Prof. Dr. Klaus Volbert

Regensburg, den 28. November 2018

Votieren und abgeben bis zum 04. Dezember 2018, 23:00 Uhr

Übungen zur Vorlesung

Algorithmen und Datenstrukturen

WiSe 2018/19

Blatt 9

Wichtige Hinweise:

– > Falls Sie bei der Bearbeitung einer Aufgabe größere Schwierigkeiten hatten und deswegen die Bearbeitung abgebrochen haben, so versuchen Sie bitte Ihre Schwierigkeiten in Form von Fragen festzuhalten. Bringen Sie Ihre Fragen einfach zur Vorlesung oder zur Übung mit!

– > Kursraum: https://elearning.uni-regensburg.de/course/view.php?id=9228

Aufgabe 1:
Ermitteln Sie die Positionen, an die die Schlüssel $61, 62, 63, 64, 65$ platziert werden, wenn eine Hashtabelle der Größe $m = 1000$ gegeben ist und die Werte mittels $h(s) = \lfloor m \cdot ((s \cdot x) \bmod_c 1) \rfloor$ mit $x = \frac{\sqrt{5}-1}{2}$ abgebildet werden.

Aufgabe 2:
Sei eine Hashtabelle der Größe m gegeben, in der n Schlüssel mittels offener Adressierung gespeichert werden sollen. Geben Sie der folgenden Formel eine sinnvolle Bedeutung:

$$P_k = \left(\frac{1}{m}\right)^k \left(1 - \frac{1}{m}\right)^{n-k} \binom{n}{k}$$

Aufgabe 3:
Implementieren Sie Hashing mit offener Adressierung in C,C++, C# oder Java anhand der Hashfunktion $\acute{h}(s) = s$ unter Verwendung folgender Varianten zur Kollisionsauflösung:

1. Lineares Probieren
2. Quadratisches Probieren mit $c_1 = 1$ und $c_2 = 3$
3. Doppeltes Hashing mit $h_1(s) = s$ und $h_2(s) = 1 + (s \bmod (m-1))$

Geben Sie jeweils die Hashtabelle der Größe $m = 11$ aus, die nach Einfügen der Werte $10, 22, 31, 4, 15, 28, 17, 88, 59$ in allen drei Varianten entsteht.

Aufgabe 4:
Betrachten Sie die beiden folgenden Varianten des Rucksackproblems:

- Eingabe: n Objekte (Werte $v_1, \ldots, v_n$, Gewichte $w_1, \ldots, w_n$), Rucksackkapazität W

- Ausgabe: Rucksackfüllung mit maximalem Wert, und zwar
  1. anteilig: $\sum_{i=1}^{n} a_i w_i \leq W$ und $\sum_{i=1}^{n} a_i v_i$ ist maximal für $a_i \in [0,1], i \in \{1,\ldots,n\}$
  2. ganzzahlig: $\sum_{i=1}^{n} a_i w_i \leq W$ und $\sum_{i=1}^{n} a_i v_i$ ist maximal für $a_i \in \{0,1\}, i \in \{1,\ldots,n\}$

Entwerfen Sie einen Algorithmus mit Laufzeit echt besser als $O(n^2)$, der Variante 1 optimal löst. Welche Laufzeit hat Ihr Algorithmus? Zeigen Sie, dass Ihr Algorithmus für Variante 2 sehr schlecht werden kann.