Draft Report

Warvin Hassan

December 17, 2024

Contents

Part I
Introduction

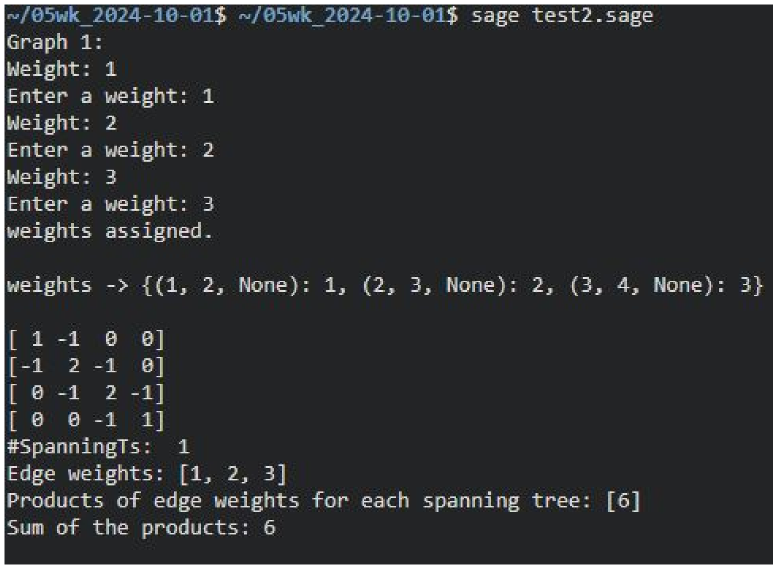
The Total Graph Value (TGV) of a graph is the sum of the products of edge weights across all spanning trees of the graph. To determine the TGV for a weighted graph, all spanning trees of the graph must first be identified. A spanning tree is a subgraph that includes all the vertices of the original graph, is connected, and contains no cycles. For each spanning tree, the product of the weights of its edges is calculated. The TGV is then obtained by summing these products over all spanning trees. This value provides insight into the relationship between the graph's structure and its edge weights. In this report, we focus on calculating the TGV for various types of graphs, with an emphasis on trees, cycles, divided cycles, and complete graphs. Specifically, we investigate how the assignment of weights to edges affects the TGV and explore methods for maximizing this value. The methods used in this study are primarily computational in nature, involving the application of combinatorial principles, graph algorithms, and symbolic computation using the SageMath software environment.

Part II
Results

1 Trees

We found that for trees the total graph value was rather trivial since there was only one spanning tree for any given tree and so the product of these graphs is the total graph value.

1.1 Code



```
~/05wk_2024-10-01$ ~/05wk_2024-10-01$ sage test2.sage
Graph 1:
Weight: 1
Enter a weight: 1
Weight: 2
Enter a weight: 2
Weight: 3
Enter a weight: 3
weights assigned.

weights -> {(1, 2, None): 1, (2, 3, None): 2, (3, 4, None): 3}

[ 1 -1  0  0]
[-1  2 -1  0]
[ 0 -1  2 -1]
[ 0  0 -1  1]
#SpanningTs:  1
Edge weights: [1, 2, 3]
Products of edge weights for each spanning tree: [6]
Sum of the products: 6
```

Figure 1: Simple tree and its results using SageMath

1.2 Proof

The following proof will show that it does not matter how one assigns weights to trees by proving that the total value of the respective graph remains the same.

Let $n \in \mathbb{Z}^+$, then a tree T_n on n vertices has $n-1$ edges and is already a tree. Since the graph is a tree the only spanning tree is the tree itself, with the total graph value being the product of the weights of all the edges in the tree

