MATH 4920 – Undergraduate Research Projects

Manitoba eXperimental Mathematics Laboratory

Exploring perfect binary trees with relation to the HK-property

Atishaya Maharjan
Supervisor: Dr. Mahsa N. Shirazi

Winter 2024

1 Introduction

For a given graph $G = (V, E)$, $V(G)$ and $E(G)$ denotes the vertex sets and edge sets of the graph G. For an arbitrary vertex, $v \in V(G)$, all vertices adjacent to v by an edge are called the neighbours of v and is denoted by $N_G(v)$. The degree of a vertex $v \in V(G)$ is the cardinality of the set of neighbours of v, and is denoted by $deg_G(v)$.

For $n \in \mathbb{Z}^+$ such that $0 \leq n \leq |V(G)|$, a path of length n in G is a sequence of distinct vertices $v_1, v_2, \ldots, v_n$ such that v_i is adjacent to v_{i+1} for $1 \leq i \leq n-1$. A cycle is an extension of a path such that the last vertex is connected to the first vertex, i.e for a path of length n, $v_n v_i \in E(G)$. As such, the length of the cycle is $n+1$.

A connected graph is a graph if for all $u, v \in V(G)$, there exists a uv-path. An coclique is a set of vertices such that no two vertices in the set are adjacent to each other. It is denoted by I. We denote a family of indpendent sets of a graph G as I_G. A family of coclique of a graph G of cardinality n is denoted by $\mathcal{I}_G^n$. For $v \in V(G)$, the family of indpendent sets, $\mathcal{I}_G^n(v) := \{A \in \mathcal{I}_G^n : v \in A\}$ is called a star of $\mathcal{I}_G^n$ and v is called its center.

A tree is a connected graph with no cycles, it is denoted by T. For a vertex $v \in V(T)$, if $deg_T(v) = 1$, it is called a leaf. A vertex that is not a leaf is called an interior vertex. The depth of a vertex is defined as length of the path from the root vertex to it. We study a more particular class of trees called binary trees, denoted by T_B, where each interior vertex v has exactly 2 children and all leaves have the same depth. Further, a perfect binary tree is a binary tree in which every vertex $v \in V(T)$ has either 0 or 2 children. A perfect binary tree is denoted by T_{PB}, however in this report we will simply drop the subscript and denote it as T for clarity. A level n of a perfect binary tree is a set of vertices such that all vertices in the set have a depth of n.

The star centers of a graph are interesting because they relate to the EKR theorem.

2 Background and Project Goals

The Erdős-Ko-Rado (EKR) theorem limits the number of sets in a family of sets that can be pairwise intersecting. The theorem states that for a family of k-sets, the maximum number of sets that can be pairwise intersecting is $\binom{n-1}{k-1}$. This theorem has wide applications in combinatorics, graph theory, probability, and other areas of statistics and mathematics.

Studying the EKR theorem, [5] Hulbert and Kamat conjectured the following:

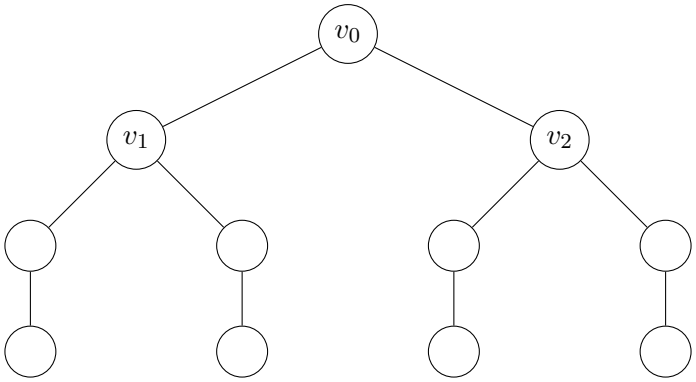


Figure 1: The largest k-star for $k \geq 5$ is centered at v_0

Conjecture 2.1 (k-EKR Conjecture)**.** *Let G be a graph, and let $\mu(G)$ be the size of its smallest maximal coclique. Then G is k-EKR for every $1 \leq k \leq \frac{\mu(G)}{2}$.*

