Aufgabenblatt 9
Quicksort

Cao, Thi Huyen(Lilli)-Kochinky,Dennis

November 22, 2016

Contents

1 Quicksort

Quicksort ist ein rekursiver Sortieralgorithmus der nach dem devide and conquer Prinzip arbeitet. Sein Vorteil ist, dass die innere Schleife sehr kurz ist, wodurch die Ausführungsgeschwindigkeit erhöht wird. Ausserdem arbeitet Quicksort in-place d.h. es wird kein zusätzlicher Speicherplatz benötigt (ausser dem Platz für den Stack für die Rekursivenaufrufe). Der Algorithmus beginnt mit der Wahl des sogenannten Pivotelements, welches auf verschiedene Arten bestimmt werden kann. Beispielsweise wird einfach das erste Element aus der Liste genommen, nun wird die Liste durch das Pivotelement in zwei Teillisten getrennt. Alle Elemente die kleiner als das Pivot sind werden auf die linke Seite verschoben, alle grösseren Elemente auf die rechte Seite. Üblicherweise wird zuerst vom Anfang der Liste beginnend ein Element, welches grösser als das Pivot ist und vom Ende der Liste beginnend ein Element, welches kleiner als das Pivot ist, gesucht und diese beiden dann vertauscht. Anschliessend wird dieser Vorgang für beide Teillisten wiederholt, bis nur noch Teillisten der Länge null oder eins übrig sind(in diesem Fall ist die Teilliste fertig sortiert).

2 Verschiedene Pivotsuchverfahren

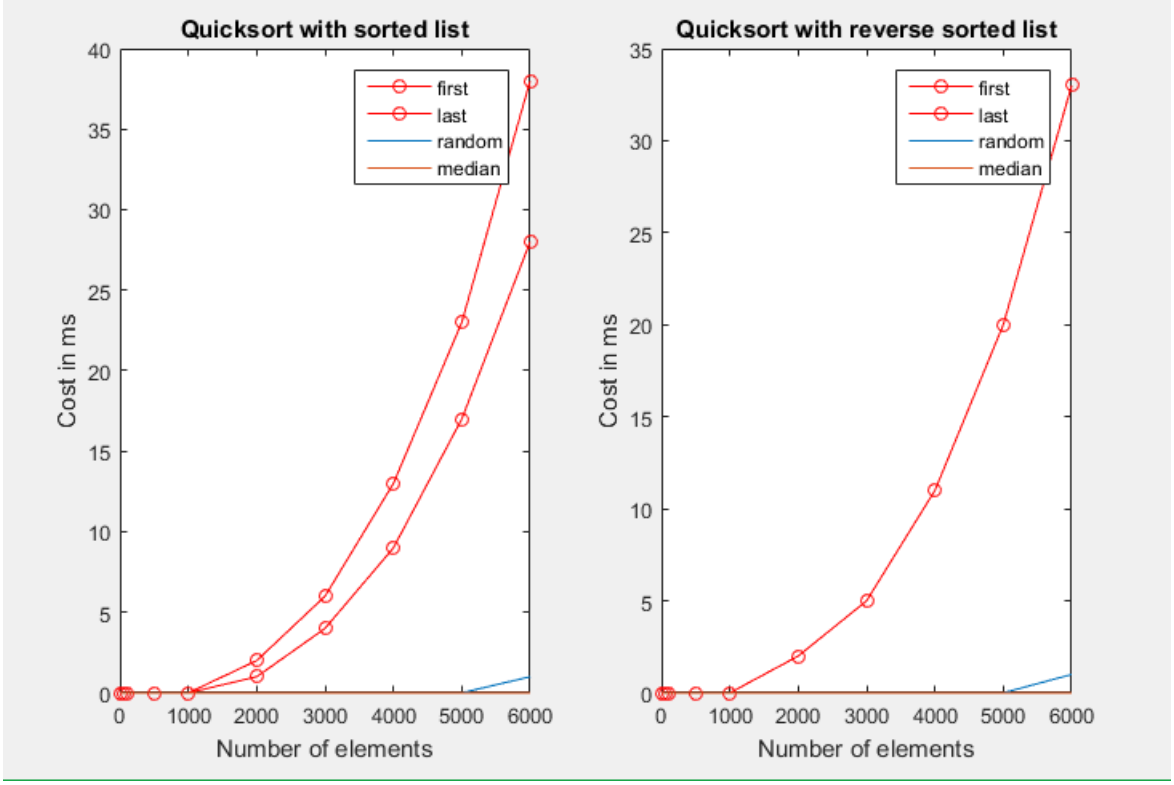
1.Erstes Element: Es wird das erste Element aus der Liste als Pivot gewählt.
2.Letztes Element: Es wird das letzte Element aus der Liste als Pivot gewählt.
3.Zufälliges Element: Es wird das ein zufälliges Element aus der Liste als Pivot gewählt.
4.Median of Three: Es wird das mittlere Element aus dem ersten, letzten und Element in der Mitte der Liste als Pivot gewählt.Anschliessen wird Pivot Element in der Mitte der Liste verschoben

