Sorting Algorithms (I)

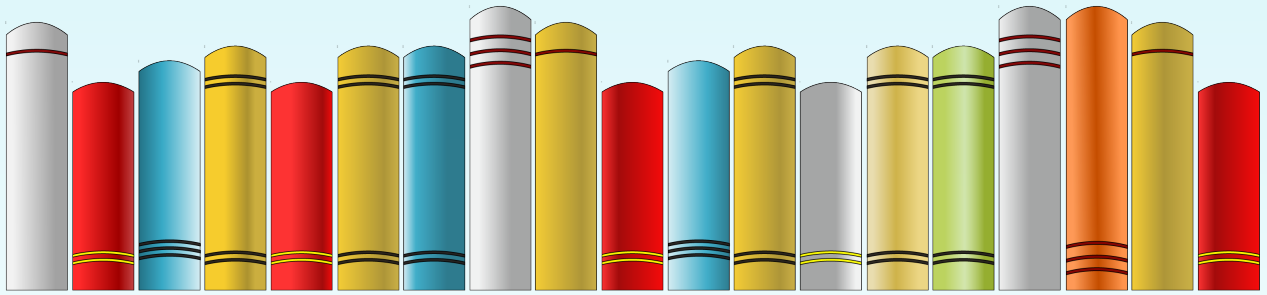
2023

Sarah Chan
sarah.chan@uwaterloo.ca

The Centre for Education in Mathematics and Computing
Faculty of Mathematics, University of Waterloo
www.cemc.uwaterloo.ca

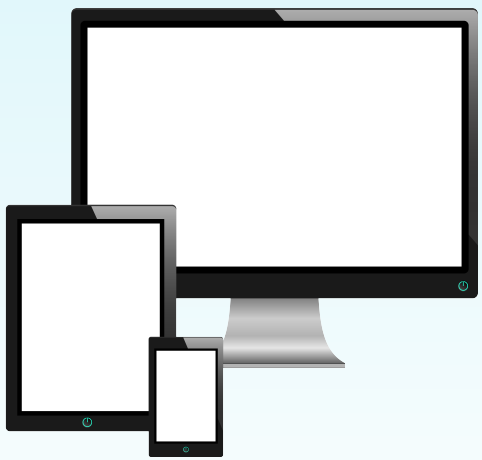
Sorting

We have all sorted items before. We might sort our clothes into different drawers. We might arrange our books alphabetically by title or by author. In the kitchen, we might separate our cutlery into forks, spoons, knives, etc.



Sorting

Computing devices sort all the time as well. Files are arranged by file name, date of creation, size, or type. Contacts may be sorted alphabetically. Emails are grouped by sender, subject, or date received. Streaming services organize programs into categories or genres.



Sorting

Sorting takes both time and effort. So why do we bother?

Sorting

Sorting takes both time and effort. So why do we bother?

When a collection is sorted it improves our ability to:

- Search for a specific item
- Identify extreme items (outliers)
- Find duplicates
- Merge two collections
- Visualize the overall collection

Sorting

Because of the huge amount of data in our world, and the advantages of working with sorted data, being able to *sort well* is an extremely important skill.

A **sorting algorithm** is a method or technique for organizing a large number of items into a specific order.

Dozens of different sorting algorithms exist.

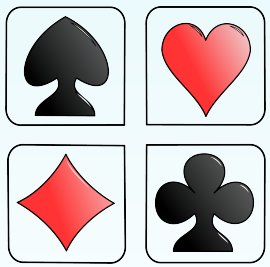
Sorting

Because of the huge amount of data in our world, and the advantages of working with sorted data, being able to *sort well* is an extremely important skill.

A **sorting algorithm** is a method or technique for organizing a large number of items into a specific order.

Dozens of different sorting algorithms exist.

How would *you* sort a deck of cards?

