Script generated by TTT

Title: groh: profile1 (24.06.2016)

Date: Fri Jun 24 09:25:07 CEST 2016

Duration: 73:02 min

Pages: 21

Teil III: Algorithmen und Datenstrukturen

basiert auf (1), (3) und (4)
(Zitierungen aus Lesbarkeitsgründen nicht überall angegeben)

Wo stehen wir?

- In Objekten sind jeweils Mengen von Attributen gekapselt („Zustand“), auf deren Werte insbesondere der Programmcode in den Methoden („Verhalten“) zugreifen kann.
- Objekte sind Instanzen von Klassen. Klassen bilden eine Hierarchie. Objekte sind polymorph.
- Objektorientierung <---> Information Hiding:
  - Objekte sollen ihre Attribute möglichst verbergen und
  - von den Methoden in der Klasse soll nur die Signatur und die Semantik, nicht aber die genaue Implementierung für andere Objekte interessant sein.
  - Methoden sollen wenn möglich keine Seiteneffekte auf andere Objekte haben.

--> Klasse als abstrakter Datentyp

Wo stehen wir?

- In Objekten sind jeweils Mengen von Attributen gekapselt („Zustand“), auf deren Werte insbesondere der Programmcode in den Methoden („Verhalten“) zugreifen kann.
- Objekte sind Instanzen von Klassen. Klassen bilden eine Hierarchie. Objekte sind polymorph.
- Objektorientierung <---> Information Hiding:
  - Objekte sollen ihre Attribute möglichst verbergen und
  - von den Methoden in der Klasse soll nur die Signatur und die Semantik, nicht aber die genaue Implementierung für andere Objekte interessant sein.
  - Methoden sollen wenn möglich keine Seiteneffekte auf andere Objekte haben.

--> Klasse als abstrakter Datentyp

Algorithmusbegriff

- Definition Problem: Gegeben: Menge von Informationen: daraus weitere (unbekannte) Information bestimmen
- Algorithmus: *Exaktes, schrittweises Verfahren zur Lösung eines Problems*:
  - Implementiert eine Funktion $f: E \to A$
    (E: Eingaben, A: Ausgaben (Repräsentationen als Daten))
    (--> „determiniert“: Gleiches E produziert gleiches A)
  - ist endliche Liste von elementar ausführbaren („effektiven“) Instruktionen, diese produzieren endliche Sequenz (--> „terminiert“) von wohldefinierten Zuständen <--> Schritten
- Nicht jede Funktion ist berechenbar (<-- es ex. ein Algorithmus)
- Programm: Formulierung eines Algorithmus in einer Programmiersprache (bspw. mit Hilfe von Java Methoden)

[8]

Algorithmusbegriff

- Definition Problem: Gegeben: Menge von Informationen: daraus weitere (unbekannte) Information bestimmen
- Algorithmus: *Exaktes, schrittweises Verfahren zur Lösung eines Problems*:
  - Implementiert eine Funktion $f: E \to A$
    (E: Eingaben, A: Ausgaben (Repräsentationen als Daten))
    (--> „determiniert“: Gleiches E produziert gleiches A)
  - ist endliche Liste von elementar ausführbaren („effektiven“) Instruktionen, diese produzieren endliche Sequenz (--> „terminiert“) von wohldefinierten Zuständen <--> Schritten
- Nicht jede Funktion ist berechenbar (<-- es ex. ein Algorithmus)
- Programm: Formulierung eines Algorithmus in einer Programmiersprache (bspw. mit Hilfe von Java Methoden)

[8] 4

Beispiel: Euklidscher Algorithmus

Problem: Gegeben: $a, b \in \mathbb{N}\ a, b \neq 0$. Gesucht: $ggT(a, b)$

Algorithmus:

1. Falls $a = b$, brich Berechnung ab, es gilt $ggT(a, b) = a$.
   Ansonsten gehe zu Schritt 2.
2. Falls $a > b$, ersetze a durch $a - b$ und
   setze Berechnung in Schritt 1 fort.
   Ansonsten gehe zu Schritt 3.
3. Es gilt $a < b$. Ersetze b durch $b - a$ und
   setze Berechnung in Schritt 1 fort.

