

Data Structures and Algorithms

Exam

Prof. Dr. M. Böhlen

June 22, 2007

9:00 - 11:00

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.

1 Linked Lists

Consider the singly-linked sorted lists in Figure 1.

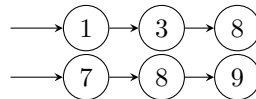


Figure 1: Two Singly-linked Sorted Lists

The goal is to develop an algorithm that merges two sorted singly-linked lists. The merged list shall contain all elements of the original lists and shall be sorted again. The algorithm shall use a constant amount of memory.

- 1a) Explain the idea of your algorithm by showing the intermediate states of the merging process.
- 1b) Implement the merging algorithm in C.
- 1c) Estimate the asymptotic complexity of your algorithm.
- 1d) Analyze the performance of your algorithm. Can it be improved? How? Provide precise arguments for your answer.

2 Binary Search Trees

Consider the binary tree in Figure 2.

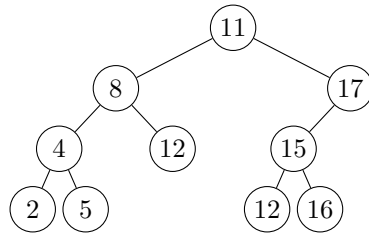


Figure 2: Binary Tree

- 2a) Is the binary tree in Figure 2 also a binary search tree? Provide precise arguments for your answer. Give the definition of a binary search tree.
- 2b) Write an algorithm that decides if a given binary tree is also a binary search tree.
- 2c) Formulate a recurrence that describes the run time behavior of your algorithm in the worst case. Solve the recurrence and determine the asymptotic complexity of your algorithm.

3 Graphs

In a directed graph a *path* is a sequence (v_0, v_1, \dots, v_k) of vertices. A *cycle* is a path that starts and ends at the same vertex. Each node in a cycle is reachable from all other nodes in the cycle.

Consider the directed graph in Figure 3.

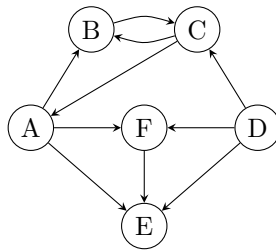


Figure 3: Directed Graph

- 3a) Mark all edges that participate in some cycle.
- 3b) Propose an algorithm that prints all edges that participate in some cycle. Illustrate the working of your algorithm on the graph in Figure 3.
- 3c) Determine the asymptotic complexity of your algorithm. Give the asymptotic complexity for the different parts of your algorithm and then combine the parts into the overall asymptotic complexity.

4 Multiplication of Matrices

A *product of matrices* is a multiplication of n matrices: $A_1 \times A_2 \times \dots \times A_n$. A product is *fully parenthesized* if it is either a single matrix or the product of two fully parenthesized products surrounded by parentheses. Since matrix multiplication is associative all parenthesizations of a product yield the same result.

- 4a) Determine the best placement of parentheses for the computation of $A \times B \times C$, where A, B , and C are the matrices given in Figure 4? What is the cost of this placement (specify also the unit of this cost)? Show the corresponding fully parenthesized product.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \\ a_{41} & a_{42} & a_{43} \end{pmatrix} \quad B = \begin{pmatrix} b_{11} & b_{12} & b_{13} & b_{14} & b_{15} & b_{16} \\ b_{21} & b_{22} & b_{23} & b_{24} & b_{25} & b_{26} \\ b_{31} & b_{32} & b_{33} & b_{34} & b_{35} & b_{36} \end{pmatrix} \quad C = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \\ c_{41} & c_{42} \\ c_{51} & c_{52} \\ c_{61} & c_{62} \end{pmatrix}$$

Figure 4: Example Matrices

- 4b) Define a function $M(i, j)$ that returns the cost of a chain multiplication of matrices $A_i \times \dots \times A_j$. Give an algorithm that computes the cost of the best parenthesization of $A_1 \times \dots \times A_n$ in $O(n^3)$ time.
- 4c) Give an algorithm that prints the best parenthesization of $A_1 \times \dots \times A_n$