Assignment 1 - COSC 3P71
Carson Cormier
7843469
ww23iu@brocku.ca

1. Introduction:

The objective of this project was to create a program that solves ARC-AGI puzzles using the Breadth-First Search, Greedy Best-First Search, and A* Search algorithms. Each 'puzzle' gives input and output grids, and the task is to find a simple program that transforms the input into the correct output. For example, given the input of [0,1] and the desired output of [1,0], the correct program might be Mirror(vertical) or Rotate(180) (depending on the context). The goal is for the algorithm to figure out which transformation produces the correct output by using a trial-and-error technique as it explores possible operations and tests them. This assignment compares how well each algorithm performs and how custom heuristics can improve those results.

2. Implementation Details:

Each puzzle solution is represented as a program tree with operations like ColorChange, Rotate, or Mirror. The combined programs in my code use Sequence(left, right) to apply one operation after another, while program complexity counts the number of operations used. I wrote the apply_program() function to run these transformations on a grid recursively. The missing functions (such as Rotate(180), Scale3x3, and many more) were completed according to the grammar. If a program uses Sequence, the left program executes first, followed by the right. I chose this method because it is what I am most familiar with and makes the most sense in this given situation (in my opinion). The search algorithms I used included Breadth-First Search, Greedy Best-First Search, and A* Search. To begin, the Breadth-First Search explores programs in order of complexity and finds the simplest one. Greedy Best-First Search uses only the heuristic (which is an estimation of how close we are to the solution) to guide the search. Finally, A*, which combines both the heuristic and program complexity to quickly find good results. All three of these search algorithms were tested on 28 ARC-AGI tasks.

3. Heuristic Design:

This project uses three heuristics to guide the informed search algorithms: Basic Heuristic (cell mismatch), as well as custom heuristics 1 and 2. The 'Basic Heuristic' is used for both GBFS and A* search; it counts the number of cell mismatches between the programs' predicted output and the target output. It provides a baseline measure of how close a program is to producing the correct input. Next, the Custom Heuristic 1 adds onto the previous mismatching method by also evaluating how many cells of each color differ from the target output. This helps the search to prefer programs that match both the individual cells and the overall color pattern. Finally, the Custom Heuristic 2 emphasizes the overall structure of the grid by looking at the border colors to find symmetry and patterns (as well as checking cell mismatches and size differences). This helps the search prefer programs that keep the general shape and edges of the target input, rather than individual cells.

4. Experimental Results:

Project outline results:

==========
ACCURACY SUMMARY
==========
Total Tasks: 28
BFS Accuracy: 12/28 (42.9%)
GBFS (Cell Mismatch) Accuracy: 15/28 (53.6%)
A* (Cell Mismatch) Accuracy: 18/28 (64.3%)
GBFS (Custom Heuristic 1) Accuracy: 16/28 (57.1%)
A* (Custom Heuristic 1) Accuracy: 20/28 (71.4%)
==========

My Results:

==========
ACCURACY SUMMARY
==========
Total Tasks: 28
BFS Accuracy: 15/28 (53.6%)
GBFS Accuracy: 15/28 (53.6%)
ASTAR Accuracy: 15/28 (53.6%)
GBFS_CUSTOM1 (Custom Heuristic 1) Accuracy: 18/28 (64.3%)
ASTAR_CUSTOM1 (Custom Heuristic 1) Accuracy: 18/28 (64.3%)
==========

As seen from these results, my BFS worked better than expected (53.6% vs. 42.9%), indicating my search implementation correctly identifies optimal programs for many basic tasks. Next, my GBFS matched the expected baseline (53.6% accuracy), which shows a consistent heuristic performance. Moreover, the ASTAR underperformed compared to the outline (53.6% vs. 64.3%), which is understandable as that was the function I struggled most with, and shows that the way I combined path costs and heuristic may not have been fully optimal. However, using Custom Heuristic 1 significantly improved performance for both GBFS and A*, proving that the custom heuristic helps improve search efficiency and accuracy using color and structure targeting methods.

5. Algorithm Analysis:

In general, informed search algorithms like GBFS and A* outperform BFS because they use a heuristic to guide the search. BFS checks all programs in order of complexity, which finds simple solutions effectively, but can often be slow and inefficient for large search spaces. On the other hand, GBFS uses its heuristic to prioritize programs that appear closer to the target, which often results in finding solutions faster with fewer program evaluations. A* improves on GBFS by combining that heuristic with program complexity, balancing speed and solution quality. The trade-off is that GBFS sometimes follows the wrong hints and misses simpler solutions, while A* is more reliable but may explore more paths. In this project, Custom Heuristic 1 improved both GBFS and A* by considering their color and structure, showing just how a well-designed heuristic can make such a huge impact on an algorithm.

6. Conclusion:

In conclusion, this project successfully implemented and evaluated three search algorithms (these being: Breadth-First Search (BFS), Greedy Best-First Search(GBFS), and A* Search (A*)) on 28 ARC-AGI puzzles to identify transformations that occurred given an input and an output. The results showed that BFS effectively solved more than half the puzzles, outperforming its expected solution rate. GBFS and A* perform even better when using Custom Heuristic 1 because it considers the color and structure in the grids. Overall, this project has shown the importance of a well-designed heuristic as it improves an algorithm's efficiency, making it faster and more accurate.

7. References:

- GeeksforGeeks. (2025). *Heuristic Search Techniques in AI*. https://www.geeksforgeeks.org/artificial-intelligence/heuristic-search-techniques-in-ai/
- Codecademy. (2023). *Greedy Best-First Search*. https://www.codecademy.com/resources/docs/ai/search-algorithms/greedy-best-first-search
- Simplilearn. (2025). *A* Search Algorithm Explained: Applications & Uses*. https://www.simplilearn.com/tutorials/artificial-intelligence-tutorial/a-star-algorithm
- GeeksforGeeks. (2025). *Greedy Best-First Search in AI*. https://www.geeksforgeeks.org/artificial-intelligence/greedy-best-first-search-in-ai/
- GeeksforGeeks. (2025). *A* Algorithm and Its Heuristic Search Strategy in Artificial Intelligence*. https://www.geeksforgeeks.org/artificial-intelligence/a-algorithm-and-its-heuristic-search-strategy-in-artificial-intelligence/
- Stack Overflow. (2011). *What are some good methods to finding a heuristic for the A* algorithm?* https://stackoverflow.com/questions/5687882/what-are-some-good-methods-to-finding-a-heuristic-for-the-a-algorithm
- Wikipedia. (2025). *A* Search Algorithm. https://en.wikipedia.org/wiki/A%2A_search_algorithm