

# Data Structures and Algorithms

## Exam

Prof. Dr. M. Böhlen

February 12, 2008

14:00 - 16:00

*The exam consists of 4 exercises. For each exercise you can get 25 points. It is important that you argue for your answers and that you present your solutions in a readable form. As auxiliary material you may use 1 A4 sheet with notes and a pocket calculator.*

### 1 Hashing

Assume the list of integers (14322, 14959, 12645, 1816564) and an empty hash table of size  $m = 4$ .

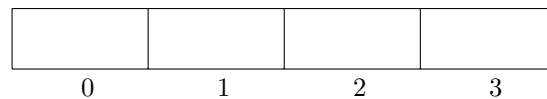


Figure 1: Empty hash table.

- 1a) Let  $\text{frac}(\cdot)$  return the fractional part of a real number. Use the hash function  $h(k) = \lfloor m(\text{frac}(k \frac{\sqrt{5}-1}{2})) \rfloor$  to insert the integers in the given order into the hash table. Use the linear probing scheme to resolve conflicts.
- 1b) Write a search algorithm that searches for a given integer in the hash table.
- 1c) What is the asymptotic complexity of your algorithm. Illustrate the complexity by selected numeric examples. Discuss how an implementation based on 1a) will match the asymptotic complexity.

### 2 Arithmetical Expression Tree

An arithmetical expression tree is a binary tree where each non-leaf node is either a multiplication “\*” or an addition “+”, and each leaf node is an integer. Fig-

Figure 2 shows the arithmetical expression tree for the expression  $(21+7)*((2*(8+3))*14)$ .

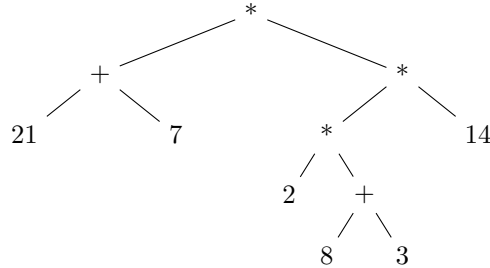


Figure 2: An example of arithmetical expression tree.

- 3a) Propose a C data structure that can represent a node of an arithmetical expression tree.
- 3b) Write an algorithm that returns the result of evaluating an arithmetic expression tree.
- 3c) Analyze the complexity of your algorithm. What are the best and worst cases?
- 3d) Explain how the commutativity and associativity laws can be used to reduce the height of the tree? Illustrate the idea on the example tree in Figure 2.

### 3 Quick Sort

Consider the array of integers in Figure 3 and the QuickSort algorithm with the following partitioning procedure:

```

Partition (A,l,r)
  x := A[l]
  i := l-1
  j := r+1
  while TRUE
    repeat j := j-1 until A[j] ≤ x
    repeat i := i+1 until A[i] ≥ x
    if i < j then switch A[i] ↔ A[j]
  else return j
  
```

- 2a) Specify a QuickSort algorithm that uses the given partitioning function.
- 2b) Show the array after each call of the Partition function of the QuickSort algorithm.
- 2c) What is the worst case complexity of the QuickSort algorithm? What is the worst case for the QuickSort algorithm? Give an example.

|       |   |    |   |   |    |   |   |   |
|-------|---|----|---|---|----|---|---|---|
|       | 4 | 11 | 7 | 6 | 12 | 1 | 8 | 5 |
| step1 |   |    |   |   |    |   |   |   |
| step2 |   |    |   |   |    |   |   |   |
| step3 |   |    |   |   |    |   |   |   |
| step4 |   |    |   |   |    |   |   |   |

Figure 3: An array of integers.

- 2d) What is an average case complexity for the QuickSort algorithm? List two other sorting algorithm with the same average case complexity.

## 4 Task Dependency Graph

In the area of project management area the concept of a task dependency graph is used. The vertices in a task dependency graph are the tasks. An edge from a vertex  $u$  to a vertex  $v$  states that task  $u$  must be done after task  $v$ .

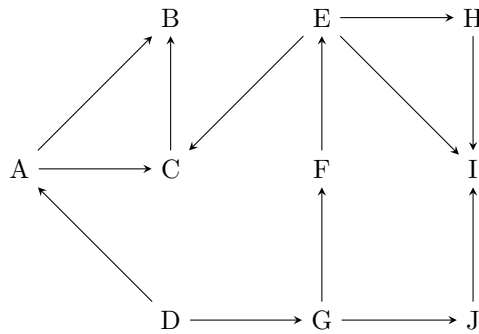


Figure 4: An example of a task dependency graph.

- 4a) For each vertex of the graph in Fig. 4 show the start and end times of a DFS traversal. Start at vertex A.
- 4b) List the vertices of the graph in Fig. 4 in such an order that for any edge  $(u, v)$  in the graph,  $u$  appears before  $v$  in the ordering.
- 4c) Is such an ordering always possible? Argue for your answer.
- 4d) Is there a unique solution? If yes argue for your answer. If no argue for your answer and give an example.
- 4e) Specify the ordering algorithm and determine its asymptotic complexity.