# RECOGNIZING K-LEAF POWERS IN POLYNOMIAL TIME, FOR CONSTANT K

Manuel Lafond, Université de Sherbrooke, Canada

UNIVERSITÉ DE SHERBROOKE

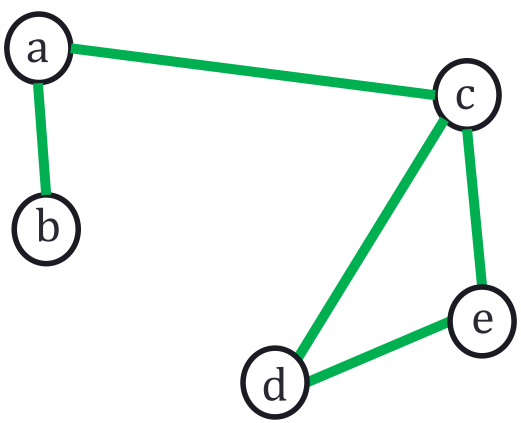
**Definition**

A graph $G$ is a ***k*-leaf power** if there exists a (rooted) tree $T$ such that:

- $\boldsymbol{L(T) = V(G)}$, where $L(T)$ is the set of leaves of $T$
- $\boldsymbol{uv \in E(G) \Leftrightarrow dist_T(u,v) \leq k}$

$3 - leaf\ power$ ?

$G$



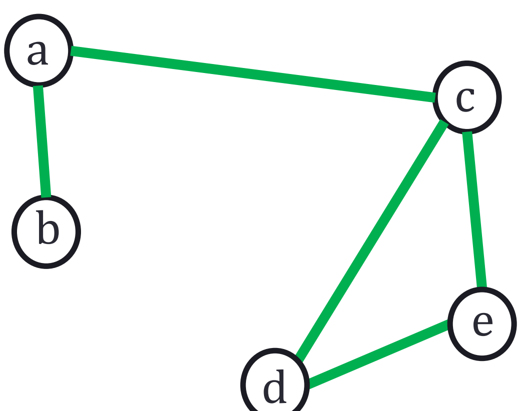
**Definition**

A graph $G$ is a ***k*-leaf power** if there exists a (rooted) tree $T$ such that:

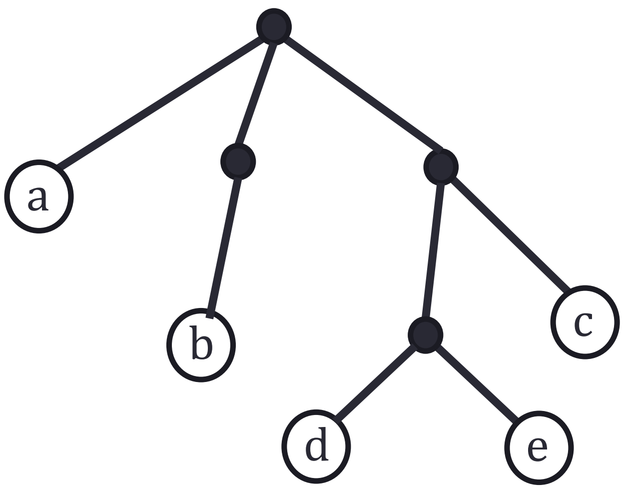
- $L(T) = V(G)$, where $L(T)$ is the set of leaves of $T$
- $uv \in E(G) \Leftrightarrow dist_T(u, v) \leq k$

$3 - leaf\ power$

$G$




$T$ :



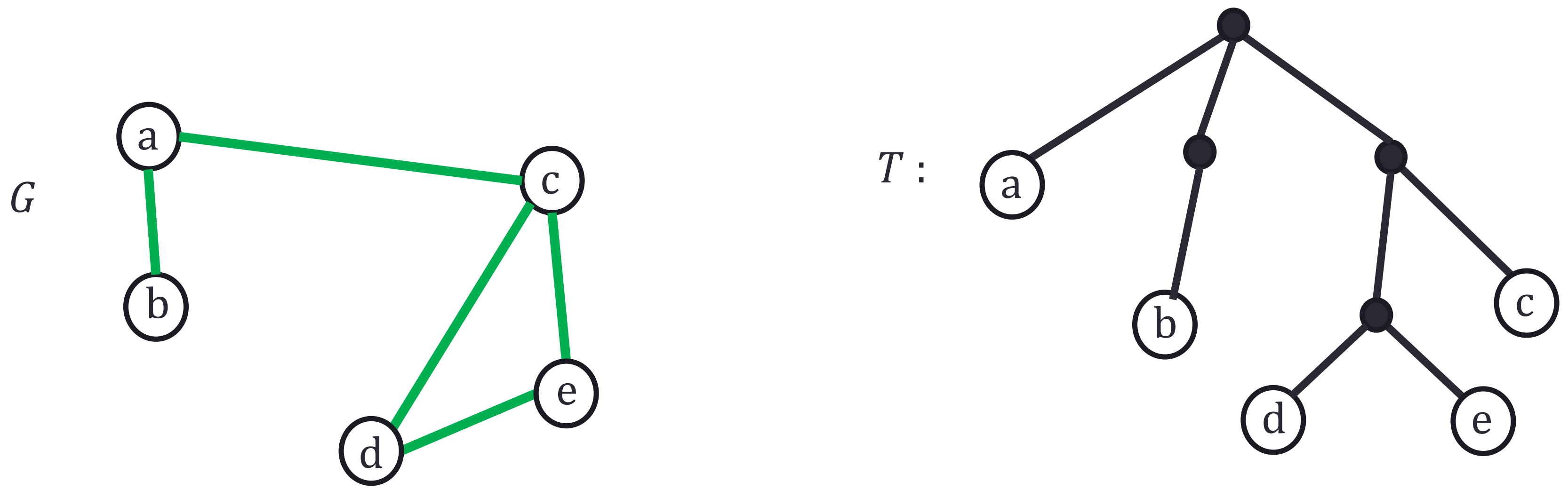
**Definition**

A graph $G$ is a ***k*-leaf power** if there exists a (rooted) tree $T$ such that:

- $L(T) = V(G)$, where $L(T)$ is the set of leaves of $T$
- $uv \in E(G) \Leftrightarrow dist_T(u, v) \leq k$

Equivalently, $G$ is a $k$-leaf power if it can be obtained by taking the $k$-th power of a tree, and taking the subgraph induced by the leaves of the tree.

$3 - leaf\ power$



**Definition**

A graph $G$ is a ***k*-leaf power** if there exists a (rooted) tree $T$ such that:

- $\boldsymbol{L(T) = V(G)}$, where $L(T)$ is the set of leaves of $T$
- $\boldsymbol{uv \in E(G) \Leftrightarrow dist_T(u, v) \leq k}$

**Open problems [Nishimura, Ragde, Thilikos, 2002]**

- Can k-leaf powers be characterized by chordal + finite set of forbidden induced subgraphs?
- Complexity of recognizing $k$-leaf powers if $k$ is in the input?
- Complexity of recognizing $k$-leaf powers if $k$ is fixed?

**Definition**

A graph $G$ is a **$k$-leaf power** if there exists a (rooted) tree $T$ such that:

- $L(T) = V(G)$, where $L(T)$ is the set of leaves of $T$
- $uv \in E(G) \Leftrightarrow dist_T(u, v) \leq k$

**Open problems [Nishimura, Ragde, Thilikos, 2002]**

- Can k-leaf powers be characterized by chordal + finite set of forbidden induced subgraphs?
  - YES for $k = 2, 3, 4$. OPEN for $k \geq 5$.
- Complexity of recognizing $k$-leaf powers if $k$ is in the input?
  - OPEN. Not known to be NP-hard or in P.
- Complexity of recognizing $k$-leaf powers if $k$ is fixed?
  - OPEN for 20 years. In P (this talk).

**Theorem**
There is an algorithm that, given a graph $G$, decides whether $G$ is a $k$-leaf power in time $O(n^{f(k)})$, where $n = |V(G)|$ and $f$ is a function that depends only on $k$.

