Algorithmen und Datenstrukturen
BAI3-AD – SoSe 2018

Prof. Dr. Marina Tropmann-Frick

BT7, Raum 10.86

marina.tropmann-frick@haw-hamburg.de

HAW
HAMBURG

Vorlesungsüberblick

1. Einführung

Motivation, Übersicht, Begriffsklärung, Größenordnungen

2. Lineare Datenstrukturen

Liste, Stack, Queue

3. Algorithmengrundlagen

Die ersten Algorithmen mit Komplexitäts-(Aufwands-)analyse

4. Sortieren

Sortieralgorithmen

5. Verzweigte Datenstrukturen

Bäume, Operationen und Implementierungsmethoden

6. Grundlegende Graph-Algorithmen

Graphen und Implementierungsmethoden, kürzeste Pfade, Breitensuche, Tiefensuche, Dijkstra's Algorithmus

7. Hash-Verfahren

Hashfunktionen, Analyse

8. Optimierung

Lineare Optimierung, Lösung Nichtlinearer Probleme

-- (Ausblick – weitere effiziente Algorithmen)*

1. Einführung

(1) Motivation

(2) Übersicht

(3) Begriffsklärung

(4) Größenordnungen

1. Einführung

(1) Motivation

1. Einführung

(1) Motivation

Wort **Algorithmus** – Abwandlung des Namens:

Abu Abdullah Muhammad ibn Musa al-Khwarizmi

(b. before 800, d. after 847 in Baghdad)

Mathematiker, Astronom, Geograph und Historiker

Lehrbuch über die indischen Ziffern
(verfasst um 825 im Haus der Weisheit in Bagdad)

"A stamp issued in honour of al-Khwarizmi by the former USSR post in 1983."

Bildquelle: www.muslimheritage.com/article/contribution-al-khwarizmi-mathematics-and-geography

(1) Motivation

Algorithmen – zentrales Thema der Informatik

Entwicklung und Untersuchung von Algorithmen zur Lösung vielfältiger Probleme – eine der wichtigsten Aufgaben der Informatik.

Algorithmen erfordern geeignete Methoden zur Strukturierung der von den Algorithmen manipulierten Daten.

Algorithmen (Alg.) und Datenstrukturen (DS) gehören zusammen!

(1) Motivation

Geeignete Alg. und DS sind die Grundlage für schnelle und in der Speichernutzung sparsame (wirtschaftliche, effiziente) Programme bzw. Software.

Algorithmus => genaue (kein Interpretationsspielraum!) Beschreibung eines systematischen Ablaufs (in den meisten Fällen) für Datenmanipulation.

Datenstruktur => systematische Anordnung von Daten.

(2) Übersicht

Zentrales Paradigma bei der Entwicklung und Anwendung von Alg. Und DS → **Abstraktion:**

Abstraktion (Abziehung, Absonderung) ist die Heraushebung eines Erkenntnisinhalts durch die willkürliche, aktive Aufmerksamkeit (Apperzeption), das willkürliche, absichtliche, zweckbewuste Festhalten bestimmter Vorstellungsmerkmale unter gleichzeitiger Vernachlässigung, Zurückdrängung, Hemmung; anderer Merkmale. Der Gegensatz zur Abstraktion im engeren Sinne, d.h. zur Erweiterung des Begriffsinhalts, ist die logische Determination.

1. Einführung

(2) Übersicht

‼ Unterscheiden von

systematischem Aufbau

und

Implementierung

Notation in

Pseudocode (Mischung mit natürlicher Sprache und math. Notation)

Syntax einer höheren Programmiersprache (meistens Java)

(2) Übersicht

‼ Spezielle Notation von Algorithmen:

Zuweisung

Z.B. „setze $j = i + 3$“: Weise der Variablen j den Wert zu, der sich durch Addition von 3 zum Wert der Variablen i ergibt, und führe die Ausführung in der nächsten Zeile fort.

Falls-Anweisung

Z.B. „falls $j > 0$“: Überprüfe, ob der Wert der Variablen j größer als 0 ist. Falls ja, so führe die Ausführung in der nächsten Zeile fort. Falls nein, führe die Ausführung hinter dem eingerückten Teil, dem so genannten *Falls-Teil*, fort.

