University of Freiburg
Dept. of Computer Science
Prof. Dr. F. Kuhn
M. Fuchs, G. Schmid

Algorithms and Datastructures
Sample Solution Exercise Sheet 1

Exercise 1: Quicksort *(10 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 2: Time Measurement *(10 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 $[n, n-1, \dots, 2, 1]$, the second kind are arrays filled with n random integers.
Repeat this for input sizes $n \in \{100, 200, \dots, 5000\}$.[1] Plot the runtimes of all 4 variants (pivot, input) into the same chart.[2] 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 probability"*, see lecture week 2) for all other variants where the input array or the choice of pivot is randomized.

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

[2] 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.

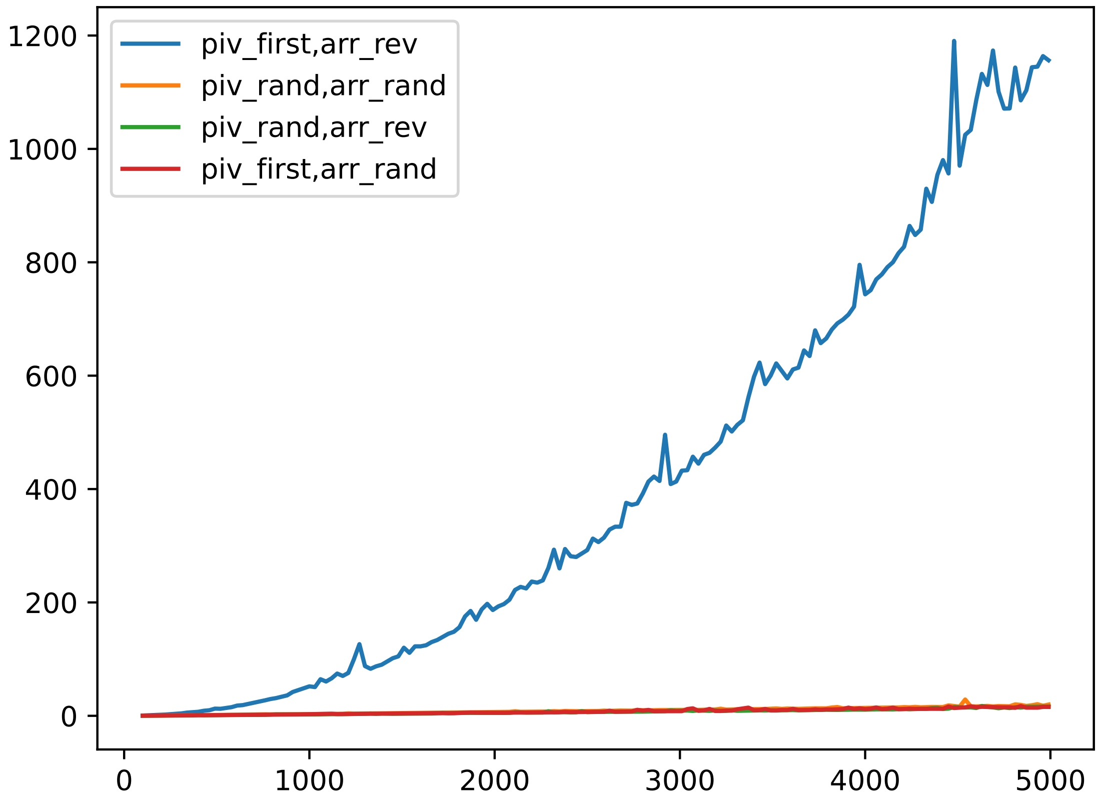

Figure 1: The first plot shows the runtimes of all requested variants of sorting algorithms for the respective inputs over the input size n.

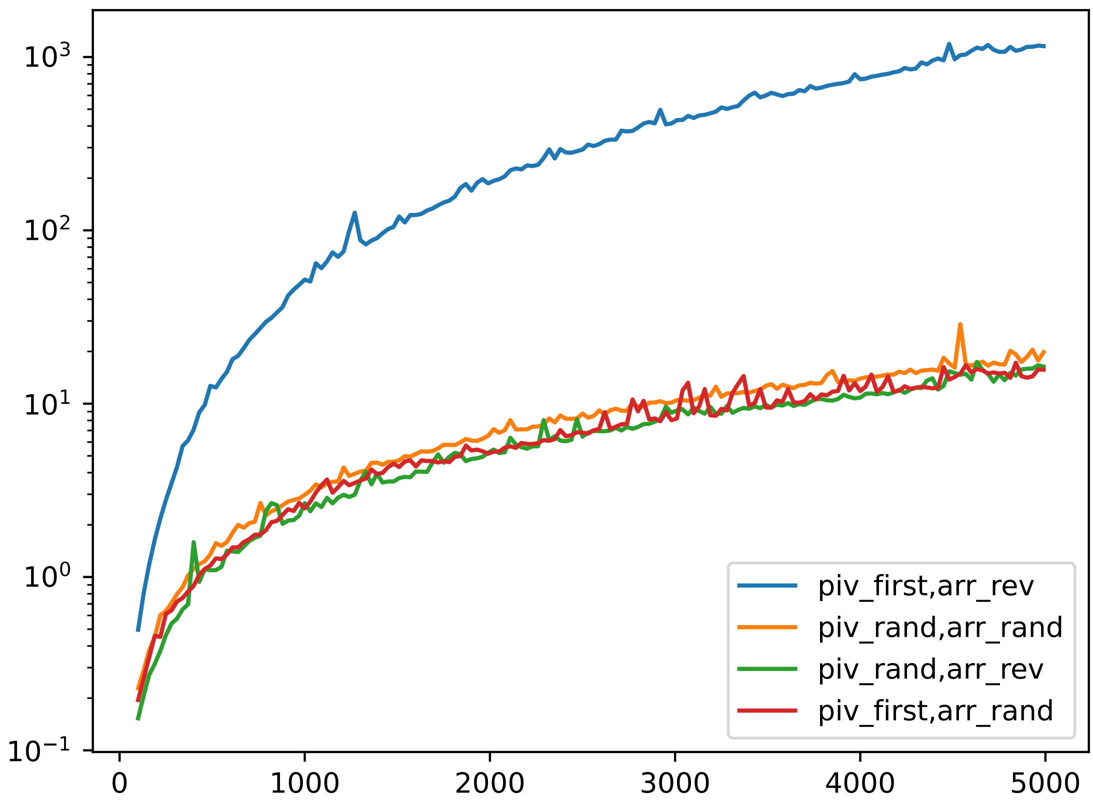

Figure 2: The second plot shows the runtimes of all requested variants of sorting algorithms for the respective inputs over the input size n. The y axis is logarithmic.