Aufwandsbetrachtung „Sortieren durch direkte Auswahl"

Wir formulieren einen funktionalen Zusammenhang zwischen dem zeitlichen Aufwand, um eine Liste von n Datenelementen der Größe nach zu sortieren, und der Anzahl n der Datenelemente.

Wertzuweisungen, Abfragen, Rechenoperationen sind elementare Anweisungen, die eine bestimmte Rechenzeit erfordern; obwohl diese Rechenzeiten mit fortschreitender Leistungsfähigkeit der Hardware immer kürzer werden, gerät man rasch an Grenzen der praktischen Durchführbarkeit eines Algorithmus, wenn die Anzahl der abzuarbeitenden Anweisungen zu stark, z. B. exponentiell, wächst.

Wesentlicher Baustein des Algorithmus „Sortieren durch direkte Auswahl" ist die Funktion **`min(x,j)`**, die das kleinste Element des arrays **`a[j], . . , a[n-1]`** ermittelt und dieses der Komponente **`a[j]`** zuweist.

```
def min(x,j):
    for i in range(j+1,len(x)):
        if x[i] < x[j]:
            temp = x[j]
            x[j] = x[i]
            x[i] = temp
```

Der Schleifenrumpf der in der der Funktion **`min(x,j)`** implementierten for-Schleife besteht aus 3 Wertzuweisungen und 1 Abfrage, die wir gedanklich als ganzes zum Anweisungsblock **A** zusammenfassen:

```
def min(x,j):
    for i in range(j+1,len(x)):
                        A
j = 0
while j <= n-2:
    min(a,j)
    j = j+1
```

Wir überlegen, wie oft der Block **A** abgearbeitet wird, indem wir zunächst die Anzahl **`z(j)`** dieser Abarbeitungen in Abhängigkeit vom Schleifenindex **`j`** notieren:

Index j	**Aufruf**	**Index i**	**z(j)**
j = 0	min(x,0)	1 ≤ i ≤ n-1	n-1
j = 1	min(x,1)	2 ≤ i ≤ n-1	n-2
j = 2	min(x,2)	3 ≤ i ≤ n-1	n-3
j = 3	min(x,3)	4 ≤ i ≤ n-1	n-4
....	....	....	....
j = n-2	min(x,n-2)	n-1 ≤ i ≤ n-1	1

Gesamtzahl z der Abarbeitungen von Anweisungsblock **A**:

$$z = z(0) + z(1) + z(2) + \ldots + z(n-2) = (n-1) + (n-2) + (n-3) + \ldots + 1$$

$$= \sum_{k=1}^{n-1} k$$

$$= \tfrac{1}{2} \cdot (n-1) \cdot n \qquad \textit{(vgl. Anmerkung)}$$

$$= \tfrac{1}{2} \cdot (n^2 - n)$$

$$\approx \tfrac{1}{2} \cdot n^2 \qquad \text{für große n}$$

Ergebnis:
Die Anzahl der abzuarbeitenden elementaren Anweisungen und damit der Zeitaufwand wachsen quadratisch mit der Anzahl n der zu sortierenden Datensätze.

Anmerkung: $\sum_{k=1}^{n} k = \frac{1}{2} \cdot n \cdot (n+1)$

19.01.2021