BGaft 2

BLATTOR HATTINGER
NIKOLIC
NOFAL

1

5)

$O(n)$

E := Elemente Z := Zyklen ~~Spz~~ Zps: sek pro Zyklus ~~Zyklen pro sek~~

$1000E = 1000Z$

Aus Ang.

$\frac{5}{1000} = Zps$

$2000E = 2000Z$

$2000 \cdot Zps = 10$

$\Rightarrow$ 2000E benötigen 10 sek

$3000E = 3000Z$

$3000 \cdot Zps = 15$ $\Rightarrow$ 3000 E benötigen 15 sek

$10000E = 10000Z$

$10000 \cdot Zps = 50$

$\Rightarrow$ 10000 E benötigen 50 sek.

$O(n^2)$

$1000E = 1000^2 Z$

$\frac{5}{1000^2} = Zps$

$2000E = 2000^2 Z$

$2000^2 \cdot Zps = 20$ $\Rightarrow$ 2000E benötigen 20 sek

$3000E = 3000^2 Z$

$3000^2 \cdot Zps = 45$ $\Rightarrow$ 3000E benötigen 45 sek

$10000E = 10000^2 Z$

$10000^2 \cdot Zps = 500$ $\Rightarrow$ 10000 E benötigen 500 sek

$O(\log(n))$

$1000\,E = \log(1000)\,Z$

$\frac{5}{\log(1000)} = Spz$

$2000\,E = \log(2000)\,Z$

$\log(2000) \cdot Spz \approx 5{,}5 \Rightarrow$ 2000 E benötigen 5,5 sek

$3000\,E = \log(3000)\,Z$

$\log(3000) \cdot Spz \approx 5{,}795 \Rightarrow$ 3000 E benötigen 5,795 sek

$10000\,E = \log(10000)\,Z$

$\log(10000) \cdot Spz \approx 6{,}\dot{6} \Rightarrow$ 10000 E benötigen $6{,}\dot{6}$ sek

$O(n \cdot \log(n))$

$1000\,E = 1000 \cdot \log(1000)\,Z$

$\frac{5}{1000 \cdot \log(1000)} = Spz$

$2000\,E = 2000 \cdot \log(2000)\,Z$

$2000 \cdot \log(2000) \cdot Spz \approx 11{,}003 \Rightarrow$ 2000 E benötigen 11,003 s

$3000\,E = 3000 \cdot \log(3000)\,Z$

$3000 \cdot \log(3000) \cdot Spz \approx 17{,}386 \Rightarrow$ 3000 E benötigen 17,38

$10000\,E = 10000 \cdot \log(10000)\,Z$

$10000 \cdot \log(10000) \cdot Spz = 66{,}\dot{6} \Rightarrow$ 10000 E benötigen 66,6

5)

$O(2^n)$

$1000E = 2^{1000} Z$

$\frac{5}{2^{1000}} = Spz$

$2000E = 2^{2000} Z$

$2^{2000} \cdot Spz \approx$ Unrechenbar und unschätzbar hohe Zahl

$= 5 \cdot 2^{1000}$

3000E: — " —

10000E: — " —

$f \in O(2^n)$: f verdoppelt sich wenn sich das Argument um eins erhöht [wikipedia]

$\Rightarrow$ Von 1000 auf 2000 erhöht sich das Argument 1000 mal, daher werden die 5 sek 1000 mal verdoppelt was zu einer unschätzbaren Zahl führt.

Erklärung anhand von $O(n^2)$

$O(n^2)$: 1000 Elemente sind 1000^2 Zyklen

weil 1000^2 Zyklen benötigen 5sec (aus Angabe)
=> 1 Zyklus folglich $\frac{5}{1000^2}$ sec.

2000 Elemente sind bei $O(n^2)$ 2000^2 Zyklen. Die Anzahl Zyklen multipliziert mit der dauer eines Zyklus ist die Gesamt-dauer des Algorithmus.
Demnach 2000^2 (Zyklen) $\cdot$ $\frac{5}{1000^2}$ (sec) ~~sind~~ ist die Gesamtdauer des ~~Al~~ Algorithmus.
Folglich bei 2000 Elemente 10 sec.

- $f_1(n) = 25n + 278\log n$

☆ nach Def von $O(n)$: $25n + 278\log n \leq c \cdot n$

wenn $c = 333$, $n_0 = 4$

dann gilt $25n + 278\log n \leq 333\, n \quad \forall n \geq n_0$

weil $25(4) + 278 \cdot 2 \leq (4)\, 333 \quad \forall n \geq 4$

$100 + 556 \leq 1332$

$\Rightarrow f_1(n) \in O(n)$ ①

☆ nach Def von $\Omega(n)$: $25n + 278\log n \geq c \cdot n$

Wenn $c = 1$, $n_0 = 4$

dann gilt $25n + 278\log n \geq n \quad \forall n \geq n_0$

$\Rightarrow f_1(n) \in \Omega(n)$ ②

aus ①, ② $\Rightarrow$ $f_1(n) \in O(n)$ und $f_1(n) \in \Omega(n)$

$\Rightarrow f_1(n) \in \Theta(n)$

$f_2(n) = (3n^2+n)^2$

$f_2(n) = 9n^4 + 6n^3 + n^2$

☆ nach def von $O(n)$: $9n^4+6n^3+n^2 \leq c \cdot n^4$

wenn $c=16$, $n_0=1$

dann gilt $9n^4+6n^3+n^2 \leq 16 \cdot n^4$

$\forall n \geq 1 \Rightarrow f_2(n) \in O(n^4)$ (1)

