Übungsblatt 3

Algorithmen und Datenstrukturen (WS 2013, Ulrike von Luxburg)

Präsenzaufgabe 1 (Quicksort) Wählen Sie in dieser Aufgabe für den Quicksort-Algorithmus stets das an der mittleren Position (bei gerader Länge links der Mitte) stehende Element als Pivotelement.

(a) Führen Sie den Quicksort-Algorithmus ($\approx$ Folie 283) auf folgendem Array durch:

$$12,\ 10,\ 21,\ 8,\ 5,\ 20,\ 8,\ 4,\ 1,\ 8,\ 9.$$

(b) Wann ist ein Pivotelement "gut", wann "schlecht"?

(c) Finden Sie eine Eingabe der Größe $n = 7$, welche die worst-case Laufzeit erzwingt, und führen Sie Quicksort darauf aus.

(d) Wie (c), aber für die best-case Laufzeit.

(e) Aus der Vorlesung ist bekannt, dass der randomisierte Quicksort worst-case Laufzeit $\mathcal{O}(n^2)$ und average-case Laufzeit $\mathcal{O}(n \log(n))$ hat. Warum könnte man diesen gegenüber einem deterministischem Algorithmus mit denselben Laufzeiteigenschaften bevorzugen?

Präsenzaufgabe 2 (Hashing) Wir haben in der Vorlesung Hashing mit verketteten Listen kennengelernt. Eine Variante ist die "offene Adressierung". Hierbei kann jede Position in der Hashtabelle höchstens *ein* Element speichern. Kollisionen werden wie folgt behandelt: Zu Beginn jeder INSERT-Operation eines neuen Elements mit Key k wird ein Kollisionszähler $i = 0$ gesetzt, welcher von der Hashfunktion $h(k, i)$ berücksichtigt wird: Ist Position $h(k, i)$ bereits belegt, dann wird i inkrementiert und die nächste Position $h(k, i)$ getestet, solange bis ein freier Platz gefunden wurde. Sei H nun eine leere Hashtabelle der Größe $s = 7$ mit den Positionen $0, \ldots, 6$. Fügen Sie die folgenden Elemente der Reihe nach in H ein: $9, 12, 2, 31, 10, 4$.

(a) ...unter Kollisionsbehandlung mittels verketteter Listen und $h(k) = k \bmod 7$

(b) ...mittels offener Adressierung und "linearer Sondierung": $h(k, i) = (k + i) \bmod 7$

(c) ...mittels offener Adressierung und "doppeltem Hashing": $h(k, i) = (h_1(k) + i \cdot h_2(k)) \bmod 7$ für $h_1(k) = k \bmod 7$ und $h_2(k) = 1 + (k \bmod 6)$

(d) Wieviele Elemente können maximal in einem Hashtable mit verketteten Listen bzw. mit offener Adressierung gespeichert werden?

(e) Beschreiben Sie die SEARCH-Operation zum Finden eines Elements anhand seines Keys.

(f) Sei m die Anzahl einzufügender Elemente. Was sind Vor-/Nachteile von $m \gg s$ und $m \ll s$?

(g) Das Verhältnis m/s heißt auch Lastfaktor. Welcher ungefähre Wert könnte sinnvoll sein?

(h) Wie funktioniert die REMOVE-Operation für Hashing mit verketteten Listen?

(i) Wie könnte eine REMOVE-Operation bei offener Adressierung konzipiert sein?

Präsenzaufgabe 3 (Stabilität)
Denken Sie sich ein Verfahren aus, mit welchem *jedes* vergleichsbasierte Sortierverfahren stabilisiert werden kann. Untersuchen Sie den zusätzlichen Speicheraufwand und die resultierende Gesamtlaufzeit.

Hausaufgaben zum 20. November, 9:00 (Vorlesungsbeginn).

Aufgabe 1 (Hashing (1+1+2+2)) Sei $\mathbb{N} := \{0, 1, 2, \ldots\}$. Wir schreiben $a\mathbb{N}+b := \{a \cdot t + b \mid t \in \mathbb{N}\}$ für $a, b \in \mathbb{N}$, beispielsweise $5\mathbb{N} + 2 = \{2, 7, 12, 17, \ldots\}$. Geben Sie in dieser Schreibweise jeweils die Menge aller Keys an (inkl. Begründung), die auf der letzten Position (Index 10) in einer Hashtabelle der Größe 11 kollidieren, wenn folgende Hashfunktion verwendet wird:

(a) $h(k) = k \bmod 11$

(b) $h(k) = 2k \bmod 11$

(c) $h(k) = (k^2 + 10) \bmod 11$

(d) $h(k) = (3^k - 1) \bmod 11$

Aufgabe 2 (Lower Bound (1+1)) Zeigen Sie ohne Anwendung der Stirling'schen Formel, dass im Beweis der worst-case Laufzeit für vergleichsbasiertes Sortieren gilt: $\log(n!) = \Theta(n \log n)$.
Tipp: Vergleichen Sie $n!$ einmal gegen n^n und einmal gegen $(n/2)^{n/2}$.

