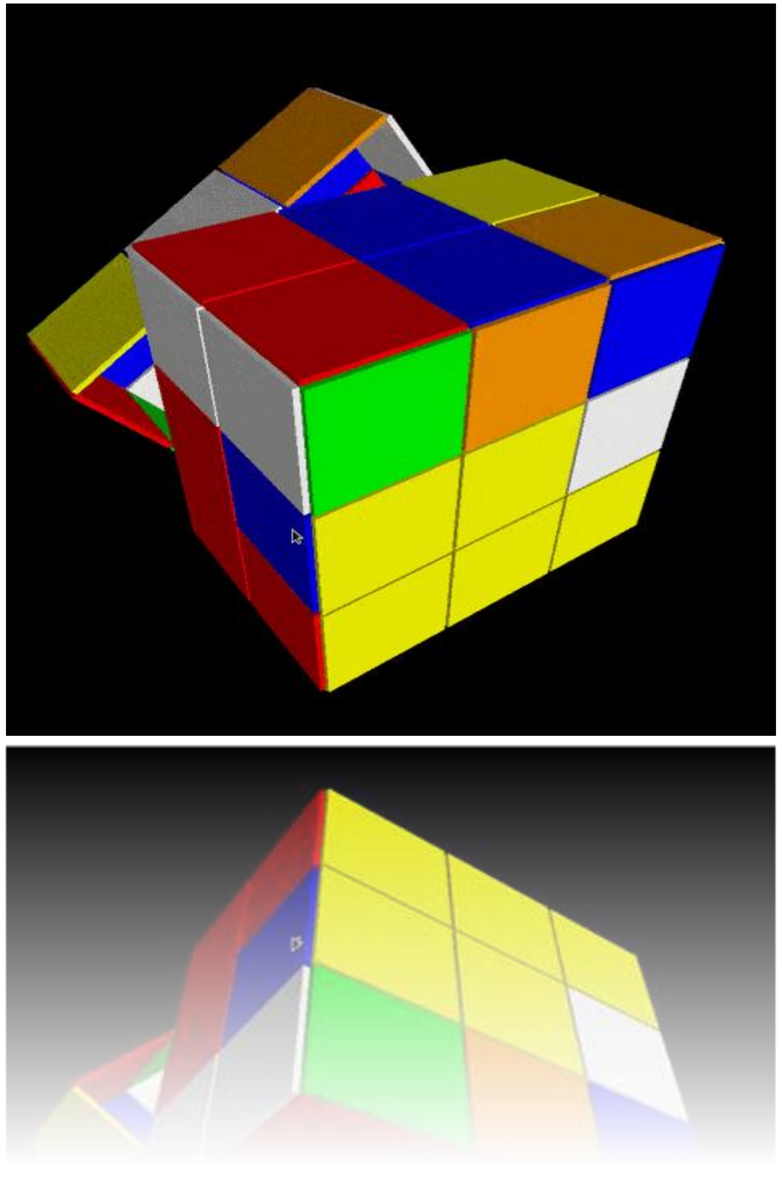
UNIVERSITY OF NEW HAVEN

SOLVING THE RUBIK'S CUBE:

