*Lecture 12*

Sorting Correctly and Efficiently

Correctness of sorting algorithms

When is a sorting algorithm correct –

* When its output is in non-decreasing order (i.e. the output is sorted according to some total order)
* And when the items in the output are a permutation of the items in the input.

Bubble sort

Example and pseudocode in slides.

Bubble sort is **stable** and **in-place**.

O(1) space complexity (because it is **in-place**)

Worst and average case time complexity is O (n ^ 2)

Comparison-based sorting algorithms

A comparison-based sorting algorithm:

* Is a sorting algorithm
* Can only gain information about the items in the input sequence by performing pairwise-comparisons.

Insertion sort, selection sort and bubble sort are comparison-based algos.

Decision trees

Look at slides. Tells us the different cases of an algorithm.

Merge sort

Example and pseudocode in slides.

It is **stable** (preserves the order of two entries with the same value).

It is **not in-place**: we need an array of at most size *n* to do the merging (space complexity is O(n)).

Merging sub-arrays is **quick** – given two arrays of size *n*, we need to perform at most *n-1* comparisons to merge them.

Recurrence relation: T(n) = 2T(n/2) + O(n) <https://www.youtube.com/watch?v=l0iXqhqfDPo>

Worst case time complexity: O(n \* log(n))

Merge sort is **asymptotically optimal.**

Quick sort

<https://www.youtube.com/watch?v=ZHVk2blR45Q>

Example and pseudocode in slides.

Shit is confusing.

Partitioning an array of size n takes Big-theta (n) operations.

When the pivot element is the smallest for each partitioning, we need n-1 partitioning rounds, i.e. O(n).

Worst case: O(n ^ 2) time complexity.

Ideally, the pivot is the median value and splits the array in half. On average, Quicksort is O(n log(n))

**Lower bound complexity of comparison-based sorts is Big-Omega (n log2(n)) ???? Lecture 13 says its Big-Omega (n log(n)).**