CMSC 122 Project Report
Algorithm Comparison

Charles Khervie Realino

December 5, 2025

I. Move-Checking Algorithm Comparison

a. Original Move-Checking Code

```
var move = game.move({
    from: source,
    to: target,
    promotion: 'q'
});

if (move === null) return 'snapback';

var moveAN = move.from + '-' + move.to;
var solutionMoves = solution.text().split(' ');

if (!solutionMoves.includes(moveAN)) {
    incorrect++;
    strikes++;
    strikText.text(`Strikes: ${strikes}`);
    newPuzzle();
} else {
    correct++;
    correctText.text(`Correct: ${correct}`);
    newPuzzle();
}
```

b. My Modified Move-Checking Code

```
var move = game.move({
    from: source,
    to: target,
    promotion: 'q'
});
```

```
if (move === null) return 'snapback';

var moveAN = move.from + '-' + move.to;
var solutionMoves = solution.text().split(' ');

if (!solutionMoves.includes(moveAN)) {
    showFeedback('x Wrong!', 'text-red-500');
    strikes++;
    strikText.text(`Strikes: ${strikes}`);

    if (strikes >= maxStrikes) {
        setTimeout(() => endGame('Maximum strikes reached'), 800);
    } else {
        setTimeout(() => newPuzzle(), 800);
    }
} else {
    showFeedback('/ Correct!', 'text-green-500');
    correct++;
    correctText.text(`Correct: ${correct}`);
    setTimeout(() => newPuzzle(), 800);
}
```

c. Explanation

Both implementations use the same fundamental method for verifying a user's move: the move is converted into `from-to` notation and compared against a list of valid solution moves.

However, the surrounding logic differs significantly:

- The original code performs an immediate puzzle reset with no user feedback.
- My version adds visual feedback through animations, clearer user communication, and delayed transitions to make outcomes understandable.
- A structured game-over state is included in my implementation, replacing the original's simple board disable behavior.
- Additional state management (timers, strike limits, UI transitions) is integrated, making the system behave like a more robust state machine.

The core algorithm remains the same, but the execution flow and user interaction algorithms were extended substantially.

II. Hint System Implementation

a. Hint System Code (No Original Version Exists)

```
function showHint() {
    var movesText = solution.text();
    if (!movesText) return;

    hintClicks++;
    var solutionMoves = movesText.split(' ');
    var firstMove = solutionMoves[0] || '';
    var parts = firstMove.split('-');
    var from = parts[0] || '';
    var to = parts[1] || '';

    if (hintClicks === 1) {
        // Step 1: show starting square
        $('#hintText').text('Hint: move the piece at ' + from);
        clearHighlights();
        highlightSquare(from, 'rgba(250, 204, 21, 0.7)');
        $('#hintButton').text('Show Solution');

    } else {
        // Step 2: reveal full solution
        solution.show();
        $('#hintText').text('');
        clearHighlights();

        if (from) highlightSquare(from, 'rgba(16, 185, 129, 0.6)');
        if (to) highlightSquare(to, 'rgba(248, 113, 113, 0.6)');

        $('#hintButton').hide();
    }
}
```

b. Explanation

The reference implementation contains no hint system or solution-request functionality. My version introduces a fully new algorithm that provides incremental guidance.

The hint algorithm operates in two stages:

- **First hint (partial assistance):** The algorithm identifies the correct piece to move by extracting the starting square of the solution. This square is highlighted on the board, and a text hint is displayed. This gives directional help while maintaining puzzle difficulty.
- **Second hint (full assistance):** On a second request, the algorithm reveals the exact move by highlighting both the origin and destination squares. The full solution text is shown, and the hint button is disabled to prevent repeated input.

Algorithmically, the hint system demonstrates:

- **State Tracking:** The `hintClicks` variable controls the progression between hint stages.
- **Selective DOM Computation:** Square highlighting requires iterating over and modifying specific board elements.
- **Incremental Revelation Strategy:** Information is disclosed gradually rather than all at once.
- **Integration With Puzzle State:** The hint always reflects the current puzzle's correct solution, ensuring accuracy.

Since the original code does not include any hint-related features, this represents a completely new algorithmic component, extending functionality and improving user accessibility while adhering to project requirements.