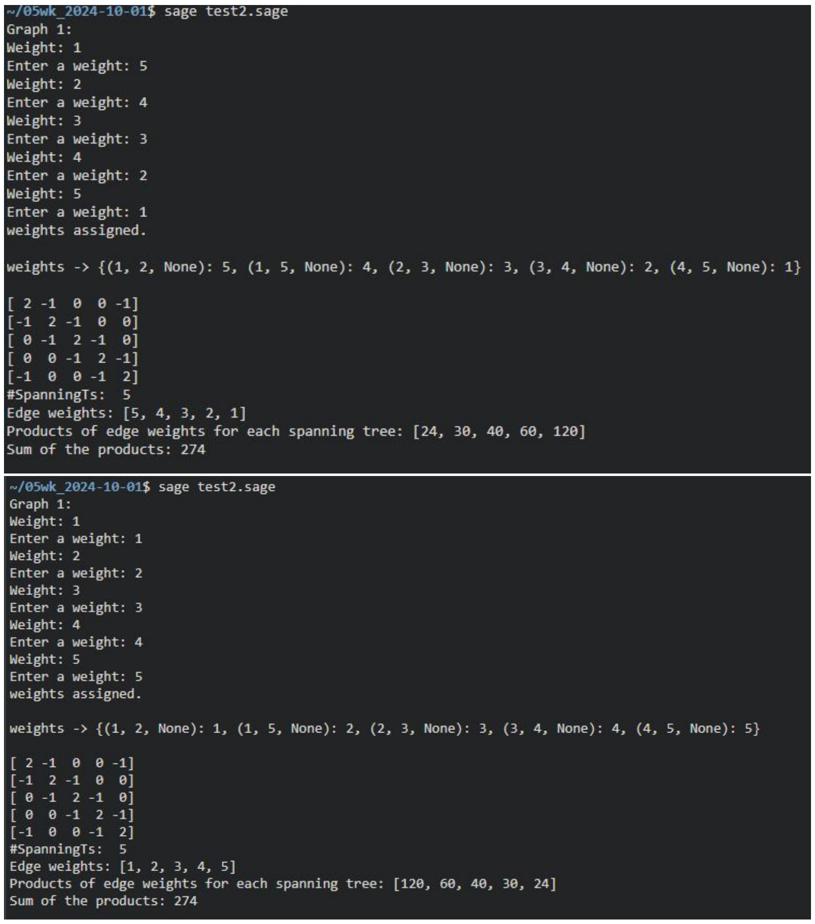
$$TGV = w_1 \cdot w_2 \cdot \ldots \cdot w_{n-1.}$$

■

2 Cycles

Cycles were also pretty trivial as we found that regardless of the way we distributed the weights on the edges we got the same total graph value and this was due to the symmetery of cyclic graphs. We noticed that the number of spanning trees of a cycle was also the same number of vertices/edges of that same cycle. This is because a spanning tree is found by removing an edge of the cycle.

2.1 Code



```
~/05wk_2024-10-01$ sage test2.sage
Graph 1:
Weight: 1
Enter a weight: 5
Weight: 2
Enter a weight: 4
Weight: 3
Enter a weight: 3
Weight: 4
Enter a weight: 2
Weight: 5
Enter a weight: 1
weights assigned.

weights -> {(1, 2, None): 5, (1, 5, None): 4, (2, 3, None): 3, (3, 4, None): 2, (4, 5, None): 1}

[ 2 -1  0  0 -1]
[-1  2 -1  0  0]
[ 0 -1  2 -1  0]
[ 0  0 -1  2 -1]
[-1  0  0 -1  2]
#SpanningTs:  5
Edge weights: [5, 4, 3, 2, 1]
Products of edge weights for each spanning tree: [24, 30, 40, 60, 120]
Sum of the products: 274
```

```
~/05wk_2024-10-01$ sage test2.sage
Graph 1:
Weight: 1
Enter a weight: 1
Weight: 2
Enter a weight: 2
Weight: 3
Enter a weight: 3
Weight: 4
Enter a weight: 4
Weight: 5
Enter a weight: 5
weights assigned.

weights -> {(1, 2, None): 1, (1, 5, None): 2, (2, 3, None): 3, (3, 4, None): 4, (4, 5, None): 5}

[ 2 -1  0  0 -1]
[-1  2 -1  0  0]
[ 0 -1  2 -1  0]
[ 0  0 -1  2 -1]
[-1  0  0 -1  2]
#SpanningTs:  5
Edge weights: [1, 2, 3, 4, 5]
Products of edge weights for each spanning tree: [120, 60, 40, 30, 24]
Sum of the products: 274
```

