```
==============================================================
============================ ___ =============================
=========================== /   \ ============================
========================== /     \ ===========================
========================= /  THE  \ ==========================
======================== /         \ =========================
======================= /   MASTER  \ ========================
====================== /             \ =======================
===================== /    PYRAMINX   \ ======================
==================== /_________________\ =====================
==============================================================


_____________________ | Requirements | _______________________

<> A computer running Linux or macOS
<> C++11 or later with its standard libraries
<> G++ or Clang++ compiler
<> A terminal to run the program and read program output from
______________________________________________________________
v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^



_____________________ | Instructions | _______________________

[1] In your terminal, change your current directory to the
    unzipped folder containing both header files and the
    "ProjStar.cpp" file.
==============================================================
[2] Run the below command to compile the program:

    (with g++ compiler)
    g++ -std=c++11 ProjStar.cpp -o ProjectStar

    (with clang++ compiler)
    clang++ -std=c++11 ProjStar.cpp -o ProjectStar
==============================================================
[3] Now, run the following command to run the program:

    ./ProjectStar
==============================================================
[4] Provide an integer value as the number of moves to
    scramble the master pyraminx puzzle in, and watch it work!
______________________________________________________________
v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^
```

## ___________ | The Expected Output; What It Means | ___________

Upon first running the program, you'll see this output:

```
==============================================================
~~~~~~~~~~~~~~~~~~~~~~~
  Before the Scramble:
~~~~~~~~~~~~~~~~~~~~~~~

               /^\                               /^\                               /^\                               /^\
              /   \                             /   \                             /   \                             /   \
             /  R  \                           /  B  \                           /  G  \                           /  Y  \
            /_______\                         /_______\                         /_______\                         /_______\
           /^\     /^\                       /^\     /^\                       /^\     /^\                       /^\     /^\
          /   \ R /   \                     /   \ B /   \                     /   \ G /   \                     /   \ Y /   \
         /  R  \ /  R  \                   /  B  \ /  B  \                   /  G  \ /  G  \                   /  Y  \ /  Y  \
        /_______V_______\                 /_______V_______\                 /_______V_______\                 /_______V_______\
       /^\     /^\     /^\               /^\     /^\     /^\               /^\     /^\     /^\               /^\     /^\     /^\
      /   \ R /   \ R /   \             /   \ B /   \ B /   \             /   \ G /   \ G /   \             /   \ Y /   \ Y /   \
     /  R  \ /  R  \ /  R  \           /  B  \ /  B  \ /  B  \           /  G  \ /  G  \ /  G  \           /  Y  \ /  Y  \ /  Y  \
    /_______V_______V_______\         /_______V_______V_______\         /_______V_______V_______\         /_______V_______V_______\
   /^\     /^\     /^\     /^\       /^\     /^\     /^\     /^\       /^\     /^\     /^\     /^\       /^\     /^\     /^\     /^\
  /   \ R /   \ R /   \ R /   \     /   \ B /   \ B /   \ B /   \     /   \ G /   \ G /   \ G /   \     /   \ Y /   \ Y /   \ Y /   \
 /  R  \ /  R  \ /  R  \ /  R  \   /  B  \ /  B  \ /  B  \ /  B  \   /  G  \ /  G  \ /  G  \ /  G  \   /  Y  \ /  Y  \ /  Y  \ /  Y  \
/_______V_______V_______V_______\ /_______V_______V_______V_______\ /_______V_______V_______V_______\ /_______V_______V_______V_______\

How many moves would you like to scramble (integer only)?
==============================================================
```

As seen above, the program presents the initial state of the master pyraminx puzzle before any scramble has been made. The faces of the pyraminx should be considered in this order in terms of the puzzle when unfolded:

```
     /^\     /^\     /^\
    / G \   / R \   / B \
   /_____\ /_____\ /_____\
           \     /
            \ Y /
             \_/
```

In this layout, consider G=GREEN, R=RED, B=BLUE, AND Y=YELLOW. Additionally, consider all of of the faces adjacent to face R to fold inward toward side R to form the puzzle's 3D shape.

