58131 Data Structures (Spring 2009)

Homework 10 (20 March–3 April)

1. A Hamiltonian path in an undirected graph is a simple path that visits each vertex of the graph exactly once. Write an algorithm which checks whether the graph given as input contains at least one hamiltonian path. What is the time complexity of your algorithm?

2. Using the algorithm presented in the lectures (Cormen page 554), construct the strongly connected components. Assume that the arcs are stored in adjacency lists in alphabetic order.
   What is the component graph of this graph? What is the smallest amount of additional arcs needed to make the original graph strongly connected.

3. (a) Prove that in every acyclic directed graph (DAG) there is a vertex to which there are no arcs from other vertices.
   (b) On the basis of this, we can produce a topological ordering as follows.

   \[
   \text{TOPO}(V, E) \\
   1 \quad \text{search } v \in V \text{ to which there are no arcs} \\
   2 \quad \text{return } v \\
   3 \quad \text{TOPO}(V - \{v\}, E - \{(v, u) \mid u \in V\})
   \]

   What is the time complexity of this method? Is this worse or better than the one presented in the lectures (Cormen page 550)?

Group homework 4

This exercise is a continuation of the previous group exercise about the Java Collections framework. As material you may again use Sun’s tutorial on the topic at [http://java.sun.com/docs/books/tutorial/collections/](http://java.sun.com/docs/books/tutorial/collections/).

Data structure and algorithm libraries can make writing complex programs easy. Libraries are usually made by experts, so it’s possible to get efficient solutions by choosing the right algorithms from the library. However, one needs to pay attention to time and space complexity of the library implementations, as choosing wrong solutions easily leads to programs that work very poorly with large inputs.

Continues on the next page!
1. Solve the following exercises using the Java Collections framework. Avoid writing loops yourself; instead, use ready-made algorithms and data structures from the library. Test your solutions and consider their time complexity.

(a) Write a method that removes duplicate entries from an array of strings. In other words, your method should take an array as input and return another array containing each string from the original array precisely once. Hint: Start by figuring out how to use arrays with the Collections framework. Use some suitable implementation of the Set interface to filter out duplicates.

(b) Write a method that takes as parameters a list of integers and an integer \( n \) and returns a new list with \( n \) random elements from the original list (erroneous values of \( n \) can be handled as you wish). The elements of the return list should be picked in such way that no element may be chosen more than once, unless it also appears many times in the original list. The result list should be ordered, starting from the largest number.

2. Familiarise yourself with the Map interface and its implementations. Use Map to implement a word counter, which counts how many times each word appears in a given text and prints out these counts. For example with text “foo bar baz foo” the result should be:

```
bar: 1
baz: 1
foo: 2
```

The program is not required to print out the words in alphabetical order.

The solutions to the group homework are to be returned no later than **Friday 3 April at 16:00**. Send your solutions by e-mail as a PDF file to the teacher of your homework group. Please list the students who participated in the group. For each exercise, you can get 0–2 homework points, so in total maximum 4 points. Feedback to the group homework will be given in the following week’s homework sessions.