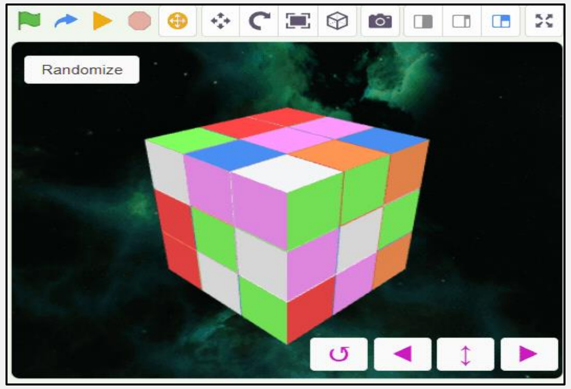
OPTIMIZED PATHFINDING AND SEARCH ALGORITHMS

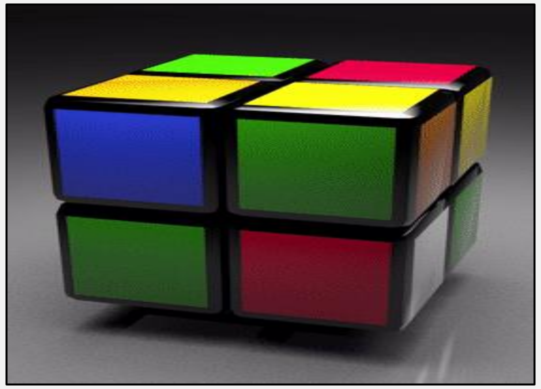
A Team Project By:-

1. **Premsai Chowdary Konanki**
2. **Pavan Sai Chaparala**



PROJECT OBJECTIVE



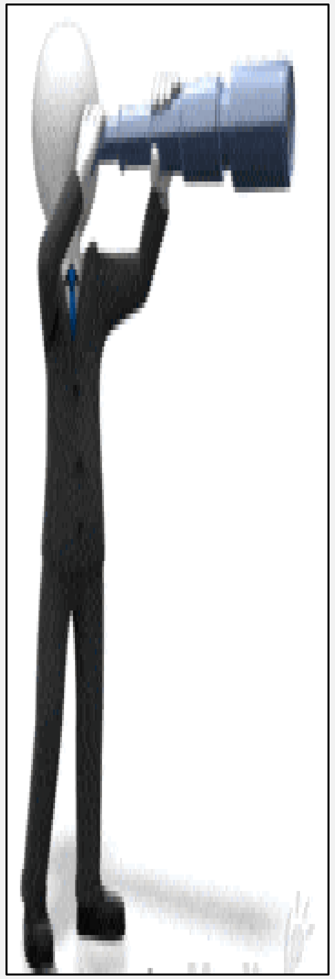


1.The project's main objective is to resolve the Rubik's cube problem. This is achieved by measuring the quality of almost-finished cubes using an efficient reinforcing process and a pattern database.

2.This project also tries to tackle the problem of the cube with n random motions applied to it and boosts its efficiency in solving the supplied Rubik's cube.

3.The 2x2x2 cube is considerably easier to solve than the more common 3x3x3 cube.

4. The project aims to integrate advanced algorithms to minimize the number of moves required to solve the cube, ensuring optimal performance and accuracy.



APPROACH

- **Breadth-First Search (BFS)**

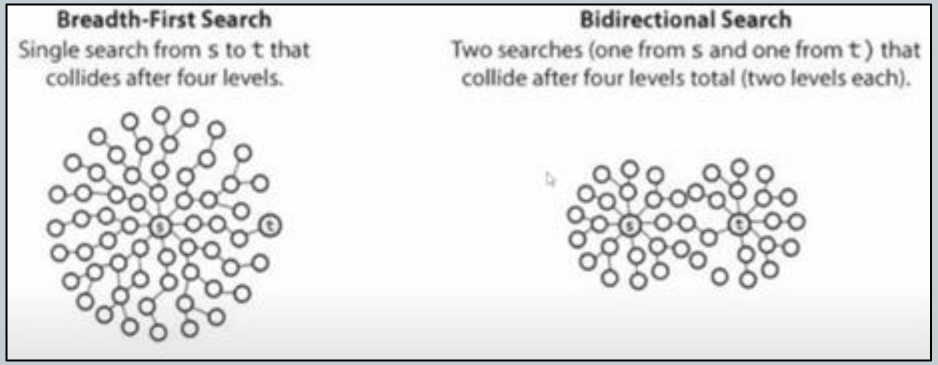
BFS is a tree traversal algorithm that systematically explores all nodes at the current level before moving to the next level. It ensures that the shortest path is found in an unweighted graph or problem space, such as solving a Rubik's Cube. This method uses a queue data structure to keep track of nodes yet to be explored.

- **Bidirectional Breadth-First Search (Bidirectional BFS)**

Bidirectional BFS is an efficient graph search algorithm that starts the search from both the starting and ending points simultaneously. This approach significantly reduces the search space, making it faster to find the shortest path than BFS.

Tools and Technologies:

- **Python** is used to implement the BFS logic.