Variante: „falls $j > 0$ … sonst“: Im Ja-Fall wird der Falls-Teil ausgeführt und dann hinter dem nach der Sonst-Zeile eingerückten Teil, dem so genannten *Sonst-Teil*, fortgefahren. Im Nein-Fall wird der Sonst-Teil ausgeführt und danach fortgefahren.

1. Einführung

(2) Übersicht

‼ Spezielle Notation von Algorithmen:

Solange-Schleife

Z.B. „solange $j > 0$“: Überprüfe, ob der Wert der Variablen j größer als 0 ist. Falls ja, so führe den eingerückten Schleifenrumpf aus. Überprüfe danach, ob der Wert von j immer noch größer als 0 ist. Falls ja, führe den Schleifenrumpf erneut aus. Und so weiter . . . Wenn die Bedingung $j > 0$ nicht (mehr) zutrifft, führe die Ausführung hinter dem Schleifenrumpf (gleiche Einrücktiefe) fort.

(2) Übersicht

‼ Beschreibung der Semantik einer Operation:

Algebraische Methode

In Anlehnung an *OCL* mit Vor- und Nachbedingungen (Pre- und Postconditions)

Object Constraint Language (OCL)

ist eine Sprache, die formal aufgebaut ist und es erlaubt, Bedingungen an Objekte und Operationen zu formulieren.

Axiomatische Methode (in wenigen Fällen)

Exakte mathematische Notation um die Semantik einer Operation zu beschreiben

Definition (Vorbedingung, Nachbedingung):

Eine **Vorbedingung** (precondition) gibt Eigenschaften der Eingabeparameter an, mit denen die Operation fehlerfrei anwendbar ist. Der Anwender eines Abstrakten Datentyps (ADT) muss sicher gehen, dass diese Eigenschaften erfüllt sind. Zur Sicherheit sollte der Entwickler des ADTs Abfragen einbauen (assert).

Eine **Nachbedingung** (postcondition) gibt die Eigenschaften des Objekts nach Beendigung der Operation an (das Ergebnis der Operation). Dies wird oft in mathematischer Schreibweise spezifiziert, da sie sehr präzise ist und man mathematische Aussagen oft leicht in ein Computerprogramm umwandeln kann.

Beispiel: (Operationsbeschreibung mit Vor-, Nachbedingung)

Operation / : float × float → float ∪ { $error$ }; $(a, b) \mapsto a/b$
pre: keine
post: a/b ist der Quotient aus den Zahlen a und b
Falls $b = 0$ wird $error$ zurückgeliefert

Übung:

Beschreiben Sie die Semantik folgender Operation mit sinnvollen Vor- und Nachbedingungen: Die Operation `concat` fügt zwei String-Objekte aneinander. Keins der beiden Objekte darf den Wert `null` haben. Die Operation liefert den zusammengefügten String zurück.

Operation `concat`: string × string → string; $(s, t) \mapsto st$
pre: s, t sind nicht `null`
post: st ist eine neue Zeichenkette, in der die ersten Zeichen von s sind, gefolgt von denen von t
(s, t bleiben durch die Operation unverändert)

(3) Begriffsklärung

Wir definieren einen *Datentyp* indem wir bestimmte Eigenschaften angeben, die jede Implementierung des Datentyps haben soll. Die Eigenschaften werden durch Operationen auf einer Wertemenge definiert.

Definition (Abstrakter Datentyp):

Ein Abstrakter Datentyp (ADT) besteht aus einer oder mehreren Objektmengen (auch Wertebereich oder Wertemenge genannt) und Operationen, die darauf definiert sind.
Die Operationen werden auch Funktionen oder Methoden genannt. Man spezifiziert die Operationen über ihre *Signatur* zusammen mit einer *Semantik.* Möchte man die Spezifikation (Signatur, Semantik) von der Implementierung unterscheiden, so wird letztere oft Methode genannt. Eine Methode ist dann die Implementierung einer Operation. Wichtig ist hier besonders die Unabhängigkeit von einer konkreten Programmiersprache oder Implementierung.

