Draft Report

Warvin Hassan

November 27, 2024

Contents

Part I
Results

1 Paths

We found that for paths the total graph value was rather trivial since there was only one spanning tree for any given path and so the product of the path 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 Path and its results using SageMath

1.2 Proof

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

Paths: Let $n \in \mathbb{Z}^+$, then a path P_n on n vertices has $n-1$ edges and is already a tree. It then follows, it has only one spanning tree, which is the path itself. Since the graph is trivially a tree the only spanning tree is the path itself, with the total graph value being the product of the weights of all the edges in the path

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

$\therefore$ TGV is an integer by the commutative laws of integer multiplication and remains the same. ■

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.

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 $w_n = e_a$ and $w_n = f_b$ where $a, b \leq n$. We will define a mapping $m : E_1 \to E_2$ such that $m(e_i) = f_i$ for $i = 1, 2, \ldots, n$. To demonstrate that this

mapping is a bjection, we must show that it is both injective and surjective. To establish injectivity, suppose there exist indices i and j such that $m(e_i) = m(e_j)$. This implies that $f_i = f_j$. Since the edges f_i and f_j correspond to distinct connections in the cycle C_2, we must have $i = j$. Therefore, the mapping m is injective. Next, to show surjectivity, consider any edge $f_k \in E_2$. There exists an edge $e_k \in E_1$ such that $m(e_k) = f_k$. This mapping ensures that every edge in E_2 is accounted for by at least one edge in E_1, thus satisfying the surjectivity condition. Having established that m is both injective and surjective, we conclude that m is a bijection. This one-to-one correspondence between the edges of C_1 and C_2 is independent of the order in which the weights are assigned to the edges. onsequently, since the edges are matched in a bijective manner, the sum of the products of the weights associated with spanning trees of C_1 and C_2 remains invariant under the assignments of weights. Thus, the total graph values for both cycles 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 wasn't 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 C6 + 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 C6 + 1 permutations with same max value.

3.2 Proof

Let $G = C_n + e$ be a graph formed by adding an extra edge e to an n-cycle C_n. This additional edge e connects two non-adjacent vertices in C_n, thereby creating a "divided cycle." Assign weights $w_1, w_2, \ldots, w_n$ to the edges of C_n and a weight w_e to the additional edge e. Our goal is to show that the maximum graph value $V(G)$, which is the sum of the products of the edge weights across all spanning trees of G, is achieved when the smallest weight is assigned to e, with the remaining weights distributed among the edges of C_n.

The graph value $V(G)$ is defined as

$$V(G) = \sum_{T \in \mathcal{T}(G)} \prod_{e \in T} w(e),$$

where $\mathcal{T}(G)$ denotes the set of all spanning trees in G and $w(e)$ represents the weight of edge e.

To calculate $V(G)$, we consider two types of spanning trees in G: those that include the extra edge e and those that exclude it. If a spanning tree includes e, it will consist of e and $n-1$ additional edges chosen from C_n to span all vertices. If a spanning tree excludes e, it must be formed entirely from the edges in C_n, which results in a spanning tree that is equivalent to a spanning tree of C_n itself.

We first calculate the number of spanning trees in each case. For the cycle C_n, there are exactly n spanning trees, as each spanning tree of a cycle graph is formed by removing one of the n edges in C_n. In G, we have two types of spanning trees: those that include e, of which there are $n-1$, and those that exclude e, of which there are n. Therefore, G has n spanning trees that exclude e and $n-1$ spanning trees that include e. Consequently, spanning trees that exclude e form the majority of spanning trees in G.

Next, we analyze the effect of weight assignment on the graph value $V(G)$. Consider the configuration where the smallest weight $w_{\min}$ is assigned to e, while the remaining, larger weights $w_1, w_2, \ldots, w_n$ are assigned to the edges of C_n. We now evaluate the contributions to $V(G)$ from the two cases of spanning trees under this weight assignment.

For spanning trees that include e, each tree's contribution to the graph value will contain the term $w_{\min}$ due to the inclusion of e, resulting in a product of the form $w_{\min} \prod_{i=1}^{n-2} w(e_i)$, where the product is taken over the $n-2$ edges chosen from the $n-1$ edges of C_n. Let $\mathcal{T}_1$ denote the set of spanning trees in G that include e. The total contribution from these spanning trees is

$$V_1 = w_{\min} \sum_{T \in \mathcal{T}_1} \prod_{e \in T \setminus \{e\}} w(e).$$

Now, consider the effect of assigning $w_{\min}$ to an edge f in C_n instead of e. If f has weight $w_{\min}$, the contributions from spanning trees that exclude e will involve products containing $w_{\min}$, while those that include e will no longer gain the benefit of a small weight on e. We compare this to the case where $w_{\min}$ is assigned to e:

Case 1: Assign $w_{\min}$ to e

The contribution V_1 from spanning trees including e is minimized due to the small $w_{\min}$, while V_2, the contribution from spanning trees excluding e, is maximized because the larger

weights are concentrated on the edges of C_n. This configuration ensures that the graph value $V(G) = V_1 + V_2$ is as large as possible.

Case 2: Assign $w_{\min}$ to $f \in C_n$

Now, spanning trees that exclude e will include f with weight $w_{\min}$. The contribution V_2 is reduced because every spanning tree in $\mathcal{T}_2$ will contain the smaller $w_{\min}$ in its product. Additionally, the contribution V_1 from spanning trees including e is not minimized as effectively, because e now has a larger weight. This results in a smaller total graph value $V(G)$.

To see this effect explicitly, note that V_2 in Case 2 is proportional to

$$w_{\min} \cdot \sum_{T \in \mathcal{T}_2'} \prod_{e \in T \setminus \{f\}} w(e),$$

where $\mathcal{T}_2'$ is the set of spanning trees in C_n that include f. Since $\mathcal{T}_2'$ contains $n-1$ trees, the factor of $w_{\min}$ uniformly reduces the contribution from all these spanning trees. By contrast, in Case 1, V_2 does not include $w_{\min}$, allowing the larger weights to dominate the product.

Thus, assigning $w_{\min}$ to any edge $f \in C_n$ reduces $V(G)$ compared to assigning it to e. This proves that placing the smallest weight $w_{\min}$ on the extra edge e maximizes the graph value $V(G)$.

■

Part II
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 its Hamiltonian cycles. 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.

```
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 5: 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.

List of Figures