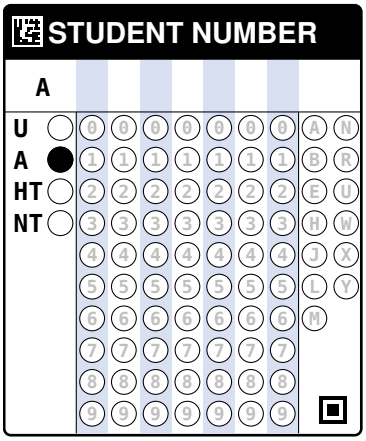
NATIONAL UNIVERSITY OF SINGAPORE

SCHOOL OF COMPUTING
Semester 1 AY2025/2026

Mid-Term Test
IT5005 Artificial Intelligence

07^{th} Oct 2025 Time Allowed: 60 minutes



INSTRUCTIONS TO CANDIDATES

1. Please write student number only. Do not write your name.
2. This assessment paper contains 8 questions and comprises 11 printed pages.
3. This is a **CLOSED BOOK** assessment
4. Only an A4 cheat sheet is allowed. Apart from calculators, electronic devices are not allowed.
5. Answer **ALL** questions and write your answers only in the space provided.
6. Psuedocodes and logical equivalences are provided in the Appendix at the end.
7. The total marks for this assessment is 50.

For Examiner's Use Only

	Score	Points
MCQs		10
6		15
7		15
8		10
Total		50

Multiple Choice Questions

Each multiple-choice question (MCQ) is worth **TWO marks** and has exactly **one** correct answer

1. In uniform-cost search (UCS), if the goal test is performed upon node generation, then UCS may return suboptimal solution.

   √ ***True*** ◯ *False*

2. Let h_1 and h_2 be admissible heuristics for a given problem. Which of the following statements is true for that problem?

   ◯ $h_1 + h_2$ is guaranteed to be admissible.

   √ $0.1 * h_1 + 0.9 * h_2$ **is guaranteed to be admissible**

   ◯ $\max(0, h_1 - h_2)$ is not admissible.

   ◯ $2 * h_1 + 0.1 * h_2$ is guaranteed to be an admissible heuristic

   ◯ None of the above choices are correct

3. Given the following statements, pick the odd one.

   ◯ $P \vee \neg P$.

   √ $(P \vee Q) \Rightarrow (P \wedge Q)$

   ◯ $(P \wedge \neg P) \Rightarrow R$

   √ $((P \Rightarrow Q) \vee (P \Rightarrow R)) \Rightarrow ((P \vee Q) \Rightarrow R)$

   ◯ $(P \wedge \neg P) \Rightarrow (P \vee \neg P)$

4. Given the sentences $P \equiv (A \vee (B \wedge C))$ and $Q \equiv ((A \vee B) \wedge (A \vee C))$, identify the correct choice.

   ◯ $P \Rightarrow Q$ is a tautology.

   ◯ When P is True, Q is also True.

   ◯ When P is False, Q is also False.

   ◯ $Q \Rightarrow P$ is a tautology.

   √ **All of the above choices are correct.**

5. A Bayesian network is a directed acyclic graph (DAG).

   √ ***True*** ◯ *False*

Section B

6. (15 points) **Search**

Consider a robot navigation problem where a robot must move through a grid-based warehouse to collect packages. The warehouse is represented as a 6×6 grid as shown in Figure 1. Note the location of the origin is at the top left $(0, 0)$, and that the X-axis is vertical, and Y-axis is horizontal. $S = (4, 0)$ is the starting position and $G = (1, 5)$ is the goal position, and dark-shaded regions indicate the obstacle (shelf or wall).