Figure 2: Same Results for Two Distinctly Distributed C5s

2.2 Proof

Let the two weighted n-cycles be denoted as C_1 and C_2. The Edges of C_1 can be represented as $E_1 = \{e_1, e_2, \ldots, e_n\}$ and the edges of C_2 as $E_2 = \{f_1, f_2, \ldots, f_n\}$. The weights assigned to the edges of both C_1 and C_2 are derived from the same set $W = \{w_1, w_2 \ldots, w_n\}$ and are assigned randomly; *i.e.*, for some weight $w_j = e_a$ and $w_j = f_b$ where $a, b \leq n$. In an

n-cycle C_n, a spanning tree is formed by removing exactly one edge from the cycle, leaving a connected graph with $n-1$ edges. Since the edges of the cycles C_1 and C_2 are in bijective correspondence through the mapping m, and the weights are derived from the same set W, the spanning trees of C_1 and C_2 are also in bijective correspondence.

For any weight assignment, the product of the weights for a given spanning tree is determined by the $n-1$ edges that remain. Because the mapping m ensures that the same weights correspond to edges in both cycles, the sum of the products of weights across all spanning trees (the total graph value, TGV) must be the same for C_1 and C_2.

Thus, the total graph values for C_1 and C_2 are equal:

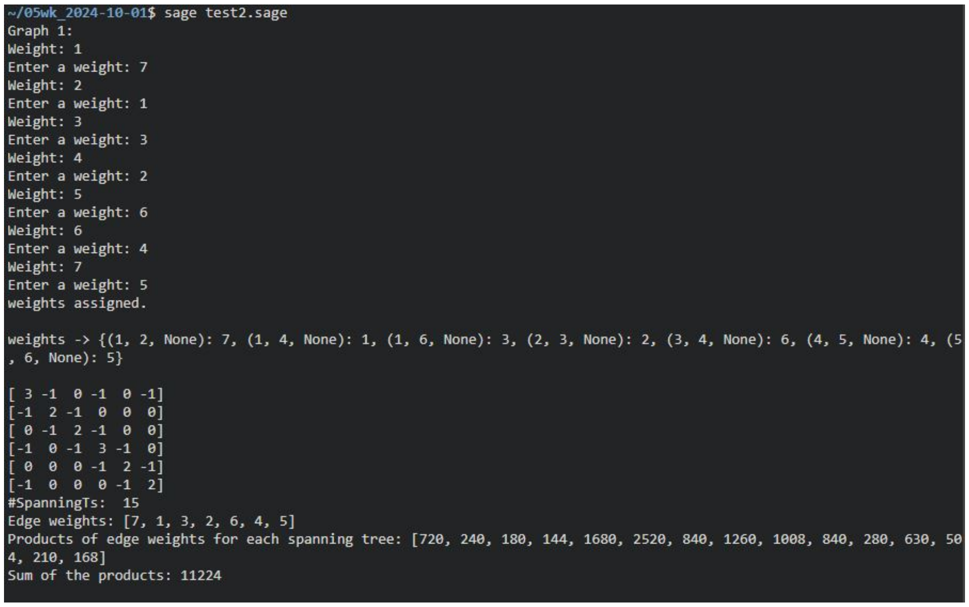
$$TGV(C_1) = TGV(C_2).$$

■

3 Divided Cycles

We found that divided cycles behaved rather predictibly since they were essentially cycles with an extra edge. The extra edge seemed to contribute insignificantly to the total overall graph value because it simply was not included in many spanning trees. With that being said, we noticed that while assigning the lowest weight to the dividing edge did indirectly contribute to the total graph value, that alone is not enough to determine the max value for any given divided cycle. Other factors, such as the distribution of weights along the cycle edges, played a significant role in determining the total graph value. Specifically, configurations that balanced the weights across the cycle edges while minimizing the impact of the dividing edge on critical spanning tree combinations tended to yield higher graph values. This suggests that the interplay between the edge weights and the structural redundancy of spanning trees in divided cycles must be carefully considered to optimize the total graph value, rather than relying solely on minimizing the weight of the dividing edge.