Abstraktionsebenen

Abstraktionsebene	Name	Beispiel
2	Abstrakter Datentyp	Integer
1	Implementierung	32-bit Integer
0	Objekt	`Int i = 5;`

Beispiel – atomare Datentypen (in Java)

Datentypen, die nicht zerlegt werden können (vs. komplex)

Bezeichnung	Datentyp	Elementare Operationen
Boolesche Wahrheitswerte	bool	+ - * / % ++ -- += -= & && | || ˆ ! < > == <= >= etc.
Ganze Zahlen	Int, byte	
Reelle Zahlen	float, double	
Unicode-Zeichen (UTF-16)	char	

Definition (Signatur):

Das Abbildungsverhalten einer Operation:

Operationsbezeichnung : Menge Inputparameter → Menge Outputparameter

Input- und Outputparameter → Elemente des Wertebereichs.
Mit Typenbezeichnungen, wie aus der Programmierung bekannt.

Beispiel:

$$\begin{aligned}
&\text{Operation } * &:\quad& \text{int} \times \text{int} \to \text{int}\\
&\text{Operation } == &:\quad& \text{int} \times \text{int} \to \text{bool}\\
&\text{Operation } ++ &:\quad& \text{int} \to \text{int}
\end{aligned}$$

(3) Begriffsklärung

Signatur beschreibt den syntaktischen Aspekt einer Operation. Es wird das Eingabe / Ausgabeverhalten definiert

Semantik kann durch die Nachbedingungen beschrieben werden, oft durch textuelle Beschreibung, in mathematischen Kontexten durch Formeln

Übung:

Schreiben Sie die Signaturen für folgende Operationen des Typs int auf:

$$*= \qquad \leq \qquad \%$$

Operation $*=$: int × int → int (a $*=$ b ist identisch zu a $=$ a $*$ b)
Operation $\leq$: int × int → bool
Operation $\%$: int × int → int

Eigenschaften von ADTs

1. **Universalität** – der Datentyp ist von Implementierung und Programmiersprache unabhängig. Innerhalb einer Programmiersprache bedeutet dies Verwendungsmöglichkeit in jedem beliebigen Programm.
2. **Präzise Beschreibung** – exakte algebraische oder axiomatische Beschreibung.
3. **Einfachheit** – von zwei Lösungen, die das gleiche Problem lösen, ist die einfachere zu verwenden.
4. **Kapselung** – die interne Form (Implementierung und Repräsentation) ist für den Anwender nicht sichtbar (Geheimnisprinzip, Kapselung). Er sieht nur das Interface.
5. **Geschütztheit** – aus des Kapselung soll sich ergeben, dass die internen Datenstrukturen nicht von außen böswillig angegriffen werden können.
6. **Modularität** – ADT sollen sich einfach kombinieren und in größere Anwendungskontexte integrieren lassen.

(3) Begriffsklärung

Operationen von ADTs

1. **Initialisierung** – Objekte werden erzeugt und auf bestimmte Weise vorbelegt
2. **Nicht-modifizierende** Operationen – Objekte werden gelesen
3. **Modifizierende** Operationen – Objekte werden verändert oder beschrieben
4. **Funktionen** – aus einem oder mehreren Objekten entsteht ein neues
5. **Zerstörung** – Beseitigen von Objekten und Freigabe des Speichers

Fragen / Anregungen ?

Orders of Magnitude – Vorbereitung zu Aufwandabschätzungen

- Betrachtung der Aufwandsabschätzungen für Algorithmen und Datenstrukturen in Abhängigkeit von der Eingabegröße
  - kleine Eingabegrößen sind eher uninteressant

- Häufig wird nur das asymptotische Verhalten des Aufwands von Interesse sein, d.h. komplexe Summen oder Rekursionsgleichungen werden mit einfachen Funktionen asymptotisch verglichen
  - eine genaue Bestimmung des Ressourcenverbrauchs ist häufig sehr aufwändig

1. Einführung

(4) Größenordnungen

Orders of Magnitude – Vorbereitung zu Aufwandabschätzungen

- Wir sehen den Ressourcenverbrauch als gleich an, wenn sich beide für große Eingaben nur um einen multiplikativen Faktor unterscheiden:

z.B. werden als gleich angesehen:

$$n \rightarrow n^2 \quad \text{und} \quad n \rightarrow \frac{1}{2}n^2 - \frac{1}{2}n$$

➡ weitere mathematische Betrachtungen zu der asymptotischen Äquivalenz folgt bei der Einführung der sog. O-Notation

Fragen / Anregungen ?