Sudoku Solver: Final Presentation

Shaikh Muhammad Irtiza, Shahram Khan, Komail Lokhandwala

May 2, 2025

Project Overview

- **Objective**: Implement a terminal-based C++ Sudoku solver.
- **Features**:
  - Input puzzles manually or via files.
  - Solve using backtracking with MRV heuristic.
  - Manage puzzles with file I/O.
  - Performance analysis.
- **Team**: Shaikh Muhammad Irtiza, Shahram Khan, Komail Lokhandwala.

Data Structures

- **Vectors**: `std::vector<std::vector<int>>` for 9x9 grid.
  - $O(1)$ access, dynamic sizing.
- **Unordered Sets/Maps**: Track constraints for rows, columns, boxes.
  - $O(1)$ average-case lookup.
- **List**: For MRV heuristic to sort empty cells.

Algorithms

- **Backtracking**: Recursively fills cells, backtracks on failure.
- **MRV Heuristic**: Prioritizes cells with fewest valid numbers.
- **Constraint Propagation**: Uses sets to enforce Sudoku rules.

Performance Analysis

Difficulty	Avg. Time (ms)	Backtracks
Easy	10	50
Medium	50	500
Hard	150	1500
Expert	300	3000

- Time Complexity: $O(9^n)$, n = empty cells.
- Space Complexity: $O(n)$ for recursion stack.

Challenges & Solutions

- **Challenge**: Slow solving for hard puzzles.
  - **Solution**: Implemented MRV heuristic to reduce backtracking.
- **Challenge**: File I/O errors.
  - **Solution**: Robust error handling with try-catch.
- **Challenge**: User input validation.
  - **Solution**: Clear input stream and validate ranges.

Future Improvements

- Add a graphical user interface (GUI).
- Implement advanced heuristics (e.g., Naked Singles).
- Support larger grid sizes (e.g., 16x16).
- Enhance database with search by difficulty.

Conclusion

- Successfully implemented a robust Sudoku solver.
- Applied key DSA concepts: vectors, sets, backtracking, file I/O.
- Demonstrated efficiency and user-friendliness.
- Ready for questions!