This question paper consists of 5 printed pages, each of which is identified by the Code Number COMP272101.

School of Computing

May 2018

COMP272101

Algorithms and Data Structures II

Answer all four questions

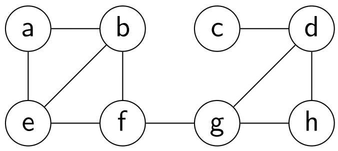
Time allowed: two hours

PLEASE DO NOT REMOVE THIS PAPER FROM THE EXAM ROOM

Question 1

This question deals with graph traversal.

(a) Execute depth-first search on the following graph. Start at vertex a and handle neighbours in alphabetical order.



Mark the edges of the DFS-tree in a drawing of the graph. For each vertex v give its DFS-number. **[6 marks]**

(b) What is the asymptotic running time of a DFS? **[2 marks]**

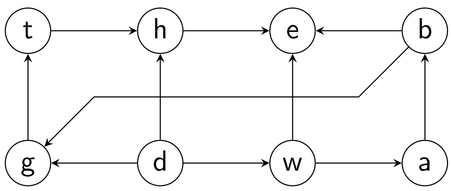
(c) Suppose you are implementing a DFS. Which data structure would you use to store the input graph? Justify your answer. **[2 marks]**

(d) How can a DFS be modified to detect whether the input graph is connected? What is the running time of the modified algorithm? **[3 marks]**

(e) Let $G = (V, E)$ be a directed graph with $n = |V|$. Define what it means for a map $\sigma\colon V \to \{1, \ldots, n\}$ to be a topological sort of G. **[2 marks]**

(f) Which directed graphs admit a topological sort? **[1 mark]**

(g) Execute topological sort on the following directed graph. If there is a choice handle vertices in alphabetic order. For each vertex v give its number $\sigma(v)$. **[4 marks]**



[question 1 total: 20 marks]

Question 2

This question is about dynamic programming for Matrix Chain Multiplication, where we want to determine the minimum number of scalar multiplications necessary to compute a product $A_1 \cdot A_2 \cdot \ldots \cdot A_n$ of n matrices $A_1, A_2, \ldots, A_n$.

(a) For $n_A, m_A, n_B, m_B \in \mathbb{N} \setminus \{0\}$, let A be an $(n_A \times m_A)$-matrix and let B be an $(n_B \times m_B)$-matrix. Which condition on n_A, m_A, n_B, m_B needs to be satisfied so that the matrices A and B can be multiplied (i.e. the product $A \cdot B$ exists)? What is the size of the resulting matrix $A \cdot B$?

[2 marks]

(b) Describe the dynamic programming algorithm for Matrix Chain Multiplication by answering questions (i)-(iv).

Recall that for all i, k with $1 \leq i \leq k \leq n$ we compute the minimum number $M[i, k]$ of multiplications necessary to compute $A_i \cdot \cdots \cdot A_k$, where

$$\begin{aligned} M[i,i] &= 0 && \text{for all } i \text{ with } 1 \leq i \leq n \\ M[i,k] &= \min\{M[i,j] + d_{i-1}d_j d_k + M[j+1,k] \mid i \leq j < k\} && \text{for all } i,k \text{ with } 1 \leq i < k \leq n. \end{aligned}$$

(i) What are the input and the output?

(ii) What are the numbers d_i in the above recurrence?

(iii) Which numbers are computed in intermediate steps and how?

(iv) How can the optimal position of the brackets be determined?

(v) What is the asymptotic running time and why?

[10 marks]

(c) Execute your algorithm for four matrices $A_1 \cdot A_2 \cdot A_3 \cdot A_4$, where A_1 is a (2×3)-matrix, A_2 is a (3×6)-matrix, A_3 is a (6×4)-matrix, and A_4 is a (4×5)-matrix. Copy the table below and fill in the intermediate results. Determine the minimum number of scalar multiplications and the corresponding positions of the brackets.

	$k = 1$	$k = 2$	$k = 3$	$k = 4$
$i = 1$				
$i = 2$				
$i = 3$				
$i = 4$				

[8 marks]

[question 2 total: 20 marks]

Question 3

(a) For each of the recurrences below, give a tight asymptotic upper bound.

(i) $T(1) = 1$, $T(n) = 4 \cdot T(n/4) + n$ **[3 marks]**

(ii) $T(1) = 2$, $T(n) = 8T(n/2) + \log_2 n$ **[3 marks]**

(iii) $T(1) = 0$, $T(n) = n + T(n-1)$ **[3 marks]**

(b) For $n \in \mathbb{N}$, let F_n denote the nth Fibonacci number, i. e. $F_0 = 0$, $F_1 = 1$, and $F_n = F_{n-1} + F_{n-2}$ for $n > 1$.

(i) Recall that for a matrix M and $n \in \mathbb{N}$, $M^n = \overbrace{M \cdot \ldots \cdot M}^{n \text{ times}}$. Prove by induction that every integer $n > 0$ satisfies $\begin{bmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n$. **[8 marks]**

(ii) Consider the algorithm below, where `M` is the matrix $\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}$. What does the function `f` compute for an input $n \in \mathbb{N}$? Give a recurrence equation for the running time of the algorithm, and provide a tight asymptotic bound. **[6 marks]**

```
M ← [[1,1],[1,0]];
function f(n)
    if n = 0 then
        return 0
    else
        tmp ← power(n, M);
        return tmp[0][1]
function power(n, M)
    if n = 1 then
        return M
    else
        P ← power(⌊n/2⌋, M);
        if n is odd then return M · P · P;
        else return P · P;
```

[question 3 total: 23 marks]

Question 4

(a) This question is about algorithm design principles.

(i) Name two well-known greedy algorithms. **[2 marks]**

(ii) Assume you are given an optimisation problem, and you want to come up with a dynamic programming algorithm for it. What will you look for first? **[2 marks]**

(b) This question is about hashing with chaining.

(i) Demonstrate what happens when we insert keys $15, 7, 2, 36, 1, 20$ into a hash table with collisions resolved by chaining. Let the table have 7 slots and let the hash function $f(x) = x \mod 7$. Assume insertions into lists are carried out at the beginning. **[6 marks]**

(ii) Assume that the hash table stores n elements. What is the worst-case running time for `lookUp`? Explain why your bound is tight. **[3 marks]**

(iii) Assume the hash table has load-factor α. What is the average-case running time for a successful search, assuming simple uniform hashing? Explain what this means in the case that the number n of elements in the table satisfies $n = \mathcal{O}(m)$, where m is the number of slots. **[4 marks]**

[question 4 total: 17 marks]

[grand total: 80 marks]