Rückkopplung beim computergestützten Lernen von Algorithmen und Datenstrukturen

Dr. Guido Rößling

Rechnerbetriebsgruppe, FB Informatik
Technische Universität Darmstadt

guido@rbg.informatik.tu-darmstadt.de

Zunächst einige Definitionen…

- **Algorithmus**: „präzise Handlungsvorschrift zur Lösung eines Problems oder einer bestimmten Art von Problemen“
- **Datenstruktur**: Art, wie man Daten ablegt und verknüpft
  - Meist mit dem Ziel, möglichst schnell darauf zugreifen zu können
  - Je nach Aufgabe bieten sich bestimmte Datenstrukturen an
- Entscheidend für beide ist das dynamische Verhalten
- Informatik betrachtet *viele* Algorithmen & Datenstrukturen
- Studierende müssen entsprechend viele lernen und begreifen
- „Klassische“ Darstellung: Quelltext, „Snapshots“, Text, …
  - Also statische Darstellung für dynamische Inhalte
  - Warum nicht rechnergestützt?
  - Aber: wie macht man das „richtig“?

Typische Themengebiete

- Algorithmen und Datenstrukturen der Informatik
  - Suchen und Sortieren
  - Kürzeste Wege
  - Daten ver- und entschlüsseln
  - Datenkompression
  - etc.
- Algorithmen / dynamische Systeme anderer Wissenschaften
  - Mathematik: Gauß'sche Elimination, LRP-Zerlegung, Matrixinversion, …
  - Physik, Chemie, Biologie: Simulation des Verhaltens von *X*
- Meist als „AV“ abgekürzt (Algorithmenvisualisierung)
  - Obwohl einiges eher Algorithmen*animation* ist

„Statische“ Beispiele zur AV

Bubble Sort


```
public void sort(int[] array)
{
  int i, j, k;
  boolean swapPerformed = true;
  for (i=array.length; swapPerformed && i>-1; i--)
    for (j=1, swapPerformed = false; j<i; j++)
      if (array[j-1] > array[j])
      {
        swap(array, j-1, j);
        swapPerformed = true;
      }
}
```

Dictionary-based Compression

LZW

Input: A B A B B A C A B

I: A

X: B

Ix: AB in Dictionary

Output: 65 66

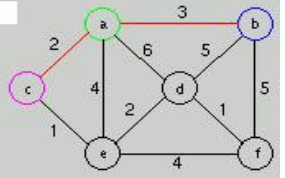
Index	Entry
65	A
66	B
67	C
255	<EOF>
256	AB
257	BA

1. Insert all 256 byte values into the dictionary at positions [0, 255]
2. Set the input I to the first input character
3. Read the next input character x ; if input end is reached, go to step 5
4. If Ix is in dictionary, set I = I x and continue with step 3
5. Print the dictionary address of I
6. Insert Ix at the next free dictionary position, if not on last character
7. Set I = x and continue with step 3, until input is finished

Graphs

Floyd's Algorithm

i : 1 Start node: b
j : 2 End node: c
x : 0 Intermediate node: a



Graph G

	a	b	c	d	e	f
a	–	3	2	6	4	–
b	3	**6**	**5**	5	–	5
c	2	–	–	–	1	–
d	6	5	–	–	2	1
e	4	–	1	2	–	4
f	–	5	–	1	4	–

```
public void floyd (int [] [] a)
{
int i, j, x;
for (x = 0; x < n; x++)
  for (i = 0; i < n; i++)
    if (a [i] [x] != NO_EDGE)
      for (j = 0; j < n; j++)
        if (a [x] [j] != NO_EDGE)
          if (a [i] [j] == NO_EDGE
            ||(a [i] [x] + a [x] [j] < a [i] [j]))
            a [i] [j] = a [i] [x] + a [x] [j];
}
```

