Module One Lesson Review

Click here for GitHub Repo

Idea one - Polya's problem-solving framework

Main topics: The Polya's problem-solving framework involves a step-by-step approach to finding solutions. Key elements include identifying the unknowns, data, and constraints of the problem and exploring partial solutions or different problem frames when a direct solution is not readily available.

Useful how: This approach is crucial as it encourages iterative problem solving and the celebration of partial solutions. The journey to solving a complex problem like tic-tac-toe begins by understanding the game's rules and characteristics, followed by brainstorming potential problem-solving algorithms.

How will I use it: The framework's value lies in its broad applicability. For instance, a heuristic function that solves tic-tac-toe could also tackle similar problem sets in chess or checkers, albeit on a smaller scale due to their complexity.

Sources/links/reading list: Author (s): Georg Polya Publisher: Princeton University Press Title: How to solve it Year of publication: 2014

Idea two - Characteristics of the tic-tac-toe game

Main topics: Tic-tac-toe is a futile, zero-sum game with perfect information. Its simplicity allows for a finite number of possible games, making it an excellent toy problem for implementing and understanding game-solving algorithms.

Useful how: Knowing that tic-tac-toe is futile (with only one final state for optimal play - a draw) and a zero-sum game can help build algorithms that consider all possible outcomes.

How will I use it: The characteristics of tic-tac-toe can serve as a foundational stepping stone for understanding and solving larger games with perfect information.

Sources/links/reading list: Introduction to AI Techniques Game Search, Minimax, and Alpha Beta Pruning June 8, 2009

Idea three - Heuristic evaluation function

Main topics: A heuristic function serves as an efficient way to decide the optimal move in a game like tic-tac-toe. It uses a game tree to assess possible outcomes and make decisions.

Useful how: Heuristic functions can dramatically reduce computational needs in large-scale problems where classical methods may prove inefficient. They are particularly valuable in AI and machine learning problems that require a vast number of decisions.

How will I use it: The knowledge of heuristic functions will prove useful in tackling and efficiently solving any new problems that come up in this unit and future AI or ML projects.

Sources/links/reading list: Readings on heuristic functions were found in online PDFs and AI/ML related websites.

Idea four - Why the code determines what kind of tests you can run

Main topics: The structure and design of the code determine the ease and effectiveness of testing. If code is not modular, as in our tic-tac-toe example, creating test cases can be challenging.

Useful how: Understanding this concept helps in realizing the limitations of testing within a given codebase and emphasizes the importance of modular code.

How will I use it: This understanding will guide my approach to refactoring and organizing code in future projects to enable more effective testing.

Sources/links/reading list: Readings on testing were found in online PDFs and developer related websites such as StackOverflow.

Reflection

Most important thing I learned?

- The importance of thinking step by step and asking myself questions that can stimulate potential solutions.
- The significance of appreciating partial solutions and building upon them.
- The process of breaking down the problem into manageable parts, specifically:
  - Identifying the unknowns.
  - Recognizing the available data.
  - Understanding the constraints.
- Once a solution is found, it's crucial to evaluate it:
  - Does the solution make sense?
  - Does it effectively solve the problem?
  - Can it be applied to resolve similar problems?

How does this relate to what I already know?

- I have implicitly applied the problem-solving framework in the past without full awareness of its structure. Gaining a thorough understanding of this framework enabled me to introspect my problem-solving process more effectively.

Why was testing troublesome?

- The original code lacked a modular design, consisting instead of a single script. This structure made testing each function individually a necessity rather than a choice.
- The board state was globally defined, and it wasn't passed to functions as an argument, further complicating the testing process.

Code updates/changes?

- I implemented an out-of-bounds check for user input to enhance the game's robustness.

Thoughts on Polya's problem-solving framework?

- I find the framework highly beneficial. While I had previously read parts of Polya's book, this exercise allowed me to apply the principles directly to my problem-solving process.
- Despite the largely math-based examples in the book, I can see their applicability in this unit and broader computational problem-solving.