3.1 Code

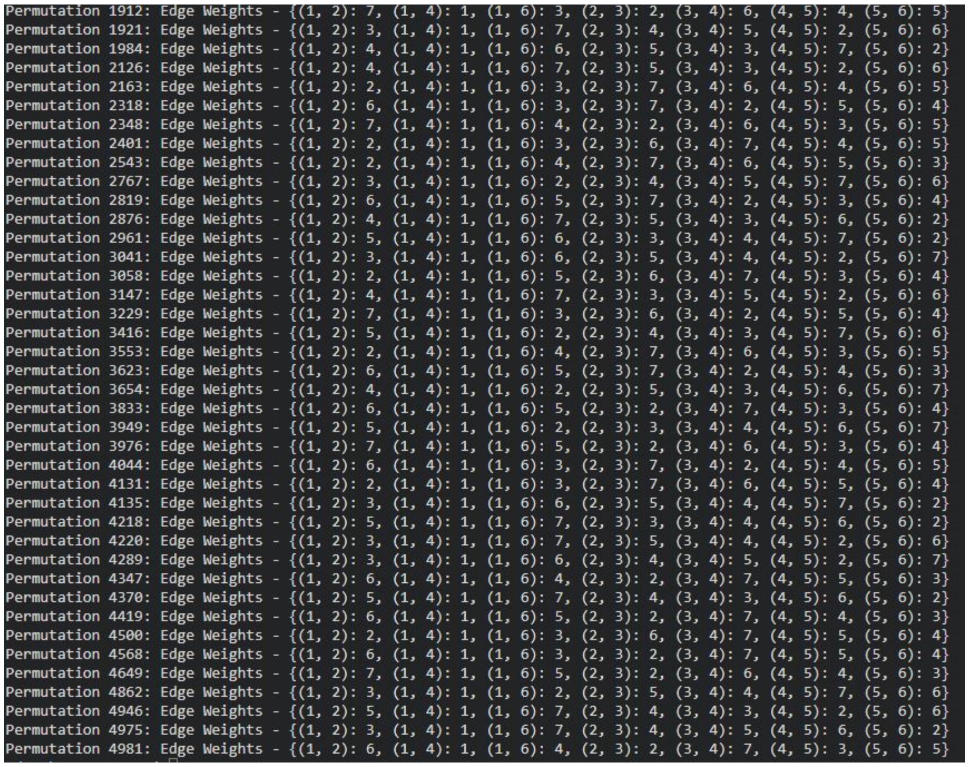


```
~/05wk_2024-10-01$ sage test2.sage
Graph 1:
Weight: 1
Enter a weight: 7
Weight: 2
Enter a weight: 1
Weight: 3
Enter a weight: 3
Weight: 4
Enter a weight: 2
Weight: 5
Enter a weight: 6
Weight: 6
Enter a weight: 4
Weight: 7
Enter a weight: 5
weights assigned.

weights -> {(1, 2, None): 7, (1, 4, None): 1, (1, 6, None): 3, (2, 3, None): 2, (3, 4, None): 6, (4, 5, None): 4, (5
, 6, None): 5}

[ 3 -1  0 -1  0 -1]
[-1  2 -1  0  0  0]
[ 0 -1  2 -1  0  0]
[-1  0 -1  3 -1  0]
[ 0  0  0 -1  2 -1]
[-1  0  0  0 -1  2]
#SpanningTs:  15
Edge weights: [7, 1, 3, 2, 6, 4, 5]
Products of edge weights for each spanning tree: [720, 240, 180, 144, 1680, 2520, 840, 1260, 1008, 840, 280, 630, 50
4, 210, 168]
Sum of the products: 11224
```

