University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
P. Bamberger, M. Fuchs,
A. Lehmann, P. Schneider

Algorithms and Datastructures
Summer Term 2022
Sample Solution Exercise Sheet 1

Due: Wednesday, May 4th, 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 with two different options of how to choose the pivot element: "Element at first position", "Element at random position". Use the template `QuickSort.py` that is provided on the website. Write a unit test for both 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 of your *QuickSort* implementation for the two variants of choosing the pivot and for two different kinds of inputs. The first kind of inputs are reversed arrays i.e. arrays of the form $[n, n-1, \ldots, 2, 1]$, the second kind are arrays filled with n random integers.
Repeat this for input sizes $n \in \{100, 200, \ldots, 5000\}$.[2] Plot the runtimes of all 4 variants (pivot, input) into the same chart.[3] Use your plots to compare the runtimes and write a short evaluation into the file `experience.txt` (c.f., Task 4).

Sample Solution

Figures 1 and 2 show plots of the running times at different scales. We make the following observations: Quicksort has a super-linear (quadratic) trend for deterministic pivot choice (first element) and input array sorted in descending order. Quicksort is much faster (more precisely: $\Theta(n \log n)$ *"with high*

[1] Your SVN-repository will be created automatically after your registration to Daphne. The URL is `https://daphne.informatik.uni-freiburg.de/ss2022/AlgoDatCond/svn/your-rz-account-name`

[2] A function to generate the arrays and the time measurements is provided in `QuickSort.py`

[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 *linear* and *logarithmic* y-axis.

probability", see lecture week 2) for all other variants where the input array or the choice of pivot is randomized.

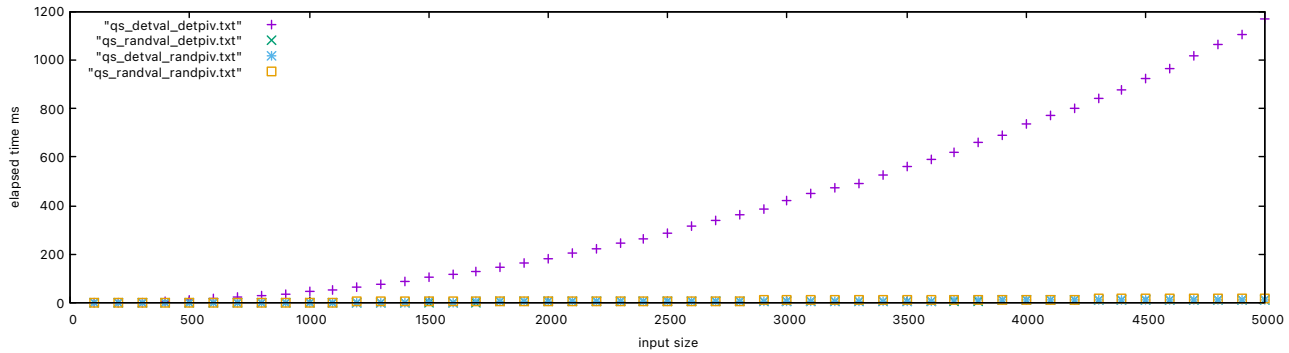

Figure 1: The first plot shows the runtimes of all requested variants of sorting algorithms for the respective inputs over the input size n.

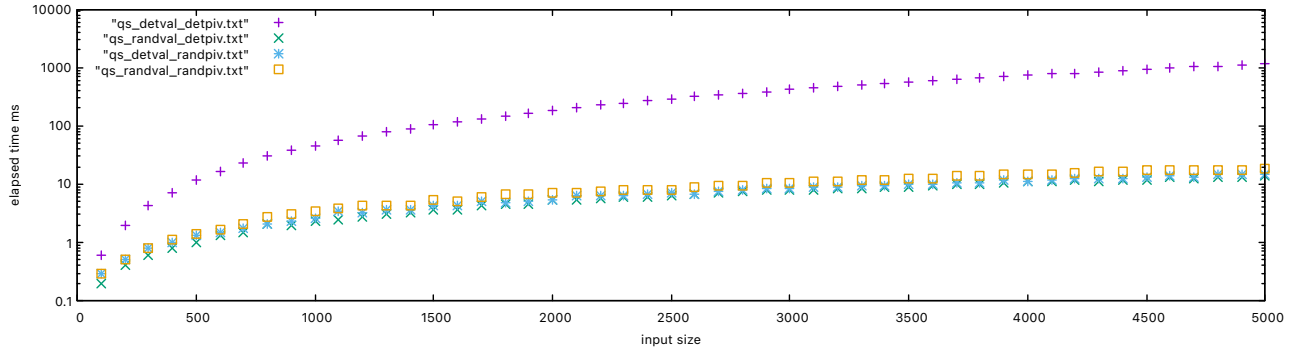

Figure 2: The second plot shows the runtimes of all requested variants of sorting algorithms for the respective inputs over the input size n. The y axis is logarithmic.

Exercise 4: Submission *(5 Points)*

Commit your code including the tests and the plots into the SVN, into a subfolder `exercise-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.