Prof. Dr. Klaus Volbert

Regensburg, den 11. November 2024

Votieren und abgeben bis zum 17. November 2024, 23:00 Uhr

Übungen zur Vorlesung

Algorithmen und Datenstrukturen

WiSe 2024/25

Blatt 6

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!

- > Musterlösungen werden bei Bedarf in den Übungen besprochen!

Aufgabe 1:
Implementieren Sie eine verbesserte Variante für Count Sort mit Laufzeit $T(n) = 2k + 3n$, wenn n ganze Zahlen aus dem Wertebereich $\{0, \ldots, k\}$ eingegeben werden können. Vergleichen Sie Heap Sort, Count Sort und Map Sort anhand von zufällig, gleichverteilt erzeugten Feldern unterschiedlicher Größe unter der Annahme, dass die Zufallszahlen im Wertebereich $\{1000, \ldots, 10000\}$ liegen. Stellen Sie dar, bei welchen Größen welcher Algorithmus die beste Laufzeit liefert. Betrachten Sie sowohl die Anzahl der ausgeführten Operationen und Vergleiche, als auch die real verbrauchte Rechenzeit.

Aufgabe 2:
Realisieren Sie eine Datenstruktur für verkettete Listen in C, C++, C#, Java oder Python und implementieren Sie eine Funktion Quicksort, die die Werte in der verketteten Liste mittels einer Quicksort-Variante sortiert. Bestimmen Sie Speicherplatz und Laufzeit Ihrer Implementierung.

Aufgabe 3:
Erstellen Sie ein Programm zur Ziehung der Lottozahlen mit Hilfe einer Ringliste. Fügen Sie dazu die Zahlen $1, \ldots, 49$ in eine Ringliste ein. Entnehmen Sie nun zufällig und gleichverteilt 6 Zahlen aus der Ringliste, indem Sie für eine Zahl eine zufällige, aber gleichverteilte Schrittweite in der Ringliste ermitteln, anschliessend die der Zielposition entsprechende Zahl ausgeben und zuletzt aus der Liste entfernen. Passen Sie die Schrittweite jeweils der entsprechenden Anzahl vorhandener Zahlen an.

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.