**Theorem**
There is an algorithm that, given a graph $G$, decides whether $G$ is a $k$-leaf power in time $O(n^{f(k)})$, where $n = |V(G)|$ and $f$ is a function that depends only on $k$.

$$f(k) \simeq 2^{k3^{k3^{k3^{\cdot^{\cdot^{\cdot^{k3^{k}}}}}}}} \quad \Bigg\} \; k \text{ times}$$

# Known results

- **2-leaf powers** = P3-free graphs [folklore]
- **3-leaf powers** = chordal + (bull, gem, dart)-free graphs [Rautenbach, Disc Maths 2006]
- **4-leaf powers** = chordal + X-free, where X is a finite set of forbidden subgraphs [Brandstädt et al., TALG 2008]
- **5-leaf powers** recognition in P [Chang & Ko, WG 2007]
- **6-leaf powers** recognition in P [Ducoffe, WG 2019]
- Recognizing *k*-leaf powers is FPT in k + degeneracy(G), and **FPT in k + treewidth(G)**. [Eppstein & Havvaei, IPEC 2018]

# Known results

- Leaf power = graphs that are $k$-leaf powers for some $k$.
- All leaf powers are **chordal**, and also **strongly chordal**
- Converse **not true** [L, WG2017; Jaffke & al., TCS2019]
- **Subclasses** of strongly chordal (interval, rooted directed, ptolemaic) graphs are **easy to recognize**

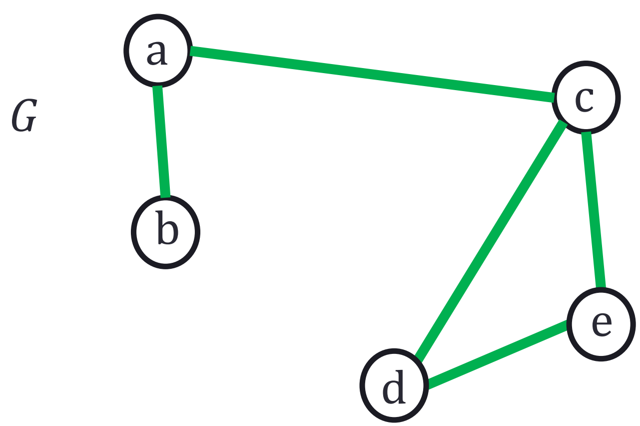
  [Brandstädt et al., LATIN2008 & DiscMath2010]
- Leaf powers have **mim-width 1** [Jaffke & al., TCS2019]
- Leaf powers with **star NeS models** in P [Bergougnoux, 2021]

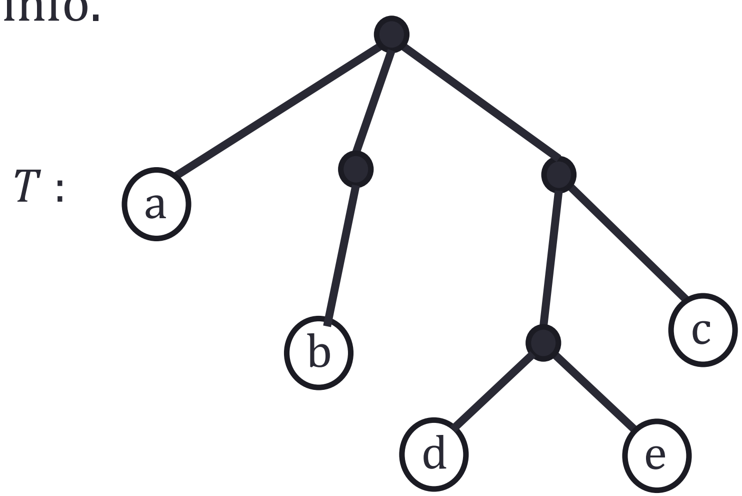
# Other tree-definable classes

- Many other tree-to-graph representations, all with similar open problems
  - Pairwise compatiblity graphs (PCG)
    - $uv$ edge iff distance in interval $[l, h]$
  - k-interval PCGs, OR-PCGs and AND-PCGs
    - Allow $k$-intervals, union/intersection of PCGs
  - Orthology graphs
    - $uv$ edge iff lca has label 1
  - Fitch graphs
    - $uv$ edge iff some edge on $u - v$ path has label 1
  - Best match graphs
  - …

# Applications

- **In computational biology:**
  - V(G) are species. Sequence data tells us that
    - **edge = 'close'** species in evolution
    - **non-edge = 'far'** species in evolution, and
    - **k = threshold** between close and far.
    - Goal = reconstruct a tree from that info.

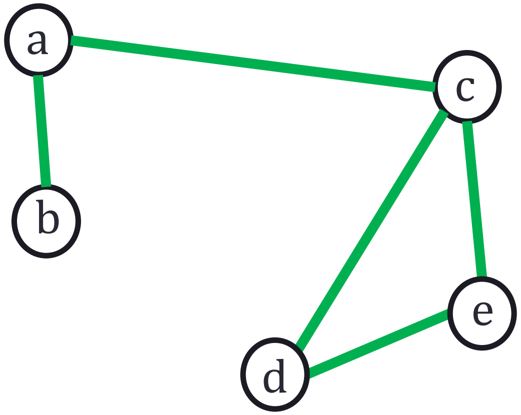


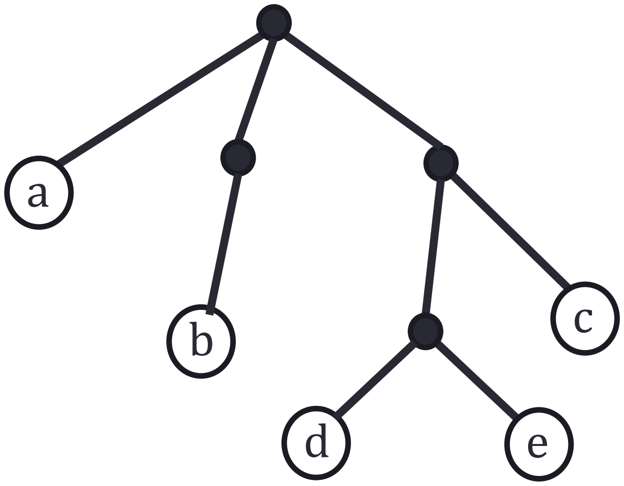
# Applications

- **In algorithms:**
  - Many problems are in P, or FPT in k for k-leaf powers. (dynamic programming on the tree)
  - Not that interesting, but also true for other tree-to-graph representations (PCGs, etc.).

$G$



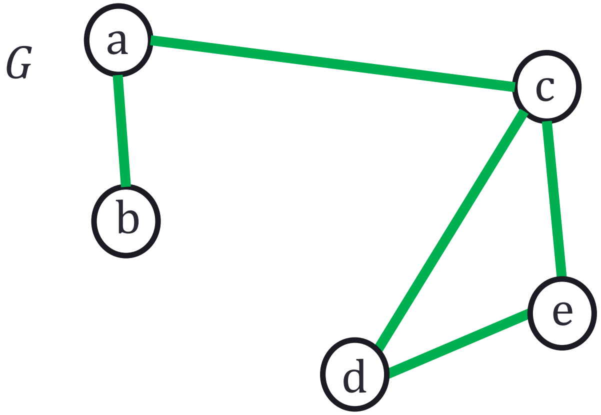

$T$ :



**Theorem**
There is an algorithm that, given a graph $G$, decides whether $G$ is a $k$-leaf power in time $O(n^{f(k)})$, where $n = |V(G)|$ and $f$ is a function that depends only on $k$.

