University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
P. Bamberger
P. Schneider

# Algorithms and Datastructures
# Summer Term 2020
# Sample Solution Exercise Sheet 1

**Due:** Wednesday, 20th of May, 4 pm.

## Exercise 1: Registration *(5 Points)*

Register for the online course system Daphne. You can also find the according link on the Website of this course. Make sure that your data is correct, specifically that you can be reached under the given email address. Then execute the *checkout* command on your SVN-repository.[1]

## Exercise 2: Quicksort *(5 Points)*

Implement the algorithm *QuickSort* from the lecture. A template `QuickSort.py` is provided on the website. Write a unit test both for the `quicksort_divide` and the `quicksort_recursive` method. The unit tests should check at least one non-trivial example. If there are critical cases that are easy to check (e.g., an empty input), you should make a unit test for these cases, too.

## Sample Solution

C.f. `Quicksort.py` in the public folder or on the website.

## Exercise 3: Time Measurement *(5 Points)*

Measure the runtime $T(n)$ of the algorithms *SelectionSort*, *MergeSort*[2] and of your *QuickSort* implementation for different input sizes $n$. You should test *QuickSort* for two different variants of choosing the pivot: Choosing the first element as pivot and choosing a random element as pivot. Repeat the experiment for two different input types: Arrays with random integers and arrays with pairwise distinct integers in descending order.
Plot the runtimes of the *four* algorithms algorithms each with the *two* different input types with input sizes $n \in \{100, 200, \dots, 5000\}$.[3] Use your plots to compare the runtimes and write a short evaluation into the file `experience.txt` (c.f., Task 4).

[1] Your SVN-repository will be created automatically after your registration to Daphne. The URL is `https://daphne.informatik.uni-freiburg.de/ss2020/AlgoDat/svn/your-rz-account-name`

[2] You can find the code for these algorithms in the public repository `https://daphne.informatik.uni-freiburg.de/ss2020/AlgoDat/svn/public`

[3] The differences in runtimes will be most distinct if they are plotted in a single chart with $n$ on the $x$-axis and the runtime $T(n)$ on a *logarithmic* y-axis.

## Sample Solution

Figures 1 and 2 show plots of the running times at different scales. We make the following observations (these are certainly not all observations one can possibly make).

- *SelectionSort* clearly has a super-linear trend (more precisely: a quadratic trend).
- *SelectionSort* is somewaht faster for randomized inputs than for inputs in reverse order (we think that this is due to the fact that in case of a reverse list the line in the inner if-clause is *always* executed, which is not always the case for a randomized input).
- For a deterministic pivot (first element) and a reverse ordered input *QuickSort* has a super-linear trend as well. In fact, *QuickSort* has the lowest of all tested running times in this case (the tested case is a worst case for *QuickSort*).
- On the other hand, *QuickSort* is much faster than all all other variants (c.f., Figure 2) if a randomization of either the input or the pivot takes place (more precisely: the runtime is $\Theta(n \log n)$ *"with high probability"*, c.f. lecture week 2).
- *MergeSort* also has a much better runtime than the algorithms with quadratic trend for *all* tested inputs (more precisely: the runtime is $\Theta(n \log n)$ *guaranteed*, c.f. lecture week 2).

## Exercise 4: Submission *(5 Points)*

Commit your code including the tests and the 8 plots into the SVN, into a subfolder `uebungsblatt-01` (German for exercise sheet 01). Make sure that there are no errors when you run your code (including style check and unit tests) on Jenkins. Commit a file `experience.txt` in which you describe your experiences with this exercise sheet and any problems that may have appeared.

