Aufwandsbetrachtung „Sortieren durch direkte Auswahl" (SelectionSort)

Wir formulieren einen Zusammenhang zwischen dem zeitlichen Aufwand, um eine Liste von n Datenelementen (z. B. Zufallszahlen, Namen) der Größe nach zu sortieren, und der Anzahl n der Datenelemente.

Wertzuweisungen, Abfragen und 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 wächst.

Wesentlicher Baustein des Algorithmus „Sortieren durch direkte Auswahl" ist der Schleifenrumpf der inneren for-Schleife (hier: rot gekennzeichnet), der das kleinste Element innerhalb des arrays **`a[j], . . , a[n-1]`** ermittelt und dieses der Komponente **`a[j]`** zuweist:

```
for j in range(0,n-1):
    min = a[j]
    for i in range(j+1,n):
       if min > a[i]:
           min  = a[i]
           a[i] = a[j]
           a[j] = min
```

Dieser rot markierte Schleifenrumpf besteht aus 3 Wertzuweisungen und 1 Abfrage, die wir gedanklich als ganzes zum Anweisungsblock **A** zusammenfassen:

```
for j in range(0,n-1):
    min = a[j]
    for i in range(j+1,n):
            A
```

Für **j=0** nimmt der Schleifenindex i der inneren Schleife alle Werte von 1 bis n–1 an, folglich wird der Anweisungsblock **A** (n–1)-mal ausgeführt.
Für **j=1** nimmt der Schleifenindex i der inneren Schleife alle Werte von 2 bis n–1 an, folglich wird Block **A** (n–2)-mal ausgeführt.
Für **j=2** nimmt der Schleifenindex i der inneren Schleife alle Werte von 3 bis n–1 an, folglich wird Block **A** (n–3)-mal ausgeführt.

In der folgenden Tabelle notieren wir für jeden Wert von **j** jeweils den Bereich, den **i** durchläuft, und die daraus sich ergebende Anzahl **z(j)**, die angibt, wie oft der Anweisungsblock **A** durchlaufen wird:

Index j	**Index i**	**z(j)**
j = 0	1 ≤ i ≤ n-1	n-1
j = 1	2 ≤ i ≤ n-1	n-2
j = 2	3 ≤ i ≤ n-1	n-3
j = 3	4 ≤ i ≤ n-1	n-4
j = 4	5 ≤ i ≤ n-1	n-5
....	....	....
....	....	....
j = n-3	n-2 ≤ i ≤ n-1	2
j = n-2	n-1 ≤ i ≤ n-1	1

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

z = z(0) + z(1) + z(2) + z(3) + ... + z(n-3) + z(n-2)
z = (n-1) + (n-2) + ... + 2 + 1
z = ½ · (n – 1) · n
z = ½ · n^2 – ½ · n ≈ ½ · n^2 für große Werte von n

Ergebnis: Bei SelectionSort wächst der Rechenaufwand zum Sortieren einer aus n Elementen bestehenden Liste quadratisch mit n.