# High-level overview

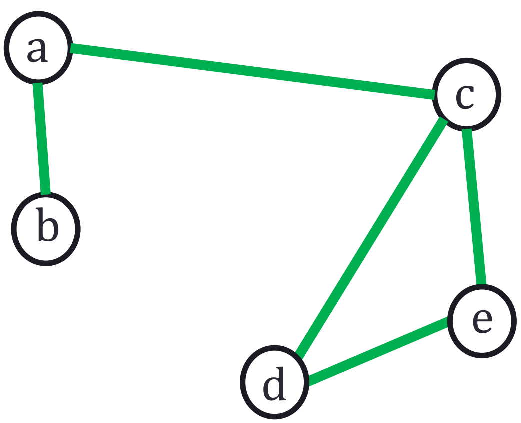
- Given a graph $G$, we must decide whether $G$ is a $k$-leaf power (assume that $k$ is fixed).



# High-level overview

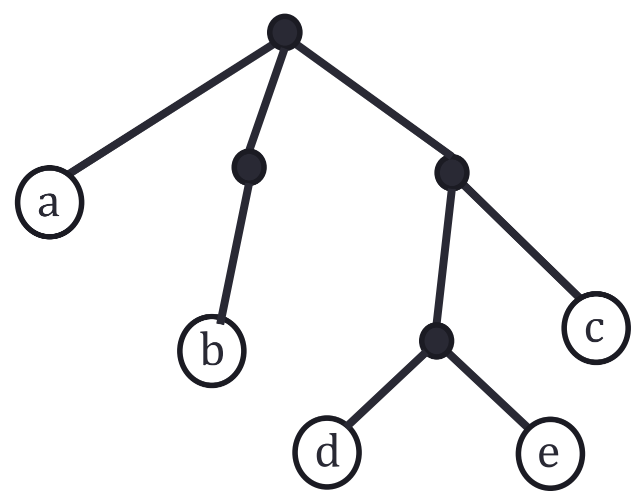
For $G$ a $k$-leaf power, a **$k$-leaf root of $G$** is a tree with $\boldsymbol{L(T) = V(G)}$ satisfying $\boldsymbol{uv} \in \boldsymbol{E(G)} \Leftrightarrow \boldsymbol{dist_T(u, v)} \leq \boldsymbol{k}$.

$G$




$3 - leaf\ root$

$T:$

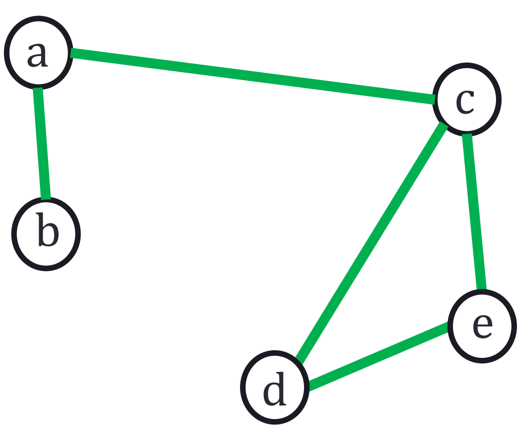


# High-level overview

For $G$ a $k$-leaf power, a ***k*-leaf root of *G*** is a tree with $\boldsymbol{L(T) = V(G)}$ satisfying $\boldsymbol{uv \in E(G) \Leftrightarrow dist_T(u,v) \leq k}$.

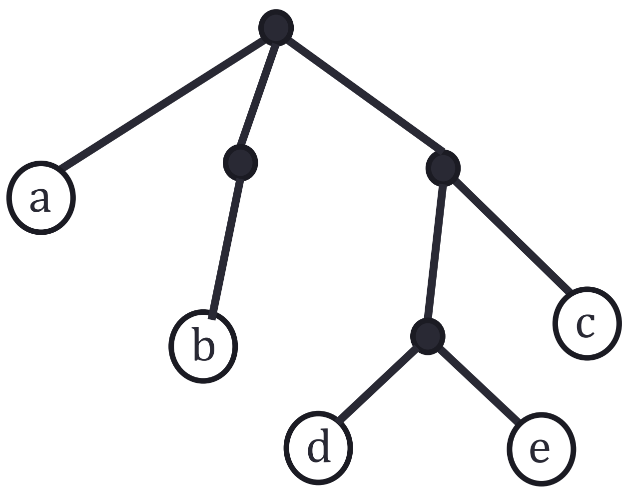
**Theorem (from Eppstein & Havvaei, 2019)**
There is a function $g$ such that one can decide in time $O(g(tw(G),k)n)$ whether $G$ is a $k$-leaf power, where $tw(G)$ is the treewidth of $G$.

$G$




$3 - leaf\ root$

$T:$

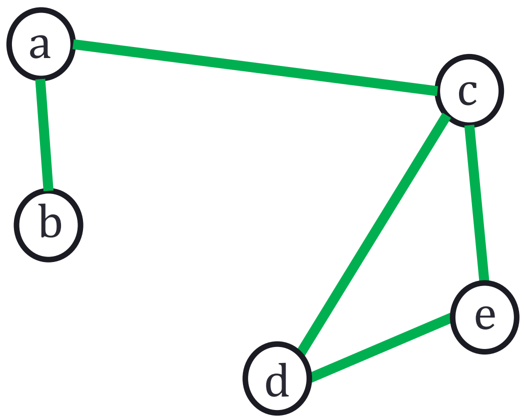


# High-level overview

For $G$ a $k$-leaf power, a ***k*-leaf root of *G*** is a tree with $\boldsymbol{L(T) = V(G)}$ satisfying $\boldsymbol{uv} \in \boldsymbol{E(G)} \Leftrightarrow \boldsymbol{dist_T(u,v)} \leq \boldsymbol{k}$.

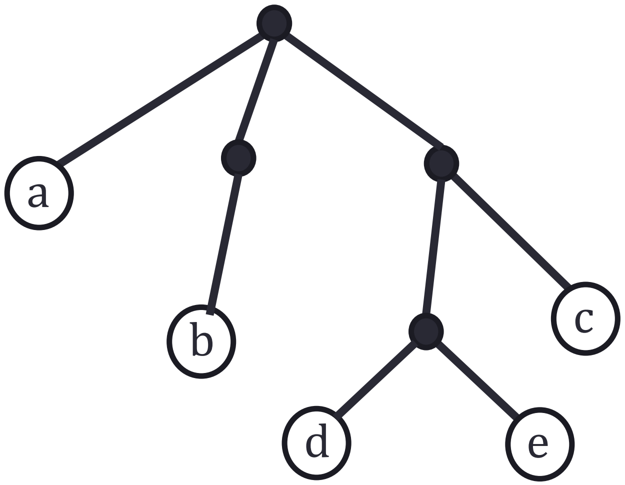
**Theorem**
Let $d, k$ be integers. Then one can decide in time $O(g(d^k, k)n)$ whether a graph $G$ admits a $k$-leaf root **of maximum degree *d***.

