

**T.C.**

**MARMARA UNIVERSITY**

**FACULTY of ENGINEERING**

**COMPUTER ENGINEERING DEPARTMENT**

CSE4082 Artificial Intelligence – Assignment 1

**Knight’s Tour Problem (KTP)**

Group Members

130319659 - Cihan Erdoğanyılmaz

150120998 - Abdelrahman Zahran

150120997 - Mohamad Nael Ayoubi

1. **Problem Definition**

The Knight's Tour Problem involves finding a sequence of moves for a knight on an n x n chessboard such that:

1. The knight visits each square exactly once.
2. The tour starts from a given square (e.g., a1), and a solution must traverse all cells on the board without revisiting any.

**Formal Definition:**

* State Space: Set of all reachable board positions.
* Initial State: Knight starting at position (0,0).
* Goal State: All board cells are visited exactly once (total moves = n \* n).
* Actions: Moves in L-shape, where a knight can move: {±2, ±1} or {±1, ±2} in the x and y directions.
* Transition Function: Valid moves must: Stay within board boundaries. Avoid revisiting already visited positions.

**Environment Classification:**

* Fully Observable: All board positions and moves are known.
* Deterministic: The next state is determined by the current state and chosen move.
* Discrete: Finite number of valid moves at each step.
* Single-agent: Only one knight moves on the board.
* Static: Board state does not change during the search process.

1. **Search Methods**

The program implements the following search algorithms using the general tree framework (treeSearch(Problem problem, String strategy) line 226 in KnightTour.java)

1. Breadth First Search: Explores all nodes level-by-level using a queue (FIFO). Ensures optimal solution (if exists) but may consume memory.
2. Depth First Search: Explores nodes depth-wise using a stack (LIFO).
3. DFS with Heuristic h1b [1]: Chooses moves that minimize the number of possible moves at the next step. (Warnsdorff's Rule)
4. DFS with Heuristic h1 [1]: Uses h1b for sorting but breaks ties based on the knight's distance to the nearest board corner.
5. **Design and Implementation**

The Java program contains the following components:

1. Node Class: Represents a single move of the knight with attributes:
   * x, y: Position on the board.
   * depth: Move count.
   * parent: Pointer to the previous node for backtracking.
   * Heuristic methods: calculateH1b() and nearestCornerDistance().
2. Problem Class: Contains the problem definition (board size, valid moves).
   * Methods: isGoal(Node node): Checks if the goal state is reached.
   * expand(Node node, heuristicType): Generates child nodes and applies heuristics for sorting.
   * isValid(Node node, int x, int y): Validates moves within board boundaries and checks for revisits.
3. Tree Search Function: Implements search strategies (BFS, DFS, DFS-h1b, DFS-h2):
   * Input: Search strategy, time constraint.
   * Output: Return status (A solution found, Timeout, Out of Memory, No solution exists). Solution path (if found). Number of nodes expanded. Execution time.
4. Main Method: Takes user inputs: board size (n), search method (a-d), and time constraint (t). Calls the appropriate search method and prints results.

General Tree Search Framework:

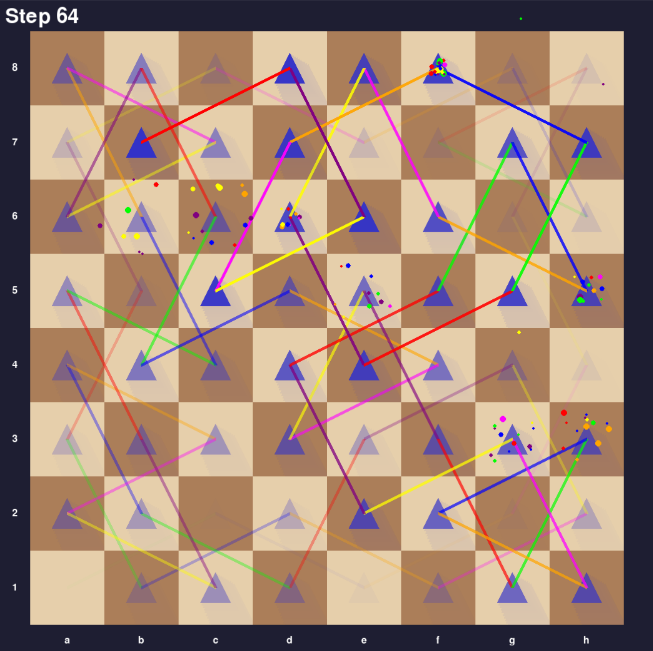
A close-up of text

Description automatically generated

1. **Results**

The results for each search method a-d, and for each board size n = 8, 16, 32, 41, and 52 with a time limit of 15 minutes are given in this section. Also, the output log files board.txt, path.txt, and moves.txt are also generated by the program and printed to terminal. The files of the runs below are included in the zip folder.

To run visualization using the python script feed the board.txt and path.txt generated by the Java code to Vizualization.py. Please make sure that these scripts are under the same folder with the mentioned text files.

Example of visualization is shown below

* 1. ***N = 8***

1. BFS
2. DFS

A number with black and orange numbers

Description automatically generated with medium confidence

A number grid with numbers

Description automatically generated

1. DFS-h1b



A number grid with black numbers

Description automatically generated

1. DFS-h2



A number grid with black numbers

Description automatically generated with medium confidence

* 1. ***N =16 (Nael)***

1. BFS



1. DFS
2. DFS-h1b

A number with black text

Description automatically generated with medium confidence

A grid of numbers

Description automatically generated

1. DFS-h2



A grid of numbers

Description automatically generated

* 1. ***N=32***

1. BFS



1. DFS



1. DFS-h1b

A number with black text

Description automatically generated with medium confidence

A grid of numbers

Description automatically generated

1. DFS-h2

A number on a white background

Description automatically generated

A grid of numbers

Description automatically generated

* 1. ***N=41 (Nael)***
  2. ***N=52***

1. **Maximum Board Size**

The maximum board size that our program can find a solution within 15 minutes using DFS-h2 is:

Output: