

**NATIONAL UNIVERSITY OF SINGAPORE**  
**IT5005 – ARTIFICIAL INTELLIGENCE**

MID-TERM TEST  
(Semester II: AY2024/25)

Time Allowed: 1 Hour

---

**INSTRUCTIONS**

1. This assessment paper contains **NINE (9)** questions in **TWO (2)** parts and comprises **SIX (6)** printed pages.
2. This is a **CLOSED BOOK** assessment.
3. Only an A4 cheat sheet is allowed. Apart from calculators, electronic devices are not allowed.
4. Answer **ALL** questions and write your answers only on the **ANSWER SHEET** provided.
5. Do **not** write your name on the ANSWER SHEET.
6. The maximum mark of this assessment is 50.
7. Pseudocodes are provided in the Appendix (Pages 5-6) at the end.

Question	Max. mark
Part A: Q1 – 5	10
Part B: Q6	5
Part B: Q7	15
Part B: Q8	15
Part B: Q9	10
<b>Total</b>	<b>50</b>

——— **END OF INSTRUCTIONS** ———

**Part A: Multiple Choice Questions** [Total:  $5 \times 2 = 10$  marks]

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

1. Which of the following statements correctly describes the depth-first search algorithms?
  - A. If the state space is a graph with unique state at each vertex, depth-first search always provides optimal solution
  - B. If the state space is a tree with unique state at each vertex, depth-first search always provides optimal solution
  - C. If the state space is a tree with unique state at each vertex, depth-first search is not guaranteed to provide optimal solution
  - D. If the state space is a graph with unique state at each vertex, depth-first search with cycle-check always provides optimal solution
  - E. None of the options (A), (B), (C), (D) are correct.
  
2. Let  $h_1$  and  $h_2$  be consistent and admissible heuristics for a given problem. Which of the following statements is true for that problem?
  - A.  $h_1 + h_2$  is guaranteed to be admissible and consistent
  - B.  $0.9 * h_1 + 0.9 * h_2$  is guaranteed to be admissible and consistent
  - C. Both  $0.9 * h_1$  and  $0.9 * h_2$  are guaranteed to be admissible and consistent
  - D. Both  $0.9 * h_1$  and  $0.9 * h_2$  are guaranteed to be admissible, but not guaranteed to be consistent
  - E. None of the options (A), (B), (C), (D) are correct.
  
3. Consider the joint probability table for three variables shown on the right. The variable *Weather* takes two values *Sunny* and *Rainy* ; the variable *Temperature* takes two values *Hot* and *Cold* ; the variable *Activity* takes two values *Picnic* and *Movie* . What is the probability of having a *Hot Temperature* , regardless of the *Weather* and *Activity*?
 

<i>Weather</i>	<i>Temperature</i>	<i>Activity</i>	<i>Probability</i>
<i>Sunny</i>	<i>Hot</i>	<i>Picnic</i>	0.10
<i>Sunny</i>	<i>Hot</i>	<i>Movie</i>	0.05
<i>Sunny</i>	<i>Cold</i>	<i>Picnic</i>	0.10
<i>Sunny</i>	<i>Cold</i>	<i>Movie</i>	0.15
<i>Rainy</i>	<i>Hot</i>	<i>Picnic</i>	0.05
<i>Rainy</i>	<i>Hot</i>	<i>Movie</i>	0.10
<i>Rainy</i>	<i>Cold</i>	<i>Picnic</i>	0.05
<i>Rainy</i>	<i>Cold</i>	<i>Movie</i>	0.40

  - A. 0.15
  - B. 0.1
  - C. 0.3
  - D. 0.25
  - E. None of the options (A), (B), (C), (D) are correct.
  
4. What does  $A \vdash_i B$  mean?
  - A.  $B$  logically follows  $A$ .
  - B.  $B$  is True if  $A$  is True.
  - C.  $A$  is True only if  $B$  is True
  - D.  $B$  is derived from  $A$  using a specific algorithm.

E. None of the options A, B, C, and D are correct.

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

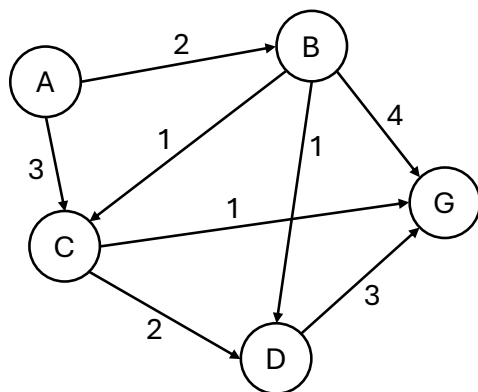
- A.  $P \Rightarrow (P \vee Q)$
- B.  $(P \wedge Q) \Rightarrow P$
- C.  $(P \Rightarrow P) \wedge Q$
- D.  $Q \vee (Q \Rightarrow P)$
- E.  $False \Rightarrow True$

**Part B: There are 4 questions in this part** [Total: 50 marks]

**6. [Total: 5 marks] Uninformed Search: Improving Best-First Search**

You are advised to use best-first search algorithm for a state space with uniform costs, i.e., the costs of all actions at all states are the same. Moreover, you are told that the state space is a tree. Briefly describe what changes would you make such that the best-first search algorithm is both optimal and efficient. Assume that you have no heuristics for this problem. The Best-First search algorithm is provided in the appendix.