Figure 3:   The Maximum value for a $C_6 + 1$.



```
Permutation 1912: Edge Weights - {(1, 2): 7, (1, 4): 1, (1, 6): 3, (2, 3): 2, (3, 4): 6, (4, 5): 4, (5, 6): 5}
Permutation 1921: Edge Weights - {(1, 2): 3, (1, 4): 1, (1, 6): 7, (2, 3): 4, (3, 4): 5, (4, 5): 2, (5, 6): 6}
Permutation 1984: Edge Weights - {(1, 2): 4, (1, 4): 1, (1, 6): 6, (2, 3): 5, (3, 4): 3, (4, 5): 7, (5, 6): 2}
Permutation 2126: Edge Weights - {(1, 2): 4, (1, 4): 1, (1, 6): 7, (2, 3): 5, (3, 4): 3, (4, 5): 2, (5, 6): 6}
Permutation 2163: Edge Weights - {(1, 2): 2, (1, 4): 1, (1, 6): 3, (2, 3): 7, (3, 4): 6, (4, 5): 4, (5, 6): 5}
Permutation 2318: Edge Weights - {(1, 2): 6, (1, 4): 1, (1, 6): 3, (2, 3): 7, (3, 4): 2, (4, 5): 5, (5, 6): 4}
Permutation 2348: Edge Weights - {(1, 2): 7, (1, 4): 1, (1, 6): 4, (2, 3): 2, (3, 4): 6, (4, 5): 3, (5, 6): 5}
Permutation 2401: Edge Weights - {(1, 2): 2, (1, 4): 1, (1, 6): 3, (2, 3): 6, (3, 4): 7, (4, 5): 4, (5, 6): 5}
Permutation 2543: Edge Weights - {(1, 2): 2, (1, 4): 1, (1, 6): 4, (2, 3): 7, (3, 4): 6, (4, 5): 5, (5, 6): 3}
Permutation 2767: Edge Weights - {(1, 2): 3, (1, 4): 1, (1, 6): 2, (2, 3): 4, (3, 4): 5, (4, 5): 7, (5, 6): 6}
Permutation 2819: Edge Weights - {(1, 2): 6, (1, 4): 1, (1, 6): 5, (2, 3): 7, (3, 4): 2, (4, 5): 3, (5, 6): 4}
Permutation 2876: Edge Weights - {(1, 2): 4, (1, 4): 1, (1, 6): 7, (2, 3): 5, (3, 4): 3, (4, 5): 6, (5, 6): 2}
Permutation 2961: Edge Weights - {(1, 2): 5, (1, 4): 1, (1, 6): 6, (2, 3): 3, (3, 4): 4, (4, 5): 7, (5, 6): 2}
Permutation 3041: Edge Weights - {(1, 2): 3, (1, 4): 1, (1, 6): 6, (2, 3): 5, (3, 4): 4, (4, 5): 2, (5, 6): 7}
Permutation 3058: Edge Weights - {(1, 2): 2, (1, 4): 1, (1, 6): 5, (2, 3): 6, (3, 4): 7, (4, 5): 3, (5, 6): 4}
Permutation 3147: Edge Weights - {(1, 2): 4, (1, 4): 1, (1, 6): 7, (2, 3): 3, (3, 4): 5, (4, 5): 2, (5, 6): 6}
Permutation 3229: Edge Weights - {(1, 2): 7, (1, 4): 1, (1, 6): 3, (2, 3): 6, (3, 4): 2, (4, 5): 5, (5, 6): 4}
Permutation 3416: Edge Weights - {(1, 2): 5, (1, 4): 1, (1, 6): 2, (2, 3): 4, (3, 4): 3, (4, 5): 7, (5, 6): 6}
Permutation 3553: Edge Weights - {(1, 2): 2, (1, 4): 1, (1, 6): 4, (2, 3): 7, (3, 4): 6, (4, 5): 3, (5, 6): 5}
Permutation 3623: Edge Weights - {(1, 2): 6, (1, 4): 1, (1, 6): 5, (2, 3): 7, (3, 4): 2, (4, 5): 4, (5, 6): 3}
