Benchmarking Sorting Algorithms In Python

INF221 Term Paper, NMBU, Autumn 2020

Jon-Mikkel Korsvik
jonkors@nmbu.no

Yva Sandvik
ysandvik@nmbu.no

ABSTRACT

In this paper, we analyse …

1 INTRODUCTION

A sorting algorithm puts the elements of a list in a certain order. It is important that these algorithms are efficient in order to optimize other complex algorithms that implement these sorting algorithms when they require lists to be in a sorted order.

This paper investigates the benchmarks of given, as well as our own, implementations of various sorting algorithms. We explore their efficiency depending on the type of elements the list contains, as well as comparing the sorting algorithms to each other.

2 THEORY

The following algorithms will be investigated in this term paper:

- Quadratic algorithms
  - insertion sort
  - bubble sort
- Sub-quadratic algorithms
  - mergesort
  - quicksort
- Combined algorithm
  - mergesort switching to insertion sort for small data
- Built-in sorting functions
  - Python 'sort()'
  - NumPy 'sort()'

Provide a brief description of the algorithms you will be investigating, including pseudocode for the algorithms. Describe in particular the expected runtime of algorithms in terms of problem size. Use a separate subsection for each algorithm.

2.1 Algorithm 1 - Insertion sort

Listing 1 Insertion sort algorithm from **?**, Ch. 2.1.

INSERTION-SORT(A)

```
1  for j = 2 to A.length
2      key = A[j]
3      i = j − 1
4      while i > 0 and A[i] > key
5          A[i + 1] = A[i]
6          i = i − 1
7      A[i + 1] = key
```

Pseudocode for the first algorithm is shown in Listing 1. Best case runtime for this algorithm is

$$T(n) = \Theta(n) \ . \tag{1}$$

It is achieved for correctly sorted input data.

2.2 Algorithm 2 - Bubblesort

Lorem ipsum …

3 METHODS

In this section, you shall described how you performed the benchmarks:

- What test-data did you use and how was it generated?
- How did you execute your benchmarks?
- How did you measure runtimes?
- What kind of computer did you use, and what software versions?
- Provide git hashes of all relevant files!
- This section should be divided into several subsections.

Table 1: Versions of files used for this report; GitLab repository https://x.y.z.

File	Git hash
`run_bench.ipynb`	`42a44d2a6`
`results.pkl`	`65342aed2`

Listing 2 Draft benchmark setup.

```
times = []
for k in range(1, 5):
    n = 2**k
    times.append(bench(func, n))
```

4 RESULTS

Here, you should present your results. Provide clear figures with tidy lettering in legible font sizes. Figures should not be resized when included, so prepare them for the column width of 84 mm. For good quality, export figures as PDF files from Python.

The results section should be divided into several subsections.

5 DISCUSSION

In this section, you should summarize your results and compare them to expectations from theory presented in Sec. 2.

ACKNOWLEDGMENTS

We are grateful to …for ….

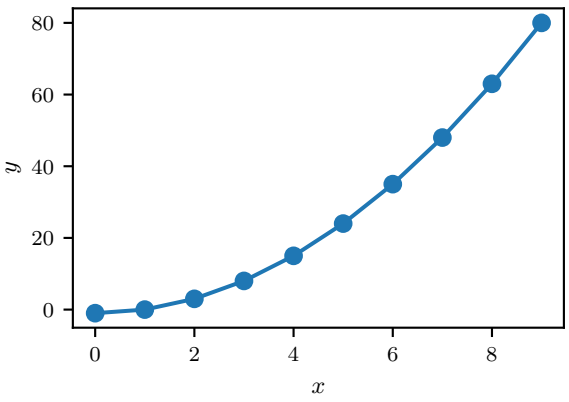

Figure 1: Benchmark results for ....