6. Aufgabenblatt zur Vorlesung

Höhere Algorithmik

WiSe 2016/17

Wolfgang Mulzer, Boris Klemz

Abgabe bis zum 2. Dezember 2016, 12 Uhr, in den Briefkasten neben Raum 111

Aufgabe 1 Caching *10 Punkte*

(a) In der Vorlesung wurde behauptet, dass man jede Ersetzungsstrategie für das Offline-Caching-Problem in eine *reduzierte* Ersetzungsstragie umwandeln kann, die Anzahl der Hauptspeicherzugriffe zu erhöhen. Dabei bedeutet reduziert, dass ein Datum erst dann aus dem Hauptspeicher in den Cache geladen wird, wenn es auch wirklich angefragt wird.

Geben Sie einen detaillierten Beweis für diese Behauptung.

(b) Normalerweise findet Caching in einem Online-Setting statt, bei dem die Zugriffsfolge nicht im Voraus bekannt ist. Eine beliebte Heuristik ist die LRU-Regel (Least-Recently-Used). LRU entfernt immer das Datum aus dem Cache, dessen letzter Zugriff am weitesten zurück liegt. Zeigen Sie, dass es beliebig lange Zugriffsfolgen gibt, bei denen LRU k mal so viele Cache-Misses verursacht wie die Furthest-in-the-Future-Regel aus der Vorlesung, welche die gesamte Zugriffsfolge kennt. Hierbei bezeichnet k die Cachegröße.

Sie können annehmen, dass der Cache anfangs die Elemente mit Adressen $1, 2, \ldots, k$ enthält, und dass die ersten Speicherzugriffe der Folge an die Adressen $1, 2, \ldots, k$ gehen.

Aufgabe 2 Union-Find *10 Punkte*

Betrachten Sie eine Folge von **Union** und **Find** Operationen der Länge m. Nehmen Sie an, dass wir mit der Partition $\{\{1\}, \{2\}, \ldots, \{n\}\}$ beginnen und dass wir den Algorithmus aus der Vorlesung mit Union-By-Rank und Pfadkompression verwenden. Zeigen Sie:

(a) Werden alle **Union**-Operationen vor allen **Find**-Operationen durchgeführt, so ist die Gesamtlaufzeit $O(n + m)$.

Hinweis: Ordnen Sie die Kosten jeder bei einem **Find** durchlaufenen Kante, bis auf die letzte, einer **Union**-Operation zu.

(b) Wenn alle **Find**-Operationen auf Mengen mit mindestens $\log n$ Elementen durchgeführt werden, so ist die Gesamtlaufzeit $O(n + m)$.

Aufgabe 3 Bitmaps *10 Punkte*

Sei n eine natürliche Zahl. Ein $n \times n$ *Bitmap* ist ein Array $B[1 \ldots n, 1 \ldots n]$ vom Typ `Boolean`, welches ein Schwarz-Weiß-Bild darstellt. Der Eintrag `true` bedeutet, dass

das Pixel an der entsprechenden Stelle schwarz ist, der Eintrag `false` bedeutet, dass das entsprechende Pixel weiß ist. Zwei Pixel sind *benachbart*, wenn sie horizontal oder vertikal nebeneinander liegen.

(a) Entwerfen und analysieren Sie einen effizienten Algorithmus, der eine größte Zusammenhangskomponente von schwarzen Pixeln in B bestimmt.

(b) Beschreiben und analysieren Sie eine Datenstruktur, welche die folgende Funktion unterstützt: `schwärze`(i, j) färbt das Pixel an der Stelle $B[i, j]$ schwarz und gibt die Größe einer größten Zusammenhangskomponente von schwarzen Pixeln zurück. Zu Beginn sind alle Pixel weiß.

Die Gesamtlaufzeit für jede Folge von m Aufrufen von `schwärze` sollte so gering wie möglich sein.

(c) Was ist die worst-case Laufzeit für einen Aufruf Ihrer Funktion `schwärze` aus (b)?

Aufgabe 4 Matroide *freiwillig, 10 Zusatzpunkte*

Sei S eine endliche, nichtleere Menge, und sei $\mathcal{I} \subseteq 2^S$ eine nichtleere Menge von Teilmengen von S. Das Paar $M = (S, \mathcal{I})$ heißt *Matroid*, wenn gilt: (i) **Vererbung**: Wenn $A \in \mathcal{I}$ und $B \subseteq A$, so ist $B \in \mathcal{I}$. Insbesondere ist $\emptyset \in \mathcal{I}$. (ii) **Austausch**: Wenn $A, B \in \mathcal{I}$ und $|A| < |B|$ ist, dann existiert ein $x \in B \setminus A$, so dass $A \cup \{x\} \in \mathcal{I}$ ist. Die Elemente von $\mathcal{I}$ heißen die *unabhängigen Mengen* von M.

(a) Eine inklusionsmaximale unabhängige Menge heißt *Basis* von M. Zeigen Sie, dass alle Basen von M die gleiche Anzahl von Elementen haben.

(b) Sei $w : S \to \mathbb{R}^+$ eine Gewichtsfunktion, die jedem Element aus S ein positives Gewicht zuordnet. Wir möchten eine Basis von M mit maximalem Gewicht finden, wobei das Gewicht einer Teilmenge als die Summe der Einzelgewichte definiert ist. Dies geht mit einem gierigen Algorithmus: Sortiere die Elemente aus S absteigend nach ihrem Gewicht. Setze $B := \emptyset$. Gehe S in der Reihenfolge durch und füge das nächste Element zu B hinzu, wenn B dadurch unabhängig bleibt. Gib die resultierende Menge B zurück.

Zeigen Sie, dass der gierige Algorithmus eine Basis von maximalem Gewicht bestimmt. Was können Sie zur Laufzeit sagen?

Hinweis: Verwenden Sie ein Austauschargument, um zu zeigen, dass sich jede optimale Basis in die gierige Basis überführen lässt, ohne dass sich dabei das Gewicht verringert.