# Properties of Sorting Algorithms

Time (worst-case) in $O(n \lg(n))$?

- QuickSort : No
- MergeSort : Yes
- HeapSort : Yes
- InsertionSort: No

In-place?

- QuickSort : Yes
- MergeSort : No
- HeapSort : Yes
- InsertionSort: Yes

Stable? (records with same keys keep their order)

- QuickSort : No
- MergeSort : Yes
- HeapSort : No
- InsertionSort: Yes

# Sorting: a Fundamental Problem

Cormen (p.158): *Many computer scientists consider sorting to be the most fundamental problem in the study of algorithms* since (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.

We shall next substantiate the last claim!

# Lower Bounds

Consider a given problem, like sorting

- until now, we have analyzed specific algorithms, giving upper/lower bounds.
- what about giving a lower bound that holds for all algorithms one may devise?

Information theory may help us.

- to always guess a number between 1 and 16 with only yes/no questions we need at least 4 questions
- to always choose among $k$ options with only yes/no questions we need at least $\lg(k)$ questions

Many algorithms are essentially exploring (binary) decision trees

- whose height measures worst-case running time
- if the algorithm must distinguish between $k$ eventual outcomes then the decision tree must have at least $k$ leaves and hence its height must be $\geq \lg(k)$

# Comparison-based Sorting

Many sorting algorithms are based on a sequence of comparisons, where the outcome of one comparison determines which two elements to compare next.

- INSERTION-SORT has an unbalanced decision tree in that it contains comparisons not likely to gain much information
- HEAP-SORT has a balanced decision tree but contains redundant comparisons

There are $n!$ outcomes so any decision tree must have height at least $\lg(n!)$ which we can estimate:

$$\lg(n!) = \lg(\prod_{i=1}^{n} i) = \sum_{i=1}^{n} \lg(i) \in \Theta(n \lg(n))$$

Theorem: all sorting algorithms that are based on comparisons will have worst-case behavior in $\Omega(n \lg(n))$.

Caveat: some sorting algorithms work without performing comparisons and may for example run in time $\Theta(n + k)$ with $k$ the largest possible key.