Beispiel: Euklidscher Algorithmus

Problem: Gegeben: $a, b \in \mathbb{N}\ a, b \neq 0$. Gesucht: $ggT(a, b)$

Algorithmus:

1. Falls $a = b$, brich Berechnung ab, es gilt $ggT(a, b) = a$.
   Ansonsten gehe zu Schritt 2.
2. Falls $a > b$, ersetze a durch $a - b$ und
   setze Berechnung in Schritt 1 fort.
   Ansonsten gehe zu Schritt 3.
3. Es gilt $a < b$. Ersetze b durch $b - a$ und
   setze Berechnung in Schritt 1 fort.

5

Worst, Average, Best Case

- Ausgangspunkt: Größe n des Problems
  - Beispiel: Problem: Gegeben: Menge M von Zahlen, gesucht: Sortierte Sequenz → Eingabegröße: $n := |M|$

- Laufzeit-Komplexität: Sei A_n die Menge aller Instanzen der Größe n eines Problems A. Sei X ein Algorithmus zur Lösung des Problems, der auf einer Instanz $a_i \in A_n$ $T_{X,n}(a_i)$ viele Schritte benötigt:
  - Worst Case Laufzeit von X für Problemgröße n:
    $$T_{X,n}^{worst} = \max_i T_{X,n}(a_i)$$
  - Average Case Laufzeit von X für Problemgröße n:
    $$T_{X,n}^{average} = \frac{1}{|A_n|} \Sigma_i T_{X,n}(a_i)$$
  - Best Case Laufzeit von X für Problemgröße n:
    $$T_{X,n}^{best} = \min_i T_{X,n}(a_i)$$

Worst, Average, Best Case

- Ausgangspunkt: Größe n des Problems
  - Beispiel: Problem: Gegeben: Menge M von Zahlen, gesucht: Sortierte Sequenz → Eingabegröße: $n := |M|$

- Laufzeit-Komplexität: Sei A_n die Menge aller Instanzen der Größe n eines Problems A. Sei X ein Algorithmus zur Lösung des Problems, der auf einer Instanz $a_i \in A_n$ $T_{X,n}(a_i)$ viele Schritte benötigt:
  - Worst Case Laufzeit von X für Problemgröße n:
    $$T_{X,n}^{worst} = \max_i T_{X,n}(a_i)$$
  - Average Case Laufzeit von X für Problemgröße n:
    $$T_{X,n}^{average} = \frac{1}{|A_n|} \Sigma_i T_{X,n}(a_i)$$
  - Best Case Laufzeit von X für Problemgröße n:
    $$T_{X,n}^{best} = \min_i T_{X,n}(a_i)$$

10

Worst, Average, Best Case

- Ausgangspunkt: Größe n des Problems
  - Beispiel: Problem: Gegeben: Menge M von Zahlen, gesucht: Sortierte Sequenz → Eingabegröße: $n := |M|$

- Laufzeit-Komplexität: Sei A_n die Menge aller Instanzen der Größe n eines Problems A. Sei X ein Algorithmus zur Lösung des Problems, der auf einer Instanz $a_i \in A_n$ $T_{X,n}(a_i)$ viele Schritte benötigt:
  - Worst Case Laufzeit von X für Problemgröße n:
    $$T_{X,n}^{worst} = \max_i T_{X,n}(a_i)$$
  - Average Case Laufzeit von X für Problemgröße n:
    $$T_{X,n}^{average} = \frac{1}{|A_n|} \Sigma_i T_{X,n}(a_i)$$
  - Best Case Laufzeit von X für Problemgröße n:
    $$T_{X,n}^{best} = \min_i T_{X,n}(a_i)$$

10

Worst, Average, Best Case

- Ausgangspunkt: Größe n des Problems
  - Beispiel: Problem: Gegeben: Menge M von Zahlen, gesucht: Sortierte Sequenz → Eingabegröße: $n := |M|$