$G$



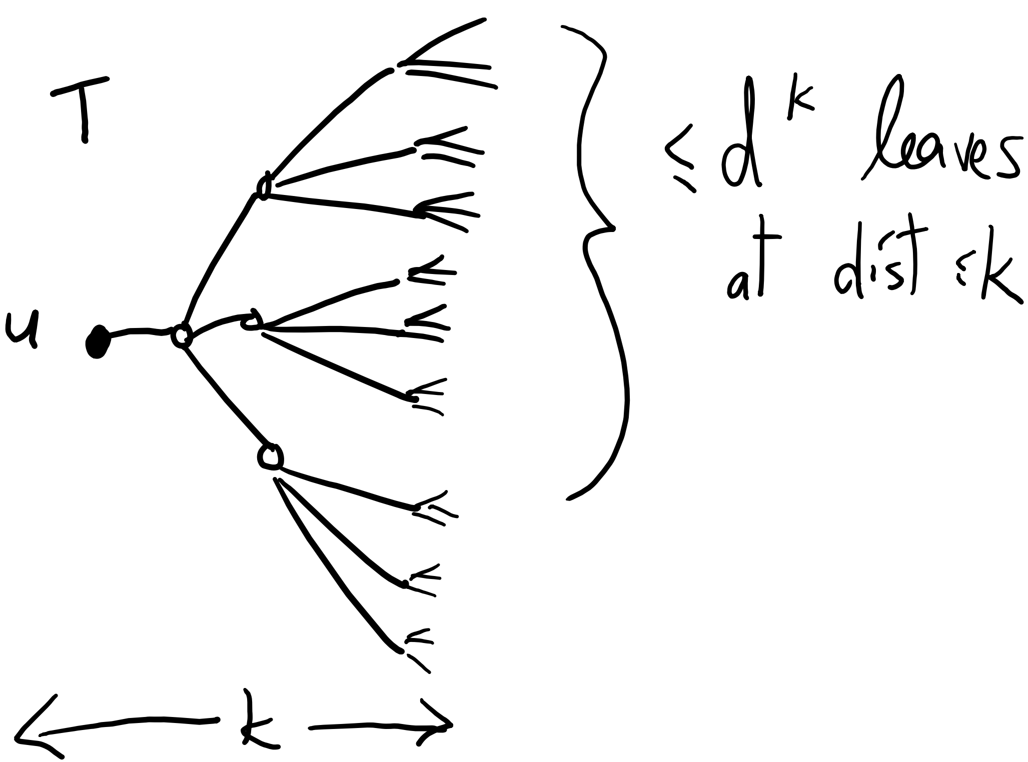

$3 - leaf\ root$

$T:$



**Theorem**
Let $d, k$ be integers. Then one can decide in time $O(g(d^k, k)n)$ whether a graph $G$ admits a $k$-leaf root **of maximum degree $d$**.

- Proof idea.
- If G admits a $k$-leaf root of max degree $d$, then $G$ has maximum degree $d^k$.




**Theorem**
Let $d, k$ be integers. Then one can decide in time $O(g(d^k, k)n)$ whether a graph $G$ admits a $k$-leaf root **of maximum degree $d$**.

- Proof idea.
- If G admits a $k$-leaf root of max degree $d$, then $G$ has maximum degree $d^k$.
- All $k$-leaf powers are chordal.
- In chordal graphs, we have $tw(G) = w(G) - 1 \leq dk$.
  - tw(G) = treewidth, w(G) = clique number

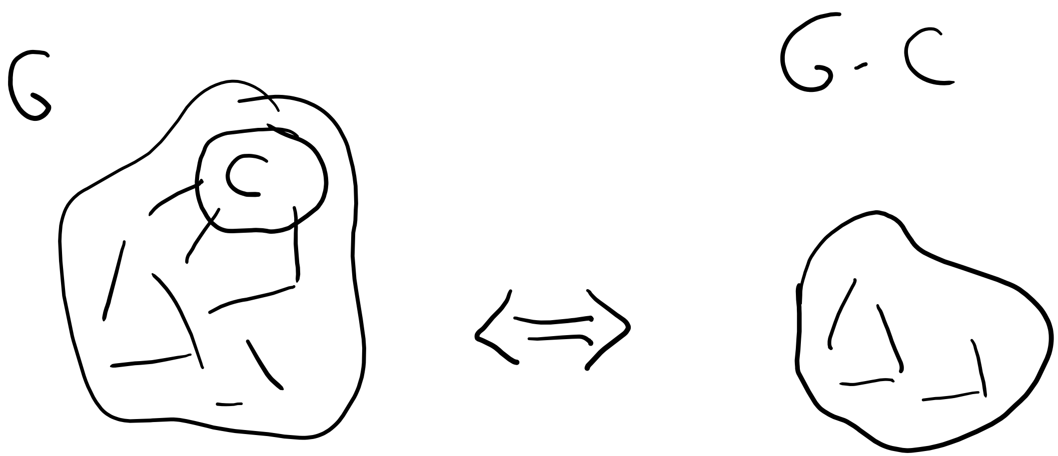
- Use Eppstein & Havvaei to decide in time $O(g(tw(G), k)n) = O(g(d^k, k)n)$ whether $G$ is a $k$-leaf power.

**Theorem**
Let $d, k$ be integers. Then one can decide in time $O(g(d^k, k)n)$ whether a graph $G$ admits a $k$-leaf root **of maximum degree $d$**.

- If $d$ is a function of $k$, problem solved.
- **Bottom-line** : the difficulty resides in $k$-leaf roots of high maximum degree.

# $k$-leaf roots with high degree

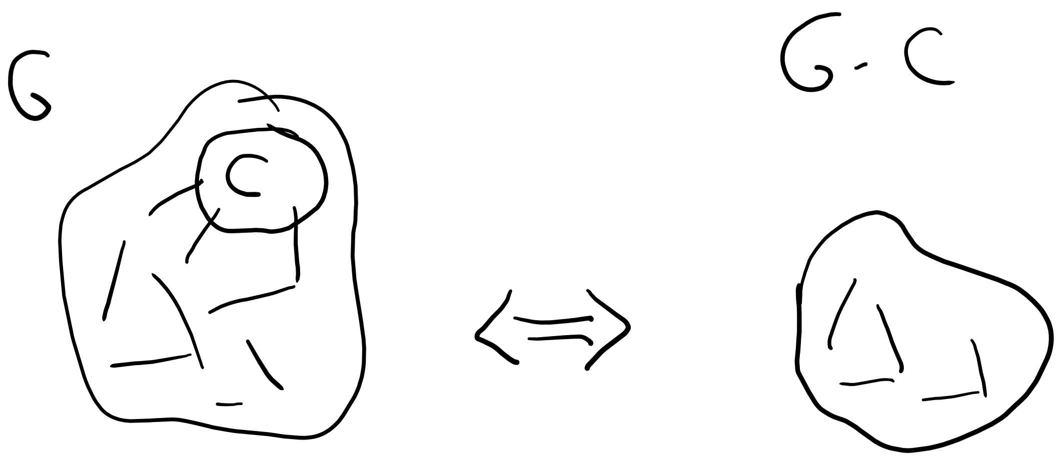
**Theorem**
There is $f$ such that if $G$ admits a $k$-leaf root of max degree $d > f(k)$, then $G$ contains a subset $C$ of vertices such that **$G$ is a $k$-leaf power if and only if $G - C$ is a $k$-leaf power**.
Moreover, $C$ can be found in time $O(n^{f(k)})$ if it exists.



# $k$-leaf roots with high degree

**Theorem**
There is $f$ such that if $G$ admits a $k$-leaf root of max degree $d > f(k)$, then $G$ contains a subset $C$ of vertices such that **$G$ is a $k$-leaf power if and only if $G - C$ is a $k$-leaf power**.
Moreover, $C$ can be found in time $O(n^{f(k)})$ if it exists.

This says that if $G$ has high-degree $k$-leaf roots, then $G$ has a redundant subset of vertices $C$ that can be found and pruned 'quickly'.



# $k$-leaf roots with high degree

**Theorem**
There is $f$ such that if $G$ admits a $k$-leaf root of max degree $d > f(k)$, then $G$ contains a subset $C$ of vertices such that **$G$ is a $k$-leaf power if and only if $G - C$ is a $k$-leaf power**.
Moreover, $C$ can be found in time $O(n^{f(k)})$ if it exists.

The algorithm:

1) Check if $G$ admits a $k$-leaf root of degree at most $d = f(k)$ using Eppstein & Havvaei. If yes, return "yes".
2) Otherwise, check if $G$ contains $C$ as described above. If not, return "no".
3) Otherwise, repeat on $G - C$.

Finishes in polynomial time, since $k$ is fixed and this is repeated at most $n$ times.

# $k$-leaf roots with high degree

**Theorem**
There is $f$ such that if $G$ admits a $k$-leaf root of max degree $d > f(k)$, then $G$ contains a subset $C$ of vertices such that ***G* is a *k*-leaf power if and only if *G* – *C* is a *k*-leaf power**.
Moreover, $C$ can be found in time $O(n^{f(k)})$ if it exists.

**Step 1 :** find lots of subsets $C_i \cup Y_i$ such that the $C_i$'s are cutsets, and all have the same neighborhood structure.

