Albert-Ludwigs-Universität
Institut für Informatik
Prof. Dr. Fabian Kuhn
P. Schneider, P. Bamberger, Y. Maus

Informatik 2 - Sommersemester 2018

Übungsblatt 3

Abgabe: Montag, 14. Mai, 14:00 Uhr

Aufgabe 1: Radix Sort *(10 Punkte)*

(a) In der Vorlesung behandelten Sie den Algorithmus *Counting Sort*. Implementieren Sie eine Variante von *Counting Sort* so, dass es ein Array A mit n Elementen[1] mit Sortierschlüsseln im Bereich $\{0,\ldots,k\}$ *stabil*[2] in $\mathcal{O}(k+n)$ sortiert. Sie *können* dazu die Vorlage `CountingSort.py` von der Vorlesungswebsite benutzen. *(5 Punkte)*

(b) Benutzen Sie Ihre Implementierung von *Counting Sort* um *Radix Sort* zum Sortieren natürlicher Zahlen gemäß der Vorlesung zu implementieren. Sie *können* dazu die Vorlage `RadixSort.py` benutzen. *(5 Punkte)*

Hinweis: Die i-te Stelle a_i einer Zahl $z \in \mathbb{N}$ in n-adischer Darstellung $z = a_0 \cdot n^0 + a_1 \cdot n^1 + a_2 \cdot n^2 + \ldots$ erhalten Sie mit der Formel $a_i = (z \bmod n^{i+1}) \operatorname{div} n^i$, wobei mod die modulo-Operation und div die ganzzahlige Division ist.

Aufgabe 2: Doubly-Linked List *(10 Punkte)*

Implementieren Sie eine doppelt verkettete Liste, in welcher jedes Element einen Schlüssel, sowie zwei Zeiger auf das Element davor und das danach enthält. Dafür werden Sie zwei Klassen schreiben müssen, `ListElement` und `DoublyLinkedList`.

Die erste Klasse enthält ein Feld `key` und zusätzlich zwei Zeiger `previous` und `next` auf die Nachbarelemente in der Liste. Die zweite Klasse implementiert eine doppelt verkettete Liste aus Elementen vom Typ `ListElement`.

Auf der Vorlesungswebsite finden Sie die Strukturdateien `DoublyLinkedList.py` und `ListElement.py`, die spezifizieren, welche Operationen Ihre doppelt verkettete Liste anbieten soll. Sie *können* diese Vorlage für Ihre eigene Lösung anpassen.

Hinweis: Vergessen Sie auch hier bitte nicht alle Operationen ihrer Datenstruktur mit Tests von typischen Eingaben und Grenzfällen zu versehen, beispielsweise indem sie Unit-tests schreiben in dem Listen erstellt und Operationen darauf ausgeführt werden.

[1] In unserer Vorlage implementieren wir Elemente als Tupel aus Sortierschlüssel und Datenwert

[2] Seien $x := A[i]$ und $y := A[j]$ die Elemente an i-ter bzw. j-ter Stelle mit gleichem Sortierschlüssel $x.key = y.key$. Sei ohne Einschränkung $i < j$ und i' und j' die jeweiligen Positionen von x bzw. y nach der Sortierung. Der Sortieralgorithmus ist stabil wenn für jeden Input A und jede Wahl von Paaren x, y wie oben beschrieben $i' < j'$ ist.