Implementation

Hashing

Quadratic Probing with Alternating Sign

alph(c):	1	19	5	1	18	3	8	5	24	1	13	16	12	5		
Input:	A	S	E	A	R	C	H	E	X	A	M	P	L	E		
alph(c) % 17:	1	2	5	1	1	3	8	5	7	1	13	16	12	5		
	1	2	5	1	1	3	8	5	7	1	13		12	5		
			0	0			4		0			16		4		
				2					2					6		
				14					14							
									5							
									9							

A total of 26 accesses, average 1.86

		Accesses
0	A	2
1	A	1
2	S	1
3	C	1
4	E	2
5	E	1
6	E	3
7	X	1
8	H	1
9	A	6
10		
11		
12	L	1
13	M	1
14	R	4
15		
16	P	1

Basic Idea:
1. Set i = 0
2. Calculate s(i, k) = (h(k) + $(-1)^i * (i/2)^2$) % m
3. If position s(i,k) unoccupied, insert element there. Otherwise, set i = i + 1 and continue with step 2

Das Gaußsche Eliminationsverfahren

$$\begin{bmatrix} 1 & 2 & -5 & 3 & 6 & 14 \end{bmatrix}$$

Zeile

$$\begin{bmatrix} 0 & 0 & 1 & 0 & -\frac{7}{2} & -6 \\ 0 & 0 & 0 & 0 & \frac{1}{2} & 1 \end{bmatrix} \quad \begin{matrix} \text{I} \\ -5 * \text{I} + \text{II} \end{matrix}$$

4. Schritt:

Das (–5)fache der ersten Zeile der Untermatrix wird zur zweiten addiert.

Unter der eben erhaltenen Eins steht nun eine Null.

Algorithmenvisualisierung (AV)

- Darstellung der Dynamik von Datenstrukturen und Algorithmen, meist mittels Computer
- Frühe Visualisierungsformen: Flussdiagramme
- Erste Filme ab ca. 1966: vor allem „Sorting Out Sorting“ von Baecker et al. (1981)
- Erstes wichtiges AV-System: BALSA (1984) & BALSA-II (1988) von Brown et al.
- „Über 150 AV-Prototypen“ (Price et al.) – Stand: 1998
- Grundlegende Referenzen: “Software Visualization”
  - MIT Press, 1998
  - Springer, 2002
- Thema auf Konferenzen und Workshops

Einige Designfragen

- 2D oder 3D?
- Welche Kontrollelemente?
  - Pause?
  - Sprünge vorwärts / ans Ende?
  - Sprünge zurück / an den Anfang?
  - Geschwindigkeitskontrolle?
  - Vergrößerungsfaktor?
- Visuelle Gestaltung
  - „puristisch schlicht aber klar“ vs. „bunt und lebendig“?
  - Mehrfache Codierung?
  - Für Nutzergruppe wirklich verständlich?
  - Lernfördernd?

Einsatz in der Lehre

- Befragung internationaler Fachdidaktiker Informatik:
  - Fast alle glauben: „Einsatz fördert das Verständnis“
  - Fast alle sagen: „Ich selbst nutze es nicht“
- Hauptargumente: der Aufwand zum…
  - Finden guter Beispiele (93%)
  - Erlernen der Systeme (90%)
  - Entwerfen von Inhalt (90%)
  - Finden / Einarbeitung in effektive Entwicklungswerkzeuge (83%)
  - Anpassen der Inhalte zum Lernstil oder Kursinhalt (79%)

