**M411 Projektarbeit**Verkettete Liste sortieren  
  
  
Project by: Adrian Cerdeira und Mario Forrer

Klasse: 17dl

Doku created by: Adrian Cerdeira

Klassenlehrperson: Kaufmann Peter  
Abgabe: 11. Januar 2019

Inhalt

[**1.** **Analyse der Sortierung** 3](#_Toc534983317)

[1.1 Wie wurde gemessen? 3](#_Toc534983318)

[1.2 Bubblesort 3](#_Toc534983319)

[1.2.1 Messungen 3](#_Toc534983320)

[1.2.2 Graphen 3](#_Toc534983321)

[1.2.3 Kurzbeschrieb der Analyse 4](#_Toc534983322)

# **Analyse der Sortierung**

## Wie wurde gemessen?

Um die Dauer der Messung, der einzelnen Funktion zu berechnen, wurde die Bibliothek time.h verwendet. Dabei sieht der Code zur Messung folgendermassen aus:

int time1, time2, thou, sec;

time1 = clock(); // Clock starten

// Testmethode hier aufrufen

createList();

time2 = clock (); // Clock für Endzeit definiren

// Berechnung der Dauer

thou = (time2-time1) \* 1000 / CLOCKS\_PER\_SEC;

sec = thou / 1000;

thou -= sec \* 1000;

// Ausgabe der Dauer

printf ("%ld.%03ld sec.\n\n", sec, thou);

## Bubblesort

### Messungen

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Anzahl Elemente** | **Dauer zur Erstellung** | **Funktioniert** | **Dauer zur Sortierung** | **Funktioniert** | **Bemerkung** |
| 5 | 1.851s | Ja | 0.010s | Ja |  |
| 50 | 1.340s | Ja | 0.100s | Ja |  |
| 500 | 1.730s | Ja | 0.200s | Ja |  |
| 50000 | 3.960s | Ja | 18.345s | Ja | CPU Auslastung 24% |
| 1000000 | 6.390s | Ja | über 30min | Nein | CPU Auslastung 30% |

### Graphen

### Kurzbeschrieb der Analyse

In meiner hochprofessionellen Analyse, habe ich mit der time.h-Bibliothek die Anfangszeit (Start der Funktion) und Endzeit (Ende der Funktion) bestimmt. Diese wurden schlussendlich subtrahiert und das Resultat per printf ausgegeben.

Dabei ist in der Analyse ausgefallen, dass die Erstellung der Elemente deutlich schneller verlief als die Sortierung. Was klar war, weil Bubblesort jedes einzelne Element überprüft und dies bei zu grosser Anzahl, lange dauert oder das Programm abbricht.

Auch die CPU Auslastung auf dem Testgerät war sehr interessant, es erreicht bei 50000 Elemente eine Auslastung von 24% bei 1 Millionen Elementen erreicht es sogar eine Auslastung von 30%.

Zusammengefasst ist Bubblesort ein einfacher Sortieralgorithmus. Codetechnisch ist er einfach einbaubar und kann für char- oder int-Typen Sortierungen prima eingesetzt werden. Leider ist Bubblesort nicht gut erweitbar und kommt bei zu grosser Anzahl von Elementen an seiner Performancegrenzen.