However, even this conjecture is hard to understand and prove, so they narrowed the class of graphs to be trees and gave the following conjecture:

Conjecture 2.2 (HK-Property)**.** *For any $k \geq 1$ and any tree T, there exists a leaf l of T such that $|\mathcal{I}_T^k(v)| \leq |\mathcal{I}_T^k(l)|$ for each $v \in V(T)$.*

The HK-property holds true for $k \leq 4$, but was proven false independetly by [1, 2, 3]. The counterexample that that they arrived at is a type of graph that is defined as a class of trees called "lobsters" [4]. This is interesting for us as the counterexample graph resembles a binary tree.

They figured out that the lobsters, while not completely obeying the HK-property, almost obey the HK-property by either having the largest stars centered around the leaves or at the root of the tree.

Henceforth, we shall give a name to the property that the largest stars are centered around the leaves or the root of the tree as:

Definition 2.3 (Partial HK-Property)**.** *If a tree T has the property that for any $k \geq 1$, there exists a leaf l of T such that $|\mathcal{I}_T^k(v)| \leq |\mathcal{I}_T^k(l)|$ for each $v \in V(T) \setminus \{r\}$, where r is the root, then we say that T satisfies the Partial HK-Property.*

Since the binary trees are very similar in structure with the lobsters, the overall goal of this research was to figure out if the HK-property or the Partial HK-Property holds for perfect binary trees.

In the paper by Estrugo and Passtine [4], they gave a couple of classes of graphs that have the HK-property. The classes of graphs that they gave were the caterpillars and the stars.

The caterpillars are trees that have a path as the spine and the leaves are attached to the spine. The spiders are trees that only have 1 vertex with degree greater than 2.

They also proved that the Lobsters satisfy the Partial HK-Property.

3 Ideas

3.1 Idea: Flip functions and escape paths

Estrugo and Passtine [4] proved all the classes of graphs that satisfy the HK-property by using a **flip function** alongside the idea of an **escape path**.

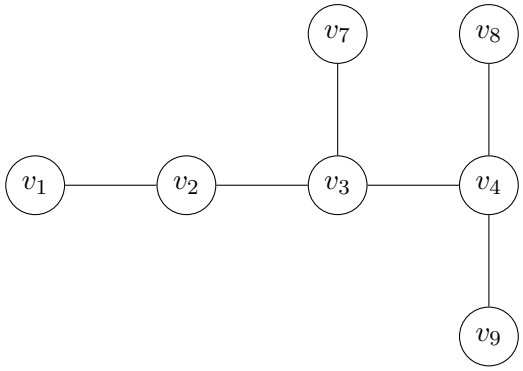


Figure 2: A caterpillar

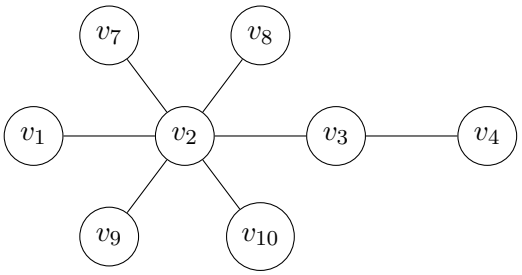


Figure 3: A Spider

Definition 3.1 (Escape path)**.** *Let G be a graph and $P = v_1, v_2, \ldots, v_n$ be a path of length n such that $P \subset G$. We say that P is an escape path from v_1 to v_n in G if $deg(v_n) = 1$ and $deg(v_i) = 2$ for all $i \in \{2, 3, \ldots, n-1\}$. If this is the case, then we say that P is an escape path from v_1 to v_n in G.*

Definition 3.2 (Flip function)**.** *Let G be a graph and $P = v_1, v_2, \ldots, v_n$ be an escape path from v_1 to v_n in G. Then the flip of P in G, $flip_P : V(G) \to V(G)$ is the function defined as follows:*

$$flip_P(v) = \begin{cases} v & \text{if } v \notin V(P) \\ v_{n+1-i} & \text{if } v = v_i \in V(P) \end{cases}$$

