Sortieren durch direkte Auswahl

Wir beschränken uns zunächst darauf, eine Liste von ganzen Zahlen (hier: Zufallszahlen) der Größe nach, und zwar aufsteigend, zu sortieren. Den Algorithmus später auf andere Datenstrukturen (z. B. Namen, Verbundtypen) zu übertragen, ist vergleichsweise einfach und bereitet keine Schwierigkeiten.

Die Python-Anweisungen `range`, `list` und `len`:

a) `range`-Anweisung

Die **`range`**-Anweisung definiert einen Bereich ganzer Zahlen.

`range(10)` definiert den Bereich 0, 1, . . . , 9
`range(4,21)` definiert den Bereich 4, 5, . . . , 20
`range(4,21,3)` definiert den Bereich 4, 7, 10, . . . , 16, 19
`range(-4,3)` definiert den Bereich -4, -3, -2, -1, 0, 1, 2

Allgemein gilt:

`range(start, stop)`
definiert den Bereich **`start`**, ... , **`stop-1`** ganzer Zahlen,

`range(start, stop, step)`
definiert den Bereich **`start`**, ... mit der Schrittweite **`step`**, wobei die Zahl **`stop`** nicht mehr enthalten ist.

b) Erstellen einer Liste ganzer Zahlen

`a = list(range(4,13))` erzeugt die Liste
`[4, 5, 6, 7, 8, 9, 10, 11, 12]`;
die (in diesem Fall 9) Elemente dieser Liste nennen wir auch Komponenten, auf die man mit **a[0], a[1], . . . , a[8]** zugreifen kann (mit Erzeugung der Liste dieses Beispiels sind die Komponenten **a[0], a[1], . . . , a[8]** in dieser Reihenfolge mit den Werten **4, 5, ... , 12** belegt). Allerdings läßt sich jeder Komponente **a[i]** eine beliebige andere ganze Zahl zuweisen.

*Bemerkung: Unter einem **Feld** oder **array** verstehen wir eine Folge von Variablen gleichen Typs; mit der Anweisung* **`a = list(range(4,13))`** *haben wir also ein array **a** erzeugt mit den Komponenten a[0], a[1], . . . , a[8].*

c) **`len(a)`** bestimmt die Anzahl der Komponenten der Liste **a**, in dem Beispiel aus b) gilt somit: **`len(a) = 9`** .

1. Erstellen einer Liste mit n Komponenten, denen Zufallszahlen zugewiesen werden (n ist eine natürliche Zahl)

Vorbemerkung:
Die Python-Anweisung **`randint`** ist eine vordefinierte Funktion des random-Moduls in Python; Syntax: **`randint(r,s)`** mit ganzen Zahlen r und s, $r \leq s$, erzeugt eine Zufallszahl aus dem Intervall [r, s].
Beispiele:
`randint(1,1000)` erzeugt eine Zufallszahl aus dem Bereich 1, ... , 1000
`randint(-7,12)` erzeugt eine Zufallszahl aus dem Bereich -7, ... , 12

Ein Algorithmus, der nach Eingabe einer natürlichen Zahl **n** eine Liste aus **n** Zufallszahlen generiert, formuliert als Python-Quelltext in der Schriftart **`Courier New`**, so daß man den Quelltext unmittelbar durch *copy* und *paste* in einen Editor für Python-Programme übernehmen kann:

```
# array mit zufallszahlen

from random import randint

n = int(input('Laenge des arrays = '))
print()

# Erzeugen des arrays mit dem Namen a
# und den n Komponenten a[0], . . . , a[n-1]
a = list(range(1,n+1))

# Zuweisung von Zufallszahlen an die Komponenten des arrays a
for i in range(0,n):
    a[i] = randint(1,1000)

# Ausgabe des arrays
for i in range(0,n):
    print(a[i])
```

2. Bestimmung des kleinsten Elements der Liste aus n Komponenten

Der Inhalt des Speicherplatzes **a[0]** wird sukzessive mit den Inhalten von **a[1], . . . , a[n-1]** verglichen; falls gilt **a[i] < a[0]**, $1 \leq i \leq n-1$, werden die Inhalte der Speicherplätze **a[i]** und **a[0]** ausgetauscht; hierzu wird, bevor **a[0]** den Wert von **a[i]** erhält, der ursprüngliche Wert von **a[0]** mittels der Hilfsvariablen **temp** gesichert und nach der Zuweisung **a[0] = a[i]** mit **a[i] = temp** an **a[i]** übergeben.