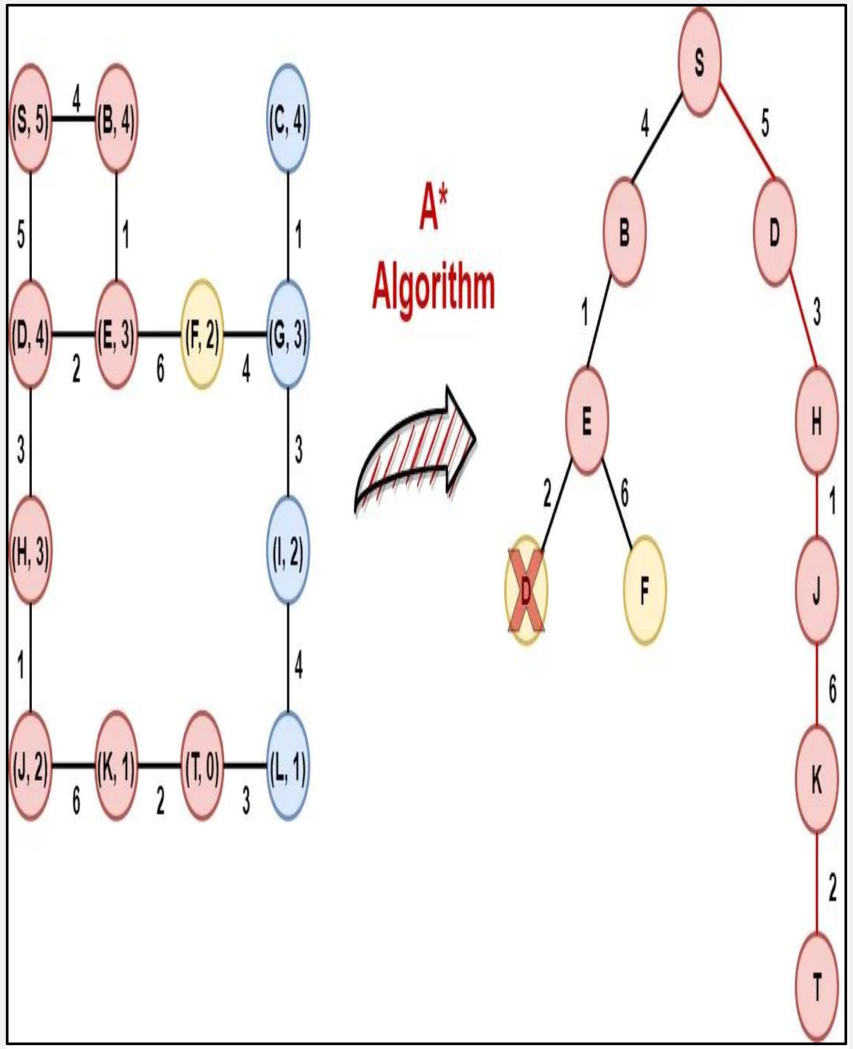
Breadth-First Search
Single search from s to t that collides after four levels.
Bidirectional Search
Two searches (one from s and one from t) that collide after four levels total (two levels each).
s
t
s
t

APPROACH

- ***A**** **Algorithm**

- The 2x2x2 Rubik's Cube can be solved optimally by using the A algorithm*. To rank the exploration of promising routes, it combines a heuristic estimate **(h(n)),** like the Manhattan Distance, with the real cost to reach a state **(g(n)).**

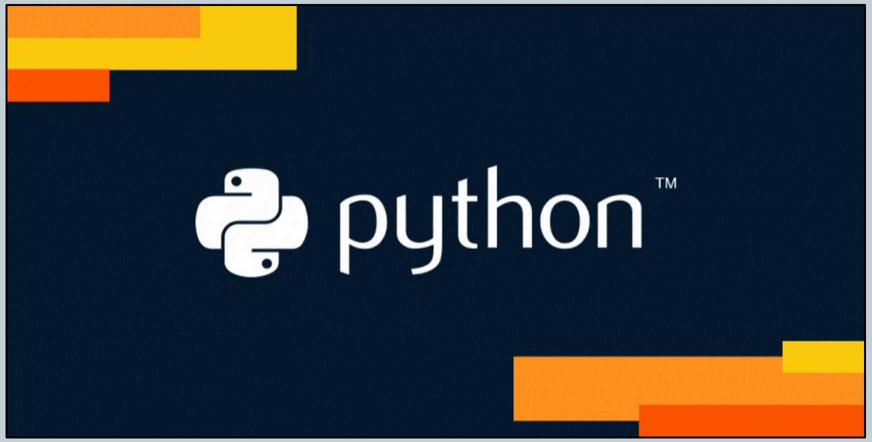
- To determine the shortest solution, the algorithm assesses potential moves and determines the overall cost **(f(n) = g(n) + h(n)).** It uses an acceptable heuristic to guarantee the best possible answer. A* minimizes the number of moves needed to solve the Rubik's Cube efficiently.



