Lecture 8

Thursday, 1 August 2013
11:08 a.m.

	Insertion sort	Merge sort	Quicksort
Parallelise	No	Yes	Yes
In-place	Yes	No	Yes
Stable	Yes	No	No
On sorted Input	O(n)	O(n.log(n))	$O(n^2)$

Parallel - is being able to be processed on multiple processes (multicore)
In place - can be sorted without creating new arrays
Stable - preserving original order when key values are equal (the left most 2 stays on the left side of all other 2's)
On sorted Input - how efficient is the sorting if we input an already sorted array.

Quicksort -more detail

The complexity function of T given

$T(1) = 1$

$T(n) - 2T(n/2) + n$

Which is ~~Θ~~(nlog(n))

But this is only if we pick the best pivot point

If the partition pocks the worst pivot (so every sub array split is [1] and [n-1], (one sub array has only 1 value and the other has the rest))
This gives the time complexity of

$T(n) = T(n-1) + n$

Which is ~~Θ~~(n^2)

This is worst case, and its as bad a insertion sort.

To get the average complexity of quicksort, consider an algorithm that always gives you a 90% split (the ratio of sub arrays is 9:1)
This gives a complexity of

$T(n) = T(9n/10) + T(n/10) + n$

Which is still ~~Θ~~(nlog(n)) (but this log is $\log_{9/10}$ instead of $\log_2$)

To make sure we don't always got bad splits we can swap the first one with a randomly selected one then use the new first (random) as the pivot. This means its random every time and with a previously sorted array it wont always have the worst pivot for every split

Or if you don't want to use random number generators you can use the median of 3 technique, Which is to look at the first 3 items and picks the median (middle) one of them and uses that as the key, this usually gives a pretty good median.