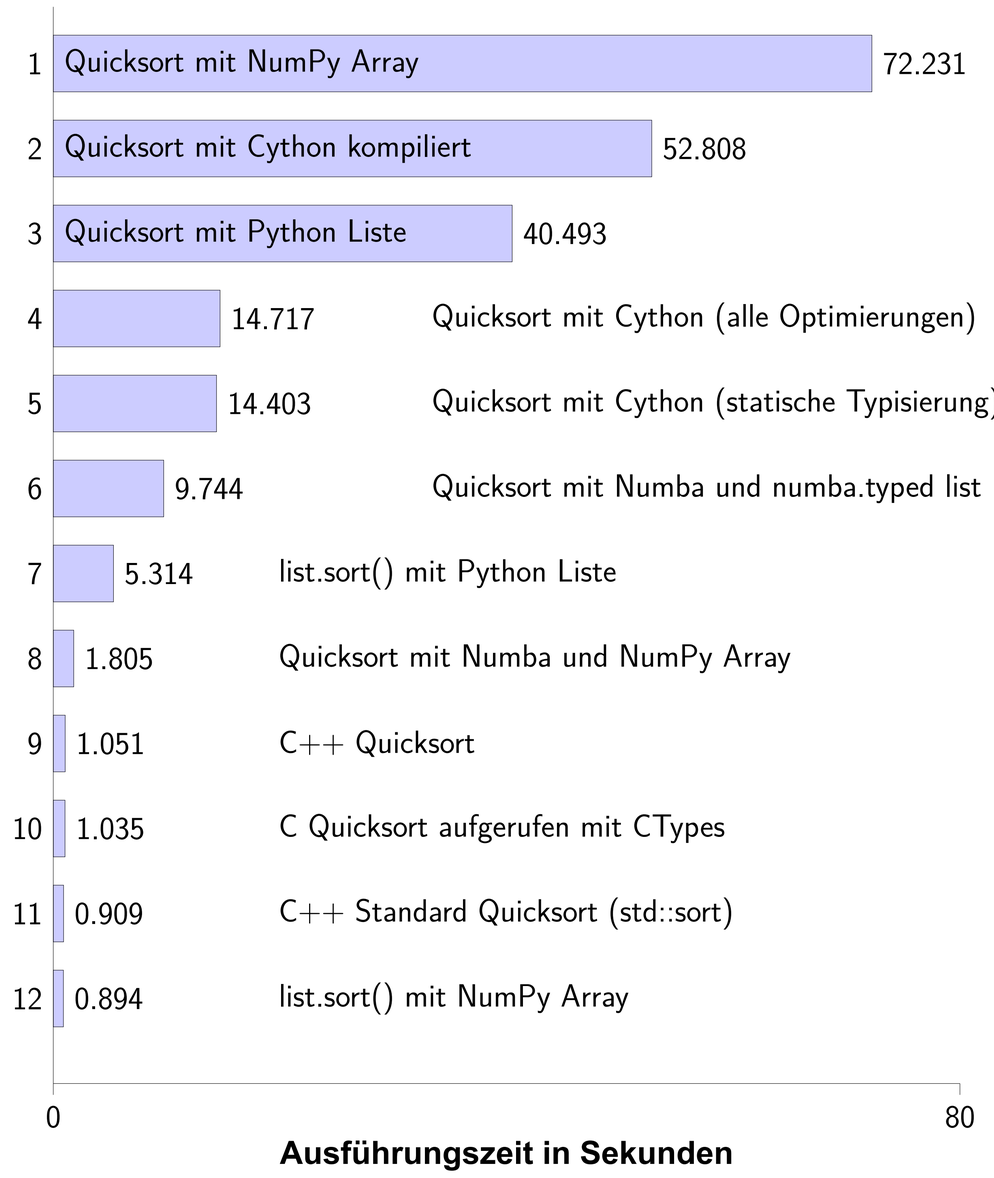
Supersonic Algorithms

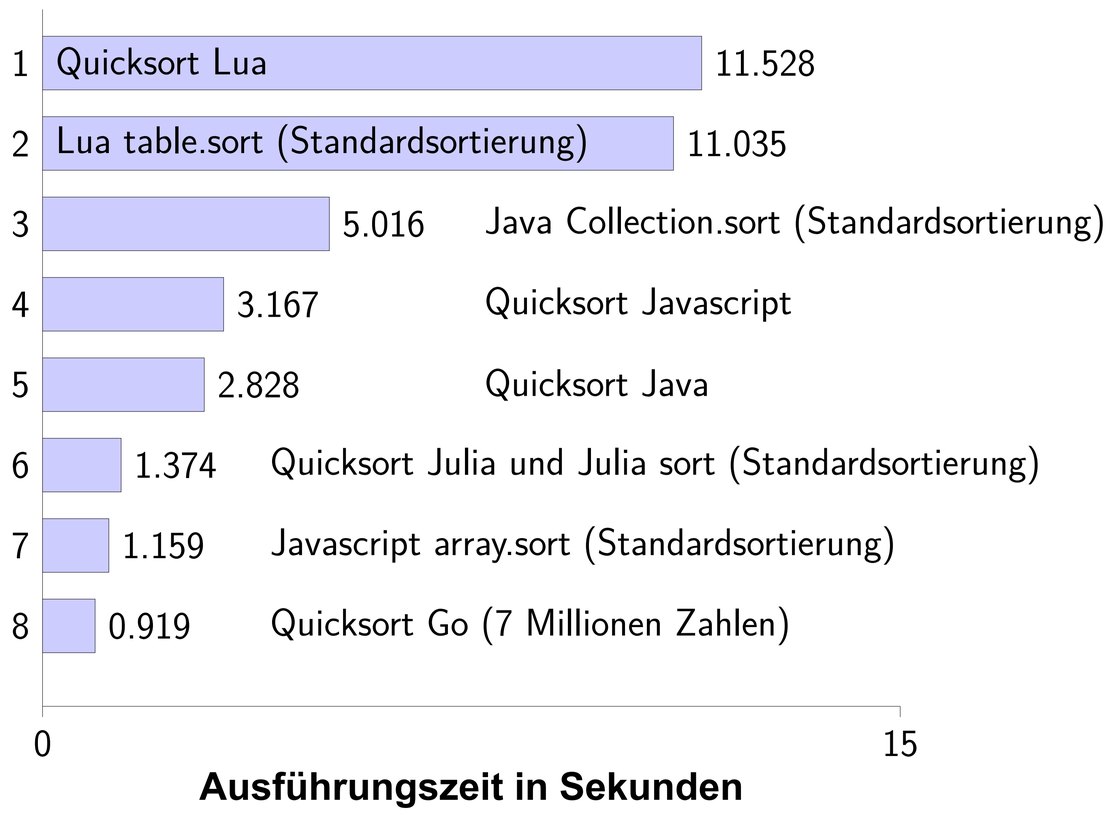
Anton Rodenwald (18), Schillerschule Hannover

Quicksort Implementationen in Python mit verschiedenen Optimierungen und Vergleich mit C++

- Ausführungzungszeit meiner verschiedenen Implementationen des Sortieralgorithmus Quicksort in Python und C++
- Programme sortierten 10 Millionen vorher generierte, zufällige Ganzzahlen auf- bzw. absteigend
- Erste Version aus dem Informatikunterricht (3) ca. 41 Sekunden
- Optimierungsversuche überraschendeweise teils langsamer (1, 2)
- Optimierungen mit Cython (4, 5) bringen deutlichen Performance Boost
- Versionen mit Numba sogar nochmals schneller (6, 8), JIT-Kompilierung sehr effektiv
- Eine Version (8) sogar schneller als list.sort(), der standardmäßigen Python Sortierfunktion
- Versionen mit C++, CTypes und NumPy Arrays nochmals eine Stufe darüber (9 - 12)
- Version mit CTypes benötigt allerdings noch zusätzlich Konvertierungszeit von ca. 2 Sekunden
- Python ähnlich schnell wie C++, was ich so nicht erwartet hatte



Quicksort Implementationen in weiteren Sprachen im Vergleich