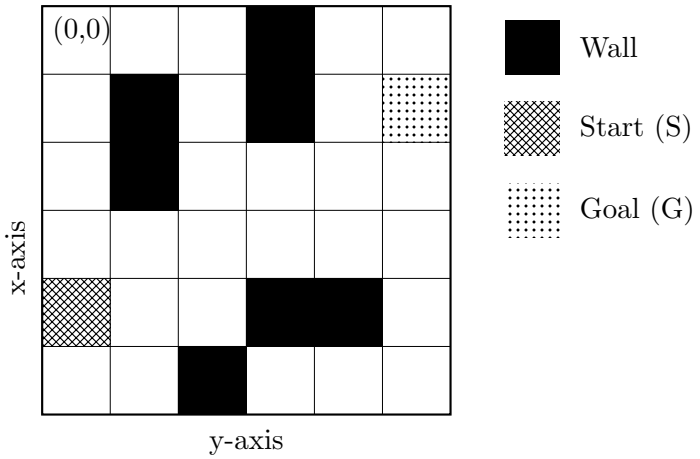


Figure 1: Maze

The robot can move in four directions: North, South, East, West (no diagonal movement). Suppose that the agent is at location (i, j). Actions North, South, East and West take the agent to the locations $(i-1, j)$, $(i+1, j)$, $(i, j-1)$, and $(i, j+1)$, respectively. Actions that take the agent out of the maze are not allowed. For example, the robot cannot take action North at $(0, 0)$. Each action costs 1 unit.

(a) (6 points) **Uninformed Search**

i. Would breadth-first search finds the optimal solution for this problem? Explain the rationale. (2 points)

Solution: Yes, since the action cost is same for all state-action pairs.

ii. Would depth-limited search with a depth limit of 12 returns the optimal solution? (2 points)

Solution: No, the depth-first search may return the solution at depth 12.

iii. Between BFS and Uniform Cost Search (UCS), which would be more memory-efficient for this problem? Justify your response. (2 points)

Solution: UCS performs delayed goal check leading to higher memory complexity than BFS

(b) (9 points) **Informed Search**

Consider two heuristic functions for position (x_i, y_i) and goal (x_G, y_G).

- Chebyshev Distance: $h_1((x_i, y_i)) = \max(|x_i - x_G|, |y_i - y_G|)$
- Minkowski Distance: $h_2((x_i, y_i)) = (|x_i - x_G|^p + |y_i - y_G|^p)^{\frac{1}{p}}$

i. Compute $h_1(S)$ and $h_2(S)$ for the Start position S. Assume $p = 2$ for the Minkowski distance. Show your calculations. (2 points)

Solution:

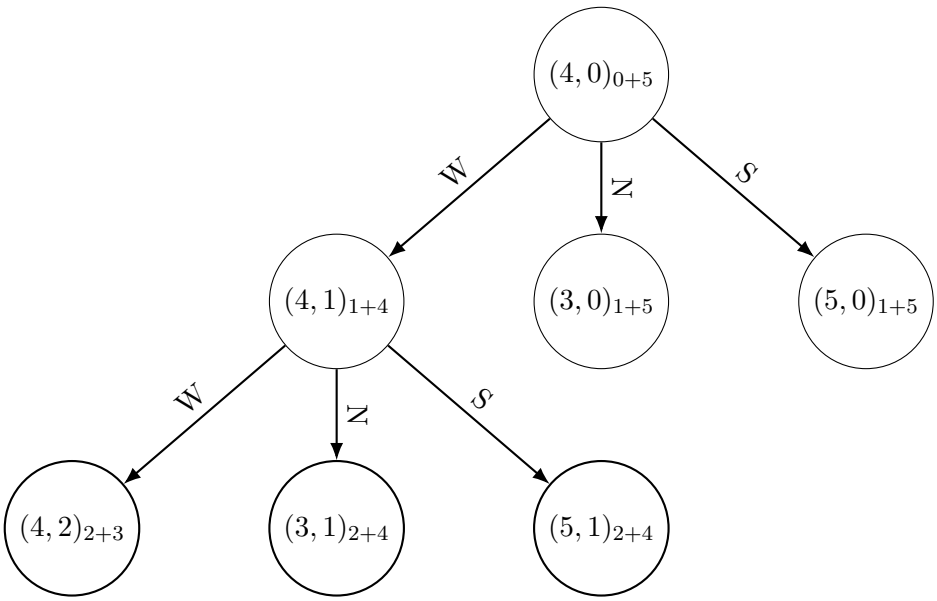
$$h_1(S) = \max(|4-1|, |0-5|) = 5$$
$$h_2(S) = \sqrt{|4-1|^2 + |0-5|^2} = 5.83$$

ii. State and explain whether the heuristic h_1 is admissible or not. (2 points)