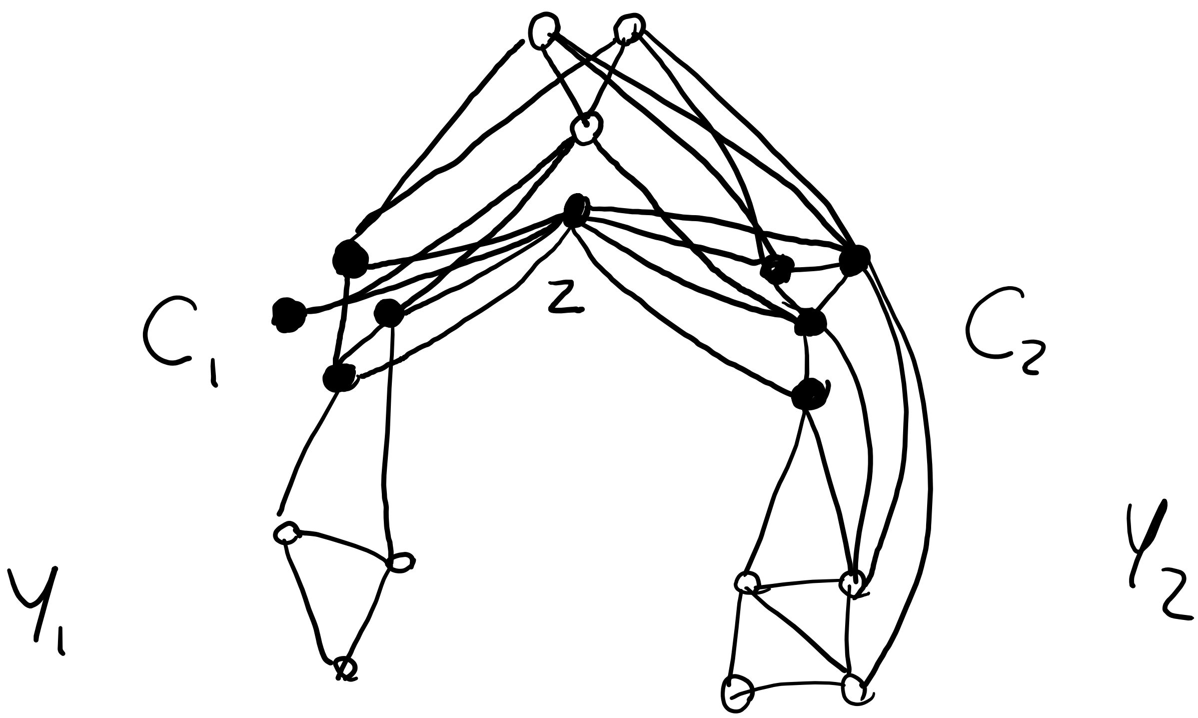
**Step 2** : argue that enough of those $C_i \cup Y_i$ admit the "same" $k$-leaf roots.

**Step 3** : argue that any such $C_i \cup Y_i$ can be removed since we can find a $k$-leaf root of $G - C_i \cup Y_i$ and embed $C_i \cup Y_i$ into it afterwards.

# Step 1 : subsets of vertices with the same neighborhood structure

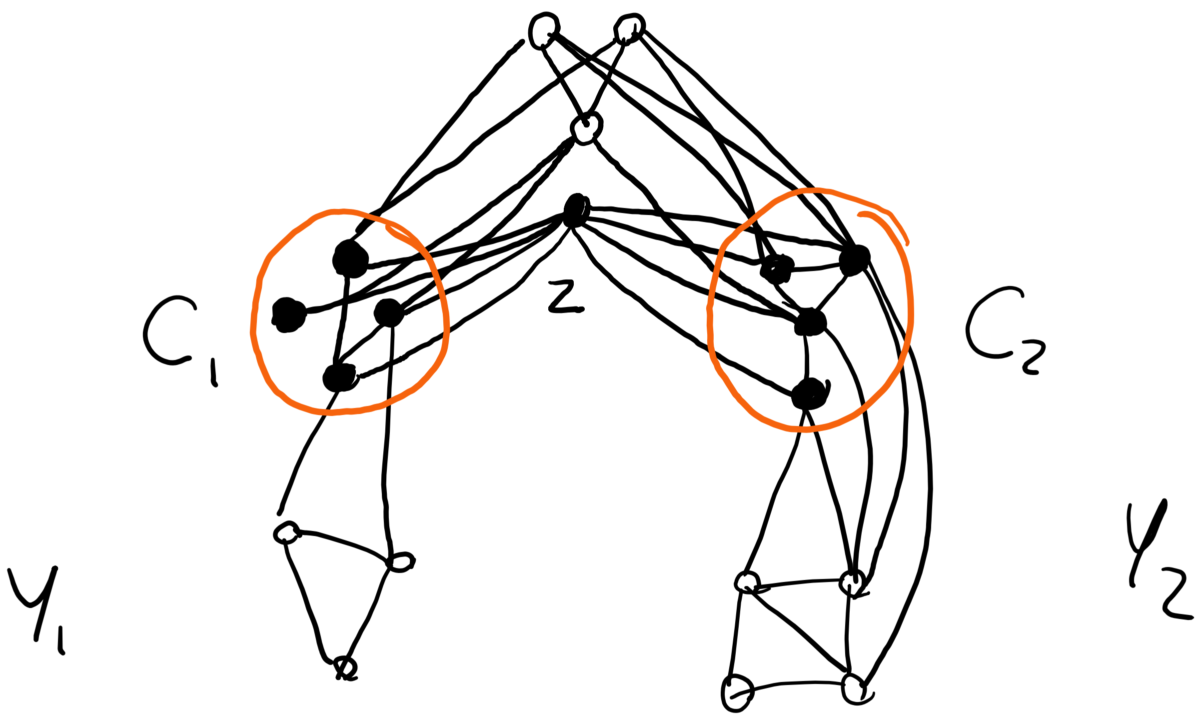
# Similar sets of vertices

- We say that $C_1 \cup Y_1$ and $C_2 \cup Y_2 \subseteq V(G)$ are **similar** if
  - There is $z$ such that $C_1 \cup C_2 \subseteq N(z)$.
  - $C_1$ cuts $Y_1$ and $C_2$ cuts $Y_2$ from the rest of the graph
  - $C_1 \cup C_2$ can be partitioned into layers $L_1, \dots, Lk$ such that **vertices in the same layer have the same neighbors** in $G - (C_1 \cup Y_1 \cup C_2 \cup Y_2)$.



