Artificial Intelligence Project 2

Shelby Stocker

To represent the cube, I made an array for each face with 9 positions. I filled in this array in the following order for each face: upper left tile, upper middle tile, upper right tile, middle left tile, middle tile, middle right tile, bottom left tile, bottom middle tile, bottom right tile. This is based off the orientation of the unfolded cube which is printed at the beginning of my program.

My program is a python script so it can be run from the terminal by the command “python 463p1-stocker.py” or in an IDE such as PyCharm.

The program begins by printing out the unfolded cube with the faces in their original positions. It then lists all the possible ways you could turn the cube along with that turns id number. It then asks the user for input as to how many turns they want to use to randomize the cube. It does not check to make sure that the user entered a number and not some other character. Once the user inputs a number, the program generates a random number from 1-12. Each of these numbers are assigned to a possible way to turn the cube. The program then makes whichever turn was generated and prints out the turn option as well as the unfolded cube following the turn. This continues until the number of turns the user wanted is reached.

Here are images of sample output:

A close up of text on a black background

Description automatically generated A close up of text on a black background

Description automatically generated

A close up of text on a black background

Description automatically generated A close up of text on a black background

Description automatically generated

HEURISTIC:

f = the face on cube N with the most duplicated tiles

X(f) = the number of rows in which all 3 tiles in the row are a tile from a pair of duplicates

Y(f) = the number of columns in which all 3 tiles in the column are a tile from a pair of duplicates

Z(f) = the number of remaining duplicate pairs

My heuristic function: H(n) = X(f)+Y(f)+Z(f)

Each “pair” of numbers is counted as 1 not 2. Also, when saying pair I am referring to a group of at least two.

Example 1: The pictures below show the cube in its initial positions, after a bottom turn and then a right turn inverted. The front face has the largest number of duplicated tiles which is 7. The front has pairs of 1, 5, and 6.

For the front face:

X(front) = 2 because the top and bottom row are both filled with numbers that are duplicated (circled in red)

Y(front) = 0 because there are no columns filled with duplicated numbers

Z(front) = 0 because all duplicated numbers were accounted for previously

H(n) = 2 + 0 + 0 = 2

The heuristic gave the correct number of moves needed to solve the cube.

A close up of a logo

Description automatically generated A close up of text on a black background

Description automatically generated A close up of text on a black background

Description automatically generated



Example 2:

The images below depict the initial starting position, a left turn, another left turn, right turn inverted, front turn, and another front turn.

A close up of text on a black background

Description automatically generatedA close up of a keyboard

Description automatically generated A close up of a keyboard

Description automatically generated

A close up of a logo

Description automatically generated A close up of text on a black background

Description automatically generated



The front face has the most duplicated numbers which is 8.

X(front) = 2

Y(front) = 2

Z(front) = 0

H(n) = 2 + 2 + 0 = 4

Example 3:

The following example depicts the cube after 8 rotations: right turn inverted, front turn inverted, bottom turn, left turn, front turn, top turn, bottom turn, top turn inverted.

A close up of text on a black background

Description automatically generated The top face has the most duplicated tiles which is 7.



X(top) = 1

Y(top) = 2

Z(top) = 0

H(n) = 1 + 2 + 0 = 3

In this case, the heuristic predicted a value less than the actual number of moves.

The maximum number of duplicate pairs that could exist on a face would be if all of the same number ended up on one face or if there was a group of say 3 pairs of 2 and a group of 3. In cases like these all 3 rows would be counted as well as all 3 columns. This would result in a heuristic value of 6. To solve this type of situation would take at least 6 moves, likely more. Therefore, this heuristic is admissible. The result produced will never be more than actual number needed to solve the cube.

My code starts by asking the user how many turns to randomize the cube, computing these random turns, and then running the a star algorithm on this randomized cube.

I implemented the a\* algorithm following code I found online. It is documented in my source code. It will not correctly run without an error. The time complexity of the a\* algorithm relies on how good the heuristic is. The number of nodes expanded grows at an exponential rate. Had my code run correctly, I would assume the graph would look like the one I created below.

![A screenshot of a cell phone

Description automatically generated]()

Before this assignment I honestly had not messed with a Rubik’s cube much so I learned about all the possible moves you can make. I assumed you could turn the middle row, but I realized that would be the same as turning the top and then the bottom. I also learned what a heuristic is. I also learned what the a\* algorithm is. I had never heard of it before this assignment. It seems to be much better than other algorithms.