An Experimental Note on Graham's Tree Reconstruction Conjecture

Kaylee Weatherspoon and Doron Zeilberger

June 2025

1 Introduction

Graph reconstruction questions, increasingly relevant to bioinformatics, network science, and cybersecurity, ask whether a given set of information suffices to uniquely determine a graph. Often, as is the case in the venerable Graph Reconstruction Conjecture ([3], 1957), the given information is a set of subgraphs. For example, it is well known that any tree is uniquely determined by its "deck," the multiset of subgraphs obtained by deleting a single vertex ([3]).

Graham's Tree Reconstruction Conjecture stands out among these problems in that it asks if a tree is reconstructible from a specific integer sequence, rather than a collection of subgraphs ([2]). Letting $L(G)$ denote the line graph of G, we refer to the following as the *Graham sequence of a tree*:

$$|V(G)|, |V(L(G))|, |V(L(L(G))|, \dots .$$

Graham's Tree Reconstruction Conjecture asks whether a tree is uniquely determined by its Graham sequence. It is easy to see that for a tree on n vertices, the first two terms are n and $n-1$. In 2018, it was shown that the number of n-vertex trees which can be distinguished by their associated Graham sequence is $e^{\Omega((\log n)^{3/2})}$ (see [1]). In this note, we rely on purely computational techniques to verify Graham's conjecture for trees on up to 10 vertices. We expect that there is a unique truncated Graham sequence of length 6 for every tree on 11 vertices but have not confirmed this.

2 Computation and Data

2.1 Code and Computation

Using Python, the authors obtained a list of all unlabeled trees on up to 16 vertices. The following two Maple functions were central to obtaining the data discussed below:

```
LineGraph:=proc(G) local E, i, taggededges, tag_e, ledges,
        n, m, j, k, L:
n:=G[1]:
E:=G[2]:

taggededges:=[]:

for k from 1 to nops(E) do:
tag_e:=convert(E[k], list):
taggededges:=[op(taggededges),[op(tag_e), k]]:
od:

#now I have a list of tagged edges

#if two edges {x,y}, {x,z} share an endvertex in G, then find the
#edges [x,y,i] and [x,z, j] and add the edge {i,j} to the edges of #L(G)

ledges:=[]:
m:=nops(E):
for i from 1 to m do
    for j from i+1 to m do
 if taggededges[i][1]=taggededges[j][1] or
        taggededges[i][1]=taggededges[j][2] or
        taggededges[i][2]=taggededges[j][1] or
        taggededges[i][2]=taggededges[j][2]
    then ledges:=[op(ledges), {taggededges[i][3], taggededges[j][3]}]:
 fi:
    od:
od:
```

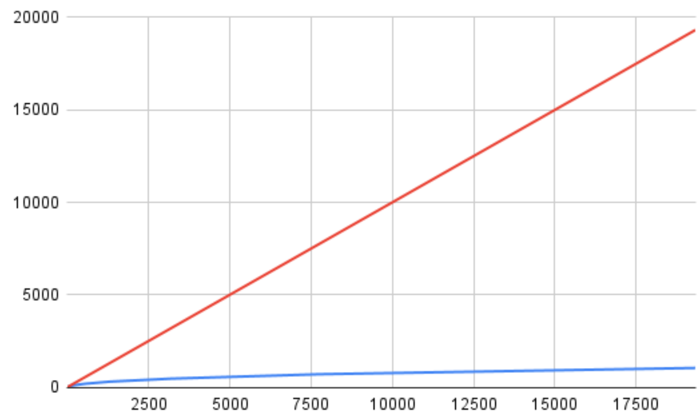
```
ledges:=convert(ledges, set):
L:=[nops(E), ledges]:

end:

GClist:=proc(T,k) local i, iterlist, newT, countlist, newG:
iterlist:=[T]:
newT:=T:

for i from 1 to k do
 newT:=LineGraph(newT):
 iterlist:=[op(iterlist), newT]:
od:

countlist:=[]:
for i from 1 to nops(iterlist) do
countlist:=[op(countlist), iterlist[i][1]]:
od:
countlist;
end:
```

2.2 Integer Sequences

From the number of distinct Graham sequences of length k on n-vertex graphs, we obtain several integer sequences. For example, these quantities for $k = 4$ and n from 1 to 16 are

$$[1, 1, 1, 2, 3, 6, 11, 20, 37, 68, 114, 188, 300, 462, 702, 1041].$$

This is now A383998 in the OEIS. We plot this and the number of unlabeled trees on n vertices below.



Another sequence of interest is the sequence D_k of discrepancies between the number of trees and the number of Graham sequences of length k. From the above,

$$D_3 = [0, 0, 0, 0, 0, 0, 0, 3, 10, 38, 121, 363, 1001, 2697, 7039, 18279, \ldots]$$

In the table below, we show the difference $D(i, j) = |\{\text{unlabled trees on } j \text{ vertices}\}| - |\{\text{distinct Graham Sequences of length } i\}|$.

	5	6	7	8	9	10	11	12	13	14	15	16
length 3	0	1	4	14	34	88	214	524	1267	3120	7695	19266
length 4	0	0	0	3	10	38	121	363	1001	2697	7039	18279
length 5	0	0	0	0	0	5	20	86	321	1148		

3 Appendix: SageMath Computation of Trees on n Vertices

The following code was created using SageMath ([4]).

```
import time
from sage.graphs.graph_generators import graphs

def UnlabeledUnrootedTrees(n):
    start = time.time()
```

```
# Generate trees using Nauty
trees = [G for G in graphs.nauty_geng(f"{n} {n-1}:c") if G.is_tree()]
canonical_graph_set = set()
for G in trees:
    # canonical form of the graph
    canonical_graph = G.canonical_label()
    # in python/sage the graph must be immutable
    #before it can be added to a set, manipulated
    immutable_graph = canonical_graph.copy(immutable=True)
    canonical_graph_set.add(immutable_graph)

end = time.time()
#print(f"Generated {len(canonical_graph_set)} unique
    #trees on {n} vertices in {end - start:.2f} seconds")
# Return the unique canonical graphs (graph objects)
return list(canonical_graph_set)  # Return as a list of graphs
```

References

[1] Joshua Cooper, Bill Kay, and Anton Swifton. "Graham's Tree Reconstruction Conjecture and a Waring-Type Problem on Partitions". In: *arXiv preprint arXiv:1109.0522* (2011).

[2] Chris Godsil and Gordon F Royle. *Algebraic graph theory*. Vol. 207. Springer Science & Business Media, 2013.

[3] Paul J Kelly. "A congruence theorem for trees." In: *Pacific Journal of Mathematics* 7 (1957).

[4] The Sage Developers. *SageMath, the Sage Mathematics Software System (Version x.y.z)*. https://www.sagemath.org. YYYY.