Solution: We know that Manhattan distance is the minimum steps needed if no obstacles existed and it is an admissible heuristic due to less restricted problem setting. Chebyshev distance is always less than or equal to Manhattan distance. Hence Chebyshev distance is also admissible.

iii. Draw the search tree with h_1 as the heuristic for A* search algorithm. Draw it till the *reached* data structure has seven elements. Annotate each node with f-cost in the following format: $(x, y)_{i+j}$ where (x, y) is the location, i is the path cost, and j is the heuristic cost. Assume that the actions are taken in the following order: East, West, North, South. If multiple nodes in the frontier has same f-cost, pop the node with least heuristic cost. If multiple nodes have the same f-cost and heuristic cost, pop the element that entered the frontier last. (5 points)

Solution:



7. (15 points) **Logic**

Aether (A), Barbara (B), and Chongyun (C) are each choosing a souvenir from X, Y, and Z. We are given only the following four pieces of information:

R1: Each person chooses exactly one souvenir

R2: Each souvenir is chosen by exactly one person.

R3: Barbara chooses Y, or Chongyun chooses Z

R4: If Barbara chooses Y, then neither Aether nor Chongyun chooses Z

R5: If Chongyun chooses Z, then Aether does not choose X.

You are supposed to use the following propositional symbols:

aX : Aether chooses X	aY : Aether chooses Y	aZ : Aether chooses Z
bX : Barbara chooses X	bY : Barbara chooses Y	bZ : Barbara chooses Z
cX : Chongyun chooses X	cY : Chongyun chooses Y	cZ : Chongyun chooses Z

Table 1: Propositional Symbols for Souvenir Problem

To help you along, we have defined the "exactly one" using the macro below. You may use it directly by substituting x_i with the propositional symbols defined above.

$$E1(x_1, x_2, x_3) = (x_1 \vee x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee \neg x_3) \wedge (\neg x_2 \vee \neg x_3) \quad (1)$$

Answer the following questions.

(a) (5 points) Translate **R1**–**R5** into propositional form using the symbols in Table 1.

Solution:

- **R1** : $E1(aX, aY, aZ) \wedge E1(bX, bY, bZ) \wedge E1(cX, cY, cZ)$
- **R2** : $E1(aX, bX, cX) \wedge E1(aY, bY, cY) \wedge E1(aZ, bZ, cZ)$
- **R3** : $bY \vee cZ$
- **R4** : $bY \implies (\neg aZ \wedge \neg cZ)$
- **R5** : $cZ \implies \neg aX$

(b) (2 points) Could you represent the above knowledge using definite clauses? Explain the rationale.

Solution: We cannot represent this knowledge in definite clausal form. For example, **R3** has two positive literals.

(c) (3 points) Convert the formulas from Question 7(a) into Conjunctive Normal Form (CNF). Formulas already in CNF do not need to be converted.

Solution:

- **R1** : Already in CNF
- **R2** : Already in CNF
- **R3** : Already in CNF
- **R4** : $(\neg bY \vee \neg aZ) \wedge (\neg bY \vee \neg cZ)$
- **R5** : $\neg cZ \vee \neg aX$

(d) (5 points) Identify the number of models of the sentence **R3**.

