COMP 6721 Applied Artificial Intelligence (Winter 2026)

Assignment #1: State Space Search, Adversarial Search and Naive Bayes

Due: Sunday, February 15th, 2026, 11:59PM

State Space Search

Question 1 Omar loves chess and also likes AI. He decides to model a chess puzzle as a **state-space search** problem.

A **knight** moves in an L-shape as shown in Figure 1 (the green squares indicate the possible positions the knight can reach from doing one jump). In Figure 2, the knight starts in the middle of a 5×5 board at position $(3, 3)$, and four pawns are placed at the four corners. Omar wants the knight to **eat all four pawns using the minimum possible number of knight moves**.

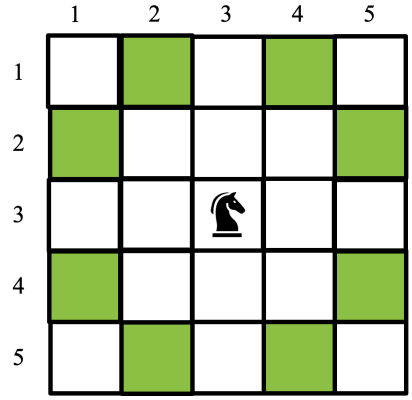

Figure 1: Knight movement (L-shape jump).

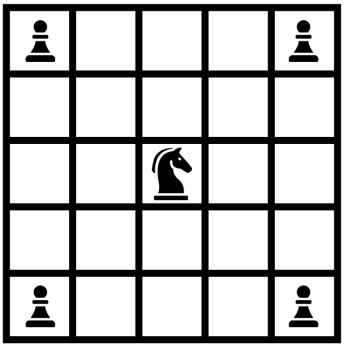

Figure 2: Initial configuration on a 5×5 board.

(a) Represent this problem as a **search problem**. Choose a suitable **state representation** as a tuple. **Note:** if a certain attribute is not impor-

tant for representing a certain state(s), you may use **X** to indicate "not important".

(b) Write down the **initial state** of the problem (as shown in Figure 2).

(c) Write down the **goal state**. Use **X** if something does not matter.

(d) Define the **actions** of the knight as a function of its current position (x, y).

(e) List all the states that exist at **level 1** of the search tree (i.e., after exactly one knight move from the initial state).

(f) Draw **one valid solution path** that allows the knight to eat all four pawns.

Question 2 Uninformed Search

Unless otherwise stated, when multiple nodes have equal priority, expand them in **alphabetical order**.

(a) **Worst-case node expansions (explicit counts + explanation).**
Consider a search problem whose state space is a **tree** with branching factor b, and assume that the shallowest goal node is located at depth d (root at depth 0). For each search strategy below, determine the **maximum number of nodes expanded** in the worst case.
Do not use Big-O notation. Instead:

- Give the explicit counting expression as a **sequence** and/or **closed form**.
- **Explain your reasoning** (which levels are fully expanded, what happens at depth d, and why this is worst-case).

(i) Breadth-First Search (BFS)

(ii) Depth-First Search (DFS)

(iii) Depth-Limited Search (DLS) with depth limit = d

(iv) Iterative Deepening Depth-First Search (IDDFS)

(b) **Graph search from S to G (show Open/Closed/Visited + final path).**
Suppose you need to find a path from **S** to **G** in the directed graph shown in Figure 4. The number on each directed edge is the cost of traversing that edge.

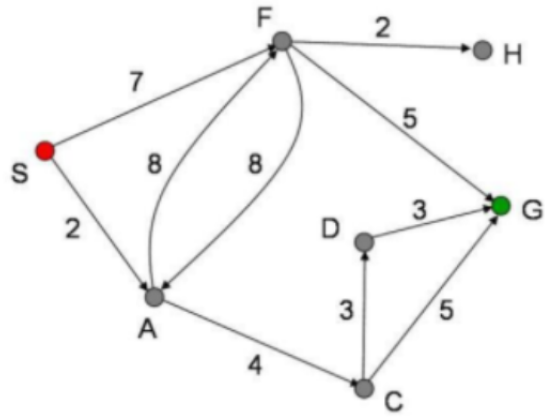

Figure 4: Directed graph for uninformed search (edge costs shown).

For each of the following search methods:

(i) Breadth-First Search (BFS)

(ii) Depth-First Search (DFS)

(iii) Iterative Deepening Search (IDS)

(iv) Uniform Cost Search (UCS)

Do NOT draw the search tree. Instead, **tabulate your work** step-by-step (in any neat table/list format you like) showing, at each step:

- **Expanded node** (the node removed for expansion),
- **Open list** (frontier, in the order it would be selected next),
- **Closed list** (expanded nodes),
- **Visited/Generated nodes** (all nodes discovered so far, including Open and Closed).

At termination, report:

- the **final path** found from S to G (list the states in order),
- the **total path cost**,
- and whether a solution was found (if not, state why it terminates).

Question 3 Informed Search

Consider the maze shown in Figure 5. From any cell, the successors include the adjacent cells to the **North, South, East, and West**, unless blocked by the maze boundary or a barrier (thick wall). Assume every move has cost 1. The goal is to find a path from cell **S** to cell **G**.

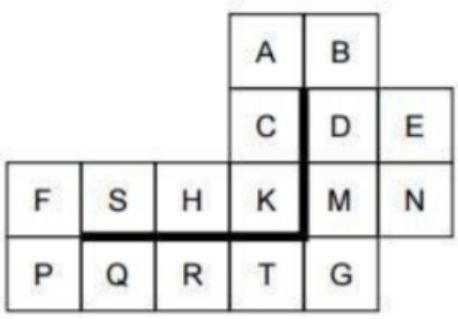

Figure 5: Maze for informed search (unit step costs).