Permutation 3654: Edge Weights - {(1, 2): 4, (1, 4): 1, (1, 6): 2, (2, 3): 5, (3, 4): 3, (4, 5): 6, (5, 6): 7}
Permutation 3833: Edge Weights - {(1, 2): 6, (1, 4): 1, (1, 6): 5, (2, 3): 2, (3, 4): 7, (4, 5): 3, (5, 6): 4}
Permutation 3949: Edge Weights - {(1, 2): 5, (1, 4): 1, (1, 6): 2, (2, 3): 3, (3, 4): 4, (4, 5): 6, (5, 6): 7}
Permutation 3976: Edge Weights - {(1, 2): 7, (1, 4): 1, (1, 6): 5, (2, 3): 2, (3, 4): 6, (4, 5): 3, (5, 6): 4}
Permutation 4044: Edge Weights - {(1, 2): 6, (1, 4): 1, (1, 6): 3, (2, 3): 7, (3, 4): 2, (4, 5): 4, (5, 6): 5}
Permutation 4131: Edge Weights - {(1, 2): 2, (1, 4): 1, (1, 6): 3, (2, 3): 7, (3, 4): 6, (4, 5): 5, (5, 6): 4}
Permutation 4135: Edge Weights - {(1, 2): 3, (1, 4): 1, (1, 6): 6, (2, 3): 5, (3, 4): 4, (4, 5): 7, (5, 6): 2}
Permutation 4218: Edge Weights - {(1, 2): 5, (1, 4): 1, (1, 6): 7, (2, 3): 3, (3, 4): 4, (4, 5): 6, (5, 6): 2}
Permutation 4220: Edge Weights - {(1, 2): 3, (1, 4): 1, (1, 6): 7, (2, 3): 5, (3, 4): 4, (4, 5): 2, (5, 6): 6}
Permutation 4289: Edge Weights - {(1, 2): 3, (1, 4): 1, (1, 6): 6, (2, 3): 4, (3, 4): 5, (4, 5): 2, (5, 6): 7}
Permutation 4347: Edge Weights - {(1, 2): 6, (1, 4): 1, (1, 6): 4, (2, 3): 2, (3, 4): 7, (4, 5): 5, (5, 6): 3}
Permutation 4370: Edge Weights - {(1, 2): 5, (1, 4): 1, (1, 6): 7, (2, 3): 4, (3, 4): 3, (4, 5): 6, (5, 6): 2}
Permutation 4419: Edge Weights - {(1, 2): 6, (1, 4): 1, (1, 6): 5, (2, 3): 2, (3, 4): 7, (4, 5): 4, (5, 6): 3}
Permutation 4500: Edge Weights - {(1, 2): 2, (1, 4): 1, (1, 6): 3, (2, 3): 6, (3, 4): 7, (4, 5): 5, (5, 6): 4}
Permutation 4568: Edge Weights - {(1, 2): 6, (1, 4): 1, (1, 6): 3, (2, 3): 2, (3, 4): 7, (4, 5): 5, (5, 6): 4}
Permutation 4649: Edge Weights - {(1, 2): 7, (1, 4): 1, (1, 6): 5, (2, 3): 2, (3, 4): 6, (4, 5): 4, (5, 6): 3}
Permutation 4862: Edge Weights - {(1, 2): 3, (1, 4): 1, (1, 6): 2, (2, 3): 5, (3, 4): 4, (4, 5): 7, (5, 6): 6}
Permutation 4946: Edge Weights - {(1, 2): 5, (1, 4): 1, (1, 6): 7, (2, 3): 4, (3, 4): 3, (4, 5): 2, (5, 6): 6}
Permutation 4975: Edge Weights - {(1, 2): 3, (1, 4): 1, (1, 6): 7, (2, 3): 4, (3, 4): 5, (4, 5): 6, (5, 6): 2}
Permutation 4981: Edge Weights - {(1, 2): 6, (1, 4): 1, (1, 6): 4, (2, 3): 2, (3, 4): 7, (4, 5): 3, (5, 6): 5}
```