Lerneffektivität

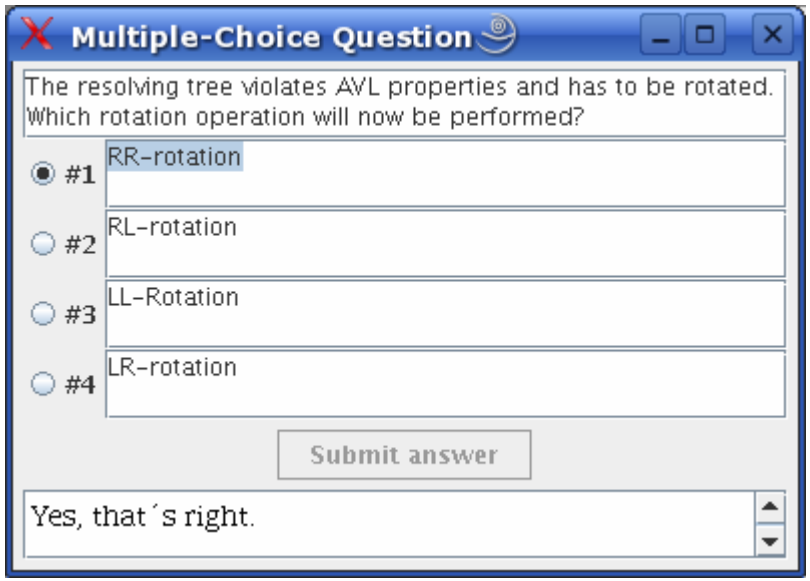
- Evaluationen ergeben weitestgehend:
  - Reines Betrachten hilft nicht oder nur kaum
- Schon Konfuzius sagte (sinngemäß) „ich sehe und vergesse, ich mache und verstehe“
- Also:
  - aktivierende Lernelemente einbetten!
  - Betrachten ist fast komplett passiv
    - Vielleicht von der „Pause“-Taste abgesehen…

Aktive Einbeziehung

- Arbeitsgruppe der ITiCSE 2002: „Engagement Taxonomy“
- Sechs Stufen zum Einsatz von AV in der Lehre & im Lernen:
  1. Kein Einsatz (wie gesagt: 90% der Lehre & Selbststudium!)
  2. Betrachten (nur minimale Interaktion für Steuerung)
  3. Fragen beantworten (meist Vorhersage nächster Schritt / Status)
  4. Anpassung von Inhalten („zielgerichtete Erstellung, so dass …“)
  5. Entwerfen eigener Inhalte
  6. Präsentation & Diskussion auf Basis AV
- Hypothesen:
  - Kein signifikanter Lernunterschied zwischen Stufe 1 & 2
  - Messbarer Lernunterschied zwischen Stufen ab Stufe 2
- Leider noch nicht hinreichend schlüssig bewiesen / widerlegt
- Problem: Seiteneffekte etc. sind auszuschließen

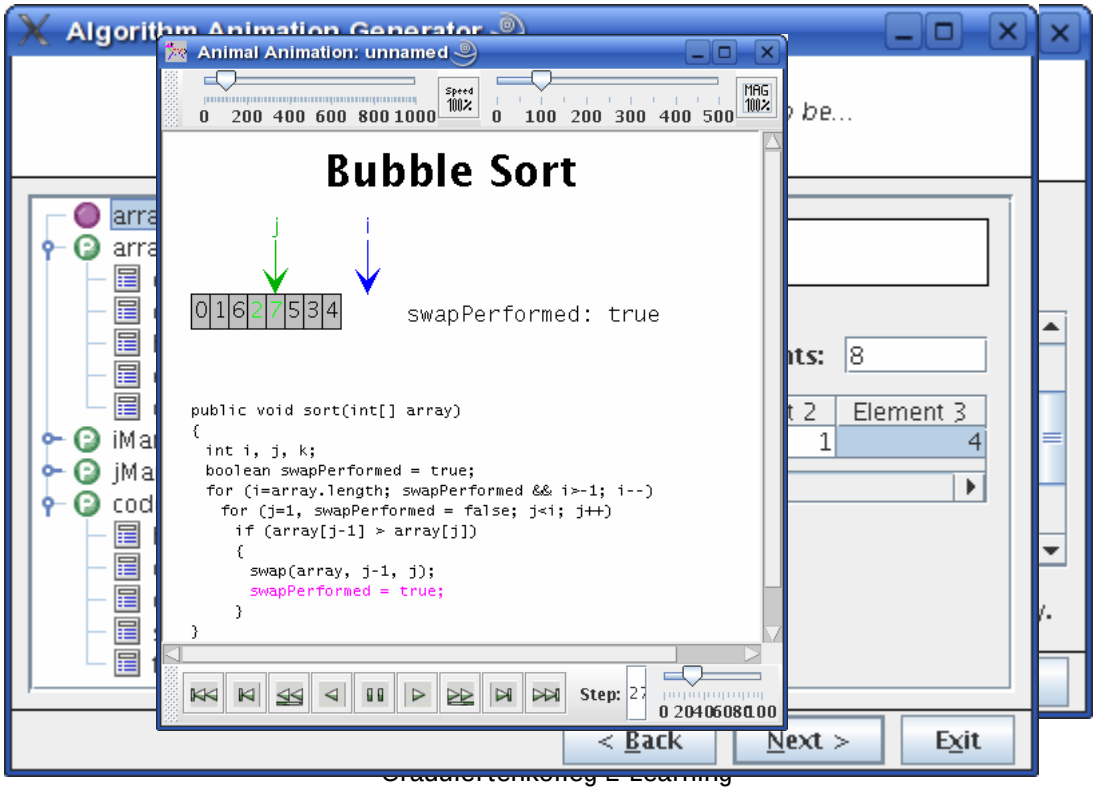
Ansätze zur Umsetzung: Beantwortung von Fragen