In the program itself, these faces are stored in a map, where:
<> map[1] contains the stickers on the side labeled R
<> map[2] contains the stickers on the side labeled B
<> map[3] contains the stickers on the side labeled G
<> map[4] contains the stickers on the side labeled Y.

Now that we have a basic idea of what the output means and looks like, let's overview the prompt asking for the number of moves to scramble the puzzle in. The best way to get an idea of both how a rotation is visualized in the terminal and how the rotation works itself is to enter "1" as the number of moves to scramble the puzzle in. Here's some example output:

```
==============================================================
How many moves would you like to scramble (integer only)? 1

Randomly rotated level 1 on axis 4 in the clockwise direction

               /^\                               /^\                               /^\                               /^\
              /   \                             /   \                             /   \                             /   \
             /  B  \                           /  G  \                           /  R  \                           /  Y  \
            /_______\                         /_______\                         /_______\                         /_______\
           /^\     /^\                       /^\     /^\                       /^\     /^\                       /^\     /^\
          /   \ R /   \                     /   \ B /   \                     /   \ G /   \                     /   \ Y /   \
         /  R  \ /  R  \                   /  B  \ /  B  \                   /  G  \ /  G  \                   /  Y  \ /  Y  \
        /_______V_______\                 /_______V_______\                 /_______V_______\                 /_______V_______\
       /^\     /^\     /^\               /^\     /^\     /^\               /^\     /^\     /^\               /^\     /^\     /^\
      /   \ R /   \ R /   \             /   \ B /   \ B /   \             /   \ G /   \ G /   \             /   \ Y /   \ Y /   \
     /  R  \ /  R  \ /  R  \           /  B  \ /  B  \ /  B  \           /  G  \ /  G  \ /  G  \           /  Y  \ /  Y  \ /  Y  \
    /_______V_______V_______\         /_______V_______V_______\         /_______V_______V_______\         /_______V_______V_______\
   /^\     /^\     /^\     /^\       /^\     /^\     /^\     /^\       /^\     /^\     /^\     /^\       /^\     /^\     /^\     /^\
  /   \ R /   \ R /   \ R /   \     /   \ B /   \ B /   \ B /   \     /   \ G /   \ G /   \ G /   \     /   \ Y /   \ Y /   \ Y /   \
 /  R  \ /  R  \ /  R  \ /  R  \   /  B  \ /  B  \ /  B  \ /  B  \   /  G  \ /  G  \ /  G  \ /  G  \   /  Y  \ /  Y  \ /  Y  \ /  Y  \
/_______V_______V_______V_______\ /_______V_______V_______V_______\ /_______V_______V_______V_______\ /_______V_______V_______V_______\

~~~~~~~~~~~~~~~~~~~~~~~
  Solving the Puzzle
~~~~~~~~~~~~~~~~~~~~~~~

Please wait as the algorithm finds a solution.....
==============================================================
```

As you can see by reading the stickers on the faces, this move rotated the corner piece containing the red, blue, and green stickers/colors on it. This piece is stated to exist on level 1 of axis 4. As you continue to experiment with puzzle scrambles, you'll begin to see which levels on which axes move which pieces. All of these rotations can be fully replicated on an actual pyraminx puzzle, given that the pyraminx has the same order of adjacent face colors in its solved state.

This puzzle-scrambling function works by using a time-based randomizer to provide random numbers ranging from 1-4 and 0-1. For each move out of “k” randomizing moves, the puzzle will be rotated about a random level from 1-4 of a random Corner of 1-4 in a random direction where 0==clockwise and 1==counter-clockwise (For the purposes of program speed/efficiency, the direction randomness was eliminated. Scrambling rotations are made clockwise and solving rotations are counterclockwise). To learn more about this “Corner” structure, read “The Data Structures Involved” after this section.

