TOPIC 6 - EinStein würfelt nicht! Puzzle Solver

Introduction

"I am convinced that He (God) does not play dice" – Albert Einstein

In 1926, Albert Einstein wrote this famous line to Max Born, to express his belief that the universe is governed by deterministic laws rather than randomness. Einstein disagreed with quantum mechanics, which was emerging as a new theory that described the world in terms of probabilities.

This quote later inspired a professor of applied mathematics in Germany, Ingo Althöfer, to design a game called *EinStein würfelt nicht!* (EWN), which means "Einstein does not play dice!" in English.

This game ironically revolves around rolling dice to move pieces on a board. However, just like what Albert Einstein said, the result of the game is not purely random, where strategy and decision making play a crucial role in determining the winner.

For more information about the rules and gameplay of *EinStein würfelt nicht!*, you may refer to its Wikipedia article.

However in this assignment, we are going to solve the EWN variant puzzle, that is single player and deterministics.

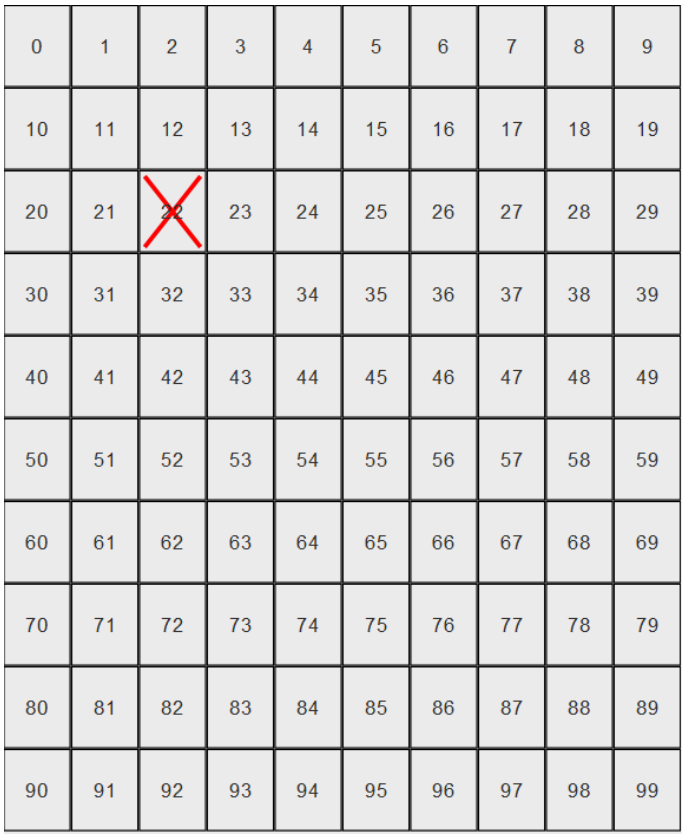
** **The material of this assignment can be obtained from this link** **

EWN Variant

Here are the rules of the EWN variant puzzles:

Board and pieces

1) Board size : 10 x 10
2) The square 22 is removed and no piece can enter
3) Pieces : 1,2,3,4,5,6



The image above shows the piece position on the board. Captured pieces are represented by -1