☆ nach def von $\Omega(n)$: $9n^4+6n^3+n^2 \geq c \cdot n^4$

wenn $c=1$, $n_0=1$

dann gilt $9n^4+6n^3+n^2 \geq n^4 \; \forall n \geq 1$

$\Rightarrow f(2) \in \Omega(n^4)$ (2)

aus (1), (2) $\Rightarrow f_2(n) \in O(n)$ und $f_2(n) \in \Omega(n)$

$\Rightarrow f_2(n) \in \Theta(n)$

MORITA

$$f_3(n) = \frac{n^3+3n}{2n^2+8}$$

nach def von $O(n)$: $\frac{n^3+3n}{2n^2+8} \leq c\,n$

$\Rightarrow$ wenn $c=1$, $n_0=1$

dann gilt $\frac{n^3+3n}{2n^2+8} \leq n \quad \forall n \geq 1$

Probe $\frac{1+3}{2+8} \leq 1 \Rightarrow \frac{4}{10} \leq 1$ ✓

$\Rightarrow$ $f_3(n) \in O(n)$ ①

nach def von $\Omega(n)$: $\frac{n^3+3n}{2n^2+8} \geq c\,n$

wenn $c = \frac{1}{10}$, $n_0=1$

dann gilt $\frac{n^3+3n}{2n^2+8} \geq \frac{1}{10} n \quad \forall n \geq 1$

probe $\frac{1+3}{2+8} \geq \frac{1}{10} \Rightarrow \frac{4}{10} \geq \frac{1}{10}$ ✓

$\Rightarrow$ $f_3(n) \in \Omega(n)$ ②

aus ①, ② $\Rightarrow$ $f_3(n) \in O(n) \wedge f_3(n) \in \Omega(n)$

$\Rightarrow$ $f_3(n) \in \Theta(n)$

$f_4(n) = 7 \cdot f_1(n) + f_2(n)$

$= 7 \cdot (25n + 278 \log n) + (3n^2 + n)^2$

$= 175n + 1946 \log n + \underbrace{9n^4 + 6n^3 + n^2}$

$= \underbrace{9n^4 + 6n^3 + n^2} + 175n + 1946 \log n$

- nach def von $O(n)$

$9n^4 + 6n^3 + n^2 + 175n + 1946 \log n \leq c \cdot n^4$

Wenn $c = 2137$, $n_0 = 4$

dann gilt:-

$9n^4 + 6n^3 + n^2 + 175n + 1946 \log n \leq 2137 \cdot n^4$

$\forall n \geq 4$

Probe $(2304 + 384 + 16 + 350 + 3092 \leq 2137 \cdot 256$

$(6946 \leq 547072)$

$\Rightarrow f_4(n) \in O(n^4)$ 1

- Nach def von $\Omega(O)$

$9n^4 + 6n^3 + n^2 + 175n + 1946 \log n \geq c \cdot n^4$

Wenn $c = \frac{1}{10}$, $n_0 = 4$

dann gilt; $6946 \geq \frac{1}{10} \cdot 256$ ✓

$\Rightarrow f_4(n) \in \Omega(n^4)$ 2

$\Rightarrow$ aus 1, 2 $\quad f_4(n) \in \Theta(n^4)$

MORITA

aufgabe 6

- die Grundidee hinter generischen typen ist eine Abstraktion um code, der für unterschiedliche Datentypen prinzipiell gleich funktioniert (bspw. Sortier oder listenalgorithmen), Unabhängig vom genauen typ implementiererен zu können.
Ein generischer typ kann dabei von einem zu Grunde liegenden Datentypen (bspw) Integer und String abstrahieren von Comparable) abstrahieren

- Wie werden sie in Java verwendet?
sobald eine Klasse / Methode oder ein interface generisch implementiert wird, dann werden erst bei verwendung die typParameter (<T>) durch konkrete Datentyp ersetzt
syntax, z.b zwei Listen von unterschiedlichen Datentypen erzeugen

```
List<Integer> ListA = new List<Integer>();
List<String> ListB = new List<String>();
```

MORITA

// Methode add funktioniert für alle listen glich

```
ListA.add(10);
ListB.add("Myname");
```

- ein einfaches beispiel
  Generische Klasse zur speicherung von 2 Koordinaten x, y

```
Public class Coordinate2D<T>{
    private T x;
    private T y;
    Public Coordinate2D(T x, T y){
        this.x = x;
        this.y = y;
    }
}
```

- Was sind die eingeschränkte typparameter und generische Methoden?

  - eingeschränkte typparameter sind typparameter die über festlegung eines ober typen eingeschränkt werden (bspw) muss bei <T extends Comparable<T>> der typ parameter T auf jeden fall untertyp des Comparable-interface sein, ist das nicht der fall, meldet der Compiler bereits einen fehler

  unter generischen Methoden versteht man methoden, die unter verwendung von generischen typen arbeiten, Dies macht wie bereits beschrieben sinn wenn eine Methode für unterschiedliche daten typen gleich funktioniert (Zb. sortieren)

```
Public <T extends Comparable<T>> voidcheckT

public <T extends Comparable<T>> void check(T p){

      }
```