Die Durchführung der Vergleiche und der ggf. erforderliche Austausch der Inhalte von **a[0]** und **a[i]** werden hier an die Funktion **min(x)** delegiert:

```
def min(x):
    for i in range(1,len(x)):
        if x[i] < x[0]:
             temp = x[0]
             x[0] = x[i]
             x[i] = temp
```

Mit dem Aufruf **min(a)** wird die Funktion **min** auf das aus den Komponenten **a[0], . . . , a[n-1]** bestehende array **a** angewendet.

```
from random import randint

n = int(input('Laenge des arrays = '))
print()

# Erzeugen des arrays mit dem Namen a
# und den n Komponenten a[0], . . . , a[n-1]
a = list(range(1,n+1))

# Zuweisung von Zufallszahlen an die Komponenten des arrays a
for i in range(0,n):
    a[i] = randint(1,1000)
```

```
# Ausgabe des arrays
for i in range(0,n):
    print(a[i])

# Bestimmen des kleinsten Elements:
# Wir definieren eine Funktion min(x), die auf
# das array a angewendet wird, das kleinste Element
# bestimmt und dieses der Komponente a[0] zuweist.

def min(x):
    for i in range(1,len(x)):
        if x[i] < x[0]:
             temp = x[0]
             x[0] = x[i]
             x[i] = temp

# Aufruf der auf das array a anzuwendenden Funktion min

min(a)

# Ausgabe der Liste mit dem kleinsten Element an der 1. Stelle
print()
for i in range(0,n):
    print(a[i])
```

Nachdem das kleinste Element der Liste **a[0], . . . , a[n-1]** dem Speicherplatz **a[0]** zugewiesen wurde, bestimmen wir das kleinste Element der „Restliste" **a[1], . . . , a[n-1]** und weisen es dem Speicherplatz **a[1]** zu.

Wenn wir dieses Verfahren sukzessive auf die weiteren „Restlisten" **a[j], . . . , a[n-1]** mit **2 ≤ j ≤ n-2** anwenden, erhalten wir ein array **a**, dessen Komponenten gemäß **a[0] ≤ a[1] ≤ . . . ≤ a[n-1]** aufsteigend sortiert sind.

Wir modifizieren die Funktion **min(x)**, indem wir einen weiteren Parameter **j** ergänzen:

```
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
```

Die mit dem Parameterwert **j** auf das array **a** angewendete Funktion **min(x,j)** ermittelt in der Liste **a[j], . . . , a[n-1]** das kleinste Element und weist es dem Speicherplatz **a[j]** zu.

3. Variante zu 2:

```
from random import randint

n = int(input('Laenge des arrays = '))
print()
```

```
# Erzeugen des arrays mit dem Namen a
# und den n Komponenten a[0], . . . , a[n-1]
a = list(range(1,n+1))

# Zuweisung von Zufallszahlen an die Komponenten des arrays a
for i in range(0,n):
    a[i] = randint(1,1000)

# Ausgabe des arrays
for i in range(0,n):
    print(a[i])

# Bestimmen des kleinsten Elements:
# Wir definieren eine Funktion min(x,j), die auf
# die Komponenten a[j], . . , a[n-1] des arrays a
# angewendet wird, das kleinste Element
# bestimmt 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

# Aufruf der auf das array a anzuwendenden Funktion min

min(a,0)

# Ausgabe der Liste mit dem kleinsten Element an der 1. Stelle
print()
for i in range(0,n):
    print(a[i])
```

4. Bestimmung der 2 kleinsten Elemente der Liste aus n Komponenten

```
from random import randint

n = int(input('Laenge des arrays = '))
print()

# Erzeugen des arrays mit dem Namen a
# und den n Komponenten a[0], . . . , a[n-1]
a = list(range(1,n+1))

# Zuweisung von Zufallszahlen an die Komponenten des arrays a
for i in range(0,n):
    a[i] = randint(1,1000)

# Ausgabe des arrays
for i in range(0,n):
    print(a[i])

# Wir definieren eine Funktion min(x,j), die auf
```

```
# die Komponenten a[j], . . , a[n-1] des arrays a
# angewendet wird, das kleinste Element
# bestimmt 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

# Aufrufe der auf das array a anzuwendenden Funktion min

min(a,0)
min(a,1)

# Ausgabe der Liste
print()
for i in range(0,n):
    print(a[i])
```

5. Bestimmung der 3 kleinsten Elemente der Liste aus n Komponenten

```
. . . . . .
. . . . . .

# Aufrufe der auf das array a anzuwendenden Funktion min

min(a,0)
min(a,1)
min(a,2)

. . . . . .
. . . . . .
```

6. Sortieren der aus den Komponenten a[0], ... , a[n-1] bestehenden Liste a

Wir sortieren das array **a** mit den Komponenten **a[0], . . . , a[n-1]**, indem wir die Funktion **min(x,j)** mit j = 0, 1, . . . , n-2 nacheinander auf das array a anwenden; die wiederholte Anwendung realisieren wir mit einer while-Schleife, deren Schleifenindex j mit dem Wert 0 initialisiert wird:

```
j = 0
while j <= n-2:
    min(a,j)
    j +=1
```

Der hier vorgestellte Algorithmus ist unter der Bezeichnung

„Sortieren durch direkte Auswahl"

bekannt.

