PLAY CHESSKOBAN

ARTIFICIAL INTELLIGENCE

GROUP_A1_84

MENU

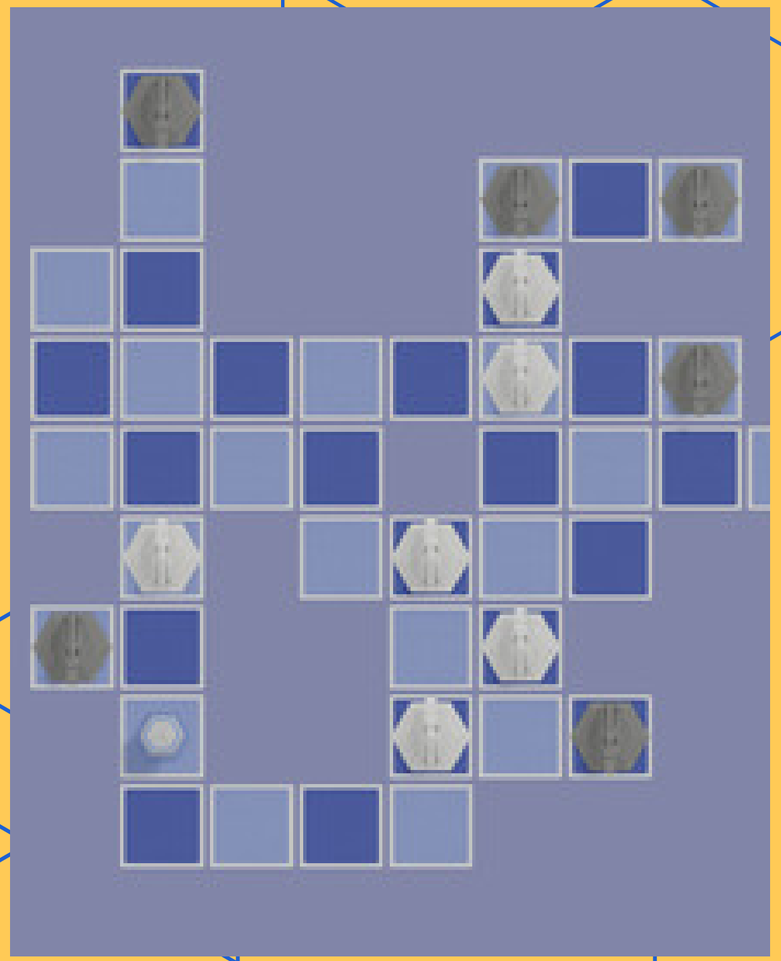
START

HIGH SCORES

MANUEL SILVA - 202108874
MARIA LARANJEIRA- 202004453
MIGUEL NOGUEIRA - 202005488

Game Definition

- Chesskoban is a single-player game that is a mix between a chess match and a pushing puzzle.
- Set on a chessboard, the aim is to use your king to push white knights into positions where they can 'capture' black knights, following the rules of chess movements.
- Victory is achieved when the white knights are strategically placed to 'check' all black knights using their classic L-shaped moves.



Related Work

- All the information related to game rules and about how the game works has been found on its official page on the steam store: https://store.steampowered.com/app/1784220/Chesskoban__Chess_Puzzles/

- In order to get started with pygame, which we employ in this project, we studied its documentation page: https://www.pygame.org/docs/

Search Problem

State Representation

The chessboard is represented as a matrix with each cell marked as a white, black, or empty space. We also have an list/tuple with the positions of the pieces - white knights, black knights, and the king.

Initial State

A variable chessboard configuration with specific starting positions for the white knights, black knights, and the king.

Objective Test

All white knights are positioned to capture all black knights, using the traditional knight's L-shaped move in chess.

Heuristic Function

A function to calculate the minimum number of moves required for the king to position the white knights optimally, enabling them to eliminate the black knights.

Search Problem

Operators

Move King

Relocate the king one square horizontally or vertically onto an empty space.

- **Preconditions:** The target square must be empty and within the bounds of the board.
- **Effects:** The king is repositioned.
- **Cost:** Typically, one move of the king counts as a single unit of cost.

Push Knight

The king moves into a square occupied by a white knight, which propels the knight in the same direction to an adjacent space.

- **Preconditions:** The adjacent square in the direction of the push must be empty.
- **Effects:** The white knight and the king occupies a new square.
- **Cost:** Each push has a cost of one unit.