Using the two definitions, they showed that the flip_P induces a one to one mapping from $\mathcal{I}_G^k(v_1)$ to $\mathcal{I}_G^k(v_n)$. By showing that the flip function is an injection, they then used the Cantor-Schröder-Bernstein theorem to show that the $|\mathcal{I}_G^k(v_1)| \leq |\mathcal{I}_G^k(v_n)|$

By that, they showed that the classes of graphs that they gave satisfy the HK-property and that the lobster graphs satisfy the Partial HK-Property.

So, our idea was to use the flip function and the escape path to show that the perfect binary trees satisfy the HK-property or the Partial HK-Property. Our initial idea was to use a "diagonal" flip function that would flip the vertices along the diagonal of the perfect binary tree. To do this, we would have to address 4 important points:

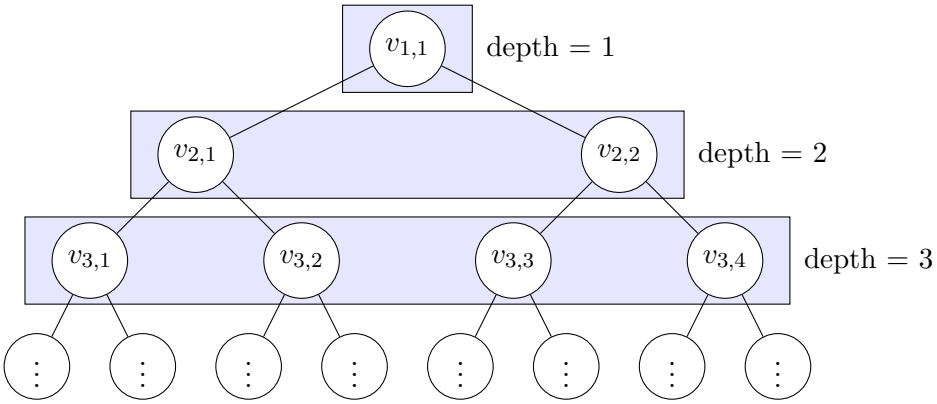
1. Enumerate the vertices of the perfect binary tree in a way that the new flip function would be able to map them around.

2. Define an escape path that would be able to escape from the root to the leaf of the perfect binary tree.

3. Define a flip function that would flip the vertices along the diagonal of the perfect binary tree.

Enumerate the vertices of the perfect binary tree in a way that the new flip function would be able to map them around.

Definition 3.3 (Depth Vertex)**.** *Let $\mathcal{V}_k \in V(T)$ be the set of vertices of depth k. We call $\mathcal{V}_k$ as the depth vertex set of depth k. Index all vertices in $\mathcal{V}_k$ from left to right as $v_{k,i}$, where k is the depth of the vertex and i is the index of the vertex in $\mathcal{V}_k$ such that $1 \leq i \leq 2^{k-1}$.*



Define a flip function that would flip the vertices along the diagonal of the perfect binary tree.

To save ourselves the trouble of defining a flip function for every path,, we decided to define an auxilary function that would flip the vertices along the vertex set so that our choice of vertices in the diagonal flip function would be arbitrary.

Definition 3.4 (Flip Function on Depth Vertex Set)**.** *Let T be a perfect binary tree and $\mathcal{V}_k$ be the depth vertex set of depth k. Then, the flip of $\mathcal{V}_k$ in T, denoted by $flip_{\mathcal{K}} : \mathcal{V}_k(G) \to \mathcal{V}_k(G)$, is the function defined as follows:*

$$flip_{\mathcal{K}}(v_{(k,i)}) = \begin{cases} v_{(k,2^{k-1}+1-i)} & \text{if } N_T(v_{(k,2^{k-1}+1-i)}) \not\subseteq \mathcal{I}_T(v_{(k,2^{k-1}+1-i)}) \\ v_{(k,2^{k-1}+1-i)}, & \text{if } N_T(v_{(k,2^{k-1}+1-i)}) \subseteq \mathcal{I}_T(v_{(k,2^{k-1}+1-i)}) \\ flip_N(v_{(k,i)}), & \text{if } N_T(v_{(k,2^{k-1}+1-i)}) \subseteq \mathcal{I}_T(v_{(k,2^{k-1}+1-i)}) \end{cases}$$

where, $flip_N$ is defined as:

$$flip_N(v_{(k,i)}) = flip_{\mathcal{K}}(u), \text{ for all } u \in N_T(v_{(k,i)})$$

