

1. Einfache Sortieralgorithmen

Sortieralgorithmen haben die Aufgabe unsortiert Datensätze und dergleichen in einen sortierten Endzustand zu verwandeln. Sortiert wird nach dem sogenannten „Schlüssel“ Dieser kann sein eine ganzzahlige Zahl, ein Buchstaben oder ein String etc. Eine unbedingt notwendige Bedingung an den Schlüssel ist, daß er für jeden Datensatz unterschiedlich ist. Ein Unterschiedlicher Datensatz darf auf keinen Fall den selben Schlüssel besitzen.

**Einfache Sortierverfahren eignen sich nur für kleine Datensätze (max. 500 DS).
Der Aufwand ist höchstens N^2 .**

Als Tafelbild eignet sich ein Beispiel eines Sortiervorganges, einer ungeordneten Liste, wie sie in den Unterlagen beschrieben werden. Auch ein Vergleich mit einem anderen ist denkbar !

1.1. Sortieren durch Minimumsuche (Maximumsuche)

Die Idee: Man sucht im ersten Schleifendurchgang, den niedrigsten Schlüssel. Diesen Datensatz vertauscht man nun mit dem ersten. Beim zweiten durchlauf sucht man nun wieder den DS mit dem niedrigsten Schlüssel jedoch ab den 2.DS. Findet man diesen so wird er nun mit dem zweiten DS vertauscht.....Die Anzahl der Durchgänge ist die Anzahl der DS-1, da beim Letzten gar nicht nach dem kleinsten gesucht werden muß.

Allgemein gesagt: nach i Schleifendurchgängen hat man ein Sortiertes Teilfeld der Länge i (von 1 bis i). Dieses Teilfeld ist auch am richtigen Platz. Deshalb sind nur $\text{anzahl}-1$ Schritte notwendig.

Bei der Maximumsuche zählt man die DS von hinten nach vorne durch und sucht jeweils den DS mit dem größten Schlüssel.

Aufwand: Vergleich zweier Schlüssel und Organisatoinarbeit

(z.B.: Indexerhöhung)
Vertauschung

Aufwand insgesamt: $N^2/2$ Aktionen+N Vertauschungen

höchstmöglicher Aufwand: $O(N)=N^2$

Der Aufwand hängt nur von der Anzahl der Datensätze ab nicht von deren Anordnung.

Beispiel des Sortiervorganges:

6	7	2	1	3	9	4	5	8
1	7	2	6	3	9	4	5	8
1	2	7	6	3	9	4	5	8
1	2	3	6	7	9	4	5	8
1	2	3	4	7	9	6	5	8
1	2	3	4	5	9	6	7	8

 Bereits sortierte Teilfolgen

1	2	3	4	5	6	9	7	8
1	2	3	4	5	6	7	9	8
1	2	3	4	5	6	7	8	9

1.2. Sortieren durch Einordnen

Mit Hilfe einer Schleife (von 2 bis N) zählt man jeden Datensatz durch. Der Schlüssel des momentanen angesprochen Datensatz j wird mit seinen „Vorgängerschlüsseln“ verglichen und jeder größere wird um einen Platz vorgeschoben. Diesen Durchgang zählt man $j-1$ runter bis 1. Der Datensatz vom Platz $j-1$ füllt dann die entstandene Lücke bei j .

Auch hier bekommt man nach dem j -ten Durchgang ein sortiertes Teilfeld mit j Datensätzen. Jedoch ist diese Feld nicht an seiner richtigen Position.

Aufwand: 1 Schlüsselvergleich
 Organisation
 Umspeicherung des vollständigen Datensatzbestandes

Aufwand insgesamt: mindestens N
 im Mittel $N^2/4$
 höchstens $n^2/2$ Aktionen

Der Aufwand hängt sehr stark von der Anordnung (Vorsortierung) ab. Je gründlicher der Datenbestand bereits sortiert ist, desto weniger wird der Aufwand. im bestem Falle mit dem minimalsten Aufwand.

Beispiel des Sortiervorganges:

6	7	2	1	3	9	4	5	8
6	7	2	1	3	9	4	5	8
2	6	7	1	3	9	4	5	8
1	2	6	7	3	9	4	5	8
1	2	3	6	7	9	4	5	8
1	2	3	6	7	9	4	5	8
1	2	3	4	6	7	9	5	8
1	2	3	4	5	6	7	9	8
1	2	3	4	5	6	7	8	9

 Bereits sortierte Teilfolgen

1.3. Sortieren durch Mischen




Beim Sortieren durch Mischen werden zwei bereits sortierte Datensatzbestände zu einem großen Datenbestand zusammengemischt. Der neuentstandene Datenbestand ist ebenfalls sortiert. Man hat je Datenbestand einen Index (x1 und x2). Nun vergleicht man den Schlüssel der beiden Datensätze. Der Datensatz mit dem niederen Schlüssel wird auf das neue Feld geschrieben. Zusätzlich wird der Index derjenigen Tabelle erhöht von der der Datensatz entnommen wurde. nun vergleicht man wieder beide,.....Hat man das Ende eines Datenbestandes erreicht, so entnimmt man nur den anderen Datenbestand die einzelnen Datensätze ohne sie zu vergleichen.

Aufwand: N Aktionen
Aufwand insgesamt: N*ldN Aktionen

Ein wesentlicher Nachteil ist der große Speicherplatzbedarf. Jedoch trotzdem eine bevorzugte Methode zum sortieren von verketteten Listen.

Beispiel des Sortiervorganges:

0	3	6	8	9					
1	2	4	5	7					
0	3	6	8	9					
1	2	4	5	7					
0									
0	3	6	8	9					
1	2	4	5	7					
0	1								
0	3	6	8	9					
1	2	4	5	7					
0	1	2							
0	3	6	8	9					
1	2	4	5	7					
0	1	2	3						
0	3	6	8	9					
1	2	4	5	7					
0	1	2	3	4					
0	3	6	8	9					
1	2	4	5	7					

-  Bereits Sortierte Folgen
-  zu vergleichende
-  Bereits eingeordnete

0	1	2	3	4	5				
0	3	6	8	9					
1	2	4	5	7					
0	1	2	3	4	5	6			
0	3	6	8	9					
1	2	4	5	7					
0	1	2	3	4	5	6	7		
0	3	6	8	9					
1	2	4	5	7					
0	1	2	3	4	5	6	7	8	
0	3	6	8	9					
1	2	4	5	7					
0	1	2	3	4	5	6	7	8	9