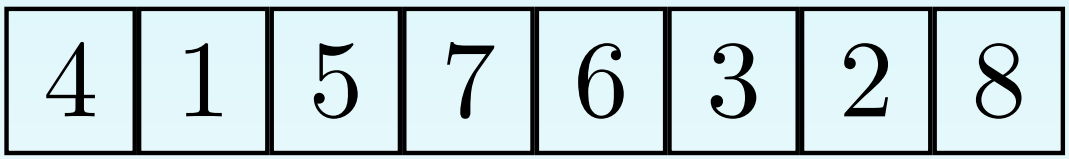


Selection Sort

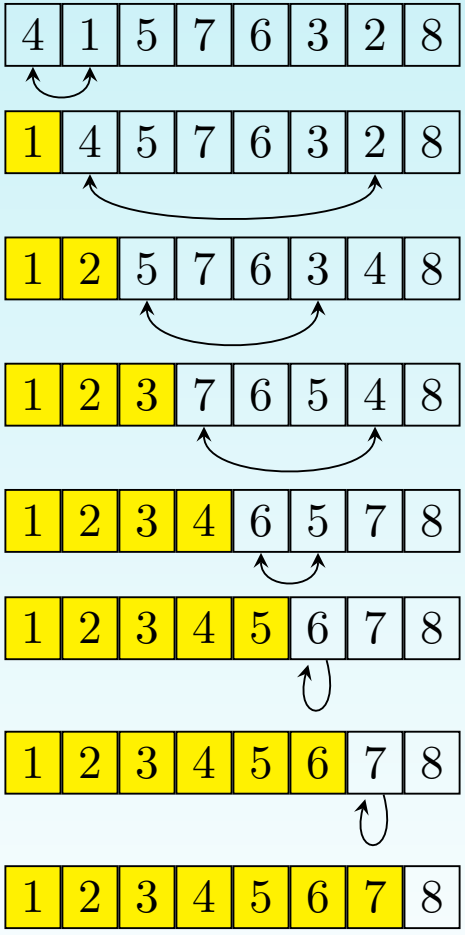
Find the minimum element in the unsorted list and swap it with the element at the front of the unsorted list. This element is now sorted and no longer part of the unsorted list.

Repeat until the unsorted list has just one element remaining.

Example:



Selection Sort: Example



Selection Sort: Problem Set

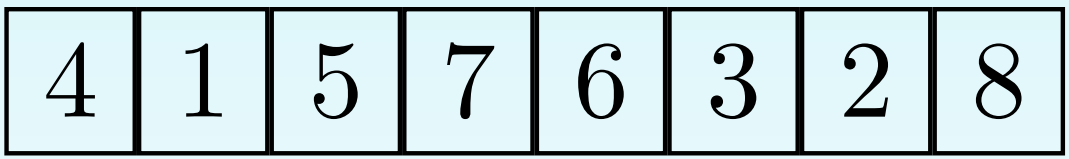
1. Sort the following list of letters in alphabetical order using selection sort: **D E A C F B**
2. If the unsorted list contains n elements, how many swaps does the algorithm make?
3. How many comparisons does the algorithm make? That is, how many times does it need to compare two elements?
4. How does the algorithm perform on data that is already sorted? How does it perform on data that is sorted in reverse?

Insertion Sort

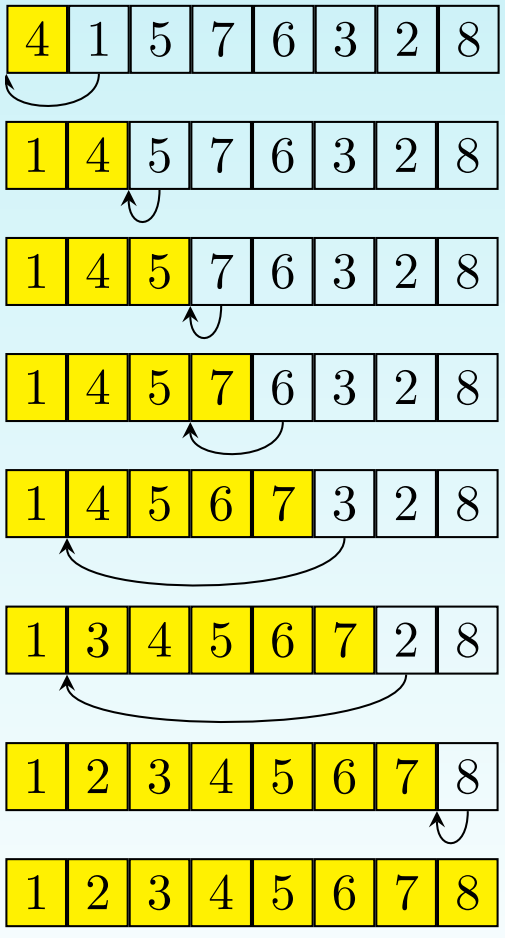
Start with the second element in the list. Shift the element left until it is in its relative proper position.