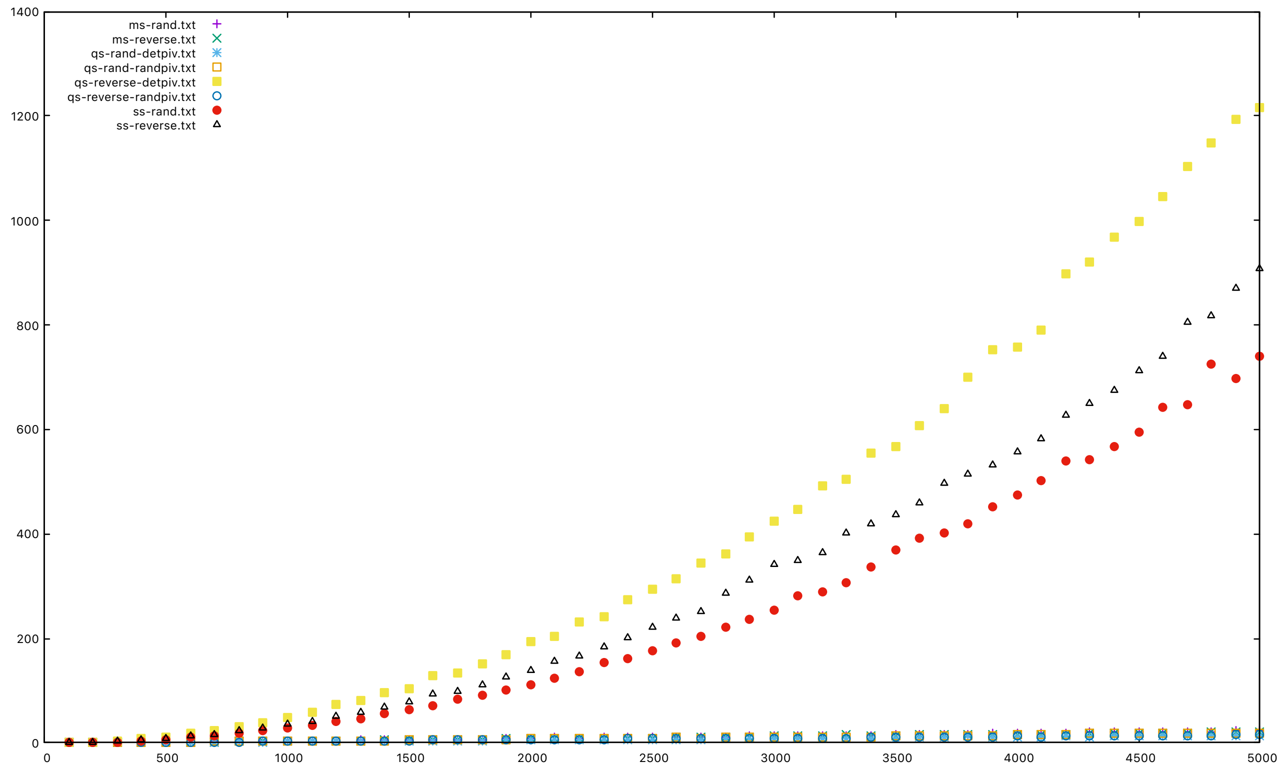


Figure 1: The first plot shows the runtimes of all requested variants of sorting algorithms for the respective inputs over the input size $n$.

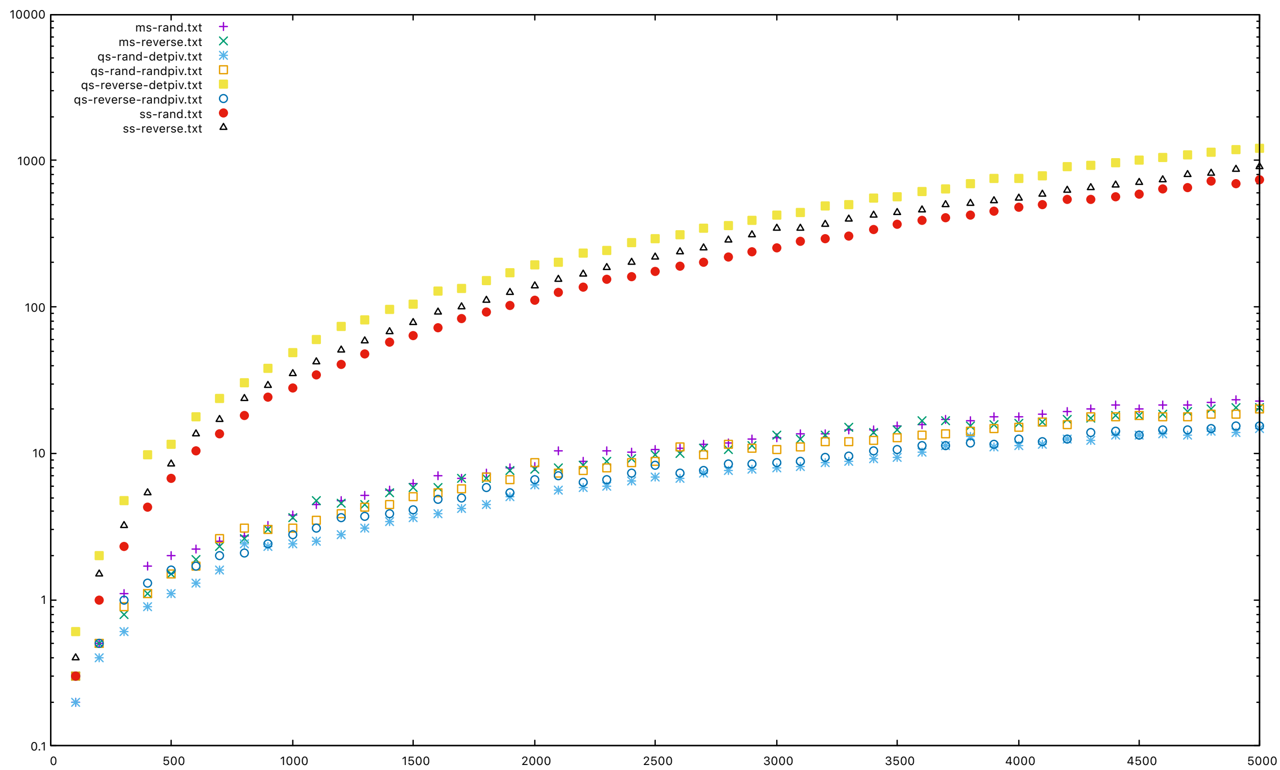


Figure 2: The second plot shows the runtimes of all requested variants of sorting algorithms for the respective inputs over the input size $n$. The $y$ axis is logarithmic.