3 Aufwandsanalyse

Die Laufzeit von Quicksort hängt hauptsächlich davon ab, wie die Liste im Laufe des Sortierens geteilt wird(und zwar von der Wahl des Pivotelements), allgemein ist
T(N) = T(Pivotauswahl) + T(Listeteilung) + T(2 rekursive Aufrufe)
Indem: T(Pivotauswahl) = constant = O(1); T(Listeteilung) = lineare Abhängigkeit von N = O(N)
T(N) = T(i) + T(N-i-1) + cN

3.1 Worst Case

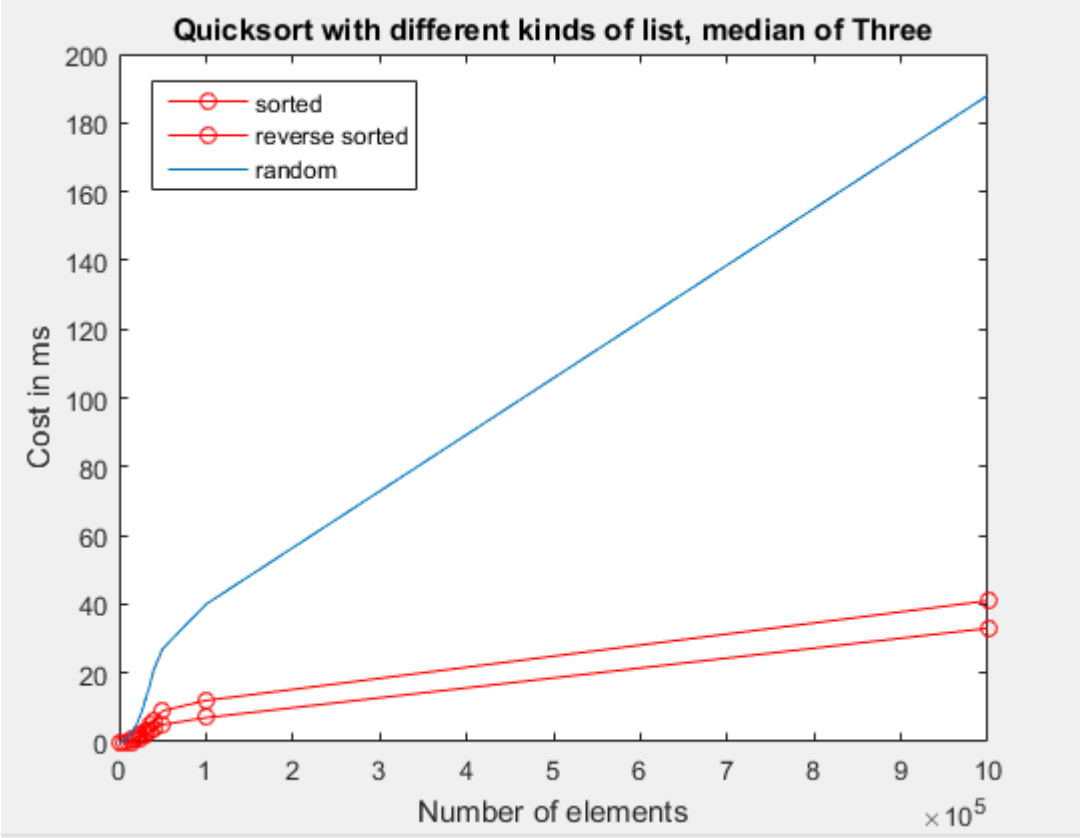
$\Longrightarrow$ Die Liste wird in ganzer Laufzeit ungleich geteilt
$\Longrightarrow$ Pivot ist immer das kleinste Element oder grösste Element
T(N) = T(N-1) +cN
T(N-1) = T(N-2) +c(N-1)
T(N-2) = T(N-3) +c(N-2)
...
T(2) = T(1) +c(2)
T(N) = T(1) +c $\sum_{i=0}^{N} i$
T(N) = T(1) + c$\frac{N(N+1)}{2}$
T(N) = O(N^2)
Zum Beispiel: Pivot ist am Anfang oder am Ende der Liste und die Liste wird schon sortiert(in richtiger Reihenfolge oder umgedrehter Reihenfolge)



