Gruppennummer: 22

Datenstrukturen und Algorithmen, Abgabe 2

Beccard, Vincent, 377979; Braun, Basile 388542; Brungs, Simon, 377281

25. April 2018

1 Funktionen sortieren

a

a.1 Zeige das $\subseteq$ transitiv ist

Es gilt $f \in O(g) \wedge g \in O(f) \Rightarrow f \in O(h)$
Laut Definition von $\subseteq$ gilt dementsprechend dann
$f \subseteq g \wedge g \subseteq h \Rightarrow f \subseteq h \Leftrightarrow f \Leftarrow O(g) \wedge g \in O(h) \Rightarrow f \in O(h)$
Damit ist die Transitivität gezeigt.

a.2 Zeige das$\subseteq$ reflexiv ist

Da $f \in O(f)$ gilt folgt laut Definition von $\subseteq$ das auch $f \subseteq f$ Damit ist die Reflexivität gezeigt.

b

$0 \subseteq 4 \subseteq 2^9000 \subseteq 2 \subseteq \log n \subseteq n\sqrt{n} \subseteq n^2 \subseteq n^2 \cdot \log n \subseteq \sum_{i=0}^{n} \frac{14i^2}{1+i} \subseteq \frac{n^3}{2} \subseteq n^3 \subseteq 2^n \subseteq n! \subseteq n^n$

2 O-Notation konkret

a Behauptung: $\frac{1}{4}n^3 - 7n + 17 \in O(n^3)$

$\limsup_{n\to\infty} \frac{1}{4}n^3 - 7n + 17 \in O(n^3) = \frac{1}{4} \geq 0,\ \frac{1}{4} \neq \infty$
Damit ist die Behauptung bewiesen. $\square$

b Behauptung: $n^4 \in O(2n^4 + 3n^2 + 42)$

$\limsup_{n\to\infty} n^4 \in O(2n^4 + 3n^2 + 42)$ (L'Hôpital)

$$= \limsup_{n\to\infty} \frac{24n^3}{48n^2} = \frac{1}{2} \geq 0$$

Damit ist $n^4 \in 0(2n^4 + 3n^2 + 42)$ bewiesen. $\square$

Beccard, Vincent, 377979; Braun, Basile 388542; Brungs, Simon, 377281;
Nummer der Übungsgruppe: 22

3 Laufzeitanalyse

a Bestimmung in Abhängigkeit von n die Best-case Laufzeit $B(n)$

Die Best-case Laufzeit ist $B(n) = 1 + 2n$.

b Bestimmung in Abhängigkeit von n die Worst-case Laufzeit $W(n)$

Die Worst-case Laufzeit ist $W(n) = 1 + 3n$.

c Bestimmung in Abhängigkeit von n der Average-case Laufzeit $A(n)$

Die Average-case Laufzeit ist $A(n) = \sum_{i=0}^{n} \frac{n!}{i!} \cdot 0.5^2 \cdot (1 - 0.5)^{n-i}$

d Äquivalenter Algorithmus mit besserer Average-case Laufzeit

Der folgende äquivalenter Algorithmus hat eine besser Average-case Laufzeit ab $n > 2$, da er nur einmal am Ende abfragt, ob m gleich 0 ist, anstatt bei jedem Element des Arrays, welches true ist. Somit spart man für jedes Element welches true enthält einen Durchlauf und nur im relativ unwahrscheinlichen Fall, wenn alle Elemente im Array false sind braucht er einen mehr.

```
1  int allTrue(bool[] E) {
2    if (E.length < 1) {
3      return -1;
4    }
5    int m = E.length;
6    int i = 0;
7    while (i < E.length) {
8      if (E[i] == true) {
9        m = m - 1;
10     }
11     i = i + 1;
12   }
13   if (m == 0) {
14     return 1;
15   } else {
16     return 0;
17   }
18 }
```

Beccard, Vincent, 377979; Braun, Basile 388542; Brungs, Simon, 377281;
Nummer der Übungsgruppe: 22

e Äquivalenter Algorithmus, dessen Worst-case Laufzeit in o(n) liegt

Nein, gibt es nicht