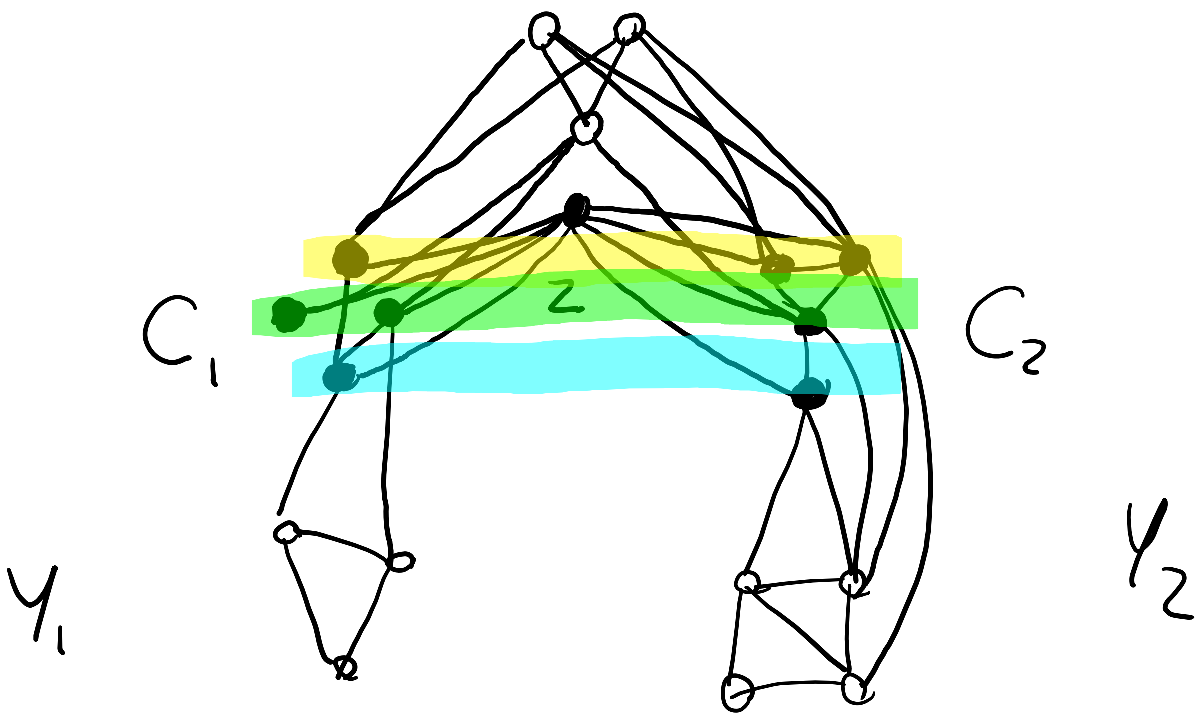
# Similar sets of vertices

- We say that $C_1 \cup Y_1$ and $C_2 \cup Y_2 \subseteq V(G)$ are **similar** if
  - There is $z$ such that $C_1 \cup C_2 \subseteq N(z)$.
  - $C_1$ cuts $Y_1$ and $C_2$ cuts $Y_2$ from the rest of the graph
  - $C_1 \cup C_2$ can be partitioned into layers $L_1, \ldots, Lk$ such that **vertices in the same layer have the same neighbors** in $G - (C_1 \cup Y_1 \cup C_2 \cup Y_2)$.



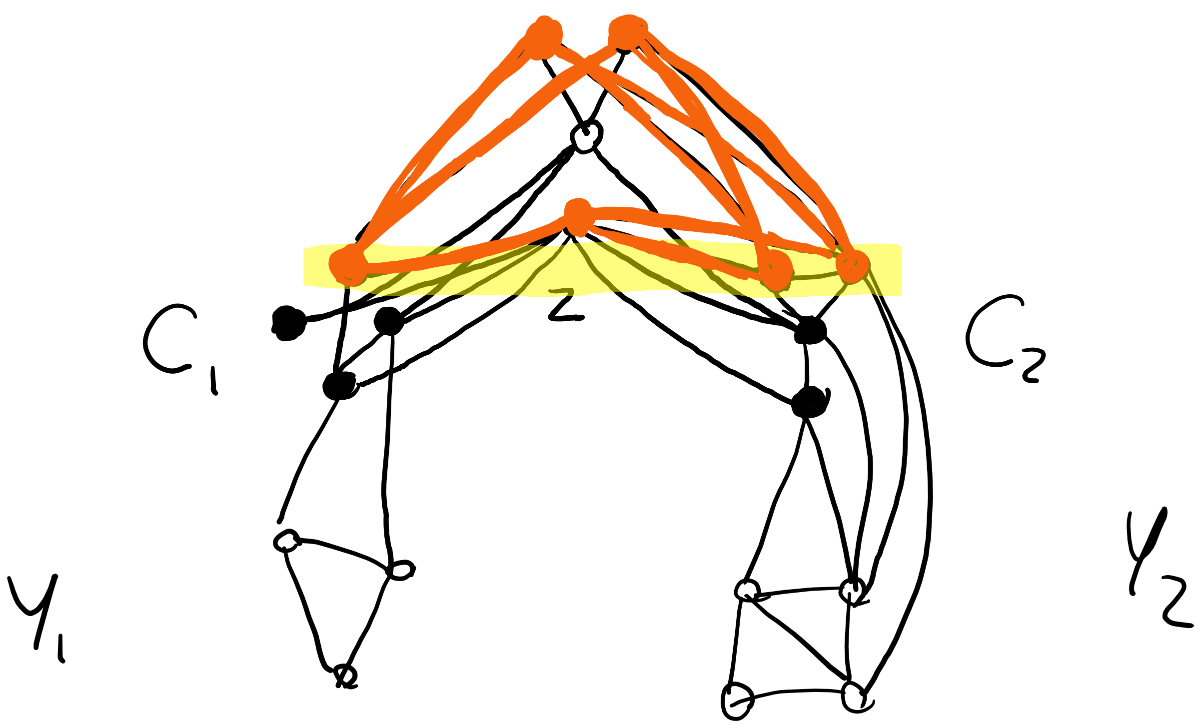
# Similar sets of vertices

- We say that $C_1 \cup Y_1$ and $C_2 \cup Y_2 \subseteq V(G)$ are **similar** if
  - There is $z$ such that $C_1 \cup C_2 \subseteq N(z)$.
  - $C_1$ cuts $Y_1$ and $C_2$ cuts $Y_2$ from the rest of the graph
  - $C_1 \cup C_2$ can be partitioned into layers $L_1, \dots, Lk$ such that **vertices in the same layer have the same neighbors** in $G - (C_1 \cup Y_1 \cup C_2 \cup Y_2)$.



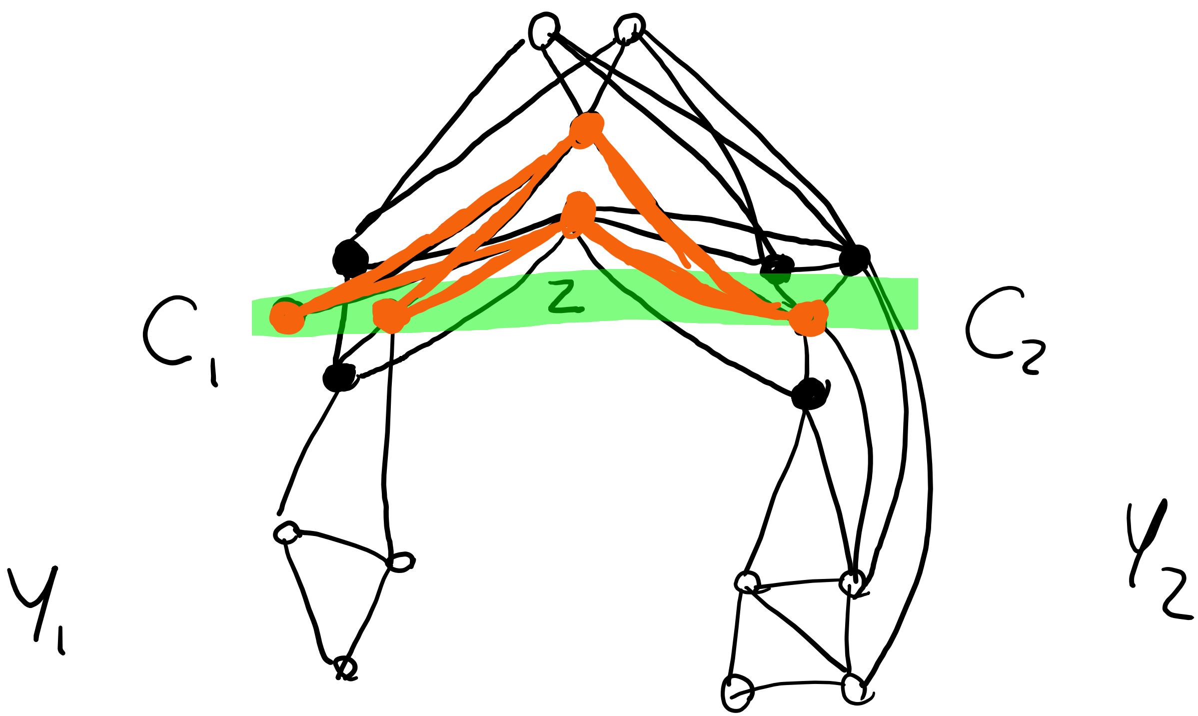
# Similar sets of vertices

- We say that $C_1 \cup Y_1$ and $C_2 \cup Y_2 \subseteq V(G)$ are **similar** if
  - There is $z$ such that $C_1 \cup C_2 \subseteq N(z)$.
  - $C_1$ cuts $Y_1$ and $C_2$ cuts $Y_2$ from the rest of the graph
  - $C_1 \cup C_2$ can be partitioned into layers $L_1, \dots, Lk$ such that **vertices in the same layer have the same neighbors** in $G - (C_1 \cup Y_1 \cup C_2 \cup Y_2)$.