**7. [Total: 10 marks] Informed Search: How do I get there?**



Node	$h_1$	$h_2$	$h_3$	$h_4$	$h_5$
A	4	3	6	4	2
B	2	2	5	1	1
C	1	1	3	1	0
D	3	3	5	2	1
G	0	0	0	0	0

Figure 1: UAV Graph and Heuristics

Path planning is critical in unmanned autonomous vehicles. In path planning, robots usually navigate from one location to another via a series of intermediate waypoints. However, the cost of moving from one waypoint to another may not be definitive, e.g., due to weather, obstacles, and other uncontrollable factors. Even so, it is possible to estimate the cost between every pair of waypoints. Let  $h$  be the estimated (heuristic cost) and  $g$  be the actual cost. Assume that if two nodes have the same cost/heuristic, then we tie break them alphabetically. For example, if C and D both exist in the priority queue with value 5 (this is an arbitrary number), then we pop C first before popping D. Suppose we have a drone to deliver a payload from A to G with the following waypoints and costs:

- (a) State whether the heuristics are admissible and consistent. If a heuristic is not admissible or inconsistent, you also need to provide the rationale; otherwise, you would get zero marks for your answer. [5 marks]
- (b) Suppose we have decided on heuristic  $h_1$  with A\* search algorithm, and we currently have expanded the nodes A and B, what is the next node to be expanded? [2 marks]
- (c) If you are using greedy best first search algorithm with  $h_1$  as heuristic, after popping node A from the priority queue, which node would be expanded next? [2 marks]
- (d) Built a dominant heuristic that is both admissible and consistent from the above heuristics. [1 mark]

### 8. [Total: 15 marks] Propositional Logic: Who Is Telling the Truth?

Three persons A, B, and C make the following statements:

**R1:** A says B is lying.

**R2:** B says C is lying.

**R3:** C says both A and B are lying.

We also know the following fact.

**R4:** Only one of the A, B, C is telling the truth.

The propositional symbols are defined in Table 1

Table 1: Propositional Symbols

Symbol	Definition
$a$	A is telling the truth
$b$	B is telling the truth
$c$	C is telling the truth

(a) Translate **R1** to **R4** into propositional forms. [4 marks]

(b) Convert the above propositional sentences to CNF [4 marks]

(c) Determine whether B is telling the truth or not using resolution-refutation. [7 marks]

### 9. [Total: 10 marks] Bayesian Networks

(a) Planners in the city of Rainville are exploring the relationship between weather conditions and traffic flow, particularly during weekends when the usual commuter patterns are less predictable. To gain further insights, they consulted a domain expert in urban traffic management. The expert suggested them to focus only on two variables *Weather* and *Traffic*. The expert further suggested the planners to restrict the domain of these variables. *Weather* is classified into two categories: *good* and *bad*. While the *Traffic* is also encoded to two values: *heavy* and *light*.

The expert believed that the *Traffic* would be *heavy* and the *Weather* would be *bad* with a probability of 0.1. He is also believed that *Traffic* would be *heavy* and the *Weather* would be *good* with a probability 0.2. While the expert was unable to provide precise probabilities for the remaining cases:

- *Weather* is *good* and *Traffic* is *light*,
- *Weather* is *bad* and *Traffic* is *light*.

However, he highlighted a crucial characteristic of the weekend traffic pattern: *Weather* and *Traffic* are independent during weekends. Given that *Weather* and *Traffic* are independent over weekend, find the probabilities that *Weather* is *good* while the *Traffic* is *light* and *Weather* is *bad* while the *Traffic* is *light*? [5 marks]

(b) Consider the Bayesian network shown in the Figure 2. [5 marks]

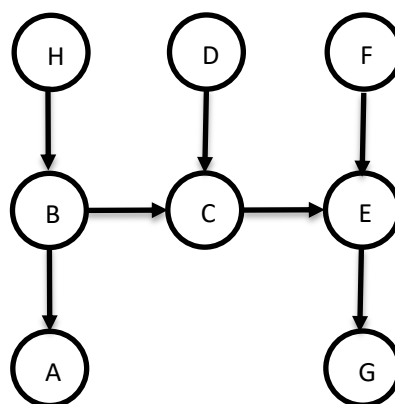


Figure 2: Bayesian Network

Prove/Disprove the following.

(i)  $H \perp F | G$

(ii)  $A \perp D | G$

**Appendix: Pseudocodes:****Search Algorithms:**

```

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

```

```

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

```

```

function BREADTH-FIRST-SEARCH(problem) returns a solution node or failure
  node  $\leftarrow$  NODE(problem.INITIAL)
  if problem.IS-GOAL(node.STATE) then return node
  frontier  $\leftarrow$  a FIFO queue, with node as an element
  reached  $\leftarrow$  {problem.INITIAL}
  while not IS-EMPTY(frontier) do
    node  $\leftarrow$  POP(frontier)
    for each child in EXPAND(problem, node) do
      s  $\leftarrow$  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 ITERATIVE-DEEPENING-SEARCH(problem) returns a solution node or failure
  for depth = 0 to  $\infty$  do
    result  $\leftarrow$  DEPTH-LIMITED-SEARCH(problem, depth)
    if result  $\neq$  cutoff then return result

```

```

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

```

### Logical Equivalences:

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

### Resolution-Refutation Algorithm:

```

function PL-RESOLUTION(KB,  $\alpha$ ) returns true or false
  inputs: KB, the knowledge base, a sentence in propositional logic
            $\alpha$ , the query, a sentence in propositional logic

  clauses  $\leftarrow$  the set of clauses in the CNF representation of  $KB \wedge \neg\alpha$ 
  new  $\leftarrow \{ \}$ 
  while true do
    for each pair of clauses  $C_i, C_j$  in clauses do
      resolvents  $\leftarrow$  PL-RESOLVE( $C_i, C_j$ )
      if resolvents contains the empty clause then return true
      new  $\leftarrow$  new  $\cup$  resolvents
    if new  $\subseteq$  clauses then return false
    clauses  $\leftarrow$  clauses  $\cup$  new

```

=== END OF PAPER ===