



Design and Analysis
of Algorithms I

QuickSort

Overview

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

The Sorting Problem

Input : array of n numbers, unsorted

3	8	2	5	1	4	7	6
---	---	---	---	---	---	---	---

Output : Same numbers, sorted in increasing order

1	2	3	4	5	6	7	8
---	---	---	---	---	---	---	---

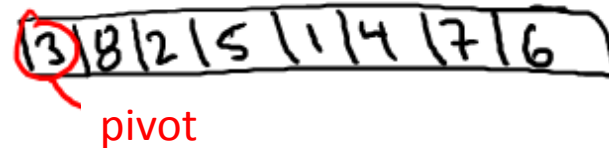
Assume : all array entries distinct.

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 => less than pivot

-Right of pivot => greater than pivot



Note : puts pivot in its “rightful position”.

Note: We do not insist anything about the relative ordering of the items less than the pivot and items more than the pivot

Two Cool Facts About Partition

1. Linear $O(n)$ time, no extra memory
[see next video]
2. Reduces problem size

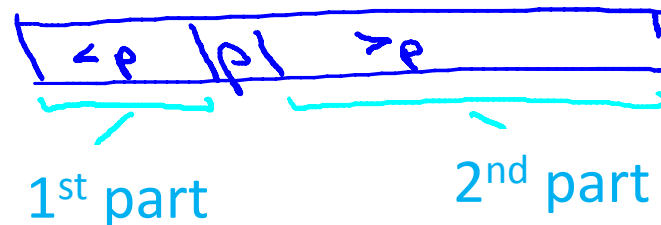
QuickSort: High-Level Description

[Hoare circa 1961]

QuickSort (array A, length n)

- If $n=1$ return
- $p = \text{ChoosePivot}(A, n)$
- 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