Solution: The sentence **R3** is true for three assignments of the variables bY and cZ as shown below.

bY	cZ	$bY \vee cZ$
T	T	T
T	F	T
F	T	T
F	F	F

There are nine variables. Remaining seven variables can take any value for each of the above three assigments. Therefore, the total number of models of the above system is $3 * 2^7 = 3 * 128 = 384$

8. (10 points) Merlimart, a popular e-commerce platform, is experiencing intermittent spikes in end-to-end latency and occasional error responses during peak hours. Incidents occur roughly a few times per week, often during marketing campaigns. The site reliability engineer (SRE) must quickly diagnose the most likely root cause to decide which mitigation to apply (scale application servers, throttle database-heavy features, or reroute traffic/adjust network QoS). Prior expertise of the SRE leads to three candidate root causes:

   - C_1: Application server overload
   - C_2: Database (DB) lock waits
   - C_3: Network congestion

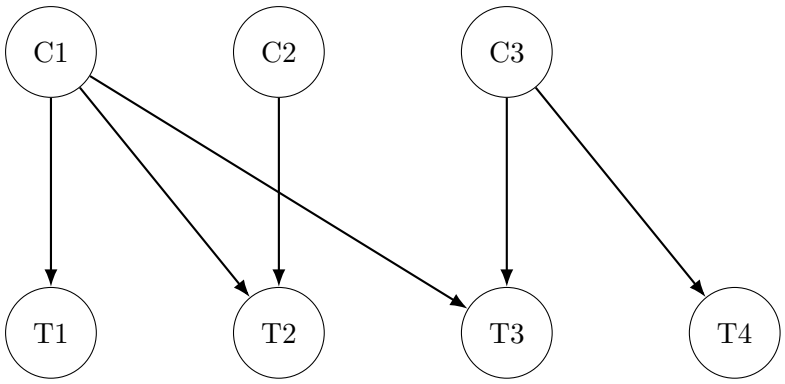
   The SRE believes that these three root causes are marginally independent. Furthermore, there are four symptoms that the engineer can check to identify the most probable cause(s).

   - T_1: CPU throttling alerts.
   - T_2: Slow page loads
   - T_3: API call timeouts
   - T_4: High packet loss

   CPU throttling alerts occur due to application server overload; slow page loads are due to DB lock waits and application server overload; API call timeouts are due to application server overload and network congestion; high packet loss is only due to network congestion.

   (a) (2 points) Draw the Bayesian network for this problem

Solution:



   (b) (2 points) Assuming the lack of prior knowledge of the dependencies or structure, identify the number of parameters required. Explain the rationale.

Solution: If prior knowledge about the dependencies among the variables is unavailable, then we need $2^N - 1$ parameters, where N is the number of nodes. Here $N = 7$. So this problem requires 127 parameters.

(c) (3 points) Identify the minimal number of parameters required to define the above Bayesian network. Show the working.

Solution:

Number of parameters needed for each node is 2^k, where k is the number of parents of the node.

C_1, C_2, and C_3 have no parents. Each of them requires one parameter.

T_1 and T_4 has one parent each. They each need two parameters

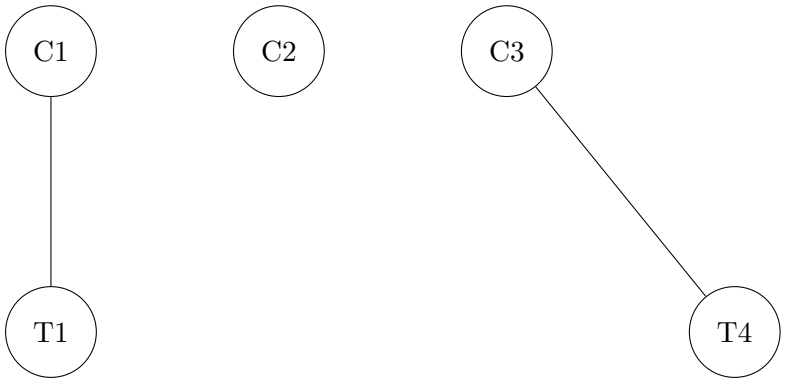
T_2 and T_3 has two parents each. They each need four parameters

So the total number of parameters needed to define this Bayesian network are 15.

