DURATION: 3 HOURS

Department Name & Course Number: Computer Science COMP/MATH 3804B
Course Instructor: Michiel Smid

Authorized memoranda: NONE

Students MUST count the number of pages in this examination question paper before beginning to write, and report any discrepancy to the proctor. This question paper has 15 pages (not including the cover page).

This examination question paper MAY NOT be taken from the examination room.

In addition to this question paper, students require:

- an examination booklet: no
- a Scantron sheet: no
Student Name:
Student Number:

<table>
<thead>
<tr>
<th>Question</th>
<th>Maximum Number of Marks</th>
<th>Marks Received</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$3 \times 3 = 9$</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>15</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$7+7=14$</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>$5+10=15$</td>
<td></td>
</tr>
<tr>
<td>Total</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>

Instructions

1. All questions must be answered on this examination paper. You may use both sides of the answer sheets for answers. There are some extra sheets attached at the end of this examination paper.

2. This is a closed book examination.

3. Please try not to ask questions during the exam. If you find a question to be ambiguous or unclear, then make, and state, whatever assumptions you feel are necessary; your mark will then be partly based on the reasonableness of your assumptions.

4. Please answer to the point and do not write everything you know on the topic. Substantial marks will be deducted if your answer is not precise.
Question 1: (3 marks for every correct answer)

Answer TRUE or FALSE:

(1.1) Dynamic programming is a technique to solve NP-complete problems in polynomial time.

Answer:

(1.2) Given a min-heap storing $n$ numbers, and given a number $x$, we can determine in $O(\log n)$ time if $x$ is stored in this min-heap.

Answer:

(1.3) Let $G$ be a connected graph with positive weights on its edges, and let $u$ and $v$ be two vertices of $G$. Dynamic programming can be used to compute the longest path between $u$ and $v$.

Answer:

Question 2: (3 marks) What does NP stand for? Circle the correct answer.

(a) non-polynomial time
(b) no parking
(c) national park
(d) non-deterministic polynomial time
Question 3: (15 marks) The function $T$ satisfies the following recurrence.

$$T(n) = \begin{cases} 
1 & \text{if } n = 1, \\
\frac{2n}{n-1} + \frac{n}{n-1} \cdot T(n-1) & \text{if } n \geq 2.
\end{cases}$$

Prove that $T(n) = O(n \log n)$.

*Hint:* You may use the fact that $\sum_{i=1}^{n} 1/i = O(\log n)$. 
Question 4: (15 marks) Let $m$ be a positive integer and let $n = 2^m$. You are given an $n \times n$ board $B_n$, all of whose cells are white, except for one, which is black. (The left part of the figure below gives an example where $n = 8$.)

A *tromino* is an L-shaped object consisting of three $1 \times 1$ cells. Each tromino can appear in four different orientations; see the right part of the figure below.

![Trominoes](image)

A *tiling* of the board $B_n$ is a placement of trominoes on the board such that

- the trominoes cover exactly all white cells (thus, the black cell is not covered by any tromino) and
- no two trominoes overlap.

Here is a tiling of the board given above:

![Tiling](image)

Describe a divide-and-conquer algorithm that

- takes as input a board $B_n$ having exactly one black cell (which can be anywhere on the board) and
- returns a tiling of this board.

You can describe the algorithm in plain English. You do not have to analyze the running time of your algorithm.

*Hint:* Look at the following figure:
Question 5: (14 marks) Let $G = (V, E)$ be a directed acyclic graph and let $s$ and $t$ be two vertices of $G$.

Describe an algorithm that computes, in $O(|V| + |E|)$ time, the total number of paths from $s$ to $t$ in the graph $G$. You can describe your algorithm in plain English. You may use any algorithm that was presented in class.
Question 6: (15 marks) You are given a sequence $x_1, x_2, \ldots, x_n$ of $n$ items, where each item $x_i$ has

- a weight $w_i$ (in kilograms) and
- a value $v_i$ (in dollars).

You are also given a positive number $W$, representing a weight (in kilograms).

You want to compute a subset of the items whose total weight is at most $W$ and whose total value is as large as possible. Thus, you want to compute a subset $I$ of $\{1, 2, \ldots, n\}$ such that

- $\sum_{i \in I} w_i \leq W$ and
- $\sum_{i \in I} v_i$ is maximum.