After the printout of the scrambled pyraminx, you’ll notice that the program is now finding a solution. Depending on the number of moves you provided, the program either immediately found a solution and generated more output or will be stuck on this message for quite while until it either completes, gets killed, or kills your machine by draining its resources. Once the program finds a solution, you should see similar output:

```
==============================================================
It takes 1 moves to solve the puzzle!

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
  Step 1: Rotate L4 of Face 2 in the counterclockwise direction.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

               /^\                               /^\                               /^\                               /^\
              /   \                             /   \                             /   \                             /   \
             /  R  \                           /  B  \                           /  G  \                           /  Y  \
            /_______\                         /_______\                         /_______\                         /_______\
           /^\     /^\                       /^\     /^\                       /^\     /^\                       /^\     /^\
          /   \ R /   \                     /   \ B /   \                     /   \ G /   \                     /   \ Y /   \
         /  R  \ /  R  \                   /  B  \ /  B  \                   /  G  \ /  G  \                   /  Y  \ /  Y  \
        /_______V_______\                 /_______V_______\                 /_______V_______\                 /_______V_______\
       /^\     /^\     /^\               /^\     /^\     /^\               /^\     /^\     /^\               /^\     /^\     /^\
      /   \ R /   \ R /   \             /   \ B /   \ B /   \             /   \ G /   \ G /   \             /   \ Y /   \ Y /   \
     /  R  \ /  R  \ /  R  \           /  B  \ /  B  \ /  B  \           /  G  \ /  G  \ /  G  \           /  Y  \ /  Y  \ /  Y  \
    /_______V_______V_______\         /_______V_______V_______\         /_______V_______V_______\         /_______V_______V_______\
   /^\     /^\     /^\     /^\       /^\     /^\     /^\     /^\       /^\     /^\     /^\     /^\       /^\     /^\     /^\     /^\
  /   \ R /   \ R /   \ R /   \     /   \ B /   \ B /   \ B /   \     /   \ G /   \ G /   \ G /   \     /   \ Y /   \ Y /   \ Y /   \
 /  R  \ /  R  \ /  R  \ /  R  \   /  B  \ /  B  \ /  B  \ /  B  \   /  G  \ /  G  \ /  G  \ /  G  \   /  Y  \ /  Y  \ /  Y  \ /  Y  \
/_______V_______V_______V_______\ /_______V_______V_______V_______\ /_______V_______V_______V_______\ /_______V_______V_______V_______\

~~~~~~~~~~~~~~~~~~~~~~~
  After the Solve:
~~~~~~~~~~~~~~~~~~~~~~~

               /^\                               /^\                               /^\                               /^\
              /   \                             /   \                             /   \                             /   \
             /  R  \                           /  B  \                           /  G  \                           /  Y  \
            /_______\                         /_______\                         /_______\                         /_______\
           /^\     /^\                       /^\     /^\                       /^\     /^\                       /^\     /^\
          /   \ R /   \                     /   \ B /   \                     /   \ G /   \                     /   \ Y /   \
         /  R  \ /  R  \                   /  B  \ /  B  \                   /  G  \ /  G  \                   /  Y  \ /  Y  \
        /_______V_______\                 /_______V_______\                 /_______V_______\                 /_______V_______\
       /^\     /^\     /^\               /^\     /^\     /^\               /^\     /^\     /^\               /^\     /^\     /^\
      /   \ R /   \ R /   \             /   \ B /   \ B /   \             /   \ G /   \ G /   \             /   \ Y /   \ Y /   \
     /  R  \ /  R  \ /  R  \           /  B  \ /  B  \ /  B  \           /  G  \ /  G  \ /  G  \           /  Y  \ /  Y  \ /  Y  \
    /_______V_______V_______\         /_______V_______V_______\         /_______V_______V_______\         /_______V_______V_______\
   /^\     /^\     /^\     /^\       /^\     /^\     /^\     /^\       /^\     /^\     /^\     /^\       /^\     /^\     /^\     /^\
  /   \ R /   \ R /   \ R /   \     /   \ B /   \ B /   \ B /   \     /   \ G /   \ G /   \ G /   \     /   \ Y /   \ Y /   \ Y /   \
 /  R  \ /  R  \ /  R  \ /  R  \   /  B  \ /  B  \ /  B  \ /  B  \   /  G  \ /  G  \ /  G  \ /  G  \   /  Y  \ /  Y  \ /  Y  \ /  Y  \
/_______V_______V_______V_______\ /_______V_______V_______V_______\ /_______V_______V_______V_______\ /_______V_______V_______V_______\

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
 Nodes expanded in last iteration: 1
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
==============================================================
```

