Let \( G = (V, E) \) be a finite graph, and let \( V = \{v_1, v_2, \ldots, v_n\} \). The Chromatic Symmetric Polynomial is a function of countably many commuting indeterminates, and is defined as:

\[
X_G(x_1, x_2, \ldots) = \prod_{e \in E} (x_1 + x_2) = \prod_{e \in E} \sum_{k} x_k^d_e \tag{1}
\]

where \( \{1, 2, \ldots, n\} \rightarrow \mathbb{N} \) is a coloring of \( G \) and the sum above is taken over all proper colorings. In [1], Stanley provides another form of \( X_G \) that we will primarily be using throughout our research. The power-sum functions, \( p_m = \sum_{i} x_i^m \) provide a basis for the space of symmetric polynomials and \( X_G \) can be written as:

\[
X_G = \sum_{S \subseteq E} (-1)^{|E(S)|} p_{\pi(S)} \tag{2}
\]

where \( \pi(S) = \{\pi_1, \pi_2, \ldots, \pi_r\} \) is an integer partition of \( n \) with each \( \pi_i \) corresponding to the number of vertices in a disjoint component of \( G \) after removing every edge not in \( S \) and where \( p_\pi(S) = p_{\pi_1} p_{\pi_2} \cdots p_{\pi_r} \) is a product of power sum functions.

Also in [1], Stanley both gives the definition above, and the following conjecture:

**Conjecture.** \( X_G \) distinguishes trees.

That is, for any two non-isomorphic trees, \( G_1 \) and \( G_2 \), we have \( X_{G_1} \neq X_{G_2} \).

It’s already known that \( X_G \) does not distinguish graphs in general (see Fig. 0).

![Figure 0: Two non-isomorphic graphs with the same Symmetric Chromatic Polynomial](image)

The representation given in (2) is much easier for computation, either by hand or by computer. From now on we'll be considering only trees, and we investigate the sets of partitions formed by taking subsets of edges as described above. The main focus of our research was to write a program to search through all trees up to 25 vertices and record the specific ones which have many matching terms of \( X_G \).

The number 25 was chosen due to the limits in processor speed. Since the number of unique trees grows approximately exponentially as \( n \) increases, our program was forced to cut down the possible number of vertices on its largest branch. For example, in fig. 3 the two centroids have weight 8. Generally speaking, when two trees have the same 2-cuts or 3-cuts one of the pair can be obtained by permuting some subset of branches about the centroids of the other. We don’t yet have a method for determining which of these branch permutations (if any) will give a new tree with identical 2-cuts or 3-cuts.

![Figure 1: The smallest pair of trees with identical 1-cuts](image)

In addition, every pair of two-centroid trees on \( n \) vertices with identical cuts can be extended to a pair of \( n+1 \) vertex trees with identical cuts simply by splitting the central edge and placing a vertex in the middle. Hence, when going from an even to an odd order the total number of non-unique trees grows relatively little. However, going from an odd to an even order there’s a much larger increase. This is most likely related to the fact that only trees with an even order can have two centroids, which have a central edge connecting them. Such a central edge makes it more likely for there to be valid branch permutations as described above.

![Figure 2: The smallest pair of trees with identical 2-cuts. Circled vertices are the centroids](image)

In addition, every pair of two-centroid trees on \( n \) vertices with identical cuts can be extended to a pair of \( n+1 \) vertex trees with identical cuts simply by splitting the central edge and placing a vertex in the middle. Hence, when going from an even to an odd order the only “new” trees with non-identical cuts are those whose branch permutations were only made valid through the addition of such a central vertex (since otherwise they’d have shown up among the previous trees with an even order).

![Figure 3: The smallest pair of trees with identical 3-cuts. Circled vertices are the centroids](image)