10. Tutorium - Algorithmen I

Nina Zimbel

24.06.2015

Heute:

- (a,b)-Bäume
- noch einmal zur Übungsklausur
- Breitensuche, Tiefensuche
- Dijkstra

Aufgabe 4 - Dynamische Arrays

Welche der folgenden Operationen eines unbeschränkten/dynamischen Arrays haben im schlimmsten Fall (ohne Amortisierung) $\Theta(n)$ Zeitkomplexität?
pushBack, Folge von n pushBack, Elementzugriff, Abfrage der Größe

Geben Sie die amortisierte Laufzeit der Operation pushBack an und zeigen Sie dies per amortisierter Analyse mit der Bankkontomethode. Nehmen Sie an, dass die Kapazität des Arrays verdoppelt wird, sobald sie nicht mehr ausreichend ist, und dass nur pushBack- Operationen ausgeführt werden.

Aufgabe 4 - Dynamische Arrays

Welche der folgenden Operationen eines unbeschränkten/dynamischen Arrays haben im schlimmsten Fall (ohne Amortisierung) $\Theta(n)$ Zeitkomplexität?
pushBack, Folge von n pushBack, Elementzugriff, Abfrage der Größe

- pushBack, Folge von n pushBack (wenn Arraygröße in O(n))

Geben Sie die amortisierte Laufzeit der Operation pushBack an und zeigen Sie dies per amortisierter Analyse mit der Bankkontomethode. Nehmen Sie an, dass die Kapazität des Arrays verdoppelt wird, sobald sie nicht mehr ausreichend ist, und dass nur pushBack- Operationen ausgeführt werden.

Aufgabe 4 - Dynamische Arrays

Welche der folgenden Operationen eines unbeschränkten/dynamischen Arrays haben im schlimmsten Fall (ohne Amortisierung) $\Theta(n)$ Zeitkomplexität?
pushBack, Folge von n pushBack, Elementzugriff, Abfrage der Größe

- pushBack, Folge von n pushBack (wenn Arraygröße in O(n))

Geben Sie die amortisierte Laufzeit der Operation pushBack an und zeigen Sie dies per amortisierter Analyse mit der Bankkontomethode. Nehmen Sie an, dass die Kapazität des Arrays verdoppelt wird, sobald sie nicht mehr ausreichend ist, und dass nur pushBack- Operationen ausgeführt werden.

- Die Operation pushBack hat amortisiert konstante Laufzeit ($O(1)$).
- Beweis mit Bankkontomethode (siehe Tafel)

Aufgabe 8 - Entwurf einer Datenstruktur

Aufgabe 8. (*Entwurf einer Datenstruktur*) [8 Punkte]

Eine Datenstruktur D soll Paare der Form $(Schluessel, Wert)$ speichern (sowohl *Schluessel* und *Wert* seien Zahlen aus $\mathbb{Z}$). Paare (x,c) und (y,c') mit $x = y$ dürfen in D **nicht gleichzeitig** vorkommen. Weiter soll D folgende Operationen mit jeweils gegebenem Laufzeitverhalten unterstützen (n bezeichne dabei die Anzahl der in D enthaltenen Paare):

- $insert(x : Schluessel,\ c : Wert)$
  fügt (x,c) in D ein. Ist schon ein Paar (y,c') mit $y = x$ in D vorhanden, so wird D nicht verändert. Der Zeitbedarf sei erwartet $O(\log n)$.

- $removeMin : Schluessel \times Wert$
  entfernt aus D ein Paar (x,c) mit **minimalem Wert** c und liefert das entfernte Paar als Ergebnis zurück. Der Zeitbedarf sei erwartet $O(\log n)$.

- $contains(x : Schluessel) : boolean$
  stellt fest, ob D ein Paar (y,c) mit $y = x$ enthält. Der Zeitbedarf sei erwartet $O(1)$.

Anwendungsbedingt sei bekannt, dass in D zu jedem Zeitpunkt höchstens m Paare gleichzeitig vorhanden sind, d. h. es gilt stets $n \leq m$. Über die Beschaffenheit der auftretenden Paare wisse man im Voraus aber nichts.

a. Skizzieren Sie, wie Sie diese Datenstruktur und die drei beschriebenen Operationen realisieren würden. [6 Punkte]