Abstract
We cover the symmetry group of the cuboctahedron and how to determine these symmetries. We discuss tiles created using a cuboctahedron as a model, how they are named, labeled, and found, in addition to proving that all unique tiles have been found. We present an algorithm, written in Java, with the purpose of finding each unique tile in its lexicographically minimal representation. Along with the program, we present a proof to demonstrate that the operations performed by the program on the individual tiles are valid and cover each member of the symmetry group of the cuboctahedron. We explain the results we got from our program and how finding each tile type is helpful in the broader scope of the project.

Problem Statement
Our goal was to identify all unique \( n \)-armed tiles, where \( n \in [0-12] \).

Results
<table>
<thead>
<tr>
<th># of Arms</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td># of Tiles</td>
<td>1</td>
<td>4</td>
<td>10</td>
<td>27</td>
<td>58</td>
<td>105</td>
<td>174</td>
<td>271</td>
<td>394</td>
<td>556</td>
<td>765</td>
</tr>
</tbody>
</table>

Lex-Minimal 3-Arm Tiles

Tile Imbalance
Note that the number of tiles with \( n \) arms and tiles with \( 12-n \) arms are unequal. This is because a single tile with 2 arms may be rotated to fit into the gap in 2 distinct 10-arm tiles, as shown below. The same holds for tiles with 3 and 5 arms, etc.

Conclusions
From this research we were able to completely classify every lexic-minimal tile. This will be vital in future research or what can be constructed from these tiles. The interactive version of our program is also capable of displaying the full orbit of all lexic-minimal n-armed tiles, given n. It is also capable of reading in the description of an arbitrary tile and displaying the lexic-minimal form of that tile as output.

Additionally, the strategies and code we employed to solve this problem once again, should future teams become interested in categorizing tiles based on geometric structures other than the cuboctahedron.

Definitions
- **Tile**: A vertex with arms attached.
- **Unique**: A unique tile type is the lexicographical representative for the group of tiles that are rotationally isomorphic to it.
- **Lexicographically Minimal**: A tile is lexic-minimal when it has its arms in positions that are in the lowest orders, like a form of alphabetical order for tiles.

Labelling of Tiles
Arms are labeled based on what plane they are in, the cuboctahedron.
- **Labeled**: \( x \), \( y \), \( z \)
- Each plane has four arms: \( a_1 \), \( a_2 \), \( b_1 \), \( b_2 \), \( c_1 \), \( c_2 \), \( d_1 \), \( d_2 \)

Rotations

Octahedral
The octahedral group is a point group of symmetries of order 48 that includes inversions. However for this project's rotational purposes it will only deal with the order 24 rotational subgroup.

- \( \alpha_1 \), \( \alpha_2 \), \( \alpha_3 \), \( \alpha_4 \), \( \beta_1 \), \( \beta_2 \), \( \gamma_1 \), \( \gamma_2 \), \( \gamma_3 \), \( \gamma_4 \)

Background

Exhaustive Case Finder

Intro
The Exhaustive Case Finder is the name of our overall program and having explained the processes taking place in the code; this next section will provide in brief the pseudo-code for our program. In order to accomplish our stated goals, we wrote four classes for our program. These four classes each play a specific role in the creation of the result and are built around each other.

Pseudo Code

- **Generators**

- **CuboctahedronSymmetry**: This performs the generator rotations in order to further group distinct tiles not distinguished between by the CuboctahedronLUT class.

- **CuboctahedronLUT**: Lookup table used to group the tiles generated by Cubes based on the angles between arm-pair. Used call to the AngleWrapper class in order to provide hashing and equality functions for sets of angles.

- **AngleWrapper**: Simply a wrapper class used for the CuboctahedronLUT class. Wraps arrays representing angles between arm-pairs to store Java’s hashable arrays wrapper, arm-comparison behavior (performs a shallow element comparison rather than a simple object reference comparisons).

Program Flowchart

Acknowledgments