A benchmark of the shell sort algorithm applied to a custom vector using allocator

Jostein Bratlie and Rune Dalmo

October 8, 2015

Abstract

This is an article template which can be used to generate a report. Several LaTeXfeatures are demonstrated, including how to display maths and algorithms, generate plots and use a bibliography to manage a list of references. Bullet point lists are used here only to indicate missing parts. Consider them as "TODO" lists.

1 Introduction

The C++ programming language comes with a standard library, the *Standard Template Library* (STL) [1], which consists of containers, methods, algorithms and more. In this report we compare custom implementations of a vector container and Shell's sorting algorithm [2] with equivalent components from the STL.

For the case of vectors we compare allocation times, whereas for the Shell sorting algorithm we benchmark and compare run-times of the sort operation when applied to test data for some selected numbers of elements.

At this point, we show how to include an image in the document (see fig. 1).

Figure 1: The famous picture of Lena

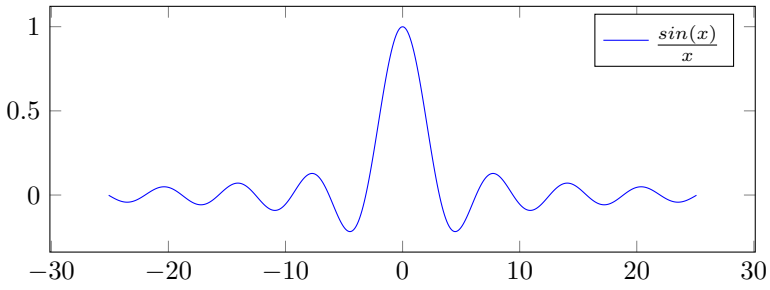

Figure 2: A plot of the function $\frac{sin(x)}{x}$, for $x \in [-25.1, 25.1]$.

2 Vector container

Vector is perhaps the most fundamental and versatile container included in the C++ STL. In this work we present a custom implementation of `vector`. The purpose of implementing an alternative vector is to study its behaviour and to be able to tweak its performance.

- Add a short description of the O-notation (with a reference to Knuth's paper from 1976)[1].
- Recall that the O-notation enables us to describe an algorithm's response to variation of the size of the input data.
- Mathematics can either be placed in the text, such as $f(x) = \frac{1}{x}$, or as equations:

$$f(x) = \int_{-\infty}^{\infty} \frac{sin(x)}{x} dx. \tag{1}$$

- Equations can be referred to. The RHS of formula (1) is a well-known indefinite integral. A plot of $\frac{sin(x)}{x}$ is provided in fig. 2.

3 Bubble sort algorithm

- Describe briefly the bubble sort algorithm (a reference or two to any standard text book on algorithms would be in order).
- Don't forget to mention Bubble sort's performance using the O-notation.

4 Shell sort algorithm

In [2] Shell introduced a high-speed sorting procedure. It has later become known as *Shell sorting*.

[1] Hint: "Big Omicron and Big Omega and Big Theta".

- Complete the short description of Shell's sorting algorithm.
- Use the example algorithm listing environment below as a starting point.

Algorithm 1 Euclid's algorithm

```
1: procedure EUCLID(a, b)                    ▷ The g.c.d. of a and b
2:     r ← a mod b
3:     while r ≠ 0 do                        ▷ We have the answer if r is 0
4:         a ← b
5:         b ← r
6:         r ← a mod b
7:     end while
8:     return b                              ▷ The gcd is b
9: end procedure
```

5 Benchmark set-up

Describe how the benchmark was performed. Be objective - don't describe too many details. The reader which this text is meant for should be able to reconstruct the experiment using his own tools. Some examples of topics for this section:

- All tests were performed on a computer with the following specifications...
- For the container we have measured how much time the CPU has spent on the `push_back()` function. The emplace operation (construct and insert element) was used to allocate the storage.
- For the sorting algorithms, a data set of N integers were sorted, where duration times were measured by using the high precision chrono device which is included with the STL.

6 Results

A graphical visualization of bytes allocated vs. allocation time in milliseconds for the vector benchmark is shown in fig. 3.

Figure 4 shows performance graphs for the Bubble- and Shell sorting algorithms, together with STL's `sort()` for reference, measured in milliseconds for some predefined numbers of items.

7 Concluding remarks

- Reflect over the method and results.

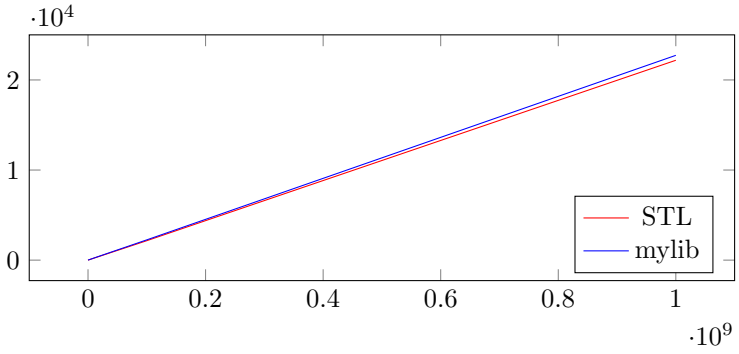

Figure 3: Run-times: *milliseconds* to allocate and construct number of objects * sizeof(int) *bytes* of data.

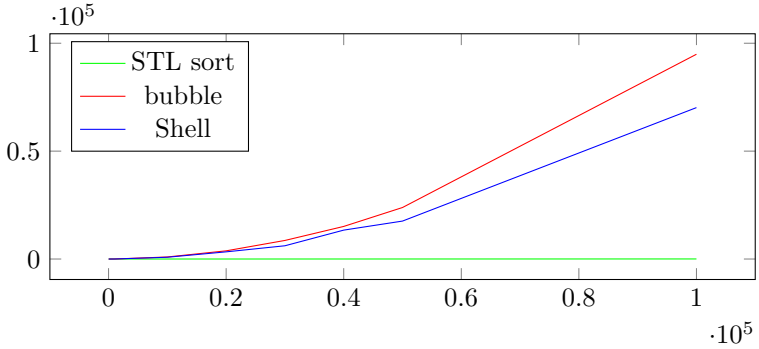

Figure 4: Run-times: *milliseconds* to sort number of objects.

- Topics for future work could be suggested here.

Example: By comparing the performance of the reserve and emplace functionality of the STL version of the vector to our custom vector implementation from *mylib* we conclude that the STL vector is slightly faster at the specified operation.

References

[1] P.J. Plauger, Meng Lee, David Musser, and Alexander A. Stepanov. *C++ Standard Template Library*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2000.

[2] D. L. Shell. A high-speed sorting procedure. *Commun. ACM*, 2(7):30–32, July 1959.