We then have the following lemma:

Lemma 3.5 ($flip_{\mathcal{K}}$ is a bijective involution)**.** *Let T be a perfect binary tree and $\mathcal{V}_k$ be the depth vertex set of depth k. Then, the flip function $flip_{\mathcal{K}}$ is an involution.*

Proof. Let $\mathcal{V}_k$ be the depth vertex set of depth k. Since the function is defined recursively, we can evaulate the flipping step and then show that the flip function on the neighbours of a vertex also results in an involution.

We first evaluate the flipping step for when the neighbours of the vertex are not in the coclique. Then, applying the flip function twice, we have:

$$\begin{aligned} flip_{\mathcal{K}}(flip_{\mathcal{K}}(v_{k,i})) &= flip_{\mathcal{K}}(v_{(k,2^{k-1}+1-i)}) \\ &= v_{(k,2^{k-1}+1-(2^{k-1}+1-i))} \\ &= v_{(k,i)} \end{aligned}$$

Hence, $flip_{\mathcal{K}}$ is an involution when the neighbours of the vertex are not in the coclique.

We now have to show that the flip function on a vertex when the neighbours of the vertex are in the same coclique is also an involution.

Consider a vertex, $v_{(k,i)}$, such that $N_T(v_{(k,2^{k-1}+1-i)}) \subseteq \mathcal{I}_T(v_{(k,2^{k-1}+1-i)})$. Then, when we apply the $flip_{\mathcal{K}}$ function twice, we obtain that:

□

We then proceed to show that the flip function on a depth vertex set maps cocliques into cocliques.

Lemma 3.6. *Let T be a perfect binary tree and $\mathcal{V}_k$ be the depth vertex set of depth k. Let $\mathcal{I}^m_{\mathcal{V}_k}$ be the family of cocliques of cardinality m. Then, the flip function $flip_{\mathcal{K}}$ maps cocliques into cocliques and induces a bijection from $\mathcal{I}^m_{\mathcal{V}_k}$ onto itself.*

In addition, $flip_{\mathcal{K}}(\mathcal{I}_{\mathcal{V}_k}(v_{k,1})) = \mathcal{I}_{\mathcal{V}_k}(v_{k,2^{k-1}})$.

Proof. Let $\mathcal{I}$ be an coclique and assume two unique vertices $v_{k,i}, v_{k,j} \in flip_{\mathcal{K}}(\mathcal{I})$, i.e $i \neq j$. Since both $v_{k,i}$ and $v_{k,j}$ belong to the depth vertex set, and since this is a perfect binary tree, we see that $v_{k,i}$ and $v_{k,j}$ are not adjacent to each other. Hence, $flip_{\mathcal{K}}(\mathcal{I})$ is an coclique. Thus, $flip_{\mathcal{K}}$ maps cocliques into cocliques.

Now, let $\mathcal{I}^m_{\mathcal{V}_k}$ be the family of cocliques of cardinality m of the depth vertex set $\mathcal{V}_k$. Then, $flip_{\mathcal{K}}$ is a bijection from $\mathcal{I}^m_{\mathcal{V}_k}$ onto itself.

Finally, we have $flip_{\mathcal{K}}(\mathcal{I}_{\mathcal{V}_k}(v_{k,1})) = \mathcal{I}_{\mathcal{V}_k}(v_{k,2^{k-1}+1-1}) = \mathcal{I}_{\mathcal{V}_k}(v_{k,2^{k-1}})$, as required.

