Visualisierung von Algorithmen und Datenstrukturen

Karsten Weicker

15. Oktober 2018

1 Zielbestimmung

Für den Einsatz in Lehrveranstaltungen sollen Datenstrukturen und darauf aufsetzende Algorithmen visualisiert werden. Für dieses Projekt existiert bereits eine Software, die zwingend weiter entwickelt werden muss.

2 Produkteinsatz

In der Vorlesung, d.h. u.U. nur mit einer Auflösung von 600×480. Grundsätzlich sollte die Oberfläche so von der Logik und der Datenhaltung getrennt sein, dass in einem späteren Projekt ein Server mit verschiedenen Frontends, z.B. auch als Android-App möglich sind.

3 Erweiterung des Produkts

- alle Datenstrukturen und Algorithmen aus ADS – es fehlt:
  - sortiertes Löschen/Einfügen in Feld und Liste
  - Sortieralgorithmen: Straight-Mergesort, Countingsort, Heapsort, Bottom-Up-Heapsort, Quicksort-Varianten
  - Algorithmen für die Skip-Liste
  - Tiefensuche mit Kantenklassifikation, Breitensuche
  - nur ganzzahlige Gewichte in Graphen
  - Dijkstra/Prim: Darstellung des resutlierenden Baums im Graph
  - Rundreise-Algorithmen
  - Operationen für Heaps
- Bei rekursiven Algorithmen oder Algorithmen mit mehreren Unteralgorithmen: Darstellung des Laufzeitstapels
- Layout-Unterstützung für Graphen
- bei Bäumen: Anzeige von Balance-Eigenschaften

- zusätzlich: Rot-Schwarz-Bäume, B-Bäume, Edmonds-Karp-Algorithmus
- Für Felder und Listen: Wechsel zwischen graphisch visualisierter Sicht und Feldsicht (vor allem auch im Ablauf von Sortieralgorithmen)
- Android-App als Client für App als Microservice

4 Ursprüngliche Produktbeschreibung

Hier ist die Beschreibung, die schon zu einem großen Teil (mit einem Schwerpunkt auf Graphen und den Dijkstra-Algorithmus) umgesetzt wurde. Das Projekt ist in Java mit einer GUI in JavaFX realisiert.

- Einzelne Visualisierungskomponenten für Datenstrukturen werden benötigt. Dabei handelt es sich um die folgenden:
  - verkettete Liste
  - Felder
  - binäre Bäume
  - optional: AVL-Bäume
  - optional: Splay-Bäume
  - optional: Skip-Liste
  - Graphen (mit unterschiedlichen Sichten: in planarer Darstellung, Adjazenzliste, etc.
  - Hash-Tabellen
  - binärer Heaps (mit den Sichten: Baum und Array)
- Die Objekte der Datenstrukturen können auf vielfältige Art initialisiert werden (0, zufällig, aus csv-Datei, interaktiv . . .)
- Per Mausklick im Objekt-Baum oder auf die visualisierte Darstellung können Basis-Operationen auf der Datenstruktur ausgeführt werden bzw. der Inhalt geändert werden.
- In den Container-Datenstrukturen sollen über einen Farbstatus der einzelnen Elemente weitere Informationen visualisierbar sein.
- Die Geschwindigkeit der Animation ist global einstellbar. Operationen werden schrittweise auf der Datenstruktur durchgeführt, dabei zeigen Pfeile an, an welcher Stelle etwas passiert oder welche Veränderung vorgenommen wird.
- Neben den einzelnen Objekten von Datenstrukturen können fest einprogrammierte Algorithmen auf den Objekten durchgeführt werden. Diese sollen auf hoher Abstraktionsebene textuell dargestellt werden – z.B. Dijkstra-Algorithmus:

```
1:  Initialisierung
2:  while (unbearbeiteter Knoten vorhanden) {
3:     entnehme Knoten mit kleinstem Abstand
4:     aktualisiere benachbarte Knoten
5:  }
```

Ein solcher Algorithmus kann schrittweise bzw. mit Halt an setzbaren Breakpoint abgearbeitet werden. Dabei werden die Änderungen auf den involvierten Datenstrukturen angezeigt – hier: Graph, Prioritätswarteschlange (Heap), evtl. Zugriffsdatenstruktur auf Prioritätswarteschlange, Tabelle mit Abständen, Tabelle mit Vorgängern. Der Dijkstra-Algorithmus soll in jedem Fall bereits umgesetzt werden.

- Durch modulare Architektur und einfache Schnittstellen soll das Programm um weitere Datenstrukturen und Algorithmen erweiterbar sein.