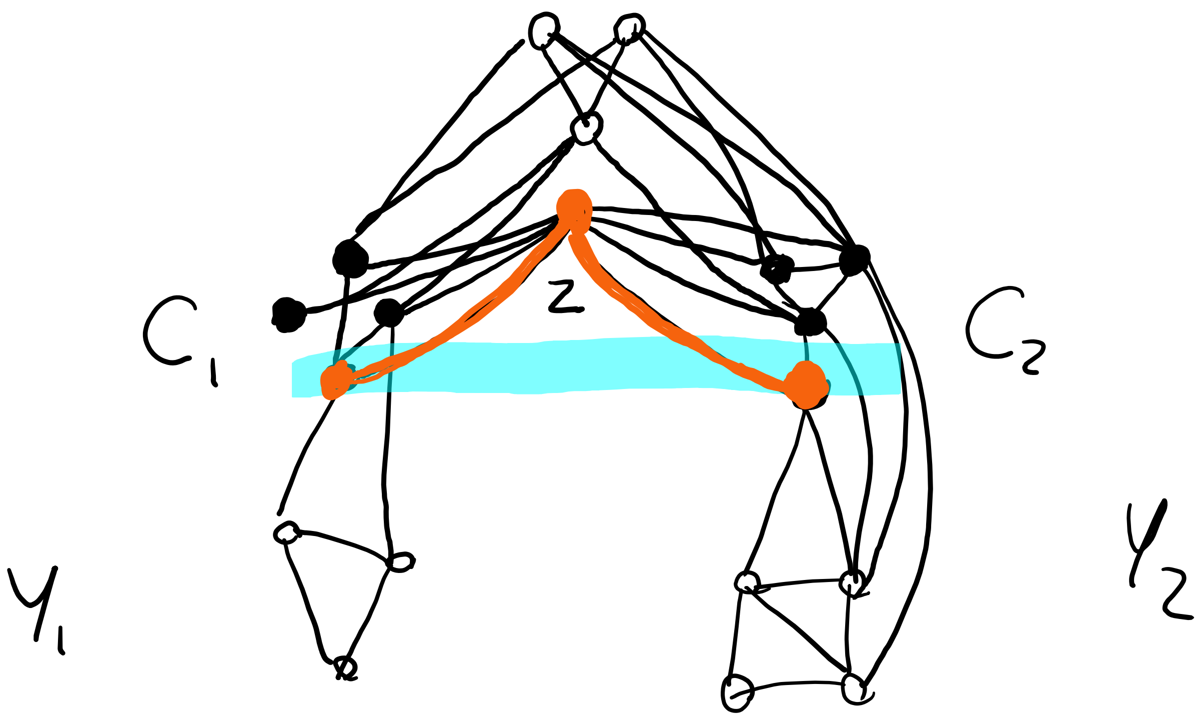
# Similar sets of vertices

- We say that $C_1 \cup Y_1$ and $C_2 \cup Y_2 \subseteq V(G)$ are **similar** if
  - There is $z$ such that $C_1 \cup C_2 \subseteq N(z)$.
  - $C_1$ cuts $Y_1$ and $C_2$ cuts $Y_2$ from the rest of the graph
  - $C_1 \cup C_2$ can be partitioned into layers $L_1, \dots, Lk$ such that **vertices in the same layer have the same neighbors** in $G - (C_1 \cup Y_1 \cup C_2 \cup Y_2)$.



# Similar sets of vertices

- We say that $C_1 \cup Y_1$ and $C_2 \cup Y_2 \subseteq V(G)$ are **similar** if
  - There is $z$ such that $C_1 \cup C_2 \subseteq N(z)$.
  - $C_1$ cuts $Y_1$ and $C_2$ cuts $Y_2$ from the rest of the graph
  - $C_1 \cup C_2$ can be partitioned into layers $L_1, \ldots, Lk$ such that **vertices in the same layer have the same neighbors** in $G - (C_1 \cup Y_1 \cup C_2 \cup Y_2)$.



A *similar structure* of a graph $G$ is a tuple $\mathcal{S} = (\mathcal{C}, \mathcal{Y}, z, \mathcal{L})$ where:

- $\mathcal{C} = \{C_1, \ldots, C_d\}$ is a collection of $d \geq 2$ pairwise disjoint, non-empty subsets of vertices of $G$;
- $\mathcal{Y} = \{Y_1, \ldots, Y_d\}$ is a collection of pairwise disjoint subsets of vertices of $G$, some of which are possibly empty. Also, $C_i \cap Y_j = \emptyset$ for any $i, j \in [d]$;
- $z \in V(G)$ and does not belong to any subset of $\mathcal{C}$ or $\mathcal{Y}$;
- $\mathcal{L} = \{\ell_1, \ldots, \ell_d\}$ is a set of functions where, for each $i \in [d]$, we have $\ell_i : C_i \cup \{z\} \to \{0, 1, \ldots, k\}$. The functions in $\mathcal{L}$ are called *layering functions*.

Additionally, $\mathcal{S}$ must satisfy several conditions. Let us denote $C^* = \bigcup_{i \in [d]} C_i$. Let $X = \{X_1, \ldots, X_t\}$ be the connected components of $G - C^*$. For each $i \in [d]$, denote $X^{(i)} = \{X_j \in X : N_G(X_j) \subseteq C_i\}$, i.e. the components that have neighbors only in $C_i$.

