9. Aufgabenblatt zur Vorlesung WS 2012/13

Helmut Alt

Algorithmen und Programmierung III

Bonuszettel aber allen Teilnehmern empfohlen

Abgabe 11.1.2013, 12 Uhr

Aufgabe 1 Sortieren *5 Punkte*

Man kann eine Folge vergleichbarer Objekte sortieren, indem man sie in eine zunächst leere Datenstruktur einfügt und anschließend deren Inhalt sortiert ausgibt. Beschreiben und analysieren Sie die entstehenden Sortieralgorithmen für folgende Datenstrukturen:

(a) Heap,

(b) binärer Suchbaum,

(c) AVL-Baum

Aufgabe 2 Median *6 Punkte*

Erweitern Sie folgende Wörterbuch-Datenstrukturen, so dass es effizient möglich ist, den Median der gespeicherten Schlüsselmenge (auf der eine lineare Ordnung definiert sein soll) zu finden. Beschreiben und analysieren Sie auch die nötigen Modifikationen für die anderen Wörterbuchoperationen:

(a) Skipliste

(b) Trie (lexikographische Ordnung)

(c) Hashing

(d) 2-3-Baum

Aufgabe 3 Merge *7 Punkte*

Seien $D_1, D_2 \subseteq U$ zwei Mengen von Schlüsseln aus einem linear geordneten Universum U, wobei alle Elemente in D_1 kleiner oder gleich allen Elementen in D_2 sind. `Merge`(D_1, D_2) sei die Operation der Vereinigung zweier solcher Mengen. Entwerfen und analysieren Sie möglichst effiziente Algorithmen für `Merge` bei den folgenden Datenstrukturen. Dabei soll davon ausgegangen werden, dass anfangs die beiden Mengen D_1, D_2 in solchen Datenstrukturen gespeichert sind, und es soll eine neue für `Merge`(D_1, D_2) entstehen, wobei die beiden alten zerstört werden dürfen.

(a) binärer Suchbaum,

(b) AVL-Baum

Aufgabe 4 greedy *6 Punkte*

Nehmen Sie an, dass sich eine Kunstgalerie in einem langen schmalen Gang befindet, der die Länge l hat. Die Positionen der n Kunstwerke seien durch reelle Zahlen $x_1, ..., x_n$ mit $0 \leq x_1 < x_2 < ... < x_n \leq l$ gegeben, die den Abstand vom Anfangspunkt des Ganges bedeuten. Geben Sie einen effizienten Algorithmus an, der daraus die minimale Zahl von Wächtern berechnet, die notwendig sind, alle Kunstwerke zu überwachen. Dabei kann ein Wächter alle Kunstwerke überwachen, die sich im Abstand ≤ 1 von ihm befinden. Erklären Sie, warum Ihr Algorithmus korrekt ist.

Aufgabe 5 Lempel-Ziv *10 Punkte*

Implementieren Sie den Lempel-Ziv-Kompressionsalgorithmus aus der Vorlesung. Dabei können Sie Schritt 1 "brute-force" implementieren, es ist nicht verlangt, einen Suffixbaum zu konstruieren. Testen Sie die Güte Ihrer Implementierung an einer Textdatei aus dem Internet. Zählen Sie dazu die vom Algorithmus produzierten Tripel (p, m, c) und rechnen Sie für p mit 16 Bit Speicherbedarf (`short`), für m und für c mit 8 Bit (`byte`).

Aufgabe 6 dynamische Programmierung *6 Punkte*

Es seien x und y Wörter mit $|x| = n$ und $|y| = m$. $b_{i,j}$ sei die Länge der längsten gemeinsamen (nicht unbedingt zusammenhängenden) Teilfolge von Zeichen im Suffix der Länge i von x und dem der Länge j von y für $0 \leq i \leq n, 0 \leq j \leq m$. Geben Sie einen Algorithmus der Laufzeit $O(nm)$ an, der alle $b_{i,j}$ berechnet.