- „Interactive Prediction“
- Komponenten unterstützen Autor und Lerner weitgehend
- Direkte Rückkopplung
  - Frage und Antwort-Spiel
  - Bewertung



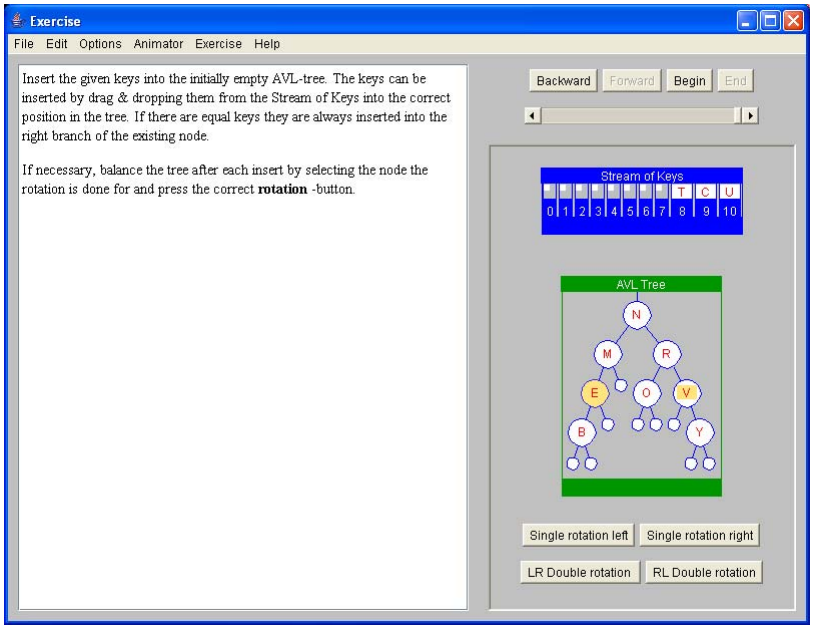
Ansätze zur Umsetzung: Anpassung von Inhalten

- Anpassung von Inhalten:
  - Bereitstellung von Generatoren
  - Nutzer kann „raten“ – dann dauert es aber potentiell lange
  - Nachdenken (Rückkopplung Nutzer / Algorithmus) beschleunigt es



