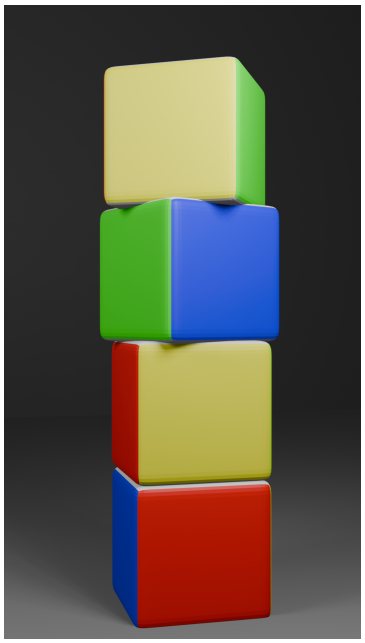
INF-2600 Assignment 1 - Search

Deadline: February 15th 2024

1 Background



In the "Cube Tower" puzzle, you are given a tower consisting of identical cubes. Each cube has the same colors on its vertical sides in a specific order (Red, Blue, Green, Yellow). The goal is to align all cubes so that all colors are the same vertically. There are two kinds of operations allowed:

1. Rotate a cube and all cubes above it 90°.

2. Rotate a cube (c1) and hold another cube (c2) above c1 so that c2 and the cubes above c2 do not rotate, but all cubes starting with c1 up to the cube just below c2 do rotate.

2 Objective

Your task is to develop algorithms to solve the Cube Tower puzzle. Implement and compare different search algorithms to find the minimum number of moves required to organize the tower.

You can use the precode as a starting point. You can also implement it in another language than Python if you prefer.

3 Requirements

3.1 Algorithm Implementation

- Implement the `rotate_cube` method to correctly perform the cube rotations as per the puzzle rules.
- Implement the following search algorithms:
  1. Depth-First Search (DFS)
  2. Breadth-First Search (BFS)
  3. A* Search
  4. Any other advanced search algorithm of your choice (e.g., Iterative Deepening, Greedy Best-First Search, etc.)
- Each algorithm should find a solution.
- Ensure your implementation correctly handles the two types of operations allowed.
- For A* Search (and any other potential informed search algorithm), you will need to decide on a heuristic.

3.2 Problem Instances

- Create at least 5 different instances of the Cube Tower problem with varying levels of complexity.
- Provide a visualization of the tower's initial and final (organized) state for each instance.

3.3 Analysis

- Analyze the performance of each algorithm in terms of time complexity, space complexity, and number of moves required to solve the puzzle.
- Discuss the optimality of the solutions provided by each algorithm.

Specifically, create plots showing the time taken, moves taken and memory used by each algorithm for each problem instance. Discuss the results in the context of the known theoretical complexities.

3.4 Report

- Write a detailed report documenting your algorithms, implementation, problem instances, the heuristic you have chosen, and analysis.
- Include a section reflecting on the challenges faced during the implementation and how you addressed them.

3.5 Bonus Challenge (Optional)

- Extend your algorithms to handle a variant of the puzzle where the cubes may have different orders of colors on their sides.
- Analyze how this variation affects the complexity and performance of your algorithms.

4 Submission

Submit your code, a set of problem instances, visualizations, and the final report (.pdf) in a zip file. Ensure your code is well-commented and follows the coding standards of the language used.