COSC326  
Etude 10  
Group Members:  
Sayyed -   
Samuel - 2955262

We are looking at a 2x2x2 cube, made of 1x1x1 cubes. These 1x1x1 cubes can be either color x or y. Thus, our cube has 8 parts of 2 colors, implying 8^2 = 256 unique combinations, without factoring in rotations.

Using python, we simulated a cube through a 3-dimensional array, initialized to zero to represent a 2x2x2 cube of one color. To add another color, we chose to change values from 0 to 1, thereby simulating a cube of 2 different colors.

To simulate rotations for every possible face, which should be 4 orientations with 6 faces, we found a code online that allowed us to simulate all rotations and orientations at <https://stackoverflow.com/questions/33190042/how-to-calculate-all-24-rotations-of-3d-array>.

Now that we had the ability to simulate a cube of any number of x and y colors and their rotations, we can generate all combinations of x and y colored cubes. For any specific combination, we can generate their permutations and iterate through them, adding one to a counter variable for every unique state. As we iterate through a permutation, we generate all their rotations and remove them from our permutations allowing us to find all unique combinations with rotations, which is 22.

Some interesting things we found while doing this was that when considering all unique combinations, pascal’s triangle was able to generate the number of unique combinations without factoring in rotations for each variant. Consider the following:

1,8,28,56,70,56,28,8,1

This is pascal’s triangle when n = 8.

Each value corresponds with the length of our lists when generating permutations from 0-8. Through this, we know that there is likely a formula that we can generate for repeated rotations, factoring them into the formula for permutations without repetition to get the values we desire.