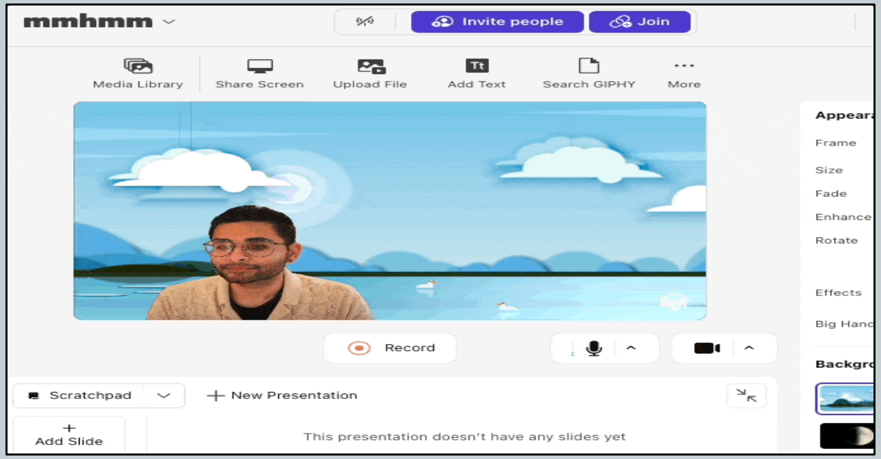
DELIVERABLES

1.A Project Proposal

2. Project Code

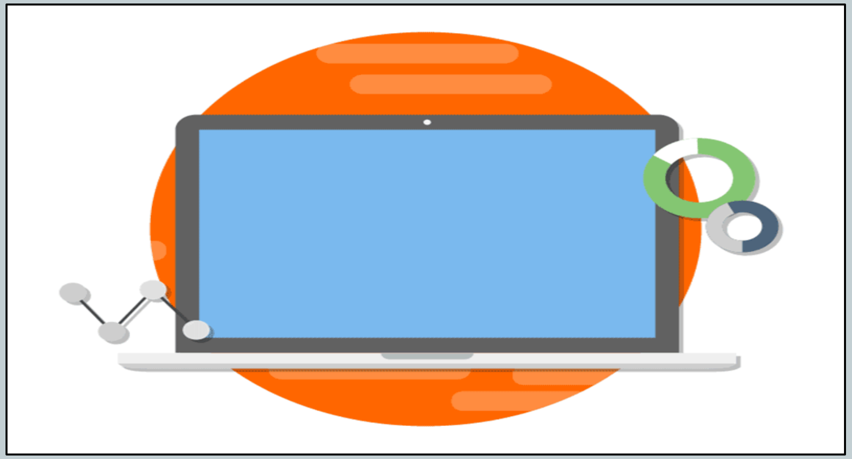


3.A video demonstration



This presentation doesn't have any slides yet

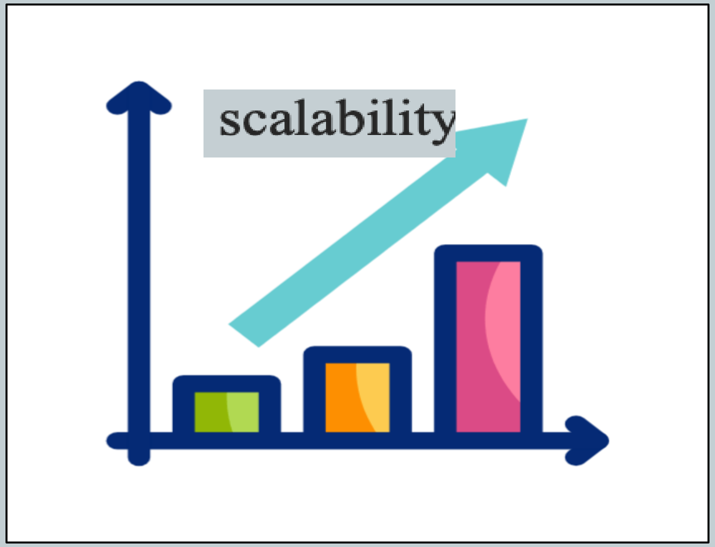
4. Review(in a report)