Hence, $flip_{\mathcal{K}}$ maps cocliques into cocliques and induces a bijection from $\mathcal{I}^m_{\mathcal{V}_k}$ onto itself. □

Now, our objective is to procure a similar result to [4] to show that our flip function induces a one to one mapping from any family of coclique $\mathcal{I}_T(v)$ for any vertex $v \in V(T)$ into any family of coclique $\mathcal{I}_T(l)$ for a leaf $l \in V(T)$. We also need to define a diagonal flip function that will flip the vertices along the diagonal of the perfect binary tree. To do that, we will need to define an escape path that would be able to escape from the root to the leaf of the perfect binary tree.

Define an escape path that would be able to escape from the root to the leaf of the perfect binary tree.

However, in our objective to properly define a diagonal flip function, we encountered a problem. The perfect binary trees does not have an escape path. The escape path is defined such that the middle vertices are all part of a pure path. This is not achievable in the perfect binary tree as all the interior vertices have 2 children, i.e. degree 3. Hence, we were unable to define an escape path that would be able to escape from the root to the leaf of the perfect binary tree.

3.2 Algorithmic Approach

To validate our conjecture, we present a simple algorithm to generate all cocliques of a perfect binary tree of cardinality k. We then compare the number of cocliques containing a vertex v and a leaf l to see if the HK-property holds for perfect binary trees.

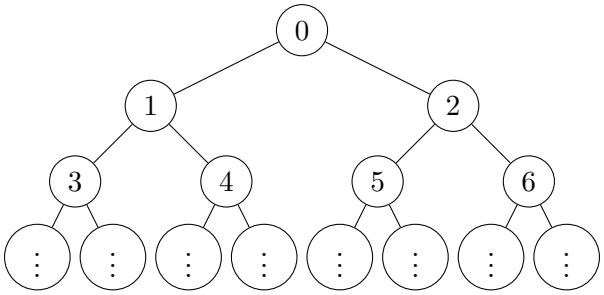
To begin with, we present the following algorithm to generate a perfect binary tree of depth n:

Algorithm 1: Perfect Binary Tree Generator

Data: $n \geq 0$, where n is the depth of the perfect binary tree
Result: A perfect binary tree graph and its leaves
Function `perfect_binary_tree_generator(`n`):`

```
if n = 0 then
    return Graph()
else
    g ← Graph();
    g.add_vertices([2^n]);
    for i in range(2^n − 1) do
        g.add_edge(i, 2 ∗ i + 1);
        g.add_edge(i, 2 ∗ i + 2);
    end
    return g
end
```

The algorithm will generate a perfect binary tree of this form:



It is easy to see that the leaves will start with the value of $\left\lfloor \frac{2^{n+1}-1}{2} \right\rfloor$, where n is the depth of the perfect binary tree.

So to generate all the leaves of the perfect binary tree of depth n, we present the following algorithm:

Algorithm 2: Perfect Binary Tree Leaves Generator

Data: $n \geq 0$, where n is the depth of the perfect binary tree
Result: A perfect binary tree graph's leaves
Function `perfect_binary_tree_generator(`n`):`
    $num\_vertices \leftarrow 2^{n+1} - 1$;
    $leaves \leftarrow []$;
    $last\_row\_start \leftarrow floor(num\_vertices/2)$;
    for *vertex in range(last_row_start, num_vertices)* **do**
        $leaves$.append($vertex$);
    end
    return *leaves*

We then use the algorithm from [6]Cliquer to generate a coclique of maximum cardinality for our perfect binary tree.

