Assignment 5

Due date: 10:00am, Monday, 13th of October 2014

General Instructions

You have to do this assignment as a **team of two students** if you are an undergraduate student, or **individually** if you are a postgraduate student. Individual work of an undergraduate student is also allowed and will be treated same as team work. Team members should be from the same tutorial group. All implementations have to be done in JAVA.

Submissions have to include coversheet including names, student ids, and your tutorial group such that submissions can get marked.

Submit your solutions (including printout of the source code) for Exercises 3 and 4.1 as well as the results of the execution of your programs to the box "ADSA" on level 4, Ingkarni Wardli (close to reception) **by the deadline**. **No late submissions will be accepted.**

In addition, submit your source code for Exercises 3 and 4.1 using the web submission system.

Exercise 1 *Hash Tables (1 point)*

Draw the 11-item hash table that results from using the hash function

$$h(i) = (2i + 5) \, mod \, 11,$$

to hash the keys

`[12,44,13,88,23,94,11,39,20,16,5],`

assuming collisions are handled by a linear probing.

Exercise 2 *Proof and Problem-solving Exercises (1+1+2 points)*

- Show an n-node directed acyclic graph that has $\frac{n(n-1)}{2}$ edges.
- Draw a directed graph with only one weakly connected component, that contains no cycles, but is not a tree.

- Attendees of a conference shake hands to greet each other, and each of the members of a program committee remembers how many times he or she shook hands. At the end of the conference, the chair of the program committee adds up the number of times that each member shook hands. Show that the result is even by proving that if $G = (V, E)$ is an undirected graph, then $\sum_{v \in V} degree\,(v) = 2\,|E|$.

Exercise 3 *Graph Implementation (4 points)*

- Create in Java a class called `AdjListGraph` that implements the adjacency list representation of a digraph given in lectures and described in section 8.3 of the Algorithms and Data Structures textbook (K. Mehlhorn and P. Sanders). Your implementation should use an array to hold the nodes of the graph, and linked lists to store the outgoing edge list of each node. (You can use Java's LinkedList class by importing java.util.LinkedList.) Include methods for:
  - constructing a new graph G with n (unconnected) nodes;
  - adding or removing edges from G (note that edges must have weights, so include a parameter to accept a weight);
  - obtaining all edges outgoing from and all edges ingoing to a given node;
  - checking for the existence of a particular edge in G;
  - printing the graph in such a way that the user can interpret easily.
- Test your implementation with a range of input graphs and operations.
- Add a method `generateDigraph(...)` that generates a random graph consisting of $n = 15$ nodes using your or your partner's ID number as a random seed. Assume uniform distribution and with probability 1/2 produce an edge that goes from node u to v for $1 \leq u, v \leq n$, $u \neq v$. Print the resulting graph as an output of your graph reporting method and redraw it by hand as a paper variant.
- Let your program compute the number of ingoing and outgoing nodes for each node v, $1 \leq v \leq n$, and print the resulting numbers in a comprehensible way. Confirm the correctness of the result manually by assigning the number of ingoing and outgoing edges to each node of the graph you have on paper.
- Let your program check whether the graph has the edges $e_{1,5}$, $e_{15,5}$, $e_{3,7}$, $e_{7,7}$, $e_{10,1}$, and $e_{8,7}$. Print results of your testing in a comprehensible way and confirm them manually by highlighting the required edges of the graph you have built on paper.

Exercise 4 *(5 + 4 + 2 points) Graph Traversals*

- Assume that a company has n computers in its office where each computer is identified my an ID from 1 to n. Furthermore, each computer is connected with at least one another computer and their set constitutes a network of the company. A message between a pair of connected computers can be transmitted in one second and in both directions. Being received by one of the computers, a message is immediately transmitted to all other connected computers. The network is organized

in such a way that there is exactly one path between any two computers. You need to define the computers from which a user can send a message in that way that the maximum delay in receiving of the message by all computers in the network is minimized.

  - Add a method `generateNetwork(...)` to generate a random network of $n = 20$ computers using your or your partner's ID number as a random seed. Assume uniform distribution for random variables and for each node u for $1 \leq u \leq n$ generate an edge between u and node v, where v is a randomly selected node with index in the interval $[1, \ldots, u-1]$. Print the resulting network in a comprehensible way and redraw it by hand on paper.
  - Extend your `AdjListGraph` implementation from Exercise 3 by adding a method `findBestNodes(...)` that uses breadth-first search to determine the computers that guarantee the fastest dissemination of a message within the network. Run your method and report the result. Verify manually the correctness of the result and highlight the desired nodes on paper.
  - What is the worst-case performance for `findBestNodes(...)`?

- Propose an algorithm for the problem above of $O(n)$ complexity. (Hint: think about an algorithm to find the center of a tree).
- Draw an adjacency list representation of a graph with six nodes and with each node having between two and four neighbours. Write the order in which the nodes are visited if the graph is traversed from its first node by depth-first search.

Submission instructions for programming code

First, type the following command, all on one line (replacing `aXXXXXXX` with your username):

```
svn mkdir --parents -m "ADSA"
https://version-control.adelaide.edu.au/svn/aXXXXXXX/2014/s2/adsa/assignment5
```

Then, check out this directory and add your files:

```
svn co https://version-control.adelaide.edu.au/svn/aXXXXXXX/2014/s2/adsa/assignment5
cd assignment5
svn add File1.java
svn add File2.java
...
svn commit -m "assignment5 solution"
```

Next, go to the web submission system at:

https://cs.adelaide.edu.au/services/websubmission/

Navigate to *2014*, *Semester 2, Adelaide*, *Algorithm and Data Structure Analysis*, then *Assignment 5*. Click *Make A New Submission For This Assignment* and indicate that you agree to the declaration. The script will then check whether your code compiles. You can make as many resubmissions as you like. If your final solution does not compile you won't get any marks for this solution.

The websubmission system for this assignment will open till 10am Monday October 13th, 2014.

End of Questions