INSTRUCTIONS TO CANDIDATES

1. This examination paper contains four (4) questions and comprises eight (8) printed pages, including this page.

2. Answer ALL questions.

3. Answer ALL questions within the space provided in this booklet.

4. This is an OPEN BOOK examination.

5. Please write your Matriculation Number Below.

MATRICULATION NO: ________________________________________________________

This portion is for examiner’s use only

<table>
<thead>
<tr>
<th>Question</th>
<th>Marks</th>
<th>Remarks</th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>/ 10</td>
<td>Leftist Heap</td>
</tr>
<tr>
<td>Q2</td>
<td>/ 20</td>
<td>Tetrahedron</td>
</tr>
<tr>
<td>Q3</td>
<td>/ 20</td>
<td>Transitive closure</td>
</tr>
<tr>
<td>Q4</td>
<td>/ 20</td>
<td>Vertex cover</td>
</tr>
<tr>
<td>Total</td>
<td>/ 70</td>
<td></td>
</tr>
</tbody>
</table>
Q1) Leftist heaps (10 points)

Propose an algorithm to insert \( k \) \((k < n)\) nodes into a leftist heap of \( n \) elements in \( O(k + \lg n) \) worst-case time.

Prove your bound.

Q2) Symmetries of a tetrahedron, Polya counting theory (20 points)

a) List all the transformations that map a regular tetrahedron onto itself under the symmetry group \( G \) of rotations in 3-d space.

b) Use Polya's counting formula to determine the number of distinct ways of coloring the faces of a regular tetrahedron with 2 colors, say white and black. For each transformation \( \pi \) of \( G \), determine the cycle index of \( \pi \), and compute the cycle index \( PG \) of \( G \).

c) Verify the result of b) by exhaustively listing all the distinct face colorings.

d) Use Polya's counting formula to determine the number of distinct ways of coloring the edges of a regular tetrahedron with 2 colors, say white and black. For each transformation \( \pi \) of \( G \), determine the cycle index of \( \pi \), and compute the cycle index \( PG \) of \( G \).

e) Verify the result of d) by exhaustively listing all the distinct edge colorings.

Q3) Transitive closure, finite state machines, constrained paths in a graph (20 points)

a) The following fsm \( M3 \) computes \( x \mod 3 \), where \( x \) is a binary integer being read "from left to right", i.e. from most significant bit to least significant bit. Prove this assertion.

![Finite State Machine M3](image)

b) Design an efficient algorithm to solve the following problem:
Given an fsm \( M = ( Q, \{0, 1\}, f: Q \times \{0, 1\} \rightarrow Q) \) with \( n \) states, and given a positive integer \( B \), compute the matrix \( R \), where \( R_{ij} \) is a regular expression that denotes all the paths in \( M \) from \( q_i \) to \( q_j \) of (exact) length = \( B \).

c) Apply this algorithm to the instance \( M3 \) and \( B = 4 \).
Q4) Vertex cover, decision and optimization problems (20 points)

**Df:** A vertex cover of a graph \( G = (V, E) \) is a subset \( C \) of \( V \) such that every edge \( (x,y) \) has at least one endpoint in \( C \).

**Df:** An independent set of a graph \( G = (V, E) \) is a subset \( I \) of \( V \) such that, for all \( x, y \) in \( I \), there is no edge \( (x,y) \) in \( E \).

a) Prove: A subset \( C \) of \( V \) is a vertex cover iff its complement \( V-C \) is an independent set. Moreover, \( C \) is a minimum vertex cover iff \( V-C \) is a maximum independent set.

b) Draw a graph \( G_1 \) that has a subset \( U \) of \( V \) such that \( U \) is both a vertex cover and an independent set. Draw a graph \( G_2 \) that has no such subset \( U \).

c) Design an algorithm to decide whether a given graph \( G = (V, E) \) has a subset \( U \) of \( V \) such that \( U \) is both a vertex cover and an independent set. Analyze the complexity of your algorithm, in particular whether it is in \( P \).

d) Consider 3 problems:

P1: Decision problem: Given \( G = (V, E) \) and \( k > 0 \), does \( G \) have a vertex cover of size \( \leq k \)?

P2: Evaluation problem: Given \( G = (V, E) \), determine the size of a minimum vertex cover

P3: Optimization problem: Given \( G = (V, E) \), construct a minimum vertex cover

d1) Assuming you can solve P1 in time \( T_1 \), show how to use this information to solve P2, and state the time complexity of the resulting algorithm that solves P2.

d2) Assuming you can solve P2 in time \( T_2 \), show how to use this information to solve P3, and state the time complexity of the resulting algorithm that solves P3.