Describe a dynamic programming algorithm that solves this problem in $O(nW)$ time. You may assume that $w_1, \ldots, w_n, v_1, \ldots, v_n,$ and $W$ are positive integers.

Hint: For integers $w$ and $j$ with $0 \leq w \leq W$ and $0 \leq j \leq n$, define $M(w, j)$ to be the maximum possible value of $\sum_{i \in I} v_i$ subject to the conditions that $I \subseteq \{1, 2, \ldots, j\}$ and $\sum_{i \in I} w_i \leq w$. The solution for $M(w, j)$ either uses item $x_j$ or does not use item $x_j$. The question asks to compute the value $M(W, n)$. 
Problem 7: (7+7 marks) Let $\varphi$ be a Boolean formula in the variables $x_1, x_2, \ldots, x_n$.
We say that $\varphi$ is in conjunctive normal form (CNF) if it is of the form
$$\varphi = C_1 \land C_2 \land \ldots \land C_m,$$
where each $C_i, 1 \leq i \leq m$, is of the following form:
$$C_i = l^i_1 \lor l^i_2 \lor \ldots \lor l^i_{k_i}.$$
Each $l^i_j$ is a literal, which is either a variable or the negation of a variable.
We say that $\varphi$ is in disjunctive normal form (DNF) if it is of the form
$$\varphi = C_1 \lor C_2 \lor \ldots \lor C_m,$$
where each $C_i, 1 \leq i \leq m$, is of the following form:
$$C_i = l^i_1 \land l^i_2 \land \ldots \land l^i_{k_i}.$$
Again, each $l^i_j$ is a literal.
We define the following two languages:
$$\text{SAT} := \{ \varphi : \varphi \text{ is in CNF-form and is satisfiable} \}$$
and
$$\text{DNFSAT} := \{ \varphi : \varphi \text{ is in DNF-form and is satisfiable} \}.$$
(7.1) Prove that $\text{DNFSAT} \in \mathbf{P}$. 
What is wrong with the following argument:

- Let $\varphi$ be an arbitrary Boolean formula in CNF-form. We can use the basic rules of logic (such as De Morgan’s Law) to rewrite $\varphi$ as an equivalent Boolean formula in DNF-form. Therefore,

  $\text{SAT} \leq_P \text{DNFSAT}$.  

- We have seen in (7.1) that $\text{DNFSAT} \in \text{P}$.  

- Since $\text{SAT} \leq_P \text{DNFSAT}$ and $\text{DNFSAT} \in \text{P}$, we have $\text{SAT} \in \text{P}$.  

- We have seen in class that $\text{SAT}$ is $\text{NP}$-complete.  

- Thus, the $\text{NP}$-complete problem $\text{SAT}$ belongs to $\text{P}$.  

- Therefore, $\text{P} = \text{NP}$.  

(7.2)
Question 8: (5+10 marks) The set cover problem is defined as follows:

\[
\text{SetCover} := \{(B, S_1, S_2, \ldots, S_m, K) : \text{ } B \text{ is a set of size } n; \text{ } S_1, S_2, \ldots, S_m \text{ are sets with } \bigcup_{i=1}^{m} S_i = B; \text{ } K \text{ is an integer; there exists a subset } I \subseteq \{1, 2, \ldots, m\} \text{ of size } K, \text{ such that } \bigcup_{i \in I} S_i = B \}\.
\]

The (0-1)-integer programming problem with \( K \) ones is defined as follows:

\[
\text{IntProgKOnes} := \{(A, K) : \text{ } A \text{ is an integer } n \times m \text{ matrix all of whose entries are in } \{0, 1\}; \text{ } K \text{ is an integer; there exists a binary column vector } x \text{ of length } m \text{ with exactly } K \text{ ones, such that } Ax \geq 1 \text{ (componentwise)} \},
\]

where \( 1 \) denotes the column vector of length \( n \), all of whose entries are equal to \( 1 \).

You are given that \( \text{SetCover} \) is \( \text{NP-complete} \).

\textbf{(8.1)} Complete the following sentence: We can prove that \( \text{IntProgKOnes} \) is \( \text{NP-complete} \), by showing that

- ........ belongs to the complexity class ........
- problem ........ can be reduced in polynomial time to problem ........

\textbf{(8.2)} Give the function \( f \) that defines the reduction that you mentioned above. It is sufficient to state the function \( f \) without explaining why it defines a correct reduction.