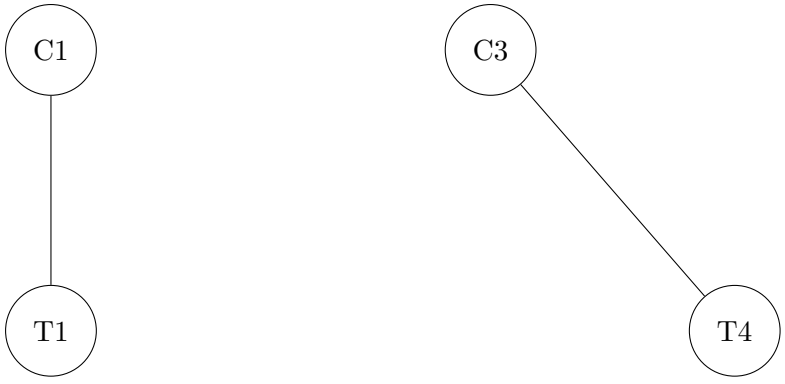
(d) (3 points) Is T_1 is independent of T_4 given C_2?

Solution:

Moralized ancestral undirected subgraph:



After removing evidence $C2$:



There exists no path between $T1$ and $T2$ after removing evidence $C2$. Hence the statement is True.

Appendix

```
function BEST-FIRST-SEARCH(problem, f) returns a solution node or failure
  node ← NODE(STATE=problem.INITIAL)
  frontier ← a priority queue ordered by f, with node as an element
  reached ← a lookup table, with one entry with key problem.INITIAL and value node
  while not IS-EMPTY(frontier) do
    node ← POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    for each child in EXPAND(problem, node) do
      s ← child.STATE
      if s is not in reached or child.PATH-COST < reached[s].PATH-COST then
        reached[s] ← child
        add child to frontier
  return failure

function EXPAND(problem, node) yields nodes
  s ← node.STATE
  for each action in problem.ACTIONS(s) do
    s' ← problem.RESULT(s, action)
    cost ← node.PATH-COST + problem.ACTION-COST(s, action, s')
    yield NODE(STATE=s', PARENT=node, ACTION=action, PATH-COST=cost)
```

Figure 2: Best-First Search Algorithm

```
function BREADTH-FIRST-SEARCH(problem) returns a solution node or failure
  node ← NODE(problem.INITIAL)
  if problem.IS-GOAL(node.STATE) then return node
  frontier ← a FIFO queue, with node as an element
  reached ← {problem.INITIAL}
  while not IS-EMPTY(frontier) do
    node ← POP(frontier)
    for each child in EXPAND(problem, node) do
      s ← child.STATE
      if problem.IS-GOAL(s) then return child
      if s is not in reached then
        add s to reached
        add child to frontier
  return failure

function UNIFORM-COST-SEARCH(problem) returns a solution node, or failure
  return BEST-FIRST-SEARCH(problem, PATH-COST)
```

Figure 3: Breadth-First Search and Uniform Cost Search

```
function ITERATIVE-DEEPENING-SEARCH(problem) returns a solution node or failure
  for depth = 0 to ∞ do
    result ← DEPTH-LIMITED-SEARCH(problem, depth)
    if result ≠ cutoff then return result

function DEPTH-LIMITED-SEARCH(problem, ℓ) returns a node or failure or cutoff
  frontier ← a LIFO queue (stack) with NODE(problem.INITIAL) as an element
  result ← failure
  while not IS-EMPTY(frontier) do
    node ← POP(frontier)
    if problem.IS-GOAL(node.STATE) then return node
    if DEPTH(node) > ℓ then
      result ← cutoff
    else if not IS-CYCLE(node) do
      for each child in EXPAND(problem, node) do
        add child to frontier
  return result
```

Figure 4: Iterative Deepening Search and Depth Limited Search

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Figure 5: Logical Equivalences

Rough Work

=== END OF PAPER ===

Rough Work