Algorithmik

Sebastian Iwanowski
FH Wedel

6. Vorlesungswoche

Algorithmik 6

Implementierung von Dictionaries

Ein Dictionary ist eine Datenstruktur für mit einem Schlüssel vergleichbare Elemente,
welche die Funktionen member (key), insert (key, newdata) und delete (key) zur Verfügung stellt

B-Bäume: Nachfahren von (2,3)-Bäumen

Ein B-Baum ist äquivalent zu einem (b/2,b)-Baum

Anwendung in der Praxis: Festplattenzugriffe

Alle 3 Dictionary-Funktionen Laufzeit Θ(log n) w.c. und a.c.

Referenzen zum Vertiefen:

Cormen Kap. 18

Algorithmik 6

Optimale binäre Suchbäume

Details zum Algorithmus von Bellman:

Algorithm 3: [Bellman, 1957] Iterative Suche nach dem optimalen Suchbaum T.

1: **for** $i = 0, \ldots, n$ **do**
2: $\quad w_{i+1,i} = q_i$
3: $\quad c_{i+1,i} = 0$
4: **end for**
5: **for** $k = 0, \ldots, n-1$ **do**
6: $\quad$ **for** $i = 1, \ldots, n-k$ **do**
7: $\quad\quad j = i + k$
8: $\quad\quad$ Bestimme m mit $i \leq m \leq j$, so dass $c_{i,m-1} + c_{m+1,j}$ minimal ist.
9: $\quad\quad r_{i,j} = m$
10: $\quad\quad w_{i,j} = w_{i,m-1} + w_{m+1,j} + p_m$
11: $\quad\quad c_{i,j} = c_{i,m-1} + c_{m+1,j} + w_{i,j}$
12: $\quad$ **end for**
13: **end for**

Bedeutung von i, j:

begrenzt die Indizes der Daten, für die Suchbaum erstellt wird

Bedeutung von k:

k+1 ist die Anzahl der Elemente, die im Suchbaum enthalten sind

Korrektheitsbeweis:

Die beiden Teilbäume eines optimalen Suchbaums sind auch optimal (Lemma 3.3.5)

Konstruktionszzeit: $O(n^3)$ (leicht beweisbar)

Verbesserung: $O(n^2)$ (in Knuth 3 detailliert beschrieben: Beschleunigung von Schritt 8)

Referenzen zum Erarbeiten:

Skript Alt S. 65 – 70 (Kap. 3.3) andere Referenzen eher unübersichtlicher

Algorithmik 6

Graphenalgorithmen: minimal spannender Baum

Algorithmus von Kruskal (Einfache Version):

Konstruktion eines minimalen spannenden Baums für ein beliebiges G:

- Beginne mit dem leeren Wald W, bestehend aus keiner Kante.
- Wiederhole für alle Kanten $e_1, e_2, ..., e_m$ des Graphen G (Reihenfolge sortiert):
  Untersuche, ob e_i zu W hinzugefügt werden kann,
  sodass W weiterhin kreisfrei bleibt:
  Falls ja, füge e_i zu W hinzu.
  bis W aus n-1 Kanten besteht (n sei die Anzahl der Ecken von G).

Satz: Der so konstruierte Wald W ist ein minimaler spannender Baum für G.

Hausaufgabe (zum 29.05.): Beweisen Sie diesen Satz durch vollständige Induktion über die Anzahl der Kanten des bereits konstruierten Waldes

Laufzeit: $O(m \log m + n^2)$ (n^2 wegen Bestimmung der Zusammenhangskomponente)

Referenzen zum Nacharbeiten und Vertiefen:

Skript Diskrete Mathematik 6, Folien 2,3,4,8,11,12,13 (graphentheoretische Grundlagen)
Turau Kap. 2.4 (Grundlagen), 3.6.1 (Kruskal)
Cormen Kap. 23

Algorithmik 6

Graphenalgorithmen: minimal spannender Baum

Algorithmus von Kruskal (Effiziente Version):

Algorithm 5: Algorithmus von Kruskal

Initialisierung: $T := \emptyset$, $Q :=$ alle Knoten, $VS := \{V_1\}, ..., \{V_n\}$
while $|VS| > 1$ do

$e = (v, w)$ ist die Kante mit minimalen Kosten aus Q
entferne e aus Q
$V = FIND(v)$
$W = FIND(w)$
IF$(V \neq W)$ THEN

$UNION(V, W)$
$T := T \cup \{e\}$

Laufzeit: O(m log m)

Entscheidende Verbesserung: Union-Find-Struktur für Partitionen S = $\{S_1,...,S_k\}$ in einem Universum aus n Elementen

O(log n) **`Find (v)`** gibt eindeutiges Referenzobjekt zurück aus dem S_i, in dem sich v befindet.
O(1) **`Union (v,w)`** vereinigt die S_i und S_j, in denen sich v bzw. w befinden.

Hausaufgabe (zum 29.05.): Implementieren Sie eine effiziente Union-Find-Struktur in einer Sprache Ihrer Wahl

Referenzen zum Nacharbeiten und Vertiefen:

Skript Alt, Kap. 4.3 (Kruskal), Kap. 3.2 (S. 56 ff., Union-Find-Problem)