Repeat by shifting the third, fourth, fifth element etc. until the last element has been shifted.

Example:



Insertion Sort: Example



Insertion Sort: Problem Set

1. Sort the following list of letters in alphabetical order using insertion sort: **E B A F C D**
2. If the unsorted list contains n elements, how many elements need to be shifted?
3. How many comparisons does the algorithm make? That is, how many times does it need to compare two elements?
4. How does the algorithm perform on data that is already sorted? How does it perform on data that is sorted in reverse?

Bubble Sort

Starting with the first two elements, make a full pass through the unsorted list and swap adjacent elements that are in the wrong relative order.

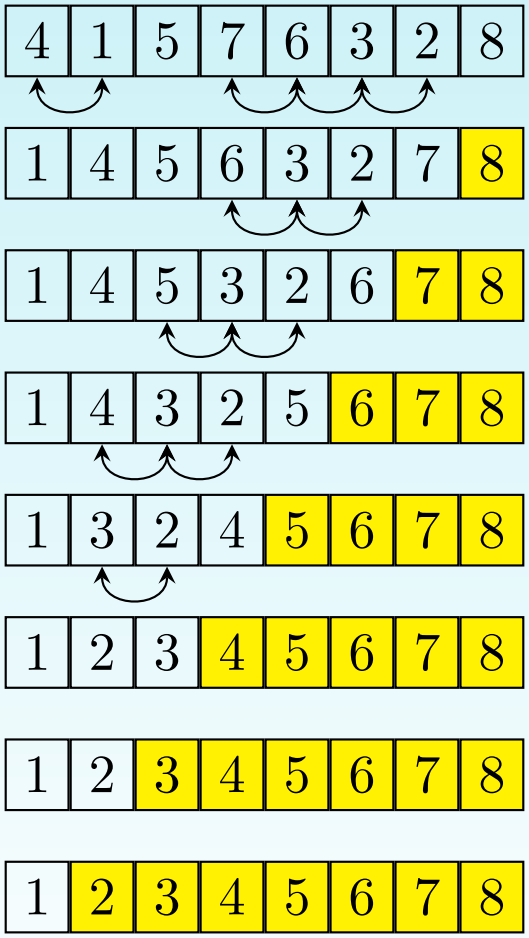
The largest element *bubbles* to the right and is now sorted and no longer part of the unsorted list.

Repeat until the unsorted list has just one element remaining.

Example:

4	1	5	7	6	3	2	8

Bubble Sort: Example



Bubble Sort: Problem Set

1. Sort the following list of letters in alphabetical order using bubble sort: **B D F C A E**
2. If the unsorted list contains n elements, how many passes does the algorithm make?
3. How many comparisons does the algorithm make?
4. How many swaps does the algorithm make?
5. How can the algorithm be improved so that already sorted or nearly sorted data has more of an advantage?

Challenge: Problem Set

1. If your data consists of only integers, **Radix Sort** is another viable sorting algorithm. Study the example below and explain how the Radix Sort algorithm works.

170	045	075	090	802	024	002	066
170	090	802	002	024	045	075	066
802	002	024	045	066	170	075	090
002	024	045	066	075	090	170	802

Hint: Consider the individual digits of each element.

Challenge: Problem Set

2. Here is an algorithm to sort four animals from lightest to heaviest:

   - Randomly pick two animals and compare their weights.
   - Compare the weights of the other two animals.
   - Compare the weights of the heaviest animal from each pair. (The "heavies")
   - Compare the weights of the lightest animal from each pair. (The "lights")
   - Arrange the animals as follows: lightest of the "lights", heaviest of the "lights", lightest of the "heavies", and heaviest of the "heavies".

   Unfortunately this algorithm will sometimes fail. When? What is the probability that this algorithm will succeed?