Mathematisch-Naturwissenschaftliche Fakultät
Wilhelm-Schickard-Institut für Informatik
Datenbanksysteme · Prof. Dr. Grust

EBERHARD KARLS
UNIVERSITÄT
TÜBINGEN

Übungen zur Vorlesung
"Datenbanksysteme II"
SS 2014
Benjamin Dietrich (b.dietrich@uni-tuebingen.de)

8. Übungsblatt

Ausgabe: 28. Mai 2014 · Abgabe: 3. Juni 2014

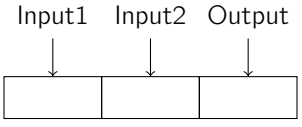
Aufgabe 1: Two-Way Merge Sort **(6 Punkte)**

Gegeben sei eine Relation, deren Tupel sich über 7 Seiten eines Heap-Files verteilen. Jede Page enthält dabei jeweils bis zu zwei einspaltige Tupel vom Typ `(INTEGER)`. Die Seiten seien wie folgt befüllt:

(3),(4)	(6),(2)	(9),(4)	(8),(7)	(5),(6)	(3),(1)	(2)

Sortieren Sies dieses File mit Hilfe des *Two-Way Merge Sort Algorithmus*. Veranschaulichen Sie das Vorgehen des Algorithmus in dem Sie:

1. Für alle Durchgänge die jeweils im Sekundärspeicher aufgebauten *runs* angeben.
2. Beispielhaft für die *merges* von 2-page runs auf 4-page runs (*Pass* 2), jeweils die Bufferbelegungen des benötigten, drei Seiten großen Hauptspeicherbereiches angeben.



Main memory buffer of Two-Way Merge Sort

Aufgabe 2: Replacement Sort (14 Punkte)

In der Vorlesung haben wir *replacement sort* als Möglichkeit kennen gelernt, um die Anzahl der initialen *runs* des External Merge Sort Algorithmus noch weiter zu reduzieren. In dieser Aufgabe sollen Sie replacement sort in Java implementieren und testen.

1. Schreiben Sie ein Java-Programm, das *replacement sort* (nur *Pass 0* des Sortieralgorithmus) implementiert. Als Parameter sollen Ihrem Programm die Größe des *current set* (in Tupeln) und der Name der zu sortierenden Datei übergeben werden. Basierend auf diesen beiden Parametern, soll Ihr Programm dann Dateien mit sortierten initialen *runs* erzeugen. (Jede Zeile einer Datei entspricht dabei einer Page mit je einem Eintrag)

   Beachten Sie hierzu unten stehenden Hinweise!

2. Nutzen Sie Ihre Implementation um exemplarisch zu zeigen, dass replacement sort initiale *runs* erzeugt, die im Durchschnitt doppelt so groß wie das *current set* sind (unter der Annahme einer Gleichverteilung der zu sortierenden Eingabewerte).

Bitte beachten Sie folgende Hinweise zur Implementation:

- Ihre Implementation soll wie folgt nutzbar sein:

  ```
  $ cd assignment08
  $ ant
  $ java -cp build Sort <buffer_size> <input_file>
  ```

  Achten Sie bitte unbedingt darauf, dass Ihr Code **fehlerfrei kompiliert**. Programme die nicht fehlerfrei kompilieren, werden mit **0 Punkten** bewertet!

- Ein Großteil der Funktionalität des Programms wurde bereits implementiert, so dass Sie sich nur um die Formulierung des eigentlichen Sortieralgorithmus kümmern müssen. Ihre Aufgabe ist es die Methode `createPass0` in der Klasse `ReplacementSort` zu implementieren. Sie nimmt eine Instanz der Klasse `BufferedReader` entgegen, welche die zu sortierende Datei darstellt, sowie die Größe des *current set*.

  ```
  /**
   * This method implements the actual replacement sort algorithm
   * @param file The input file that is to be sorted.
   * @param bufferSize The size of the sorting buffer (number of tuples in
   * current set)
   */
  public static void createPass0(BufferedReader file, int bufferSize)
    throws IOException
  ```

  Weitere Funktionen, wie etwa zum Erstellen einer Datei für einen neuen run (`createNewRun`), sind bereits vorhanden.

  Die Quellen können Sie direkt von unserer Homepage beziehen:

  http://db.inf.uni-tuebingen.de/staticfiles/teaching/ss14/db2/assignment08.zip

- Um eine unsortierte Eingabedatei mit zufälligen Strings zu erzeugen steht Ihnen das Programm `TupleGen` zur Verfügung, welches den Quellen beigefügt ist. `TupleGen` kann wie folgt genutzt werden:

  ```
  $ cd assignment08
  $ ant
  $ java -cp build TupleGen <nr_tuples> <tuple_size> <output_file>
  ```

- Bitte reichen Sie als Ergebnis das **gesamte Projekt** und *nicht* nur `ReplacementSort.java` ein!