Algorithms and datastructures
Sorting algorithms

March 6, 2020

1 Introduction

In this and the next lab session, we will compare the speed of several sorting algorithms. We will implement each sorting algorithm as a class that implements the function

`void operator(std::vector<T>& v) const override`.

An object of such a class is called a functor. This operator will take one parameter, an STL vector `std::vector<T>`. This vector is then sorted by the operator according to the implemented algorithm. All those classes (one for each sort method) inherit from the abstract class `Sorteermethode<T>`, which only contains the function `operator(std::vector<T>& v)`, and the function `meet(...)` responsible for measuring the sorting performance.

Sorting methods that generally work fast under certain circumstances (e.g. with small `int`), can be very slow otherwise (e.g. with large `Telefoonboek`). You should find that in your measurement results. Thus, to allow different types of data in the vector to be sorted, the classes must be parameterized, i.e. they must be template classes.

2 Measuring performance

To measure execution times, you are provided with the class `chrono`. However, knowing absolute numbers is not very interesting — so we go for a table of different measurement results that we can compare. If we want to confirm that a method is $O(n^2)$ or $O(\log n)$, then we let $n(=$ length of the vector) vary between two limits (`kortste` and `langste`, with $n_{i+1} = 10\ n_i$). It is also interesting to compare between the initial state of the vectors (random, sorted, inverted, many doubles, ...).

To visualize your results you may want to export them in a `.csv` file. A helper class for creating such files can be found in `csv.h`. Note: the default value of the separator used is '.'. This is not suitable for spreadsheet programs with a non-English-language setting, where floating point numbers are represented with a floating comma. You can find how to produce a `.csv` file where the dot has been replaced by a floating comma in the API of the class `CsvData`:

The class `CsvData` is defined in the `csv.h` header. Each object of the class corresponds to one `.csv` file, in which number values are stored. Since most spreadsheets can create graphs with different data sets, with each data set stored by default in a column, each `CsvData` also stores data in this way.

Finally, copying and comparing keys takes time. The results of the measurement therefore depend on the type of keys that you sort, e.g. `int` or `double`. Now we want to repeat the measurement with strings. However, the problem is that the class `Sortvector` tries to insert random integers into the vector. Now create a new class `Intstring` (numeric string). Its implementation is provided in `intstring.h`. This is a normal string, but you can assign an integer to it. The string is then the number written out in decimal numbers. Compare the results of measurement when using these numbers with the ones you get with `int`s.

3 Assignment

Study the given source code `main.cpp` and use the provided header files to

1. Implement the Sortvector class (`sortvector.h`). Use the widely accepted Marsenne-Twister pseudo random number generator. You may use STL methods as helpers.
2. Implement STL sort (`stlsort.h`). This is simply a method that uses the standard library sorting method.
3. Implement Insertion sort (`insertionsort.h`).
4. Implement Shell sort (`shellsort.h`).
5. Implement Merge sort (`mergesort.h`).

Note that 2.-5. are classes that publicly inherit from the class `Sorteermethode<T>`.

Write an efficient code and avoid copy operations as much as possible!