Figure 4:   Many $C_6 + 1$ permutations with same max value.

3.2 Proof

Let C_n be a cycle graph with n vertices, and let the edges of C_n be labeled $e_1, e_2, \ldots, e_n$, with corresponding edge weights $w_1, w_2, \ldots, w_n$. Let e_d denote the dividing edge, which connects two non-adjacent vertices of C_n and "divides" the cycle. The total graph value $T(G)$ is defined as the sum of the products of edge weights over all spanning trees of G. To analyze the impact of assigning weights, we partition the spanning trees into three disjoint sets.

Fix an edge on the cycle e_j. The first set, S_1, consists of all spanning trees that either include both e_d and some edge e_j, or exclude both e_d and e_j. Notice that swapping the weights of e_d and e_j does not alter the contribution of the spanning trees in S_1, as the product terms involving w_d and w_j remain identical. S_2 is the set that includes either e_j or e_d but not both and $S_2 \neq \emptyset$ Finally, S_3 is the converse set of S_2, *i.e.*, the set does not take one edge from each side of the divider but those that may take two from one side.

Since $w_j > w_d$ by assumption, the sign of the change depends on the difference of the spanning trees that include e_d and those that include e_j. The key observation is that the contribution of spanning trees that include e_j is larger because e_j is a cycle edge, and cycle edges appear in more spanning trees compared to e_d, which connects non-adjacent vertices.

Notice that W.L.O.G. each tree that contains e_d, but not e_j has a TGV that corresponds to a tree that contains e_j but not e_d and all of the other edges remain the same. However the converse of this is not true. Refer to figures 5 and 6 to see this.

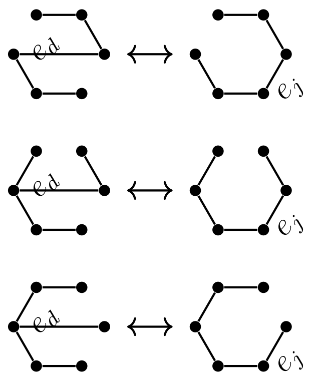


Figure 5: Spanning trees of S_2.

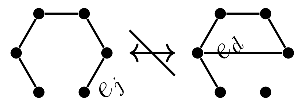


Figure 6: Counterexample: A spanning tree of S_3.

Further more, this implies that swapping w_d with w_j reduces the total graph value $T(G)$. To maximize $T(G)$, w_d must be minimized, ensuring its impact on S_2 is as small as possible.

Assigning the smallest weight to e_d achieves this, leading to the maximization of the total graph value.

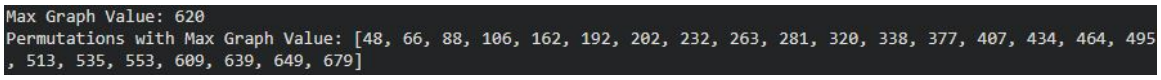
$\blacksquare$

Part III
Comments on Complete Graphs and others

We were unable to find a general solution for complete graphs but we were able to find some interesting results that are still worth mentioning.

4 K_4 Results

For K_4 we were able to determine total graph value and all its permutations via brute force. We conjectured a connection with its Hamiltonian cycles, but we were not able to prove this. Due to its symmetry we believe that all of these max permutations are isomorphic to each other but did not prove this rigorously enough to arrive at a satisfactory answer either.