- Laufzeit-Komplexität: Sei A_n die Menge aller Instanzen der Größe n eines Problems A. Sei X ein Algorithmus zur Lösung des Problems, der auf einer Instanz $a_i \in A_n$ $T_{X,n}(a_i)$ viele Schritte benötigt:
  - Worst Case Laufzeit von X für Problemgröße n:
    $$T_{X,n}^{worst} = \max_i T_{X,n}(a_i)$$
  - Average Case Laufzeit von X für Problemgröße n:
    $$T_{X,n}^{average} = \frac{1}{|A_n|} \Sigma_i T_{X,n}(a_i)$$
  - Best Case Laufzeit von X für Problemgröße n:
    $$T_{X,n}^{best} = \min_i T_{X,n}(a_i)$$

10

Asymptotische Komplexität: Landau Symbole

- Neben $O(f)$ („…höchstens so schnell wie f“) gibt es auch noch die Klassen $\Omega(f)$ („… mindestens so schnell…“), $\Theta(f)$ („… genauso schnell…“), $o(f)$ („… echt langsamer…“), $\omega(f)$ („… echt schneller…“).
- Beispiele:
  - $h(n) = 15n^3 + 200n^2 + 267854 \in O(f(n) = n^3)$ oder kurz $\in O(n^3)$
  - $h(n) = 11n^2 + 854 \in O(n^2)$
  - $h(n) = 11n^2 + 854 \in O(n^3)$
  - $h(n) = 133 \log n + 54 \in O(\log n)$
- Schreibweisen:

  $h: n \to 12n^3 \in O(f: n \to n^3) \leftrightarrow h(n) = 12n^3 \in O(f(n) = n^3) \leftrightarrow 12n^3 \in O(n^3) \leftrightarrow 12n^3 = O(n^3) \leftrightarrow h \in O(f) \leftrightarrow h = O(f)$

Asymptotische Komplexität: Landau Symbole

- Neben $O(f)$ („…höchstens so schnell wie f“) gibt es auch noch die Klassen $\Omega(f)$ („… mindestens so schnell…“), $\Theta(f)$ („… genauso schnell…“), $o(f)$ („… echt langsamer…“), $\omega(f)$ („… echt schneller…“).
- Beispiele:
  - $h(n) = 15n^3 + 200n^2 + 267854 \in O(f(n) = n^3)$ oder kurz $\in O(n^3)$
  - $h(n) = 11n^2 + 854 \in O(n^2)$
  - $h(n) = 11n^2 + 854 \in O(n^3)$
  - $h(n) = 133 \log n + 54 \in O(\log n)$
- Schreibweisen:

  $h: n \to 12n^3 \in O(f: n \to n^3) \leftrightarrow h(n) = 12n^3 \in O(f(n) = n^3) \leftrightarrow 12n^3 \in O(n^3) \leftrightarrow 12n^3 = O(n^3) \leftrightarrow h \in O(f) \leftrightarrow h = O(f)$

12

Asymptotische Komplexität: Landau Symbole

- Neben $O(f)$ („…höchstens so schnell wie f“) gibt es auch noch die Klassen $\Omega(f)$ („… mindestens so schnell…“), $\Theta(f)$ („… genauso schnell…“), $o(f)$ („… echt langsamer…“), $\omega(f)$ („… echt schneller…“).
- Beispiele:
  - $h(n) = 15n^3 + 200n^2 + 267854 \in O(f(n) = n^3)$ oder kurz $\in O(n^3)$
  - $h(n) = 11n^2 + 854 \in O(n^2)$
  - $h(n) = 11n^2 + 854 \in O(n^3)$
  - $h(n) = 133 \log n + 54 \in O(\log n)$
- Schreibweisen:

  $h: n \to 12n^3 \in O(f: n \to n^3) \leftrightarrow h(n) = 12n^3 \in O(f(n) = n^3) \leftrightarrow 12n^3 \in O(n^3) \leftrightarrow 12n^3 = O(n^3) \leftrightarrow h \in O(f) \leftrightarrow h = O(f)$

