# CIS 575. Introduction to Algorithm Analysis
# Material for March 6, 2024

## Perspectives on Sorting



## 1 Comparison of Sorting Algorithms

We have at this point seen four sorting algorithms, of quite different flavor, and analyzed their resource use. While running time and space use are important attributes for any algorithm, sorting algorithms are often required to be *stable* where a **stable** algorithm is one that keeps the *order* of items with *identical* keys.

InsertionSort can easily be made stable (don't swap two elements if their keys are equal), as can MergeSort.

But HeapSort is not stable: to see this, consider the sequence $R7a, R7b, R5$ where the records $R7a$ and $R7b$ both have 7 as their key whereas $R5$ has key 5. This already forms a heap, and thus the root $R7a$ will be placed at its ultimate destination at the end, eventually yielding the sorted sequence $R5, R7b, R7a$.

Also QuickSort will not be stable if it uses the Dutch National Flag algorithm which we have already seen is not stable.

Our results can be summarized in the table

| | QuickSort | MergeSort | HeapSort | InsertionSort |
|---|---|---|---|---|
| time (worst-case) in $O(n \lg(n))$? | No | Yes | Yes | No |
| in-place? | Yes | No | Yes | Yes |
| stable? | No | Yes | No | Yes |

We see that each of the 4 sorting algorithms has 2 of the 3 desirable properties, except for QuickSort which has only one, but can take comfort in being typically the fastest...

## 2 Sorting is Highly Relevant

Why are we so focused on this particular task? The *Cormen* textbook, in the introduction to its part II, writes (p.158) that

> Many computer scientists consider sorting to be the most fundamental problem in the study of algorithms

and lists several reasons, such as (paraphrased)

- real-world applications almost always make use of sorting, to present the results and/or internally to facilitate processing;

- there are a variety of sorting algorithms, developed using a wide range of techniques;
- the problem is "solved" in the sense that we cannot hope for a new algorithm that is asympotically faster than the existing ones.

The last claim may appear surprising: how can we know there doesn't exist say a linear-time sorting algorithm?

- Indeed, there *does* exist sorting algorithms that under certain assumptions run in linear time
- but as we shall soon prove, a sorting algorithm based on *comparisons* must run in time $\Omega(n \lg(n))$, in particular *not* in linear time.