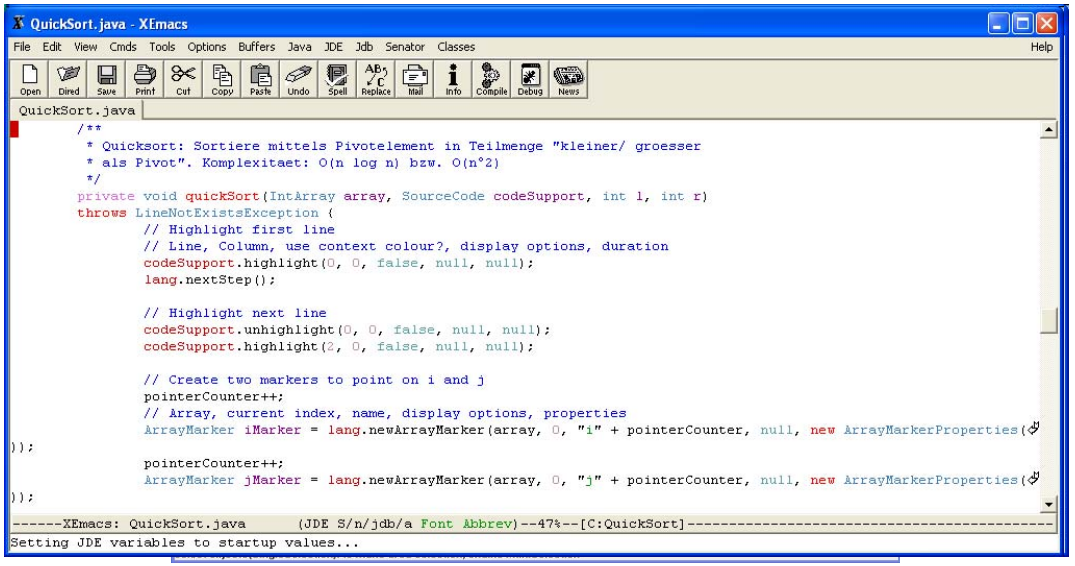
Ansätze zur Umsetzung: Anpassung von Inhalten

- Sonderform: “Algorithmensimulationsübung”
  - Algorithmus vorgegeben, Daten werden vor- oder eingegeben
  - Nutzer muss Algorithmus ausführen
  - System kommentiert Ausführung bei Fehlern
- Direkte Rückkopplung bei Fehlern



Ansätze zur Umsetzung: Entwerfen eigener Inhalte

- Für „Einsteiger“ mit grafischer Nutzeroberfläche
- Für „Fortgeschrittene“ mit Skriptnotation (Textdatei)
- Für „Java-Programmierer“ mit Java-API
- Rückkopplung beim Nachdenken…
  - „wie war das noch einmal genau?“
  - „wie stelle ich es verständlich dar?“



Ansätze zur Umsetzung: Präsentation & Diskussion

- Fragen und Annotationen des Publikums sammeln
- Pro Animationsschritt auswerten: was wurde wo gefragt?
- In das Re-Authoring aufnehmen
- Direkte Rückkopplung mit Publikum
- Dann Rückkopplung mit Inhalten im Redesign

- Hier gibt es noch *wenig bis keine* Forschung im Gebiet AV…

Hypertextbooks

- Kombination der Aktivierung mit Lernmaterialien
  - *Textbook*: „die Bibel der Vorlesung“
  - *Hyper*medien: dynamische Verlinkung
  - Dazu: dynamische Inhalte (z.B. AV!), dynamische Navigation, …
- Indirekte Rückkopplung: Artefakt passt sich dem Nutzer an

Zusammenfassung

- Visualisierung “hilft” vermutlich nur, wenn Nutzer aktiv wird
- Aktivierende Elemente regen gleichzeitig Rückkopplung an
- Es handelt sich meist um schnelle Rückkopplung
  - Das System regt den Nutzer zum Handeln an (Fragen, Anpassen)
- Iterative Rückkopplung findet sich vor allem beim Anpassen
- Bei Erstellung und Präsentation / Diskussion kann auch Meta-Rückkopplung auftreten

- Vielen Dank für die Aufmerksamkeit!
  - Weiß jemand, wie es gerade im Spiel steht?