QuickSort

Overview

Design and Analysis
of Algorithms I

QuickSort

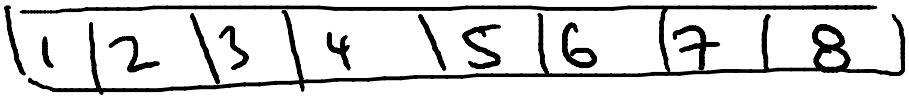
- Definitely a "greatest hit" algorithm
- Prevalent in practice
- Beautiful analysis
- $O(n \log n)$ time "on average", works in place
  - i.e., minimal extra memory needed
- See course site for optional lecture notes

Tim Roughgarden

The Sorting Problem

Input: array of n numbers, unsorted.

Output: Same numbers, sorted in increasing order.



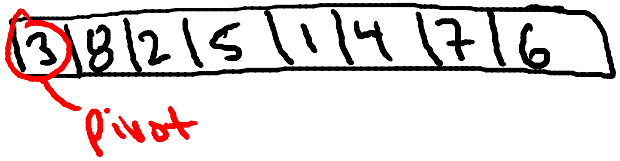
Assume: all array entries distinct.

Exercise: extend QuickSort to handle duplicate entries.

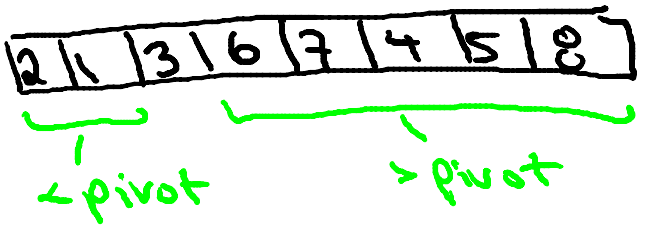
Partitioning Around a Pivot

Key idea: partition array around a pivot element.

- pick element of array



- rearrange array so that:
  - left of pivot $\Rightarrow$ less than pivot
  - right of pivot $\Rightarrow$ greater than pivot



Note: puts pivot in its "rightful position".

Two Cool Facts About Partition

(1) linear ($O(n)$) time, no extra memory
[see next video]

(2) reduces problem size

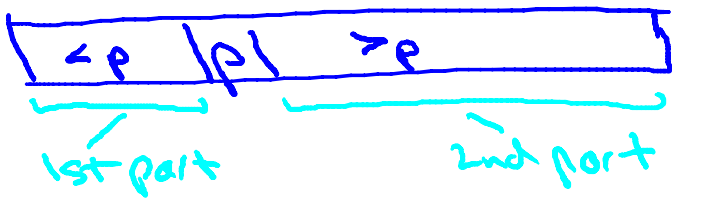
QuickSort: High-Level Description

[Hoare circa 1961]

QuickSort(array A, length n)

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

[currently unimplemented]



Outline of QuickSort Videos

- The Partition subroutine
- Correctness proof [optional]
- Choosing a good pivot
- Randomized QuickSort
- Analysis
  - A Decomposition Principle
  - The Key Insight
  - Final Calculations