Let the heuristic be

$$h(state) = \text{Manhattan distance from } state \text{ to } G$$

computed **as if there were no barriers**. For example, $h(K) = 2$ and $h(S) = 4$.

(a) **Greedy Best-First Search** using h. **Tabulate** step-by-step (any neat format) the **Expanded node**, **Open list** (ordered by h), **Closed list**, and **Visited/Generated nodes**. Give the **order of expansions** (include G if found), and report the **final path**.

(b) **Hill-Climbing Search** using h. Show the **sequence of selected/visited nodes** (what you move to each step) and the heuristic values. State whether the method reaches the goal or gets stuck (local minimum / plateau / ridge). If it reaches G, report the final path.

(c) **A* Search** using h. Use $f(n) = g(n) + h(n)$ and **remove redundant states** (do not expand a state more than once). **Tabulate** step-by-step the **Expanded node**, **Open list** (ordered by f, break ties alphabetically), **Closed list**, and **Visited/Generated nodes**. Give the **order of expansions** (include G if found), and report the **final path** and its **total cost**.

(d) Is h an **admissible** heuristic? Justify your answer.

(e) Is $h_2(state) = \min(2,\ h(state))$ an **admissible** heuristic? Justify your answer.

(f) Is $h_3(state) = \max(2,\ h(state))$ an **admissible** heuristic? Justify your answer.

Adversarial Search

(a) Consider the game tree shown in Figure 1. The static evaluation scores are given at the **leaf** nodes. Trapezoids that point **up** represent decision nodes for the **maximizing** player (MAX), while trapezoids that point **down** represent decision nodes for the **minimizing** player (MIN).

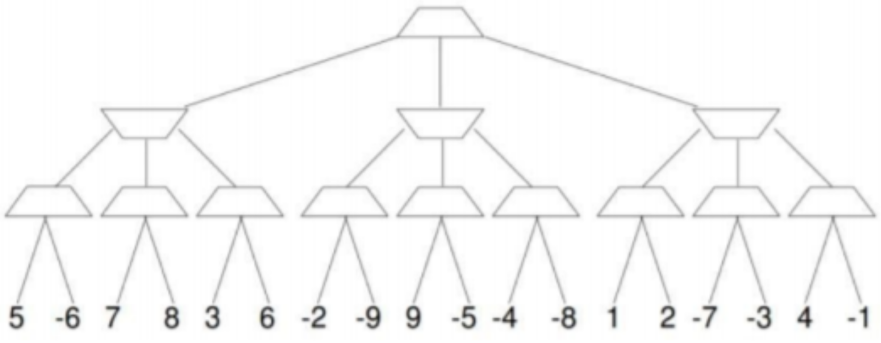

Figure 1: Game tree for Minimax and Alpha-Beta pruning.

(i) Apply the **Minimax** algorithm to the game tree. Clearly show the minimax value computed at each internal node, and indicate which move MAX should choose at the root.

(ii) Apply **Alpha-Beta pruning** to the same game tree. Clearly indicate which nodes do **not** need to be examined (pruned). Assume that children are examined strictly from **left to right**.

(b) Consider the two-player game shown in Figure 2. The game consists of a line of 4 spaces labeled $1, 2, 3, 4$ and two tokens (Player A and Player B). Use the following rules:

- Player A moves first.
- The players alternate turns.
- On each turn, a player must move their token to an **adjacent open** space (left or right), if such a move exists.
- If the opponent occupies an adjacent space, a player may **jump over** the opponent to the next open space (if any). For example, if A is on 3 and B is on 2, then A may move to 1 (if it is open).
- The game ends when one player reaches the **opposite end** of the board.
- If Player A reaches space 4 first, the game value for A is +1.
- If Player B reaches space 1 first, the game value for A is −1.

Figure 2: Starting position of the two-player token game.

(i) Draw the **complete game tree** for this problem starting from the initial state in Figure 2. Put each **terminal state** in a **square box** and write its game value (+1 or −1) in a **circle** next to it.

(ii) Apply the **Minimax** algorithm to your game tree and determine the optimal move for Player A from the initial state. Clearly show the minimax value at each internal node.

Bayes Classifier

(a) Consider the following hypothetical data concerning student characteristics in the department of Computer Science and whether or not each student was hired for an internship opportunity.

Name	Grades	Personal Coding Projects	Was Hired?
Emily	Low	Yes	Yes
Charles	Average	No	No
Sam	Average	No	No
Diana	Average	Yes	Yes
Melissa	High	Yes	Yes
Katie	High	Yes	No
Jacob	High	Yes	No
Henry	Low	No	No

Use a Naive Bayes classifier to determine whether or not someone with low grades and has personal coding projects will be hired for an internship opportunity.

(b) Consider the following set of recipes, where each ingredient can be classified as low, medium, or high in quantity:

Ingredient 1	Ingredient 2	Ingredient 3	Delicious Recipe?
low	high	low	Yes
high	low	high	Yes
low	medium	high	No
medium	high	low	No
high	high	medium	No

Using Naive Bayes, would the following proportion of ingredients lead to a delicious recipe?

Ingredient 1 = low, Ingredient 2 = high, Ingredient 3 = medium

(c) In this question, you will implement a Categorical Naive Bayes classifier from scratch to predict fraudulent transactions. To complete this question, you are provided with both a template Jupyter notebook file and the dataset in CSV format. All features in the given dataset are categorical. Complete the Jupyter notebook included with this assignment to implement your classifier. All the instructions needed to complete the code are provided in the notebook. Please insert screenshots of your classifier's performance metrics (accuracy scores and classification report) in your assignment PDF.

Important: To receive full marks for this question you MUST submit your completed Jupyter notebook (.ipynb file) alongside your PDF file for this assignment.