Aufgabe 3 (Quicksort (3+1+1*)) Es werde Quicksort auf einem Array aus n Zahlen ausgeführt.

(a) Betrachten Sie eine Quicksort-Variante, bei welcher als Pivotelement stets der Median der aktuellen Eingabesequenz ermittelt und verwendet wird (in worst-case Laufzeit $\mathcal{O}(n)$ mittels "Median-of-Median-Algorithmus"). Bestimmen Sie eine scharfe Schranke für die asymptotische worst-case Laufzeit.

(b) Warum vermuten Sie wird diese Variante in der Praxis nicht bevorzugt verwendet?

(c) Betrachten Sie eine Quicksort-Variante, welche als Pivotelement dasjenige Element der Eingabesequenz A verwendet, welches am nächsten am arithmetischen Mittelwert der Zahlen aus A liegt. Kann auf diese Weise die asymptotische worst-case Laufzeit von $\mathcal{O}(n^2)$ unterschritten werden? Beweisen oder widerlegen Sie. (dieser Aufgabenteil ist etwas kniffliger)

Aufgabe 4 (Zufällige Auswahl (2+2+2)) In randomisierten Algorithmen können wir in einem Zeitschritt eine faire Münze werfen, und erhalten so mit gleicher Wahrscheinlichkeit entweder den Wert 0 oder 1. Sei nun A ein Array der Länge $n > 0$:

(a) Sei $n = 2^k$. Nutzen Sie derartige Münzwürfe um ein zufälliges Element aus A zu wählen, wobei jedes Element mit exakt derselben Wahrscheinlichkeit $\frac{1}{n}$ gewählt werden soll (Begründung!). Die Anzahl notwendiger Münzwürfe soll $\mathcal{O}(\log n)$ garantieren (mit 100% Wahrscheinlichkeit).

(b) Erweitern Sie Ihre Lösung auf beliebige $n > 0$ derart, dass die Anzahl Münzwürfe immernoch $\mathcal{O}(\log n)$ garantiert. Dabei begnügen wir uns aber damit, dass die Gleichverteilung nur ungefähr erreicht wird (idealerweise beliebig genau approximierbar).

(c) Erweitern Sie Ihre Lösung aus (a) auf beliebige $n \in \mathbb{N}_{>0}$ derart, dass jedes Element weiterhin mit *exakt* derselben Wahrscheinlichkeit geliefert wird (Begründung!). Die Anzahl Münzwürfe darf diesmal zufällig schwanken, soll aber im Erwartungswert $\mathcal{O}(\log n)$ sein. Tipp: Wenn ein Ereignis mit Wahrscheinlichkeit $p > 0$ eintritt, dann ist die erwartete Anzahl notwendiger unabhängiger Versuche bis zum ersten Eintritt gleich p^{-1} (geometrische Verteilung).

Aufgabe 5 (Berechnungsbaum (3+1)) Seien a, b, c drei vergleichbare Elemente, und das Array $[a, b, c]$ die Eingabe für MERGESORT.

(a) Konstruieren Sie den Berechnungsbaum (gemäß ≈ Folie 240), welcher alle von MERGESORT durchgeführten Vergleichsabläufe für alle möglichen Wertordnungen von a, b, c beinhaltet. Tipp: An der Wurzel steht "$b \leq c$".

(b) Wieviele Blätter hätte dieser Baum, wenn das Eingabe-Array stattdessen $[a, b, c, d]$ wäre? Wieviele für $[a, b, c, d, \ldots, w, x, y, z]$?

Aufgabe 6 (Algorithmusentwurf (2))
Entwerfen Sie einen deterministischen Algorithmus GETSORTEDMINS(A, k), der ein unsortiertes Array mit n Zahlen als Eingabe erhält, und die kleinsten $k \leq n$ Zahlen daraus in aufsteigend sortierter Reihenfolge zurückliefert. Die Gesamtlaufzeit soll $\mathcal{O}(n \log k)$ betragen. Begründen Sie.