As you can see above, the program will tell you how many moves are necessary in order to solve it, along with a step-by-step guide on how it rotate it from scrambled-state to solved-state. After it shows you the solved-state puzzle, it will inform you of the nodes that were expanded in the last iteration of the IDA* algorithm before it found the goal node.

*** In the case that you noticed the solution does not match with the example output of the puzzle’s scrambled state, please note that the solution output is not correlated with the scrambling example output provided earlier. These outputs were taken from different runs of the program.

v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^

## ______________ | The Data Structures Involved | ______________

There are 2 primary data structures involved in masterminx.h:

<> A private map of the master pyraminx's four faces, each represented by an array of 16 chars, labeled “PYRAMINX”

<> A custom "Corner" struct that contains 27 pointers to specific positions on the master pyraminx.

While the map of four arrays is fairly simple, the Corner struct is a bit more complicated. when making this program, I wanted to find a way to minimize the number of rotation functions necessary to move every part of the puzzle. In order to accomplish this, I created this struct and made four instances of it in the program. This struct allows me to view the pyraminx from the perspective of four separate axes, each of which include both stickers/pieces unique to each axis and stickers/pieces shared among multiple axes. Each struct represents a top-down view of three horizontal layers/slices of the master pyraminx, with the top level, level 1, being a corner piece itself. Given that these layers can only turn clockwise or counter-clockwise, one can say that they are rotating about an imaginary fixed axis that pierces through a corner piece and the center of the corner piece's opposite-side face (if I explained that poorly, imagine just sitting the puzzle on a desk and then poking a needle straight through the top and center of the it).

This structure includes a single rotation function that handles which level to rotate on the axis and which direction to rotate it, clockwise or counter-clockwise.
==============================================================

There are 2 primary data structures involved in starpower.h:

<> A custom “Pnode” struct that contains essential information That allows it to be usable in a tree and by IDA*

<> A custom “PQ” struct that represents a priority queue that uses a min-heap that sorts Pnodes by order of its f-values.

Both of these structures are essential to the program for many reasons. Let’s start by breaking down the Pnode struct (in its full name, “Pyraminx Node"). This structure contains many different variables:

<> A configuration of the puzzle
<> The last rotation that created its configuration

<> A pointer to the Pnode’s parent
<> An array of pointers to the Pnode’s children
<> The h-value, supplemented by a built-in heuristic function
<> The g-value, which is the Pnode’s distance from the root
<> The f-value, which is the sum of the h-value and g-value.

As can be seen above, a Pnode is equipped for the sole purpose of being placed in a tree and used by the A* algorithm. Now, let’s break down the PQ struct’s contents:

<> A std::vector that contains Pnode children pointers (HEAP)
<> An integer storing the size of the heap (HEAPSIZE).