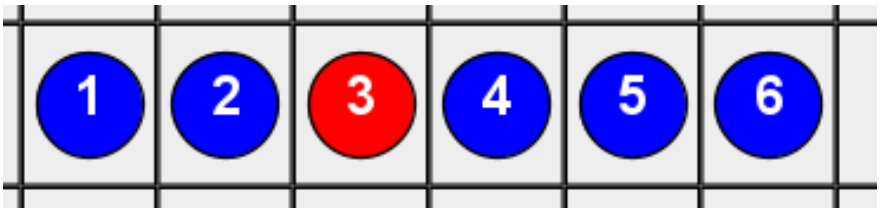
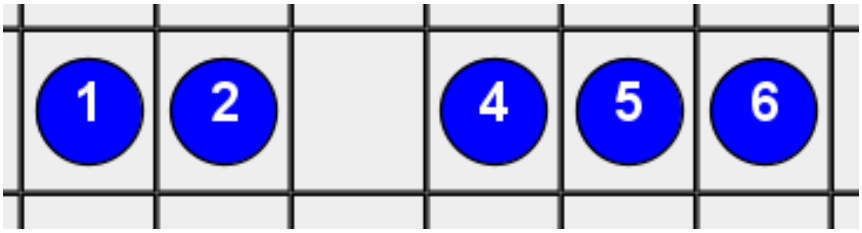
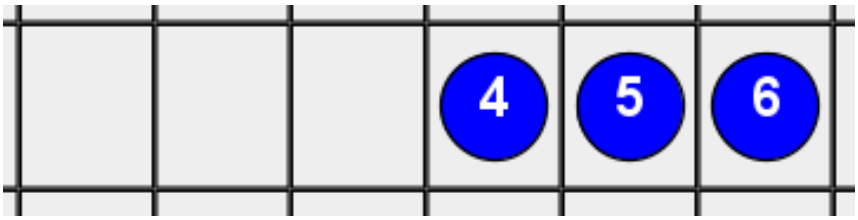
Determining the movable piece

1) The fixed dice sequence is given for each level.
   For each turn, the dice number is given by

   $$diceNumber = diceSequence[turnNumber]$$

2) If the dice number matches a piece that still exists on the board, the player can only move the matched piece.

3) If the dice number does not match any existing piece, the player can choose to move either one of these pieces:
    a) The piece that its number is the smallest but bigger than the dice number (If such piece exists)
    b) The piece that its number is the biggest but smaller than the dice number (If such piece exists)

Existing pieces	Dice Number	Movable piece(s)
	3	Only 3
	3	2 and 4
	3	Only 4

Movement Rules

1) In each turn, only one move can be made.
2) The movable piece can move to any of the 8 adjacent squares (similar to the King in chess).
3) If the destination square contains another piece, that piece is captured and removed from the board. The piece position of a captured piece will be -1.

Winning and Losing Conditions

1) Win: The given target piece reaches the square 0 within 30 moves.
2) Lose: The given target piece does not reach the square 0 after 30 moves.

Problem Statement

In this assignment, you are required to develop a simulator for the EWN variant puzzle. The simulator should be able to:

- Accept the input of the player mode and create the corresponding player object
- Accept the input of the level and load the relevant files
- Obtain the moves from the selected player
- Display whether the player has successfully solved the puzzle

You are provided with partially completed Java classes. Your task is to complete the functions as described below to make the simulator fully functional.

Basic Requirements (8 marks)

1) GameLoader.java - Constructor (0.5 mark)
    - Read data from the given filename and stores them in the appropriate variables in the GameLoader class

2) GameLoader.java - printGameDetails() (0.5 mark)
    - Print game setup details into “moves.txt”.

3) GameState.java - generatePossibleMoves() (2 marks)
    - Generate all possible moves based on the current piece positions and dice roll

4) GameState.java - isWinning() (1 mark)
    - Check whether the current pieces position fulfill the winning criteria

5) HumanPlayer.java - chooseMove() (1 mark)
    - Prompt the human player to choose the next move

6) RandomPlayer.java - chooseMove() (1 mark)
    - Randomly choose the moves to solve the puzzle

7) Player.java - printMove() (1 mark)
    - Print the chosen move of the player into the “moves.txt” file

8) GameMain.java - main() (1 mark)
    - Prompt the user to choose the mode (Human Player, Random Player and AI Player) and create a player object based on the selected player.