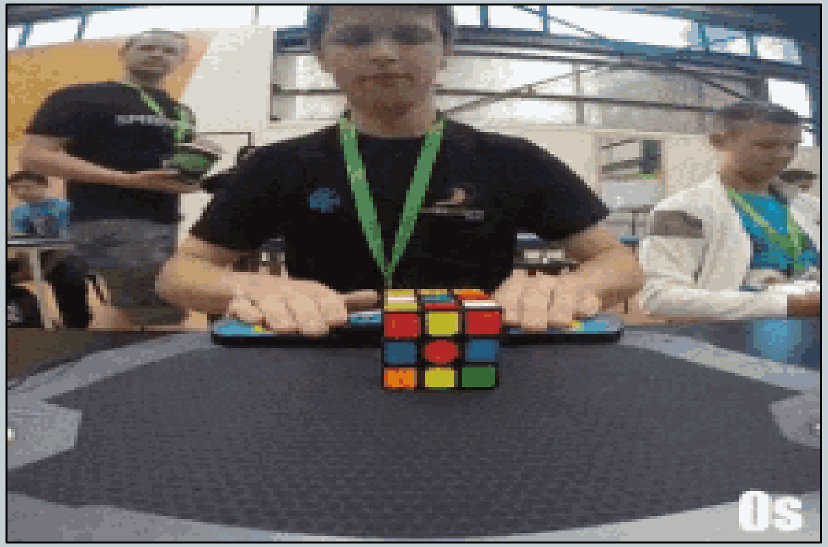
EVALUATION

1. The project outcome should be assessed based on its capability to efficiently solve a simple Rubik's Cube, a 2x2x2 or 3x3x3 configuration.

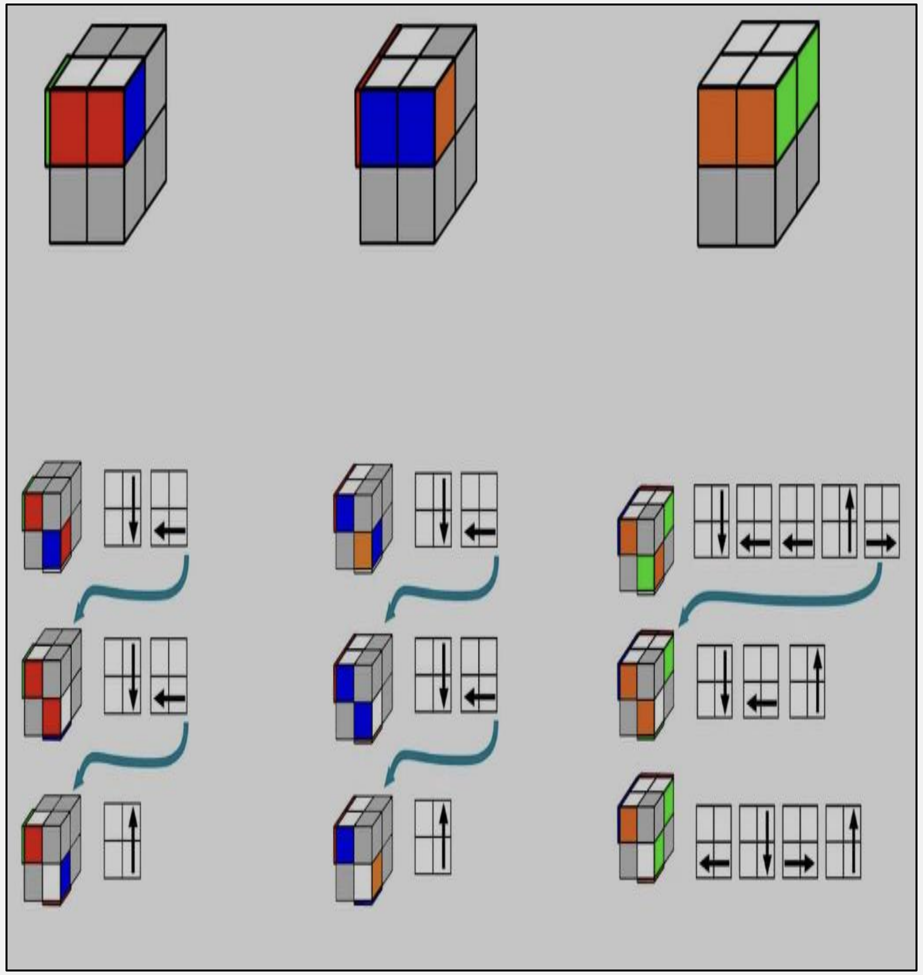
2. The evaluation of the algorithm's performance should take into account its completeness, efficiency, memory utilization, scalability, and any optimizations or heuristics implemented to improve its effectiveness.



