Sorting:
– Intro: aspects of sorting, different strategies
– Insertion Sort, Selection Sort,
– Quick Sort
– Heap Sort,
– Merge Sort
– Theoretical lower bound for comparison-based sorting,
– Digital Sorting: BucketSort, RadixSort

The Sorting Problem
Input:
• A list \( L \) of data items with keys
Output:
• A list \( L' \) of the same data items in increasing order of keys,

Caution!
• Don’t over use sorting!
• Do you really need to have it sorted, or will a dictionary do fine instead of a sorted array?

Aspects of Sorting:
• Internal / External
• In-place / With additional memory
• Stable / unstable
• Comparison-based / Digital

Strategies
• Insertion sorts:
  For each new element to add to the sorted set, look for the right place in that set to put the element...
  Linear insertion, Shell sort, ...
• Selection sorts:
  In each iteration, search the unsorted set for the smallest (largest) remaining item to add to the end of the sorted set
  Straight selection, Heap sort, ...
• Exchange sorts:
  Browse back and forth in some pattern, and whenever we are looking at a pair with wrong relative order, swap them...
  Quick sort, Merge sort...
### Linear insertion sort

"In each iteration, insert the first item from the unsorted part to its proper place in the sorted part."

In-place $A[0..n-1]$!

Data stored in $A[0..n-1]$ from $i=1$ to $n-1$:
- Sorted data in $A[0..i-1]$
- Unsorted data in $A[i..n-1]$

- Scan sorted part for index $s$ for insertion of the selected item
- Increase $i$

### Selection sort

"In each iteration, search the unsorted set for the smallest remaining item to add to the end of the sorted set."

In-place $A[0..n-1]$!

Data stored in $A[0..n-1]$ from $i=1$ to $n-1$:
- Sorted data in $A[0..i-1]$
- Unsorted data in $A[i..n-1]$

- Find index $s$ of smallest key
- Swap places for $A[i]$ and $A[s]$