- Auch Implementierte ich Quicksort in anderen Programmiersprachen, um deren Geschwindigkeit zu erforschen
- Nicht besonders optimietiert, da ich kaum Erfahrung in diesen Sprachen hatte
- Lua relativ langsam (1, 2)
- Bereitgestellte Sortierfunktion von Java (3) komischerweise langsamer als eigene Quicksort in Java (5)
- Javascript Quicksort im Mittelfeld (4)
- Standardsortierung und eigene Implementation in Julia ähnlich schnell (6)
- Javascript array.sort am schnellsten (7)
- Go Quicksort Implementation zwar über Javascript, sortierte aber nur 7 Millionen Zahlen

C++ Radixsort Implementationen

Um eine niedrig möglichste Zeit zu erreichen schaute ich mir noch die Radixsort als anderen Sortieralgorithmus an, die allerdings nicht direkt mit der Quicksort vergleichbar ist. Implementationen dieser brauchten zuerst 2.269 Sekunden (1), doch ich konnte die Performance noch steigern (2, 3) und die Quicksort überholen. Dies zeigte mir, dass korrekte Implementation und Wahl des Algorithmuvon eine wichtige Rolle spielen. Auf die resultierende Frage, wieso die Radixsort nicht häufiger genutzt wird, fand ich bisher noch keine Antwort.