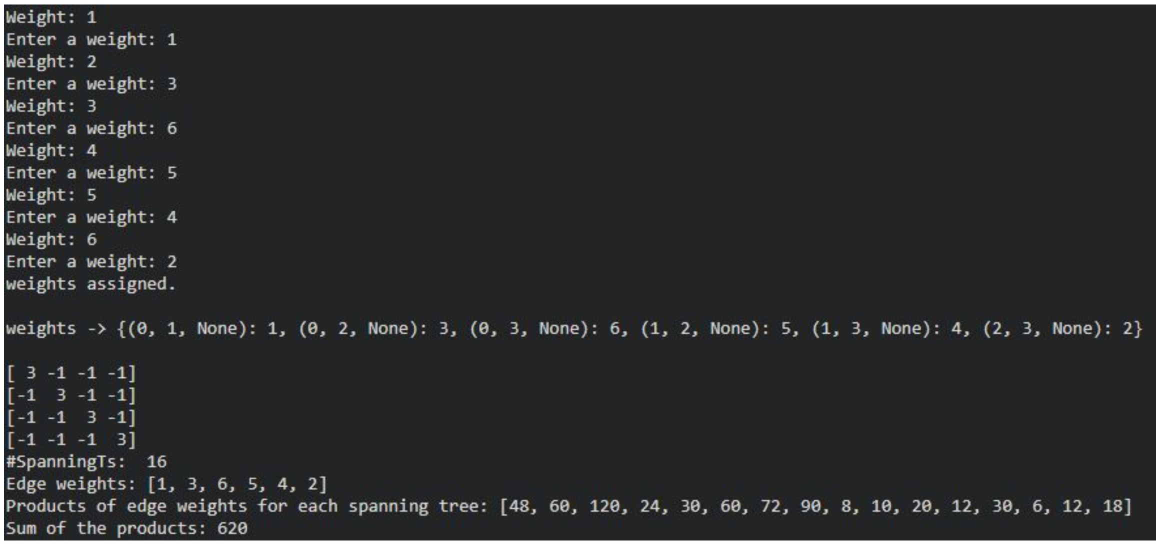


```
Max Graph Value: 620
Permutations with Max Graph Value: [48, 66, 88, 106, 162, 192, 202, 232, 263, 281, 320, 338, 377, 407, 434, 464, 495
, 513, 535, 553, 609, 639, 649, 679]
```

Figure 7: All Permutations of K4 with Max Value.

5 K_5 Limitations

For K_5 we ran into hardware limitations as the processing load became too massive to fully compile in a reasonable amount of time. While we were unable to determine the maximum graph value for K_5, we were able to work around this slightly by using a pseudo-random permutation generator and fixed weights to lessen the burden on the processor. There are $\binom{5}{2} = 10$ edges for K_5, if we assign a distinct weight to each edge, there are $10! = 3,628,800$ ways to arrange these weights among the edges, the conjectured TGV for K_5 that we found was 102701.

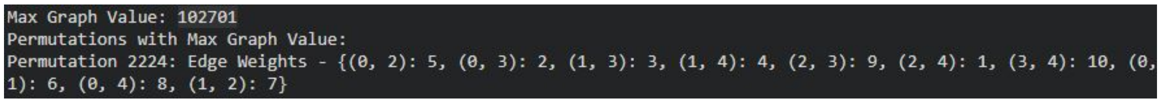


```
Weight: 1
Enter a weight: 1
Weight: 2
Enter a weight: 3
Weight: 3
Enter a weight: 6
Weight: 4
Enter a weight: 5
Weight: 5
Enter a weight: 4
Weight: 6
Enter a weight: 2
weights assigned.

weights -> {(0, 1, None): 1, (0, 2, None): 3, (0, 3, None): 6, (1, 2, None): 5, (1, 3, None): 4, (2, 3, None): 2}

[ 3 -1 -1 -1]
[-1  3 -1 -1]
[-1 -1  3 -1]
[-1 -1 -1  3]
#SpanningTs:  16
Edge weights: [1, 3, 6, 5, 4, 2]
Products of edge weights for each spanning tree: [48, 60, 120, 24, 30, 60, 72, 90, 8, 10, 20, 12, 30, 6, 12, 18]
Sum of the products: 620
```

Figure 8: Permutation 48: (1, 3, 6, 5, 4, 2) with Graph Value = 620 Edge weights: (0, 1): 1, (0, 2): 3, (0, 3): 6, (1, 2): 5, (1, 3): 4, (2, 3): 2.



```
Max Graph Value: 102701
Permutations with Max Graph Value:
Permutation 2224: Edge Weights - {(0, 2): 5, (0, 3): 2, (1, 3): 3, (1, 4): 4, (2, 3): 9, (2, 4): 1, (3, 4): 10, (0,
1): 6, (0, 4): 8, (1, 2): 7}
```

Figure 9: Conjectured TGV for K_5.

List of Figures