Assignment 8

**Valeria Fionda, Mouna Kacimi,
Werner Nutt, Simon Razniewski**

Hashing

Instructions: Your assignment should represent your own effort. However, you are not expected to work alone. It is fine to discuss the exercises and try to find solutions together, but each student shall write down and submit his/her solutions separately. It is good academic standard to acknowledge collaborators, so if you worked together with other students, please list their names.
Be prepared to present your solution at the lab. If you are not able to explain your solution, this will be considered as if you had not done your work at all.
You can write up your answers by hand (provided your handwriting is legible) or use a word processing system like Latex or Word. Experience shows that Word is in general difficult to use for this kind of task.
For a programming task, your solution must contain (i) an explanation of your solution to the problem, (ii) the Java code, in a form that we can run it, (iii) instructions how to run it. Also put the source code into your solution document. For all programming tasks, it is not allowed to use any external libraries ("import") or advanced built-in API functions (for example, `String.indexof("a")` or `String.substring(1,5)`), if not stated otherwise.
Please, include name, matriculation number and email address in your submission.

1. Optimizing Chaining

Instead of using lists for storing entries with the same hash values, it is also possible to use binary search trees. Explain your answer to each of the following two questions:

1. What are in this case the worst-case running times for Insert, Find and Delete? Do they improve?

2. What are in this case the average running times for Insert, Find and Delete? Do they improve?

(8 Points)

2. Hashing for Word Counting

Counting different words is a basic task in linguistic analysis. For example in the first two lines of this exercise, there are 27 words, of which 23 are different.

1. Explain how one can effectively count the number of different words in a text using a hash table.

2. Implement a hash table of your choice (with open addressing or chaining) with 2^{14} cells and test it with the following three hash functions for strings:

   - Java's `String.hashcode()`
   - Java's `String.substring(0,4).hashcode()` (that is, the method `String.hashcode()` is only applied to the first 4 characters of the argument string)
     Hint: Ensure that this function also does the right thing for strings of less than 4 characters
   - a hash function chosen by yourself, that may only use the single characters of the word (you may use the function `Character.hashcode()`).

3. Test each of your functions on the following three texts, which can be found at Wikisource:

   - the book Genesis of the bible [1]
   - Romeo and Juliet by Shakespeare [2]
   - the European regulation about banana quality standards [3].

4. For each text, count how many different words are in the text. Also count the percentage of different words wrt. the total number of words in each text (e.g., 6.000 different words in 8.000 words in total $= 75\%$). Also count the number of collisions for each hash function and explain the differences that you find. How well did your own hash function perform?

Remarks:

- It is advisable to copy the texts from Wikisource into text files, then read the text files linewise into Java (using classes such as `BufferedReader`, `InputStreamReader` or similar).

- You may use the Java class `StringTokenizer` to identify the words in the text.

- ignore sentence delimiters (`.,!?`), numbers and other special characters. Also ignore cases, you may use the function `String.toLowerCase()` for that purpose.

[1] `http://en.wikisource.org/wiki/Bible_%28King_James%29/Genesis`
[2] `http://en.wikisource.org/wiki/The_Tragedy_of_Romeo_and_Juliet`
[3] `http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31994R2257:EN:HTML`

(22 Points)

Submission: Tue, 18 May 2012, 8:30 am. Preferrably by email.

If you want to submit a hand-written solution, hand it over to your lab tutor or in the lecture. However, all code has to be submitted in electronic form.