1.Algorithm used
2. Optimized path
3. Memory Utilization
4. Performance and Completion



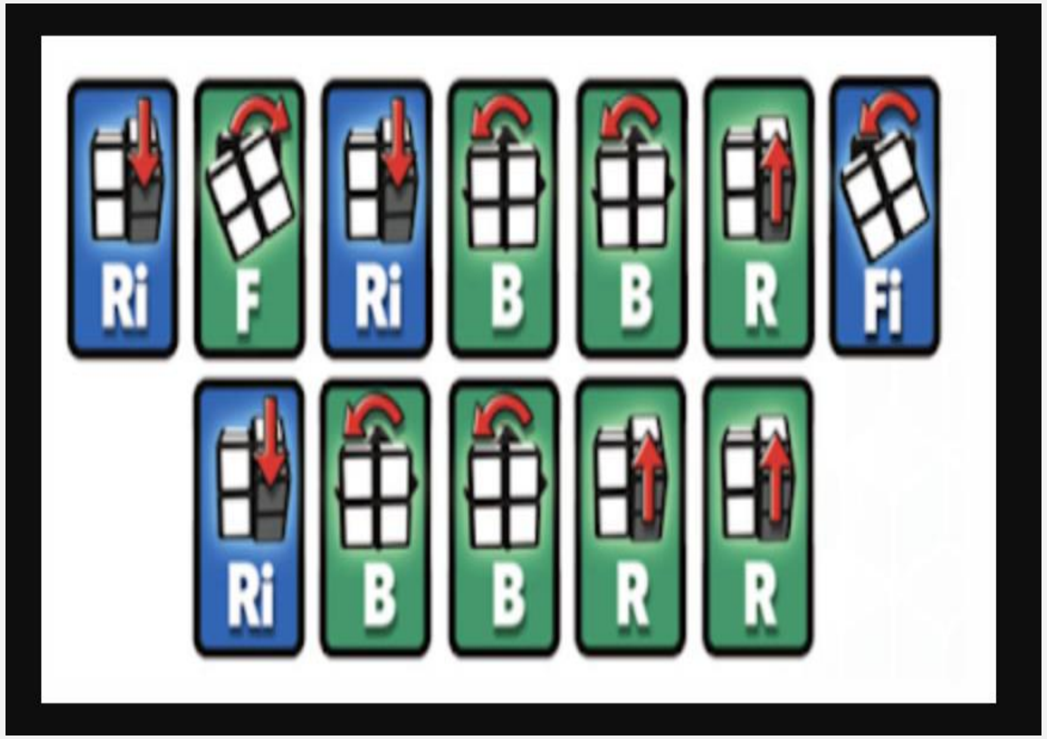
IMPLEMENTATION



- A tuple of 24 integers that each represent the color of a sticker on a different face of the cube is used to represent the Rubik's Cube.
- In its resolved form, the sticker order corresponds to the configuration. To effectively investigate possible solutions, the algorithm uses distinct data structures for both forward and backward search processes, such as parent dictionaries, distance metrics, and queues.
- The quickest path to solving the challenge is found more quickly because to this bidirectional strategy, which also helps to shrink the search space.
- To maximize the solution time, the forward and backward searches also operate in tandem with the goal of arriving at a shared state.

PATH

- Using the start parent and end parent dictionaries, the path is reconstructed by tracing from the end state back to the start state. With the aim of meeting in the midway, this demonstrates the bidirectional exploration of states, in which the start and end points are investigated concurrently.



OUTPUT

- The goal of our project is to find the shortest path while improving search efficiency.
- Although it may take longer, we first use BFS to search the issue space and determine the shortest path.
- After that, A* is used, which speeds up the pathfinding process by optimizing the search by fusing heuristic estimates with actual expenses.
- In conclusion, bidirectional BFS is employed since it usually results in the shortest path with the least time complexity by drastically reducing the search space.
- The most effective strategy, which offers the best outcome in our situation, is bidirectional BFS.

CONCLUSION

1. Bivariate BFS optimizes search processes through a dual-front strategy, significantly reducing the search space while maintaining an efficient balance between time and space complexity.

2. This method is particularly well-suited for shortest-path problems, ensuring both computational efficiency and solution optimality.

3. In the context of solving the Rubik's Cube, Bivariate BFS represents a groundbreaking advancement, showcasing its capability to address complex, real-world challenges.

4. Its potential extends beyond this specific application, offering impactful contributions to reinforcement learning and the broader domain of advanced algorithmic problem-solving.