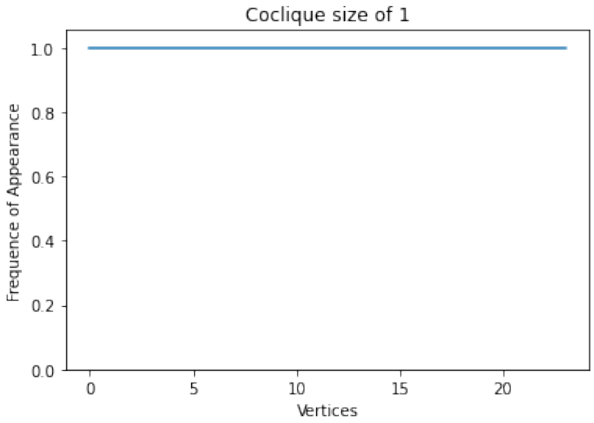
Algorithm 3: Maximum Indpendent Set Algorithm

Data: A perfect binary tree graph T
Result: A maximum coclique of T
Function `maximum_coclique(`T`):`
    $cliquer \leftarrow Cliquer(T)$;
    return *cliquer.get_maximum_coclique()*

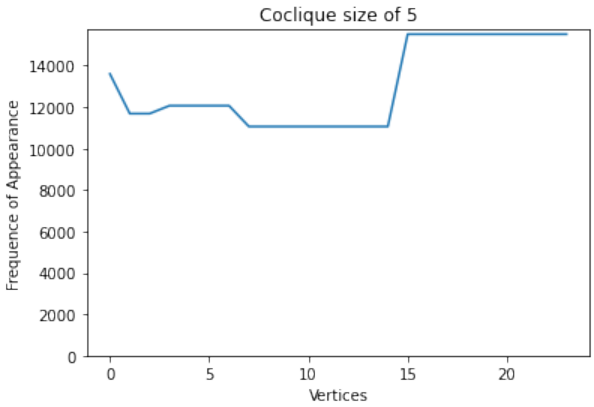
The next page shows the results of the algorithm for a perfect binary tree of depth 5. The X-axis denotes the vertice's labels (not the actual numbers) and the Y-axis denotes the cardinality of the stars centered around the vertices.

The data shown in the figures above verifies that the Partial HK-Property holds for perfect binary trees of depth 5. The next step would be to verify this for perfect binary trees of depth 6 and 7.

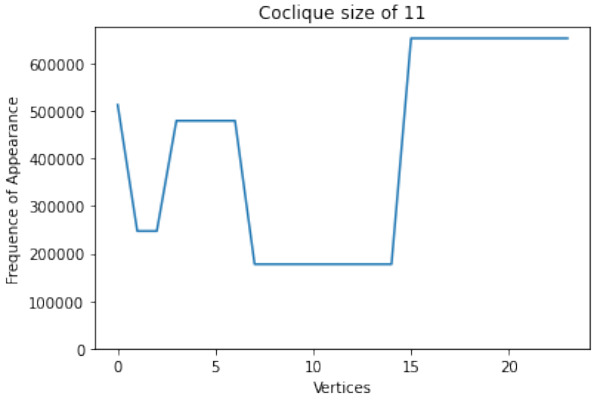
However, the algorithm is very slow and inefficient and it scaled exponentially. Hence, running the algorithm for perfect binary trees of depth 6 and 7 would be very computationally expensive.