For the sake of priority queues and min-heaps, these variables are fairly straightforward in what their purposes are. The additional functions that are included in this struct are used to build the heap/queue into a min-heap, ordering the Pnodes by their corresponding f-values, the Pnodes with the lowest f-values having greatest priority.

v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^

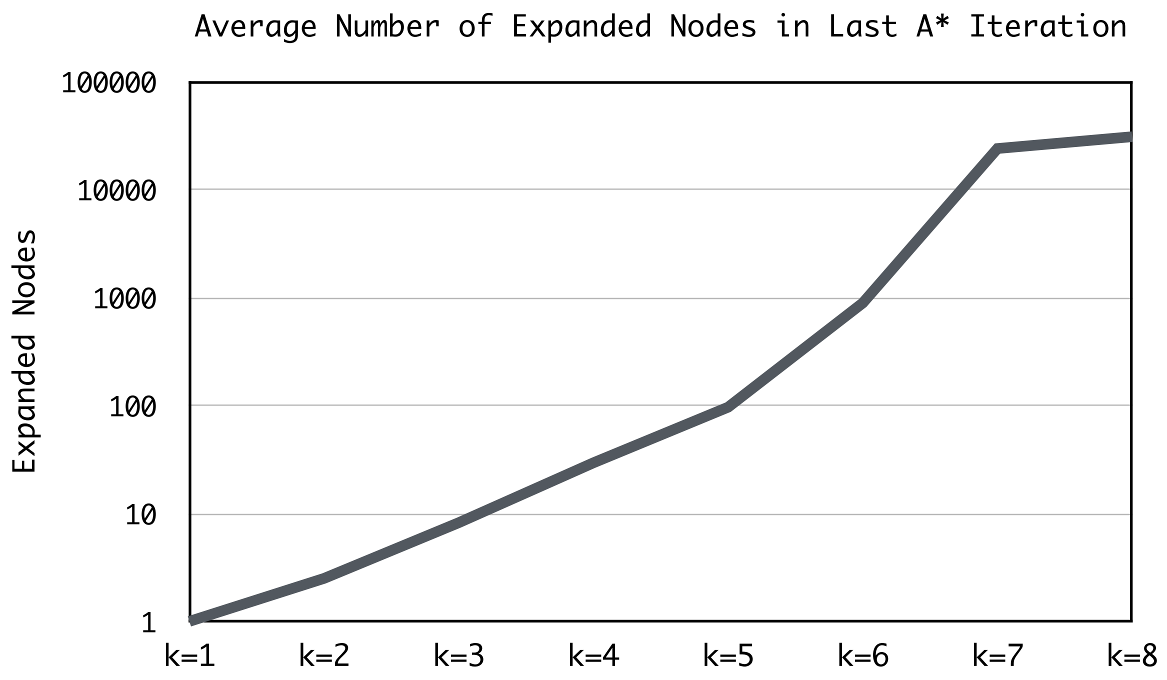
## | The Heuristic Used |

Going back to the built-in heuristic function implemented in the Pnode struct, the heuristic I chose to use may not be the best, but it does work: max(# colors found on each face) - 1. In other terms, this heuristic function looks at each face of the pyraminx and sees how many colors it finds on it. After counting this total for each face, it picks the maximum total found among the puzzle’s faces and subtracts 1 from it. At the very least, this shows how many colors are not on the side they need to be and thus the LEAST number of rotations that will be used to solve it (for example, rotating one level of the pyraminx puzzle will leave three sides with two colors. 2-1=1, which we obviously know is the number of moves to fully solve the puzzle.). Additionally, a heuristic of 0 tells us that the puzzle has no stickers out of place and therefore is in a solved state.

v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^

## | The PNode Plot |

==============================================================



==============================================================

As seen in the graph above, the average number of expanded nodes in the last IDA∗ iteration dramatically increases as the number of randomizing moves, k, increases. This is likely due to the exponentially-growing nature of the tree containing many possible configurations of the pyraminx. Though IDA∗ can help in conserving memory usage, it can only do so much in the context of how effective the whole program and heuristic is.

v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^

## | What I Learned |

After completing this second assignment, I gained extensive knowledge of heuristics, their importance in certain algorithms, and the usefulness of algorithms such as A∗, Iterative Deepening A∗ (IDA∗), and other related algorithms.

Additionally, I feel as though I have a greater understanding of just how algorithms can be incorporated to accomplish different tasks and find solutions to problems based on the the data its given.

v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^v^