Der folgende in Python codierte Algorithmus sortiert aufsteigend ein array **a** der Länge **n**, dessen Komponenten **a[0], . . . , a[n-1]** Zufallszahlen aus dem Bereich 1, . . . , 100000 zugewiesen wurden:

```
# sorting by direct selection

from random import randint

n = int(input('Laenge des arrays = '))
print()

# Erzeugen des arrays mit dem Namen a
# und den n Komponenten a[0], . . . , a[n-1]
a = list(range(1,n+1))

# Zuweisung von Zufallszahlen an die Komponenten des arrays a
for i in range(0,n):
    a[i] = randint(1,100000)

# Ausgabe des arrays
for i in range(0,n):
    print(a[i])

# Die auf die Komponenten a[j], . . , a[n-1] des arrays a
# angewendete Funktion min(x,j) bestimmt das kleinste Element
# und weist dieses der Komponente a[j] zu.

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

# Aufrufe der auf das array a anzuwendenden Funktion min

j = 0
while j <= n-2:
    min(a,j)
    j +=1

# Ausgabe der sortierten Liste

print()
print('Sortierte Liste:')

for i in range(0,n):
    print(a[i])
```

Selbach
update 26.01.2021

SelectionSort mit Ermittlung des Zeitbedarfs zur Laufzeit:

```
# sorting by direct selection
# Nach Eingabe einer natuerlichen Zahl n wird ein
# aus n Komponenten bestehendes array sortiert.

from random import randint
import time

n = int(input('Laenge des arrays: '))
print()

# Erzeugen des arrays mit dem Namen a
# und den n Komponenten a[0], . . . , a[n-1]
a = list(range(1,n+1))

# Zuweisung von Zufallszahlen an die Komponenten des arrays a
for i in range(0,n):
    a[i] = randint(1,1000000)

# Ausgabe des arrays
r = int(input('Wieviele Elemente sollen angezeigt werden? '))
print()
for i in range(0,r):
    print(a[i])

# Die auf die Komponenten a[j], . . , a[n-1] des arrays a
# angewendete Funktion min(x,j) bestimmt das kleinste Element
# und weist dieses der Komponente a[j] zu.

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

# Aufrufe der auf das array a anzuwendenden Funktion min
# mit Ermittlung des Zeitbedarfs zur Laufzeit

start = time.time()

j = 0
while j <= n-2:
    min(a,j)
    j +=1

end = time.time()


# Ausgabe der sortierten Liste

print()
print('Sortierte Liste:')
print()

for i in range(0,r):
    print(a[i])

print()
print('Zeitaufwand zum Sortieren von',n,'Elementen: {:7.3f} s'.format(end-start))
```

```
# sorting by direct selection
# Nach Eingabe einer natuerlichen Zahl n wird ein
# aus n Komponenten bestehendes array sortiert.

from random import randint
import time

n = int(input('Laenge des arrays: '))
print()

# Erzeugen des arrays mit dem Namen a
# und den n Komponenten a[0], . . . , a[n-1]
a = list(range(1,n+1))

# Zuweisung von Zufallszahlen an die Komponenten des arrays a
for i in range(0,n):
    a[i] = randint(1,1000000)

# Ausgabe des arrays
r = int(input('Wieviele Elemente sollen angezeigt werden? '))
print()
for i in range(0,r):
    print(a[i])

# Die auf die Komponenten a[j], . . , a[n-1] des arrays a
# angewendete Funktion min(x,j) bestimmt das kleinste Element
# und weist dieses der Komponente a[j] zu.

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

# Aufrufe der auf das array a anzuwendenden Funktion min
# mit Ermittlung des Zeitbedarfs zur Laufzeit

start = time.time()

j = 0
while j <= n-2:
    min(a,j)
    j +=1

end = time.time()


# Ausgabe der sortierten Liste

print()
print('Sortierte Liste:')
print()

for i in range(0,r):
    print(a[i])

print()
print('Zeitaufwand zum Sortieren von',n,'Elementen: {:7.3f}
s'.format(end-start))
```