Musterlösungen Übungsblatt 5

Thomas Graf
EF / WF Informatik 2018-2019
Algorithmen

22. Oktober 2018

1 Insertion Sort ausführen ($\star$)

Die folgende "Tabelle" zeigt die einzelnen Schritte von *Insertion Sort* auf der gegebenen Eingabe:

31	41	59	26	41	58
31	41	59	26	41	58
31	41	59	26	41	58
26	31	41	59	41	58
26	31	41	41	59	58
26	31	41	41	58	59

2 Grosse Unterschiede bei der Laufzeit ($\star\star$)

Zu diesem Zeitpunkt (Oktober 2018) beträgt die peak performance des schnellsten Supercomputers 187′659.3 TFlop/s. Dies entspricht $187'659.3 \cdot 10^12$ Gleitkomma-Operationen pro Sekunde, also $187'659.3 \cdot 10^12 \cdot 60 = n_1$ Operationen pro Minute. Der Supercomptuer kann den Algorithmus für eine maximale Eingabegrösse von $n = \sqrt{n_1} = 3355526000$ berechnen.

Für die Eingabe von $n = 20$ für Algorithmus B benötigt der Computer $60 \cdot (n!)/n_1 \approx 13$ Sekunden. Für die Eingabe $n = 32$ hingegen $60 \cdot (n!)/n_1 \approx 1.4 \cdot 10^18$ Sekunden. Dies entspricht ungefähr 3.2 mal dem Alter des Universums.

3 Untersuchung eines Algorithmus ($\star\star$)

Der gegebene Algorithmus ist bekannt als *Bubble Sort*. Es handelt sich dabei um einen Sortieralgorithmus. Es werden immer zwei benachbarte Elemente verglichen und getauscht, falls das linke Element grösser ist als das rechte Element. Eine Animation der Funktionsweise von Bubble Sort kann hier gefunden werden:
https://de.wikipedia.org/wiki/Bubblesort#/media/File:Bubble-sort-example-300px.gif.

Aufgrund des doppelten for-loops ist klar, dass der Algorithmus nicht linear, sondern quadratisch in n ist.

4 Perfekte Zahlen (schnell) ($\star\star$)

Wir zeigen zuerst die Aussage aus dem Tipp.
Falls n eine Primzahl ist, dann ist die Aussage trivialerweise erfüllt, da n nur 1 als echten Teiler hat. Deshalb würde bereits die Suche in der Menge $\{1\}$ alle echten Teiler von n liefern.
Nehmen wir nun an, dass n keine Primzahl ist. Dann existiert für n eine Zerlegung $n = dd'$, mit $d, d' \in \mathbb{N}$, aber dann muss die kleinere der beiden Zahlen von oben durch $\sqrt{(n)}$ beschränkt sein. Da die Teiler d und d' von n ganzzahlig sind, ist die obere Schranke sogar durch $\lfloor\sqrt{n}\rfloor$ gegeben sein.

```python
import numpy as np

def is_perfect_fast(n):
    if n == 1: return(False)
    sum = 1
    for zahl in range(2,int(np.sqrt(n))+1):
        if n%zahl == 0:
            sum += zahl + n/zahl
    return(sum == n)

def is_perfect_slow(n):
    sum = 0
    for i in range(1,n):
        if n % i == 0:
            sum += i
    return(sum == n)

def perfect_numbers(n1, n2, method = 'slow'):
    """ prints all the perfect numbers in [n1,n2] (n1 < n2) in
        increasing fashion, using either a fast or a slow method
        (default method is 'slow')
    """
    if method == 'fast':
        fun = is_perfect_fast
    else:
        fun = is_perfect_slow

    for k in range(n1,n2+1):
        if fun(k) == True:
            print(k)

perfect_numbers(1,100000,method='fast')
```