Implementation work

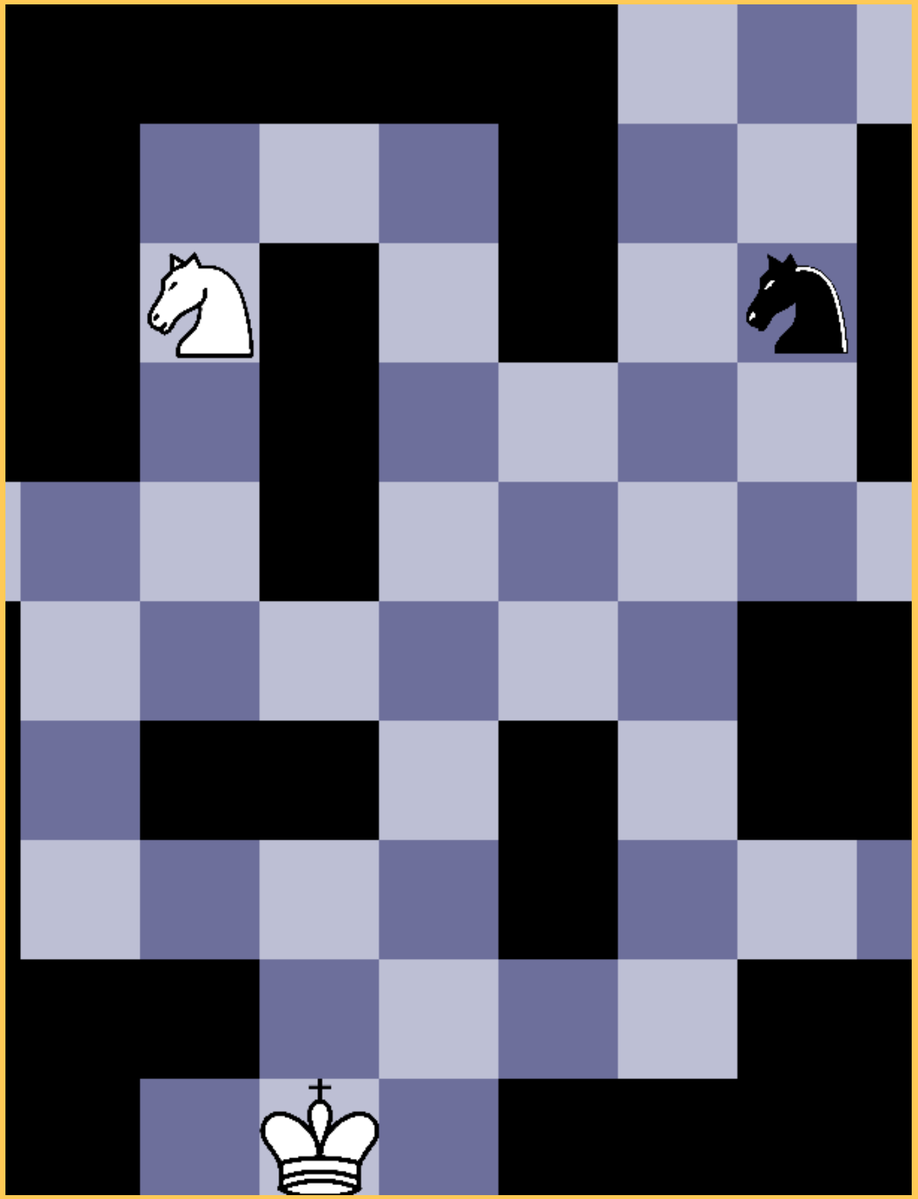
Programming Language and Environement

Development in **VSCode**, using **Python** and the **Pygame** library for graphical representation and interaction.

Data Structures and Work Developed

- Use of matrices to represent the chessboard and lists/tuples for piece positions.
- Initial progress includes implementing the board visuals, king movement mechanics, and the game menu.
- Currently refining the algorithm for the movement of knights and the victory state.

```
LEVEL2 = [
    [None, 'w', None],
    ['w', 'b', 'w'],
    [None, 'w', None]
]
```



Approach

- The game has three different difficulty levels, with a possible solution.
- After finishing a level, players can view the number of moves made during a gameplay and the corresponding number of moves and time calculated by different algorithms.
- Informed algorithms, with a better heuristic often yield more efficient results by strategically estimating the cost to reach the goal.

Implementated Algorithms

Informed Search

- **A* Algorithmn:**

Optimizes pathfinding for the horse using the Manhattan heuristic to balance between directness and exploration.

Uninformed Search

- **Depth First Search (DFS):**

Navigates complex, open boards by exploring paths deeply before backtracking.

- **Breadth First Search (BFS):**

Always finds the shortest path in dense scenarios by expanding outward from the start node.

Experimental Results

Level 1

```
A* Time: 0.00016810
DFS Time: 0.00015140
BFS Time: 0.00017110
```

LEVEL 1

Your Moves: 7

A* Moves: 7

DFS Moves: 7

BFS Moves: 7

Level 2

```
A* Time: 0.00022500
DFS Time: 0.00035490
BFS Time: 0.00028690
```

LEVEL 2

Your Moves: 19

A* Moves: 17

DFS Moves: 23

BFS Moves: 17

Level 3

```
A* Time: 0.00032490
DFS Time: 0.00036580
BFS Time: 0.00298070
```

LEVEL 3

Your Moves: 29

A* Moves: 29

DFS Moves: 45

BFS Moves: 29

Conclusions

The experimental analysis reveals the trade-offs between algorithm complexity and performance. A* and BFS consistently find the shortest path, with A* showing better performance in larger boards due to its heuristic guidance. DFS, while not always efficient, proves effective in sparser boards.

This investigation underlines the importance of choosing the right algorithm based on the specific requirements and constraints of the game environment, balancing between efficiency, speed, and path optimality.

THANK YOU FOR PLAYING!

PLAY AGAIN?

YES

NO