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

16 September 2014

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

Instructions for students:

Write <u>First Name,</u> <u>Last Name,</u> <u>Student Number</u> and <u>Signature</u> 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.

You are allowed to use a pencil for this exam.

Data Structures and Algorithms
– Exam –

Werner Nutt

16 September 2014

- The exam comprises 5 questions, which consist of several subquestions. You will have 2 hours time to answer the questions.
- Write code that is asymptotically as efficient as possible.
- 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, 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.

1 Asymptotic Analysis

Consider the following functions:

1. $f(n) = n^{3/2}$
2. $g(n) = 2^{3 \log_2 n}$
3. $h(n) = 2 \cdot 3^n$
4. $k(n) = 3 \cdot n^2$
5. $l(n) = 3 \cdot 2^n$

(i) Order these functions according to their asymptotic complexity, from the function having the smaller asymptotic complexity to the function having the larger one (i.e., such that $f_1 = O(f_2)$; $f_2 = O(f_3)$; etc.). Point out functions that are asymptotically equivalent.

There is no need to explain your answer.

(15 points)

2 Most Frequent Number in Array

We consider arrays of integers where elements can occur multiple times. For example, $[3, -1, 7, 5, -1, -5, 3, -1, 7, 6, 3]$ is such an array.

We want to develop an efficient procedure `mostFrequentNum`(A) that returns the most frequent number if given such an array A. In case there are several numbers that occur with the same frequency, the procedure should output the least among those numbers in the array.

In the example above, both the numbers 3 and -1 occur three times each. Since $-1 < 3$, our procedure should output -1 if the input is the example array.

(i) Write up your idea for a solution to this task.

(10 points)

(ii) Write down pseudocode for `mostFrequentNum`(·)!

(12 points)

(iii) What is the asymptotic running time of `mostFrequentNum`$(\cdot)$?
Explain your answer.

(8 points)

3 Divide And Conquer

We consider **sorted arrays** where **duplicate** occurrences of the same integer are possible.

Your task is to develop an **efficient** algorithm `findFirstPos`(A, k) that, given such an array and an integer key k, returns the first (= least, smallest) index i sucht that $A[i] = k$, if k occurs in the array A, and that returns -1 otherwise.

(i) Explain in words how an efficient algorithm for this task can work.

Hint: Define a recursive function `findFirstPosAux`(A, k, l, r) that takes as input, in addition to A, the left and right boundary of a segment of A that contains the first occurrence of k in A.

(10 points)

(ii) Write down pseudocode for a **recursive** algorithm `findFirstPos`(A, k).

(12 points)

(iii) Analyze the running time $T(n)$ of the algorithm.

Hint: Formulate a recursion for $T(n)$ and apply the Master Theorem.

(8 points)

(iv) Explain in words how one has to change the procedure from above to define a procedure `findLastPos`(A, k) that returns the **last index** of A where k occurs, or -1 if k does not occur in A.

(6 points)

(v) Write pseudocode for `findLastPos`(A, k).

(8 points)

(vi) Write pseudocode for an iterative version `findFirstPosIter`(A, k) that is at least as efficient as the recursive version.

(12 points)

4 Lists

Let the class `IList` (like “integer list”) be defined as

```
class IList {
    head INode;}
```

The nodes of these lists are instances of the class

```
class INode{
    int key;
    INode next;}
```

Consider the method

```
    int length()
```

for this class that returns the number of all nodes of such a list. If the list is empty, it returns 0.

(i) Write pseudocode for the method `length()`.

(You can choose whether to write a recursive or an iterative version.)

(10 points)

Again for the class `IList`, consider the static method

```
int[] iListToArray(IList L)
```

that takes as input an IList L and that returns an integer array that has the same length as L such that the i th element of the array is the same number as the key of the i th node of the list.

(ii) Write pseudocode for the method `iListToArray(·)`.

(You can choose whether to write a recursive or an iterative version.)

(10 points)

Consider now the method for the class `IList`

```
    IList reverseCopy()
```

that returns a new list with new nodes, having the same keys as the old list, such that the order of keys in the new list is the reverse of the order in the old list.

(iii) Write pseudocode for the method `reverseCopy()`.

(13 points)

5 Graph Algorithms

Consider the directed graph $G = (V, E)$ with node set

$$V = \{ s, a, b, c, d, e, f, g \}$$

and edge set

$$E = \{ (s, a), (s, c), (g, c), (g, e), (a, b), (b, d), (c, d), (c, a), (e, d), (e, f), (f, d) \}$$

(i) Make a drawing of the graph with sufficient space among the nodes. Run the DFS (=Depth-First Search) algorithm on that graph. Annotate each node with the start time and end time of DFS visiting it. Mark the edges of the resulting depth-first forest in bold.

(12 points)

(ii) Explain what is a topological sorting. For which kind of graphs do topological sortings exist?

(5 points)

(iii) Write down the topological sorting resulting from the traversal in (i).

(3 points)