Comp 251: Assignment 1

Answers must be returned online by February 2nd (11:59pm), 2015.

- Your solution must be returned electronically on MyCourse.
- The only format accepted for written answers is PDF. PDF files must open on SOCS computers.
- The solution of programming questions must be written in java. Your program should compile and execute of SOCS computers. Java files that do not compile on SOCS computer will not be graded.
- To some extent, collaborations are allowed. These collaborations should not go as far as sharing code or giving away the answer. You must indicate on your assignments the names of the persons with who you collaborated or discussed your assignments (including members of the course staff). If you did not collaborate with anyone, you write "No collaborators" at the beginning of your document. Importantly, if asked, you should be able to orally explain your solution to a member of the course staff.
- Unless specified, all answers must be justified.
- Violation of these rules are subject to penalties.
- Partial answers will receive credits.
- The course staff will answer questions about the assignment during office hours or in the online forum at `https://cs251qanda.cs.mcgill.ca/`. We urge you to ask your questions as early as possible. No questions will be answered 24h or less prior to the submission deadline.

Exercise 1 (40 points) We wish to experiment hashing with the division and multiplication methods seen in class:

$$f(k) = k \mod d \text{ (division method)}$$
$$g(k) = (A \cdot k \mod 2^w) >> (w - d) \text{ (multiplication method)}$$

Where A is a random number s.t. $2^{w-1} < A < 2^w$, and d and w two integer s.t. $w > d$. Let n be the number of key to insert, and m the number of slots in the hash tables. We remind you that the value $\alpha = \frac{n}{m}$ is called the load factor.

We want to estimate the number of collisions with respect to the choice of the values of w, d, m, and α. More specifically, we want to write a java program that will take as input (i.e. in the javacommand line) the values w, d, and m (in this order). The values of α will vary from 0 to 2 with a step of 0.1. In this experiment, you will simulate the insertion of n keys and count the number of collision for each method (i.e. division and multiplication).

We provide you a template file `COMP251HW1.java` that you will complete. In particular, this file contains a function `generateRandomNumberInRange(double min, double max)`

that will enable you to generate random number within a specified range. We also provide you a jar file `JavaPlotBuilder.jar` to visualize your results (you can run this file with the command line `java -jar JavaPlotBuilder.jar`).

Your first task is to implement the two java methods `divisionMethodImpl` and `multiplicationMethodImpl` in `COMP251HW1.java`. These methods will generate n random keys in the range $[0, 2^w[$ and return the number of collisions. In the case of `multiplicationMethodImpl`, it is worth noting that the value of A must also be adjusted.

A typical command line to run this program will be `java COMP251HW1 32 8 100`, where $w = 32$, $d = 8$, and $m = 100$. The output data will be stored in a csv file generated by the class `COMP251HW1`. You will be able to visualize your data using the jar file `JavaPlotBuilder.jar` (run the command line `java -jar JavaPlotBuilder.jar` and open the csv file with the GUI).

Your second task will be to try several (well chosen) set of values for w, d, and m. Use your results to decide (and justify with graphs) which method is the best. Discuss the behaviour of the hash functions with variation of the parameters w, d, m, and α.

Exercise 2 (15 points) Show the execution of $heapsort$ on $A = \langle 6, 4, 3, 5, 1, 2 \rangle$. Heaps must be represented as arrays. Indicate the operation made on the heap for each step of the algorithm, and show the full status of the array and variables.

Exercise 3 (15 points) We use a random hash function h to hash n distinct keys in an array T of size m. What is the expected number of clashes? Write your results using the big-oh notation and the load factor.

Exercise 4 (15 points) We implement binary trees with a dynamic data structure such that each node `x` is represented by a object that has a field `father[x]` pointing to the father, `left[x]` pointing to the left child, and `right[x]` to the right child. Let `B` be a binary tree and `x` one node of this tree. Write the pseudo-code for the procedure `RotateLeft(B, x)` seen in class that replace `x` by its left subtree, and place `x` and its right sub-tree at the root of the tree (See the slides of the lectures on AVL trees).

Exercise 5 (15 points) Give a recurrence for the number of possible binary search trees with n keys. You may assume the keys are $1, 2, \cdots, n$.
Hint: the root of the binary search tree can be any one of the n keys. You need to consider each of these n possiblities.