3.2 Best Case

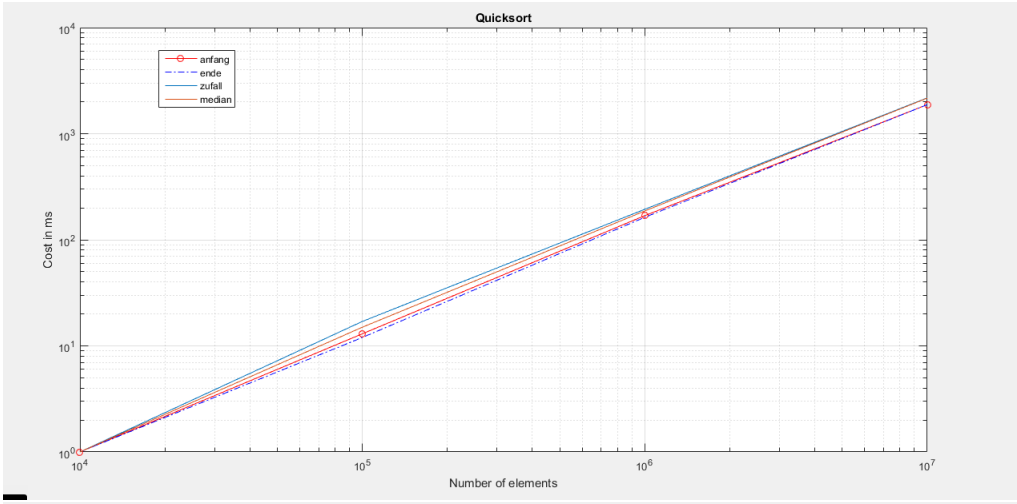
$\Longrightarrow$ Die Liste wird in der ganzen Laufzeit gleich geteilt
$\Longrightarrow$ Pivot ist immer das Element in der Mitte (median of the array)
T(N) = T(N/2) + T(N/2) + cN = 2T(N/2) + cN

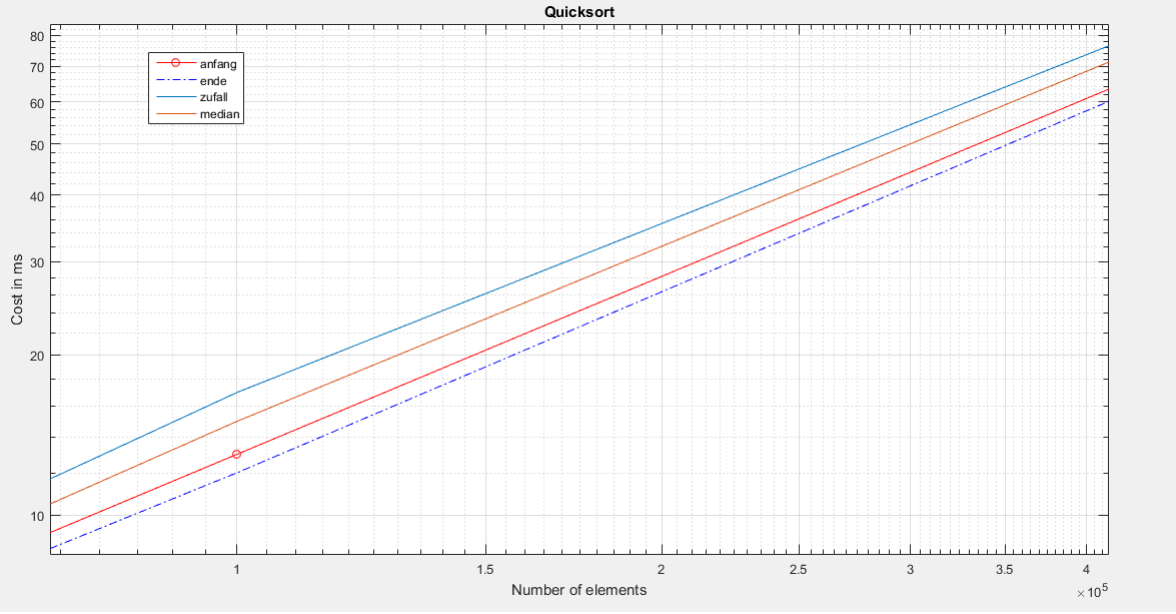
T(N/2)= 2T(N/4) + cN/2
T(N/4)=2T(N/8) + cN/4)
T(N/8)=2T(N/16) + cN/8)
...
T(N)= 2(2 (2 (2(2....)+cN/8)+cN/4)+cN/2)+cN
T(N) = $2^k T(\frac{N}{2^k}) + kcN$
Indem $\frac{N}{2^k}$ =1
$N = 2^k$
k=$log_2 N$
T(N)=$2^{log_2 N} T(1) + log_2 N * cN$
T(N) = NT(1) + cN*$log_2 N$
T(N) = N$log_2 N$
T(N) = NlogN
Zum Beispiel: Die Liste ist sortiert(in richtiger Reihenfolge oder umgedrehter Reihenfolge) und Pivot ist das Element durch Methode median of Three



3.3 Average Case

Zufällige Liste mit verschiedenen Grössen wird getestet. Das Ergebnis wird in den folgenden Graphiken dargestellt.





N	ANFANG	ENDE	ZUFALL	MEDIAN	INSERTION
N= 10::::	0:	0:	0:	0:	0
N= 100::::	0:	0:	0:	0:	0
N= 1000::::	0:	0:	0:	0:	0
N= 10000::::	16:	0:	0:	0:	0
N= 100000::::	16:	0:	16:	15:	0
N= 1000000::::	141:	156:	188:	140:	0
N= 10000000::::	1688:	1750:	1971:	1719:	0

$T(N) = (2\text{-}3*10^{-5})N\log N$
$T(N) = O(N\log N)$

3.4 Vermeiden Worst Case

Um Worst Case zu vermeiden wird in praktischen Implementierungen ein zufälliges Element als Pivot jedes Mal neu gewählt. Eine andere Technik ist median of Array zu suchen und das als Pivot zu benutzen.