1.4 Algorithms

Description

Do we have a section on data and/or information at a general level?
What is data and/or information?

As mentioned, a program being executed carries out a number of actions for accomplishing a task or solving a problem. The procedure realised by a program is often referred to as an algorithm.

An *algorithm* is a set/sequence of instructions for accomplishing a task, solving a problem, or performing a computation. An algorithm acts as an exact list of instructions that conduct specified actions step by step.

A *sequential algorithm* is an algorithm where the instructions are carried out in sequence. The actions thus happens in a totally ordered sequence.

A *parallel algorithm* is an algorithm where some of the instructions may be carried out at the same time by different actors. This actions thus happens in a sequence that is partially ordered.

Algorithms may be described in natural languages, programming languages, flow charts, pseudo code, etc.

The best way to understand an algorithm is to think of it as a recipe that guides you through a series of well-defined actions to achieve a specific goal. Just like a recipe produces a replicable result, algorithms ensure consistent and reliable outcomes for a wide range of tasks in the digital realm.

One way to understand an algorithm is to think of its as a *cooking recipe* that guides you through a series of well-defined actions to achieve the goal making a given dish. A *knitting pattern* is another example of an algorithm the lines a number of well-defined stations here to produce some knitting.

In general there is not just one algorithm that solves a given problem; different algorithms can be designed to solve the same problem, with each using a distinct approach, but achieving the same result. For a given dish like pasta Carbonara there are also many different recipes, however, not all leading to the the exact same dish!

In computing a main property of an algorithm is its efficiency with respect to execution time and and use of data space.

An algorithm makes use of entities that either are transformed by the algorithm to the results of the algorithm or are used to perform the actions of the algorithm. For a cooking recipe, the entries are the ingredients for the dish and the kitchen equipment and of course the chef being used to make the dish.

For a computer algorithm the entities are the objects manipulated by the algorithm. For algorithms performing a computation, the entities are the *input data* en the restart/result is the *output data* computed by the algorithm.

As an example of algorithm, consider a deck of cards that we want to sort in the order of Clubs, Spades, Hearts and Diamonds, and within each suit, they should be sorted in the order of the number on the card.

Club 7 is before Club 10 (Club 7 < Club 10), Spades 10 is before Hearts 3 (Hearts 3 > Spades 10), etc.

We use a sorting method called *bubble sort*. It makes use of three card piles, a *left pile*, a *right pile* and a *sorted pile*.

Here is a *pseudocode* description of the bubble sort algorithm: (from https://runestone.academy/ns/books/published/mobilecsp/Unit5-Algorithms-Procedural-Abstraction/Sorting-Algorithms.html)
. Linket virker ikke mere!?

Skal sikkert omskrives til en anden stil og en figur vil være fin!
FX en figure med de 3 piles og en hånd

Bubble sort a deck of 13 * 4 cards:

```
1. Place the unsorted deck, face down, in the right hand pile.
2. repeat 13 * 4 times
    1. Put the top card of the right pile in your hand.
    2. repeat until there are no more cards in the right pile.
        ▪ if the card in your hand > the top card on the right pile
          Place top card on the left pile.
        ▪ else Place the hand card on the left pile.
    3. When the pass is finished, put the card left in your hand on the sorted pile.
    4. Move the left pile to the right pile.
```

This pseudocode algorithm contains two steps:

- Step (1), the unsorted deck is placed in the right hand pile.
- Step (2) is a so-called loop statement specified by the keyword **repeat** and a number 4 * 13. This means that the body of the loop is repeated 52 times corresponding to the number of cards.
  The body of the loop contains 4 steps:
  - Step (1), the top card of the right pile is placed in your hand.
  - Step (2) is a so-called conditional statement specified by the keywords **if** and **else**.
    - It consist of a condition that selects between two actions.
    - If the condition *the card in your hand* is greater than *the top card on the right pile*, then the top card is placed on the left pile
    - Otherwise, the hand card is placed on the left pile.
  - Step (3), the card left in your hand is placed on the sorted pile.
  - Step (4) the left pile is moved to the right pile.

The algorithm is called bubble sort since on each pass through the unsorted cards, the card with the highest value bubbles to the top – i.e the one left in the hand of the sorter.

As can be seen, the algorithm makes 52 passes through the unsorted cards. This is an inefficient way of sorting. If we apply the bubble sort technique to large number of data, millions or billions it may be very time consuming. There are sorting algorithms that are more efficient. We encourage the reader to take courses in the design and analysis of algorithms as a necessary supplement of this book.