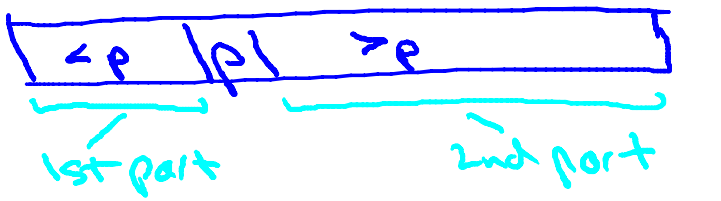
QuickSort: High-Level Description

[Hoare circa 1961]

QuickSort(array A, length n)

- if n = 1 return
- p = ChoosePivot(A,n)
- Partition A around p
- recursively sort 1st part
- recursively sort 2nd part

[currently unimplemented]



The Importance of the Pivot

Q: Running time of QuickSort?

A: depends on the quality of the pivot.

Suppose we implement QuickSort so that ChoosePivot always selects the first element of the array. What is the running time of this algorithm on an input array that is already sorted?

- Not enough information to answer question
- $\theta(n)$
- $\theta(n \log n)$
- $\theta(n^2)$

Reason:

Run time: $\geq n + (n-1) + (n-2) + \cdots + 1$

$= \Theta(n^2)$

let $n/2$ terms all at least $\frac{n}{2}$

Suppose we run QuickSort on some input, and, magically, every recursive call chooses the median element of its subarray as its pivot. What's the running time in this case?

- ○ Not enough information to answer question
- ○ $\theta(n)$
- ○ $\theta(n \log n)$
- ○ $\theta(n^2)$

Reason: let $T(n)$ = running time on arrays of size n.

Then: $T(n) \leq 2T(\frac{n}{2}) + \Theta(n)$ (because pivot = median; choose pivot, partition)

$\Rightarrow T(n) = \theta(n \log n)$ [like MergeSort]

Random Pivots

Key question: how to choose pivots? BIG IDEA: RANDOM PIVOTS!

That is: in every recursive call, choose the pivot randomly.
(each element equally likely)

Hope: a random pivot is "pretty good" "often enough".

Intuition: ① if always get a 25-75 split, good enough for $O(n \log n)$ running time. [non-trivial exercise: prove via recursion tree]

② half of elements give a 25-75 split or better

Q: does this really work?

Average Running Time of QuickSort

Quicksort Theorem: for every input array of length n, the average running time of Quicksort (with random pivots) is $O(n \log n)$.

Note: holds for every input. [no assumptions on the data]

- recall our guiding principles!
- "average" is over random choices made by the algorithm (i.e., pivot choices)