

Assignment 1

|  |  |
| --- | --- |
| **Course** | **Applied Artificial Intelligence** |
| **Instructor** | **Ms. Rushda Muneer** |
| **Section** | **BSE-6A** |

# ***Important Guidelines:***

1. Copying assignments from others is strictly prohibited. If an assignment shows more than 10% similarity to any source (the maximum allowable limit), it will result in a score of zero for both current and previous assignments.
2. Submit your assignment on time. Late submissions will incur a 25%-mark deduction if submitted within 6 hours, a 50%-mark deduction if submitted within 24 hours, and submissions beyond 24 hours will result in a score of zero.
3. No marks will be awarded to individuals who do not submit the file on Google Classroom (GCR).
4. Only submit your python files in a zipped folder. First, place all python files in a single folder (do not create any subfolders), rename the folder with your roll number, then zip it. **Failure to follow this format will result in a score of zero.**
5. In case of confusion please feel free to contact onemail [***l215843@lhr.nu.edu.pk***](mailto:l215843@lhr.nu.edu.pk)**or *rushda.muneer@lhr.nu.edu.pk***

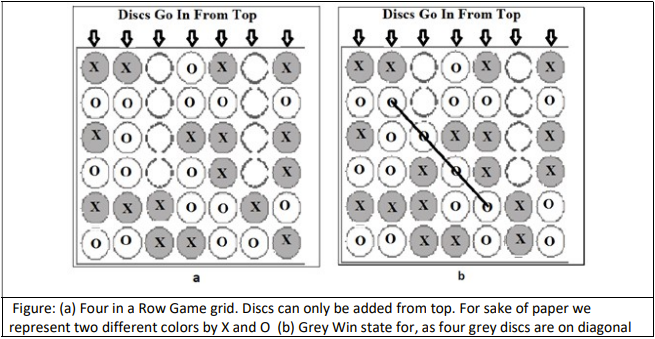
**Four in a Row Game - State Space Search Implementation**

**Background**

Four in a Row is a two-player connection game played on a 7-column, 6-row vertical grid. Players take turns dropping colored discs into columns, with pieces falling to the lowest available position.

**Game Rules**

1. Board is 7 columns × 6 rows
2. Players alternate turns, dropping one disc per turn
3. Discs fall to the lowest empty position in chosen column
4. First to connect 4 discs (vertical, horizontal, or diagonal) wins
5. Draw if board fills with no winner



**Implementation Tasks**

**Part 1: State Space Representation [20]**

Design a representation for:

* Game board state
* Valid moves
* State transitions (making a move)
* Goal state checking (win detection)

**Part 2: Search Algorithm Implementation [25x3]**

Implement three uninformed search strategies:

1. Breadth-First Search (BFS)
2. Depth-First Search (DFS)
3. Iterative Deepening Search (IDS)

Each implementation should:

* Find a winning sequence of moves if one exists
* Track number of nodes explored
* Track maximum depth reached
* Track time taken to find solution

**Part 3: Analysis Questions [30]**

1. Calculate the approximate number of nodes in the complete game tree for the first move (given average branching factor of 5)
2. Compare the performance of BFS, DFS, and IDS in terms of:
   * Memory usage
   * Time to find solution
   * Number of nodes explored
3. How will you store the game states and search results in files? Design a format specification.

**Part 4: File Operations [25]**

Implement functions to:

1. Save/load game states
2. Save search results including:
   * Search algorithm used
   * Solution path found
   * Performance metrics
   * Number of moves needed