12

Asymptotische Komplexität: Landau Symbole

- Neben $O(f)$ („…höchstens so schnell wie f“) gibt es auch noch die Klassen $\Omega(f)$ („… mindestens so schnell…“), $\Theta(f)$ („… genauso schnell…“), $o(f)$ („… echt langsamer…“), $\omega(f)$ („… echt schneller…“).
- Beispiele:
  - $h(n) = 15n^3 + 200n^2 + 267854 \in O(f(n) = n^3)$ oder kurz $\in O(n^3)$
  - $h(n) = 11n^2 + 854 \in O(n^2)$
  - $h(n) = 11n^2 + 854 \in O(n^3)$
  - $h(n) = 133 \log n + 54 \in O(\log n)$
- Schreibweisen:

  $h: n \to 12n^3 \in O(f: n \to n^3) \leftrightarrow h(n) = 12n^3 \in O(f(n) = n^3) \leftrightarrow 12n^3 \in O(n^3) \leftrightarrow 12n^3 = O(n^3) \leftrightarrow h \in O(f) \leftrightarrow h = O(f)$

12

Asymptotische Komplexität: Landau Symbole

- Neben $O(f)$ („…höchstens so schnell wie f“) gibt es auch noch die Klassen $\Omega(f)$ („… mindestens so schnell…“), $\Theta(f)$ („… genauso schnell…“), $o(f)$ („… echt langsamer…“), $\omega(f)$ („… echt schneller…“).

- Beispiele:
  - $h(n) = 15n^3 + 200n^2 + 267854 \in O(f(n) = n^3)$ oder kurz $\in O(n^3)$
  - $h(n) = 11n^2 + 854 \in O(n^2)$
  - $h(n) = 11n^2 + 854 \in O(n^3)$
  - $h(n) = 133 \log n + 54 \in O(\log n)$

- Schreibweisen:

$h: n \to 12n^3 \in O(f: n \to n^3) \leftrightarrow h(n) = 12n^3 \in O(f(n) = n^3) \leftrightarrow$
$12n^3 \in O(n^3) \leftrightarrow 12n^3 = O(n^3) \leftrightarrow h \in O(f) \leftrightarrow h = O(f)$

Asymptotische Komplexität: Landau Symbole

- Neben $O(f)$ („…höchstens so schnell wie f“) gibt es auch noch die Klassen $\Omega(f)$ („… mindestens so schnell…“), $\Theta(f)$ („… genauso schnell…“), $o(f)$ („… echt langsamer…“), $\omega(f)$ („… echt schneller…“).

- Beispiele:
  - $h(n) = 15n^3 + 200n^2 + 267854 \in O(f(n) = n^3)$ oder kurz $\in O(n^3)$
  - $h(n) = 11n^2 + 854 \in O(n^2)$
  - $h(n) = 11n^2 + 854 \in O(n^3)$
  - $h(n) = 133 \log n + 54 \in O(\log n)$

- Schreibweisen:

$h: n \to 12n^3 \in O(f: n \to n^3) \leftrightarrow h(n) = 12n^3 \in O(f(n) = n^3) \leftrightarrow$
$12n^3 \in O(n^3) \leftrightarrow 12n^3 = O(n^3) \leftrightarrow h \in O(f) \leftrightarrow h = O(f)$

Asymptotische Komplexität: Landau Symbole

- Neben $O(f)$ („…höchstens so schnell wie f“) gibt es auch noch die Klassen $\Omega(f)$ („… mindestens so schnell…“), $\Theta(f)$ („… genauso schnell…“), $o(f)$ („… echt langsamer…“), $\omega(f)$ („… echt schneller…“).

