Quicksort

Chapter 7

Problem of the Day

- Given two sets N and M, write an algorithm to determine if these sets are disjoint
- What is the running time of your algorithm?

Sorting

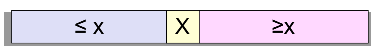
- What's the running time for:
  - Insertion Sort
  - Merge Sort
  - Heapsort
- Which of these algorithms sort in place?

Quicksort

- The Basic version of quicksort was invented by C. A. R. Hoare in 1960
- Divide and Conquer algorithm
- In practice, it is the fastest in-place sorting algorithm

Divide and Conquer

- Divide: Partition the array into two subarrays around a pivot x such that elements to the left are <= x and elements to the right are >= x



- Conquer: Recursively sort the two subarrays
- Combine: Trivial!

Key?

Quicksort Pseudocode

QUICKSORT(A, p, r)

- What's the call to sort the entire array?

Partitioning the Array

PARTITION(A, p, r)

Example

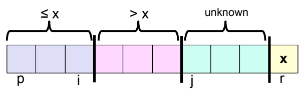
1	2	3	4	5	6	7	8
5	3	9	1	8	2	4	7
p							r

Example

1	2	3	4	5	6	7	8
5	3	1	2	4	7	8	9
p				i			r

Correctness of Partition

- During the execution of PARTITION there are four distinct sections of the array:



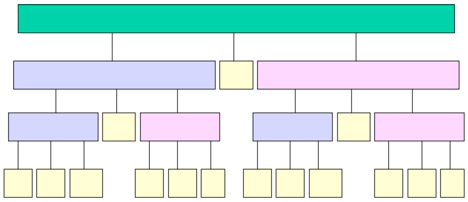
Exercise - Partition the Following

44	**75**	**23**	**43**	**55**	**12**	**64**	**77**	**33**	**41**

Analysis of Partition

- What is the running time of PARTITION?

Quicksort in Action



Exercise

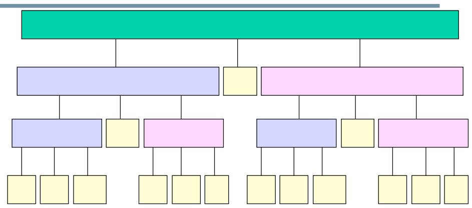
- Sort the following array using quicksort

3	4	2	5	1

Performance of Quicksort

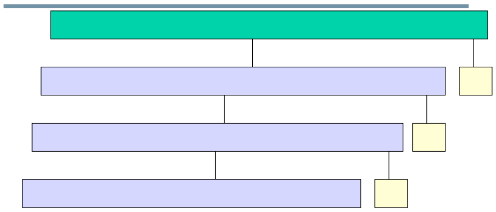
- What does the performance of quicksort depend on?
- What would give us the best case?

Best Case of Quicksort



- The total partitioning on each level is O(n), and it take lg n levels of perfect partitions to get to single element subproblems. When we are down to single elements, the problems are sorted. Thus the total time in the best case is O(n lg n).

Worst Case of Quick Sort



- Suppose our pivot element splits the array as unequally as possible. Thus instead of n/2 elements in the larger half, we get zero, meaning that the pivot element is the biggest element in the array. Now we have n-1 levels, instead of lg n
- T(n) = T(n-1) + O(n), which has a solution of $O(n^2)$

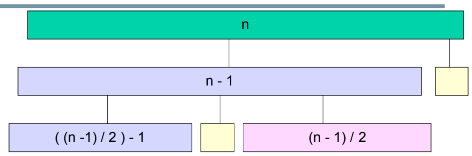
Quicksort Analysis

- To justify its name, Quicksort had better be good in the average case.
- Showing this requires some intricate analysis.

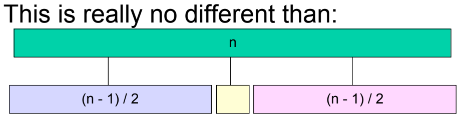
Average Case Analysis

- Let's look at this by intuition
- Running quicksort on a random array is likely to produce a mix of balanced and unbalanced partitions
- It has been shown that 80% of the time partition produces good splits and 20% of the time it produces bad splits

Average Case Analysis



- This is really no different than:



- Thus, the O(n -1) of the bad split can be absorbed into the O(n) of the good split

Average Case Analysis

- The running time of quicksort when alternating good and bad splits is like the running time for good splits alone
- O(n lg n) but with a slightly larger constant hidden by the O-notation
- More of that next time!!!!