FREIE UNIVERSITÄT BOZEN
LIBERA UNIVERSITÀ DI BOLZANO
FREE UNIVERSITY OF BOZEN · BOLZANO

Fakultät für Informatik | Facoltà di Scienze e tecnologie informatiche | Faculty of Computer Science

Data Structures and Algorithms

Written Examination

22 February 2013

FIRST NAME		**LAST NAME**	
STUDENT NUMBER		**SIGNATURE**	

Instructions for students:

Write First Name, Last Name, Student Number and Signature where indicated. If not, the examination can not be marked.

Do not speak to any other student during the examination. If you speak to another student, your examination will be cancelled.

Use a pen, not a pencil.

Write neatly and clearly.

Data Structures and Algorithms

– Exam –

Werner Nutt

22 Februar 2013

- The exam comprises 4 questions, which consist of several subquestions. You will have 2 hours time to answer the questions.
- There is a total of 150 points that can be achieved in this exam. Marking will be out of 120. That is, to achieve a final mark of 30, however, it will suffice to obtain 120 points.
- Please, write down the answers to your questions in the exam booklet handed out to you.
- For drafts use the blank paper provided by the university.
- If the space in the booklet turns out to be insufficient, please use the university paper for additional answers and return them with the booklet.
- **No questions** will be answered during the exam. If you are not sure about interpreting a question, you may write down additional assumptions you made in order to proceed with your solution.

1 Divide And Conquer

Consider an array A of integers. Suppose that $A.length$ returns the length of such an array.

(i) Write down an iterative algorithm in pseudocode that returns true if the array is sorted and false otherwise.

(8 points)

(ii) Write a recursive algorithm following the divide-and-conquer approach that tests whether A is sorted. Start from the fragment below.

```
boolean sorted (array A, int l, int r)
  if l = r then return true
  else if r = l + 1 then ...
  else ...
```

(12 points)

(iii) Briefly explain why your algorithm is correct.

(3 points)

(iv) Analyze the running time $T(n)$ of the algorithm. Formulate a recursion for $T(n)$ and apply the Master Theorem.

(3 points)

2 Sorting with Lists

(i) Describe in your own words the idea of the Insertion Sort algorithm for arrays.

(6 points)

(ii) Write down a Java class declaration for doubly linked lists consisting of elements with an integer as key.

(6 points)

(iii) Write down code (in Java or in pseudocode) for an operation `insert` that takes as input

- a sorted doubly linked list
- an element like those in the list

and inserts the element in the list such that the resulting list is sorted.

(8 points)

(iv) Write down code (in Java or in pseudocode) for a version of Insertion Sort that sorts doubly linked lists.

(8 points)

3 Binary Tree Algorithms

We consider binary trees that have integers as key values.

(i) Write down a Java class declaration for such binary trees.

Hint: There is no need to distinguish in the data structure between internal nodes and leaves. All are nodes of the same kind.

(6 points)

We want to compute for a given binary tree the sum of all key values.

(ii) Write pseudocode for an algorithm `NodeSum` that takes a tree as input and returns the sum of the keys of all nodes of that tree.

Hint: You may want to write the algorithm as a recursive function.

(10 points)

(iii) Explain why your algorithm `NodeSum` is correct.

(3 points)

(iv) What is the asymptotic running time of `NodeSum` for a tree with n nodes? Explain your answer.

(3 points)

Next, we want to compute for a given binary tree the maximum of all key values. Note that we do not assume that the input trees are search trees and remember that all keys are positive.

(v) Write pseudocode for an algorithm `NodeMax` that takes a tree as input and returns the maximum of the keys of all nodes of that tree.

Hint: You may want to write the algorithm as a recursive function.

(10 points)

(vi) Explain why your algorithm `NodeMax` is correct.

(3 points)

(vii) What is the asymptotic running time of `NodeMax` for a tree with n nodes? Explain your answer.

(3 points)

4 Graph Algorithms

A *graph* is a pair $G = (V, E)$, where V is the set of vertices (or nodes) and E the set of edges of the graph. In a *directed graph,* or digraph, the edges have a direction. In a *weighted* directed graph, a number is attached to each edge.

(i) Write down Java class definitions to implement weighted directed graphs.

(10 points)

Dijkstra's algorithm finds shortest paths in weighted directed graphs. Consider the directed graph G_1 with the set of vertices $\{A, B, C, D, E\}$ and eight weighted edges, given by the following triples:

$$(A, B, 9),\ (A, C, 2),\ (A, D, 6),\ (C, D, 3),\ (C, E, 1),\ (D, B, 2),\ (E, B, 3),\ (E, D, 1).$$

For example, the triple $(A, B, 9)$ says there is an edge from A to B with weight 9.

(ii) Make a drawing of the graph with sufficient space among the vertices.
Run Dijkstra's algorithm on this graph with start vertex A and target vertex B.
Visualize the execution of the algorithm, like we did in the lecture.
It should become clear in which order the different steps were executed.
In particular, it should be clear how many steps were needed
and what is the cost of the shortest path from A to B.

(8 points)

From now on we consider graphs without edge weights. Moreover, we concentrate on graphs without cycles, that is, *directed acyclic graphs* (DAGs).

(iii) Give an example of a directed graph that is a DAG and another example of a directed graph that is not a DAG.

(3 points)

For a DAGs, one can *topologically sort* the graph vertices.

(iv) Explain what is a topological sort of the vertices of a DAG.

(4 points)

Consider the directed graph G_2 with the set of vertices $\{A, B, C, D, E, F, H\}$ and the edges

$$(A, B),\ (A, C),\ (A, D),\ (A, E),\ (C, F),\ (D, E),\ (D, H),\ (F, H),\ (H, B),\ (H, E)$$

(v) Write down an ordering of the vertices that is a topological sort for the graph G_2.

(4 points)

Since we do not consider weights anymore, the length of a path is now the number of edges on the path. (Clearly, this is the same as assuming that all edges have weight 1.)

In some applications it is important to find the *longest path* in a DAG. The longest path problem is, given a DAG $G = (V, E)$ and a start vertex s in V, to label every vertex v in V with the length $v.len$ of the longest path from s to v.

(vi) Explain what it means that a problem has the optimal substructure property! Does the longest path problem have the optimal susbstructure property? Explain your answer.

(4 points)

(vii) Describe an algorithm that, given a DAG and a start vertex s, returns for each vertex v the length of the longest path from s to v. You need not write pseudo-code and can describe the steps of the algorithm in words.

Hint: You may want to first topologically sort the vertices of the graph.

(20 points)

(viii) Explain how your algorithm finds the length of the longest path in the graph G_2 above from A to B, from A to C, and from A to D.

(5 points)