- Beispiele:
  - $h(n) = 15n^3 + 200n^2 + 267854 \in O(f(n) = n^3)$ oder kurz $\in O(n^3)$
  - $h(n) = 11n^2 + 854 \in O(n^2)$
  - $h(n) = 11n^2 + 854 \in O(n^3)$
  - $h(n) = 133 \log n + 54 \in O(\log n)$

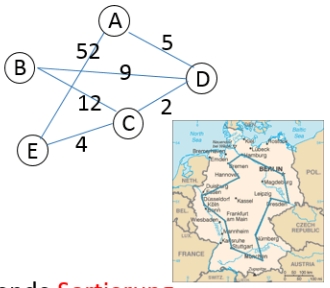
- Schreibweisen:

$h: n \to 12n^3 \in O(f: n \to n^3) \leftrightarrow h(n) = 12n^3 \in O(f(n) = n^3) \leftrightarrow$
$12n^3 \in O(n^3) \leftrightarrow 12n^3 = O(n^3) \leftrightarrow h \in O(f) \leftrightarrow h = O(f)$

Beispiele für Probleme

- SAT: Gegeben aussagenlogische Formel mit n Variablen (bspw. $(a \vee b) \wedge (\neg c \vee (e \wedge f))$), gesucht: gibt es eine Belegung der Variablen aus $\{true, false\}^n$, die die Formel wahr macht? (NP-vollständig, keine Lösung in P bekannt)

- TSP (Traveling Salesperson): Gegeben gewichteter Graph $G(V, E, w)$, gesucht: eine Besuchsreihenfolge der Knoten (Städte), so dass die Gesamtreisestrecke (Entfernung d zwischen Städten V_1 und V_2 : $d = 1/w(e(V_1, V_2))$) minimal ist. (keine effiziente (d.h. worst case polynomiell zeitkomplexe) Lösung bekannt)

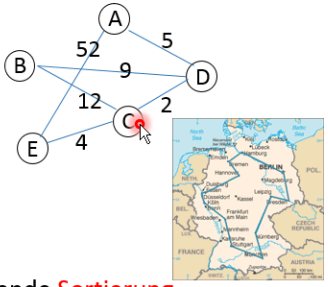


- Gegeben eine Menge von Zahlen, gesucht: aufsteigende Sortierung. (Ohne Zusatzannahmen über Maximalgröße der Zahlen und nur unter Zuhilfenahme von Vergleichen): Worst-Case-Zeitkomplexität: $O(n \log n)$

Beispiele für Probleme

- SAT: Gegeben aussagenlogische Formel mit n Variablen (bspw. $(a \vee b) \wedge (\neg\, c \vee (e \wedge f))$), gesucht: gibt es eine Belegung der Variablen aus $\{true, false\}^n$, die die Formel wahr macht? (NP-vollständig, keine Lösung in P bekannt)

- TSP (Traveling Salesperson): Gegeben gewichter Graph $G(V, E, w)$, gesucht: eine Besuchsreihenfolge der Knoten (Städte), so dass die Gesamtreisestrecke (Entfernung d zwischen Städten V_1 und V_2 : $\mathrm{d} = 1/\mathrm{w}(\mathrm{e}(\mathrm{V}_1, \mathrm{V}_2))$) minimal ist. (keine effiziente (d.h. worst case polynomiell zeitkomplexe) Lösung bekannt)



- Gegeben eine Menge von Zahlen, gesucht: aufsteigende Sortierung. (Ohne Zusatzannahmen über Maximalgröße der Zahlen und nur unter Zuhilfenahme von Vergleichen): Worst-Case-Zeitkomplexität: $O(n\ log\ n)$

Datenstrukturen, abstrakte Datentypen

- Datenstruktur: Formal definierte Organisationsform von Daten im Hinblick auf effiziente Verarbeitung.
  - Bsp: Graph, gewichteter gerichteter Graph, Binärbaum, Baum, Array (Feld), doppelt verkettete Liste, Hashmap….

- abstrakter Datentyp: Menge von Daten eines Typs + Definition von zugehörigen Operationen auf diesen Daten
  - Operationen werden ihrer Semantik nach definiert. Auf Implementierung wird nicht eingegangen.
    → Kapselung durch Definition einer Schnittstelle (Syntax und Semantik)
  - werden oft mit Hilfe von Datenstrukturen implementiert
  - Bsp: Klassen in Java (aber auch primitive Typen) implementieren abstrakte Datentypen