Einführung in die Informatik II

28.02. und 02.03.2020

1 Radixsort

Das Sortierverfahren Radixsort kann dazu verwenden werten Mengen von Schlüsseln fester Länge aufsteigend zu sortieren. Jeder Schlüssel ist eine Folge von m Stellen.

Zum Sortieren verwendet Radixsort k Fächer, je ein Fach pro möglichem Stellenwert.

Beispiele:

Beispiel 1: Die Schlüssel sind Worte, die aus m Kleinbuchstaben (= Stellen) bestehen. Hierbei sind k=26 Fächer anzulegen (ein Fach pro Buchstabe).

Beispiel 2: Als Schlüssel werden Zahlen mit m Ziffern (= Stellen) verwendet. Dies führt zu k=10 Sortierfächern für die Ziffern 0..9.

Abweichend von der Vorlesung implementieren wir die einzelnen Fächer als Keller.

Die Vorgehensweise von *Radixsort* lässt sich wie folgt in Pseudocode darstellen, wobei sich die zu sortierenden n Schlüssel in dem Array keys befinden:

```
k Kellerfaecher anlegen
2
3
   //Alle Positionen der Schluessel von hinten durchlaufen
4
   for (pos <- m - 1 to 0 by -1) {
5
6
     //1. Phase: Verteilen des Inhalts von keys auf die Kellerfaecher
7
     for (i <- 0 to keys.length - 1) {</pre>
8
       Ablegen von keys (i) auf das Kellerfach mit dem Index,
9
       den man keys(i) an der Stelle pos entnimmt
10
11
     //2. Phase: Leeren der Kellerfaecher und Schreiben nach keys
12
13
     for (j \leftarrow k - 1 \text{ to } 0 \text{ by } -1) {
14
       Leeren des Kellerfachs mit dem Index j;
15
       dabei keys von hinten fuellen
16
     }
17 | }
```

Hinweis: Die Position pos wird (Scala-üblich) von 0 beginnend gezählt. Mit pos = 2 wird daher auf die Einerstelle einer dreistelligen Zahl (also bei m = 3) zugegriffen!

a) Wir verwenden dieses Verfahren auf Zahlen mit maximal m=3 Stellen an. Die Zahlen stehen rechtsbündig im Schlüssel und werden bei Bedarf mit führenden Nullen ergänzt. Hierbei sei keys wie folgt initialisiert: keys = Array (978, 100, 7, 391, 57, 831, 110, 470, 612, 217)

Nachfolgend finden Sie den Inhalt der Kellerfächer für die letzte Position (pos = 2) nach Abschluss der 1. Phase:

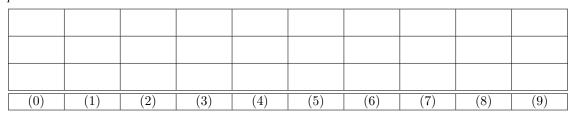
470							217		
110	831						57		
100	391	612					7	978	
(0)	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)

Nach dem übertragen der Kellerfächer (2. Phase) sieht keys wie folgt aus: keys = Array (100, 110, 470, 391, 831, 612, 7, 57, 217, 978)

Beachten Sie, dass sich sowohl beim "Einkellern" in Phase 1 als auch bei der Entnahme in Phase 2 die Reihenfolge eines Fachs umkehrt. Nach dieser zweimaligen Umkehrung stehen die Elemente in keys wieder in der gleichen Reihenfolge wie beim *Radixsort* aus der Vorlesung.

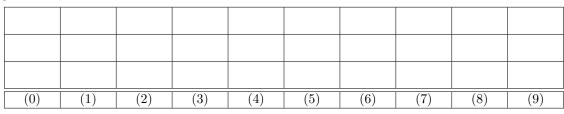
Tragen Sie nun analog für die Positionen pos = 1 und pos = 0 jeweils den Wert der Kellerfächer nach Abschluss der 1. Phase und den Wert von keys nach Abschluss der 2. Phase ein.

pos = 1:



keys = ____

pos = 0:



keys = ____

b) Welcher Komplexitätsklasse lässt sich *Radixsort* zuordnen (*m* entspricht der Länge der Schlüssel, *n* ihrer zu sortierenden Anzahl)?

```
\square \ O(n \cdot \log(n))
```

 $\square \ O(m \cdot n)$

 $\square O(m+n)$

 $\square O(n^2)$

Begründen Sie Ihre Antwort kurz in Stichworten.

c) Zur Realisierung der Kellerfächer wird der folgende Typ Stack verwendet:

```
6  def emptyStack : Stack = new Stack
7  
8  //Pruefen ob Stack leer
9  def isEmpty(stack : Stack) : Boolean = stack.s == null
```

Implementieren Sie folgende Funktionen und Prozeduren. Lösen Sie, wo geboten, eine IllegalArgument Exception mit geeignetem Fehlertext aus!

- d) Definieren Sie eine Prozedur getNumPos, welche bei gegebenen m maximalen Stellen pro Schlüssel, an der Stelle pos, den Wert von num zurückgibt. Zum Beispiel soll für m=3, pos=2, num=216 das Ergebnis 6 sein.
- e) Definieren Sie nun eine Prozedur radixsort, welche einen Array keys: Array [Int] nach dem *Radixsort*-Verfahren sortiert. Überlegen Sie sich vorab, welche Parameter dieser Prozedur zu übergeben sind.