User Manual for The Knight's Tour (10x10 Board)

Explore The Knight's Tour with Lisp

Follow these steps to embark on The Knight's Tour adventure on a 10x10 board using your Lisp interpreter.

1. Open Lisp Interpreter:

Launch your Lisp interpreter in the terminal or command prompt. Here are some popular Lisp implementations to consider:

1. **SBCL (Steel Bank Common Lisp):**

- A high-performance, open-source Common Lisp implementation known for its speed and active maintenance.

2. **Clozure CL (CCL):**

- A fast, mature, and open-source Common Lisp implementation supporting native threads and providing good performance.

3. **GNU CLISP:**

- A notable implementation with portability across various platforms, featuring an interactive environment (REPL) and a bytecode compiler.

4. **ECL (Embeddable Common Lisp):**

- A lightweight, embeddable Common Lisp implementation designed for easy integration into other applications, supporting multiple backends.

5. **Allegro CL:**

- A commercial Common Lisp implementation recognized for its speed and advanced features, including a native compiler and multi-threading support.

6. **LispWorks:**

- A commercial implementation offering a comprehensive development environment, supporting various platforms, and providing advanced debugging and profiling tools.

These are just a few examples; there are many other Lisp implementations and dialects. The choice depends on factors such as performance requirements, platform compatibility, and personal preferences.

2. Load and Compile Files:

Load and compile all the necessary files with the following command:

```
CL-USER> (load "~/Desktop/IA/proj2/interact.lisp")
```

3. Run the Program:

Initiate the main program using the following command:

```
CL-USER> (main :time-limit 10 :depth 15)
```

- **time limit**: The specified time limit (in seconds) within which the alpha-beta algorithm should provide the next move. Choose any value greater than 0.
- **depth**: The depth limit, indicating how deeply the alpha-beta algorithm should explore potential moves. Adjust this parameter based on your preferences and system capabilities.

4. Choose the Game Mode:

Upon executing the main function, a prompt application will launch, prompting you to select your desired game mode:

```
GAME MODE:

1) Player vs AI
2) AI vs AI

Enter a number >
```

5. Player vs AI

When opting for the Player vs AI game mode, the player engages in a match against the alpha-beta algorithm facilitated by a prompt application that manages the game flow. Upon any modification to the game board or adjustments to both players' scores, the updated information is promptly displayed in the terminal, providing the player with a comprehensive overview of the ongoing game.

During the player's turn, all moves made by the player are showcased on the terminal interface, allowing for clear visibility of each move. The player is required to input the desired move to advance the game. On the AI's turn, the algorithm invokes the alpha-beta algorithm to compute the optimal next move and promptly presents it in the terminal using the same structured format. This ensures a seamless and informative gaming experience for the player.

```
   | A  B  C  D  E  F  G  H  I  J
---|------------------------------
01 | 00 01 02 03 04 05 06 07 08 ▣1
02 | 10 11 12 13 14 15 16 17 18 19
03 | 20 21 22 23 24 25 26 27 28 29
04 | 30 31 32 33 34 35 36 37 38 39
05 | 40 41 42 43 44 45 46 47 48 49
06 | 50 51 52 53 54 55 56 57 58 59
07 | 60 61 62 63 64 65 66 67 68 69
08 | 70 71 72 73 74 75 76 77 78 79
09 | 80 81 82 83 84 85 86 87 -- 89
10 | -- 91 92 93 94 95 96 97 98 ▣2

Player score: 09
Ai score: 99

############## PLAYER ##############

1) I3 (17)
2) H2 (28)

Enter a number >
```

6. AI vs AI

If the AI vs AI game mode is selected, the program will initiate a match between two versions of the alpha-beta algorithm, with each AI taking turns. Similar to the Player vs AI game mode, after each player move, the updated board and their respective scores will be displayed on the terminal.

```
Move player1:

   | A  B  C  D  E  F  G  H  I  J
---|------------------------------
01 | 00 01 02 03 04 05 06 07 08 --
02 | 10 11 12 13 14 15 16 ▣1 18 19
03 | 20 21 22 23 24 25 26 27 28 29
04 | 30 31 32 33 34 35 36 37 38 39
05 | 40 41 42 43 44 45 46 47 48 49
06 | 50 51 52 53 54 55 56 57 58 59
07 | 60 61 62 63 64 65 66 67 68 69
08 | 70 -- 72 73 74 75 76 77 78 79
09 | 80 81 82 83 84 85 86 87 -- 89
10 | -- 91 92 93 94 95 96 97 98 ▣2

player1 score: 26
player2 score: 99

Move player2:

   | A  B  C  D  E  F  G  H  I  J
---|------------------------------
01 | 00 01 02 03 04 05 06 07 08 --
02 | 10 11 12 13 14 15 16 ▣1 18 19
03 | 20 21 22 23 24 25 26 -- 28 29
04 | 30 31 32 33 34 35 36 37 38 39
05 | 40 41 42 43 44 45 46 47 48 49
06 | 50 51 52 53 54 55 56 57 58 59
07 | 60 61 62 63 64 65 66 67 68 69
08 | 70 -- 72 73 74 75 76 77 ▣2 79
09 | 80 81 82 83 84 85 86 87 -- 89
10 | -- 91 92 93 94 95 96 97 98 --

player1 score: 26
player2 score: 177

( . . . )
```

This pattern continues, showcasing the progression of the AI vs AI match with each move.

7. Metrics

All game logs will be written to the file `log.dat` with the following structured format:

```
######################################
### Number of Nodes Analyzed: --
### Number of Min Cuts: --
### Number of Max Cuts: --
### Time (s): --
######################################

   | A  B  C  D  E  F  G  H  I  J
---|------------------------------
01 | 00 01 02 03 04 05 06 07 08 ▣1
02 | 10 11 12 13 14 15 16 17 18 19
03 | 20 21 22 23 24 25 26 27 28 29
04 | 30 31 32 33 34 35 36 37 38 39
05 | 40 41 42 43 44 45 46 47 48 49
06 | 50 51 52 53 54 55 56 57 58 59
07 | 60 61 62 63 64 65 66 67 68 69
08 | 70 71 72 73 74 75 76 77 78 79
09 | 80 81 82 83 84 85 86 87 -- 89
10 | -- 91 92 93 94 95 96 97 98 ▣2

######################################

( . . . )
```

Each block represents the metrics of a single execution of the alpha-beta algorithm. The placeholders such as "--" indicate where specific metrics values would be recorded, providing a clear structure for logging and analyzing the algorithm's performance.