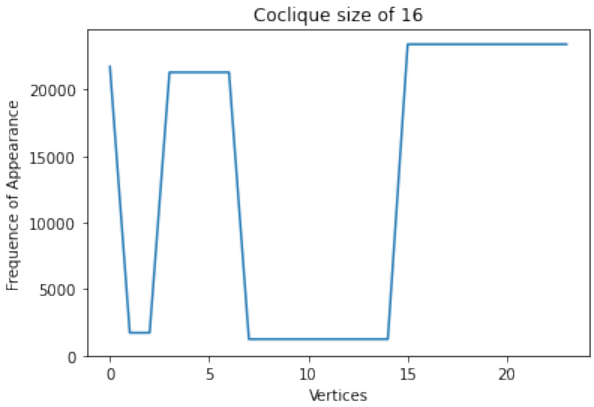
(a) coclique of size 1



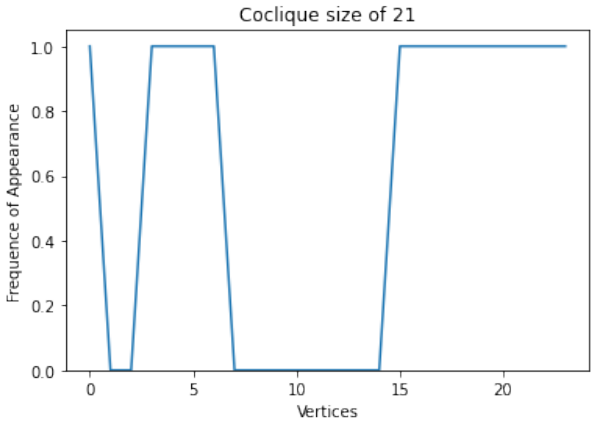
(b) coclique of size 5



(c) coclique of size 11



(d) coclique of size 16



(e) coclique of size 21

3.3 Inductive Conjectures

From our observations from the algorithm, we note the following:

1. The maximum coclique always seems to contain the last depth vertex set (All the leaves).
2. The maximum coclique always seems to contain the root.
3. The maximum coclique seems to start from the last depth vertex set and then move towards the root alternately.

In essence, we observe that that maximum coclique number seems to follow this pattern:

If T is our perfect binary tree of depth d, then

$$\alpha(T) = 2^d + 2^{d-2} + \dots$$

When d is even, we see that

$$\alpha(T) = 2^d + 2^{d-2} + \dots + 2^0$$

When d is odd, we see that

$$\alpha(T) = 2^d + 2^{d-2} + \dots + 2^1$$

From our observations, we conjecture the following:

Conjecture 3.7. *For any perfect binary tree T of depth d, the maximum coclique $\alpha(T)$ is given by*

$$\alpha(T) = \sum_{i=0}^{\lfloor \frac{d}{2} \rfloor} 2^{d-2i}$$

If this conjecture holds true, then we claim that:

Claim 3.1. *There is a unique maximum coclique set that contains all the leaves.*

Then, we can further conjecture that if we induct on d and k where $k \leq \alpha(T)$, we have:

Conjecture 3.8. *Let T be a perfect binary tree of depth d. Let r be the root of T. Then, for all possible values of d and k, there exists a leaf l of T such that $|\mathcal{I}_T^k(v)| \leq |\mathcal{I}_T^k(l)|$ for each $v \in V \setminus r$.*

Note that 3.8 is basically the statement of the Partial HK-Property specifically for perfect binary trees.

4 Conclusions

To conclude, we believe that the Partial HK-Property holds for perfect binary trees. We have shown that the maximum coclique of a perfect binary tree follows a certain pattern and we have shown that the maximum coclique always seems to contain the leaves and the root.

We do have an idea of the proof and a proof sketch, however proving it will take some time and we plan to work on it in the future.

In addition, noting that a perfect binary tree is a special case of a perfect k-ary tree, we can conjecture that the Partial HK-Property holds for perfect k-ary trees as well, if our inductive conjecture holds true and is generalizable.

We also aim to explore other classes of k-ary trees and see if the Partial HK-Property holds for them as well in the future.

5 References

[1] R. Baber. *Some results in extremal combinatorics.* ProQuest LLC, Ann Arbor, MI, 2011. Thesis (Ph.D.)–University of London, University College London (United Kingdom).

[2] Peter Borg. Stars on trees. *Discrete Math.*, 340(5):1046–1049, 2017.

[3] Peter Borg and Fred Holroyd. The Erdős-Ko-Rado properties of various graphs containing singletons. *Discrete Math.*, 309(9):2877–2885, 2009.

[4] Emiliano J.J. Estrugo and Adrián Pastine. On stars in caterpillars and lobsters. *Discrete Appl. Math.*, 298:50–55, 2021.

[5] Glenn Hurlbert and Vikram Kamat. Erdős-Ko-Rado theorems for chordal graphs and trees. *J. Combin. Theory Ser. A*, 118(3):829–841, 2011.

[6] Sampo Niskanen and Patric R. J. Östergård. Cliquer user's guide, version 1.0. 2003.