- Prompt the user to enter the name of the human player (Random Player and AI Player use default name, “Random Player” and ”AI Player” respectively)
- Prompt the user to select the level
- Call the chooseMove function of the player
- Show the result of the game (puzzle is solved or not)

Sample Input & Output

Sample Input file (level1.txt) :

```
2
21 32 43 34 23 12
3 3 5 4 1 3 2 3 1 6 6 6 1 4 3 3 6 4 2 5 6 2 4 3 2 6 3 2 6 2
```

Line 1 => Target piece
Line 2 => Initial position of all the 6 pieces
Line 3 => Dice Sequence

Sample Output file (moves.txt):

```
AI Player
5 4 4 6 4 4 1 2 4 1 1 1 6 5 2 1 5 2 4 1 2 1 6 4 2 2 2 1 1 6
3
76 44 74 46 77 73
-1 44 74 46 76 73
-1 44 74 35 76 73
```

Line 1 => Player name
Line 2 => Dice Sequence
Line 3 => Target Piece
Line 4 => Initial position of all the 6 pieces
Line 5 onwards => Position of the pieces after each move (Move1, Move 2, …) until the game ends

Visualization Tools

You can use the EinStein würfelt nicht! Viewer to visualise the puzzles.

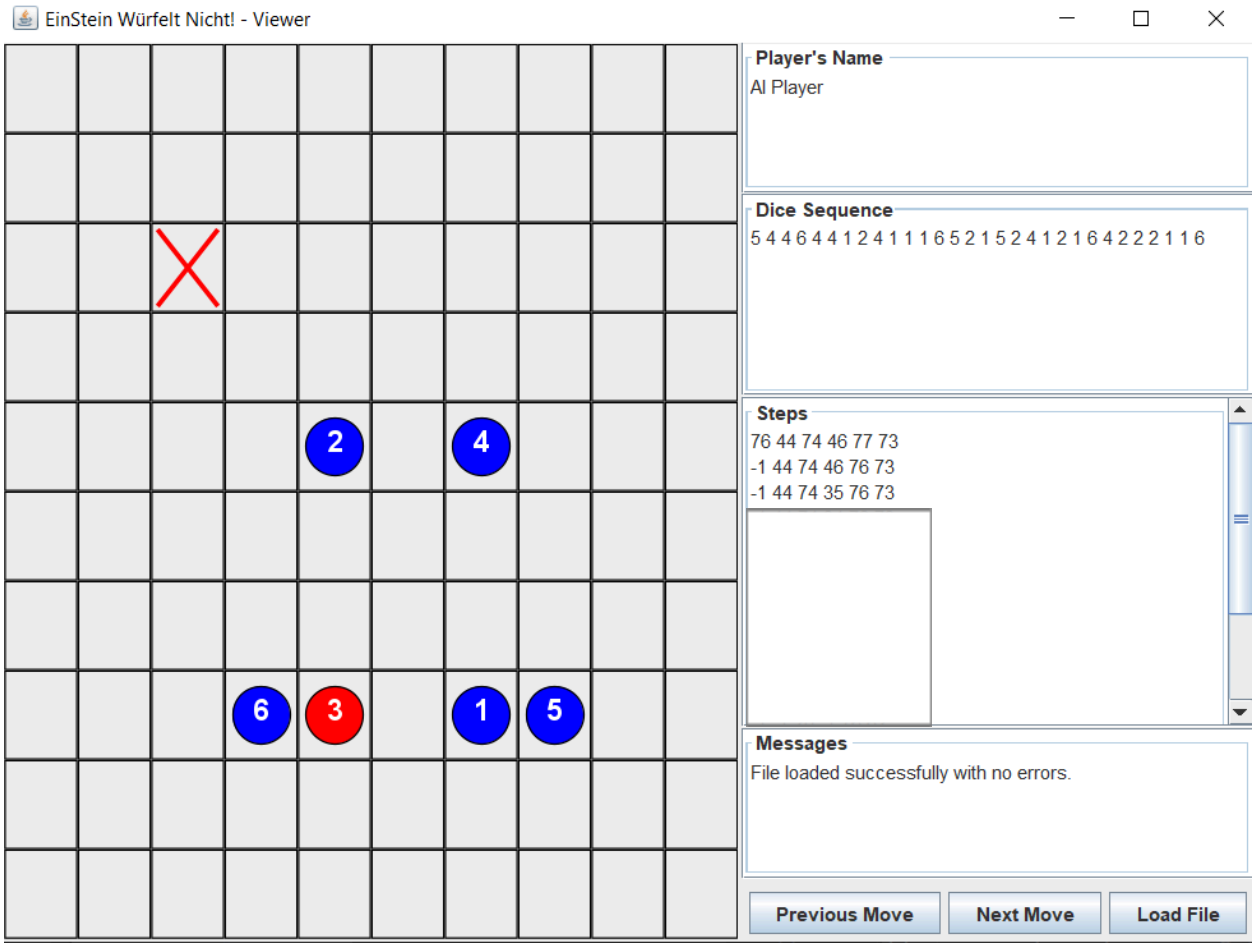
The program reads the "moves.txt" file based on the output format described above and displays the following information:

- Player's name
- Dice Sequence
- Pieces Position of the initial position and after each moves
- Message
- Piece positions on the board
- Target piece (highlighted in red)

The Message field will show the error message if there are illegal moves in your steps, invalid dice sequence, or any other errors are detected.

If no errors are detected, the message "File loaded successfully with no errors" will be shown.

Click the "Load File" button whenever there are any changes to the "moves.txt" file to update the piece positions.



Extra Challenges

Design and implement your own logic to automatically solve the given EWN puzzles.

Instructions:

1) Implement the chooseMove() function in the AIPlayer class from AIPlayer.java
2) You may use any suitable AI or algorithmic approach to solve the problem, including
    - Rule-based strategy
    - Dynamic programming
    - Greedy algorithm
    - Search-based algorithm
3) Hardcoding is strictly prohibited. The program must be able to make decisions dynamically based on the current game state.
4) Four puzzle levels are provided in the following files
    - level1.txt
    - level2.txt
    - level3.txt
    - level4.txt

Evaluation Criteria:

1) The puzzles must be solved within the maximum number of moves allowed for each level:

Level	Maximum Moves	Mark(s) allocation
Level 1	6	1
Level 2	10	1
Level 3	10	1
Level 4	15	1

2) Each level solved with execution time under 15s and not more than the maximum number of moves will be awarded 1 mark.

Comments

I was first introduced to the EWN game in a course that I have taken during my long term student exchange programme. The course focused on building the AI to play the single and double player board game. One of the homeworks was similar to this assignment, where the students were asked to design an algorithm written in C language that could solve the puzzles in under 10 seconds with extremely hard test cases.

This assignment is inspired by that experience, but don't worry, I have shifted the focus from algorithm design to building a puzzle simulator and the given test cases are at an easy level. It should be much easier and a great way to enhance your understanding of programming concepts.

Here are some tips and notes:

1) If you already have an idea of how to write the code, then you may use LLM or any other AI tools to **assist** you. Make sure that you **understand** the code and the logic before you use it. AI should help you, not replace your thinking.
2) The basic requirements should be solvable on your own without using AI tools.
3) The extra challenges are open ended and feel free to use any tools or resources. Again, **understand** the code before using it.
4) Discuss with your teammates to gather more ideas to solve the questions. No one should be a free rider in the group.
5) If you encounter any bugs or unclear instructions in the materials, please inform me.
6) Use **Git** to organise your code and collaborate with your teammates.
7) Write clean, well structured, and readable code with proper comments so that others can easily understand your work.
8) Any updates or corrections to the assignment materials will be announced on this GitHub link.

For any queries, you can contact me (NG ERN YI) through:

1) WhatsApp/ Telegram (017-4213195)
2) Email (22004779@siswa.um.edu.my)