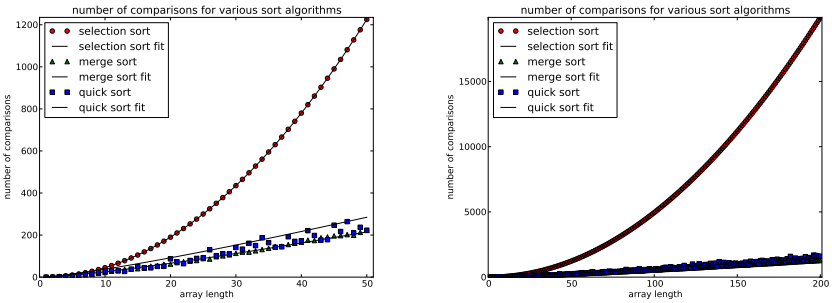
1 Sortierverfahren

Vergleich der Sortierverfahren selectionSort, mergeSort und quickSort

1.1 Implementation

Die Implementation ist in sort.py zu finden. Die Anzahl der benötigten Vergleiche sind bis zu einer Arraylänge von 50 bzw. 200 Einträgen dargestellt. Wie erwartet ist die Zahl der Vergleiche bei quickSort kleiner als bei selectionSort und etwas größer als bei mergeSort (vgl. Abb. 1). Die Wahl der Konstanten ist in Tabelle 1 zu sehen.



(a) n = 50 (b) n = 200

Abbildung 1: Anzahl der Vergleiche für 50 (a) bzw. 200 (b) Einträge.

Algorithmus	Anzahl der Vergleiche
selectionSort	$\frac{N^2}{2} - \frac{N}{2}$
mergeSort	$1.1\,N\,\log N$
quickSort	$1.2\,N\,logN + N$

Tabelle 1: Zahl der Vergleiche in Abhängigkeit von der Arraylänge für verschiedene Sortieralgorithmen

1.2 Zeitmessung in Abhängigkeit von der Arraygröße

Die Zeiten für die Sortierverfahren sind bis zu einer Arraygröße von 100 Einträgen in Abb. 2 aufgetragen. Der funktionale Zusammenhang aus 1.1 ist hier nicht mehr gültig, da dort nur die Vergleiche gezählt wurden. Tatsächlich muss aber auch noch die Zeit für swaps und das Anlegen neuer Arrays (bei mergeSort) beachtet werden.
Entgegen der Erwartung ist quickSort deutlich langsamer als die anderen Sortierverfahren. MergeSort ist ab ca. 50 Arrayeinträgen schneller als selectionSort.

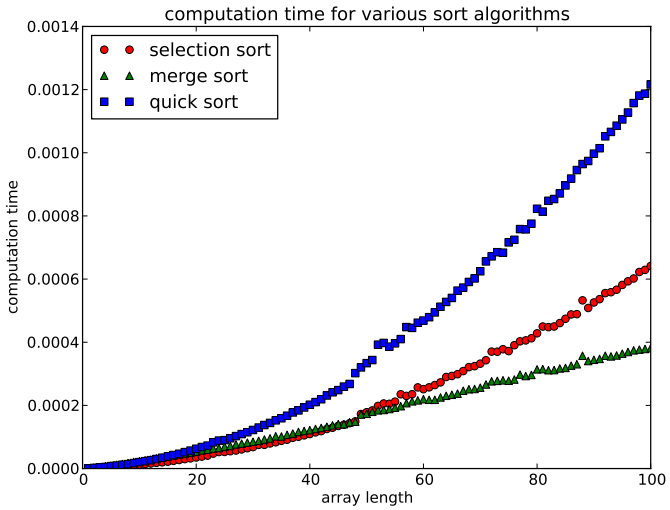


Abbildung 2: Benötigte Zeit der Sortieralgorithmen in Abhänigkeit von der Arraygröße

1.3 Überprüfungsalgorithmus

Der Algorithmus wurde in sort.py als checkSorting implementiert und hat die Richtigkeit der Sortierung bestätigt. Die Überprüfung von Länge, Elementen und Reihenfolge der Elemente ist im Code durch Kommentare gekennzeichnet.