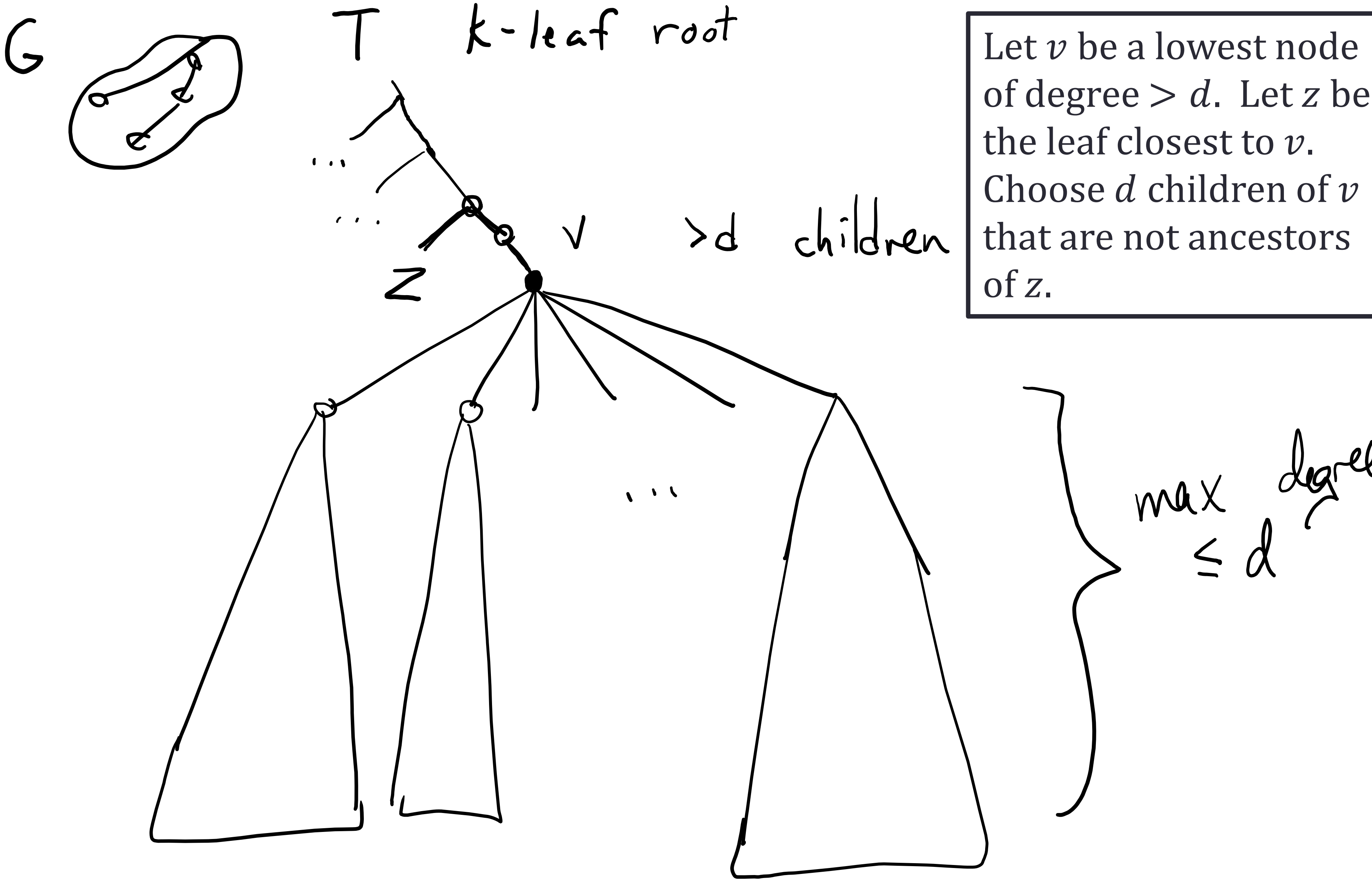
Then all the following conditions must hold:

1. for each $i \in [d]$, $Y_i = \bigcup_{X_j \in X^{(i)}} X_j$ ($Y_i = \emptyset$ is possible);
2. there is exactly one connected component $X_z \in X$ such that for all $i \in [d]$, $N_G(X_z) \cap C_i \neq \emptyset$. Moreover, $z \in X_z$ and $C^* \subseteq N_G(z)$;
3. for all $X_j \in X \setminus \{X_z\}$, $X_j \subseteq Y_i$ for some $i \in [d]$. In particular, $X_z$ is the only connected component of $G - C^*$ with neighbors in two or more $C_i$'s;
4. the layering functions $\mathcal{L}$ satisfy the following:
   (a) for each $i \in [d]$, $\ell_i(z) = 0$. Moreover, $\ell_i(x) > 0$ for any $x \in C_i$;
   (b) for any $i, j \in [d]$ and any $x \in C_i, y \in C_j$, $\ell_i(x) = \ell_j(y)$ implies $N_G(x) \setminus (C_i \cup Y_i \cup C_j \cup Y_j) = N_G(y) \setminus (C_i \cup Y_i \cup C_j \cup Y_j)$. Note that this includes the case $i = j$;
   (c) for any $i, j \in [d]$ and any $x \in C_i, y \in C_j$, $\ell_i(x) + \ell_j(y) \leq k$ implies $xy \in E(G)$. Note that this includes the case $i = j$.
   (d) for any *two distinct* $i, j \in [d]$ and any $x \in C_i, y \in C_j$, $\ell_i(x) + \ell_j(y) > k$ implies $xy \notin E(G)$. Note that this does *not* include the case $i = j$

# Similar sets of vertices

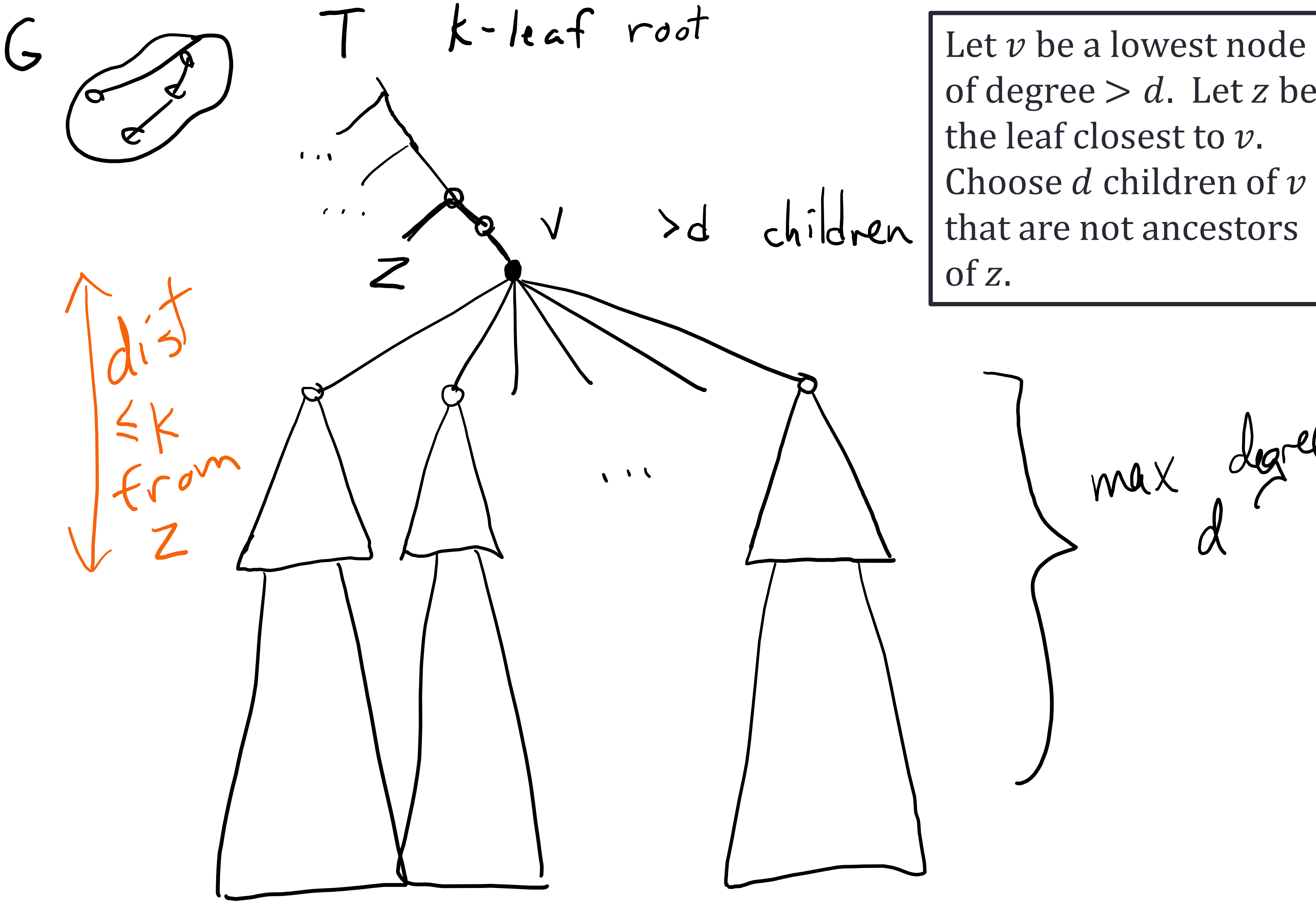
- We say that $C_1 \cup Y_1$ and $C_2 \cup Y_2 \subseteq V(G)$ are **similar** if
  - There is $z$ such that $C_1 \cup C_2 \subseteq N(z)$.
  - $C_1$ cuts $Y_1$ and $C_2$ cuts $Y_2$ from the rest of the graph
  - $C_1 \cup C_2$ can be partitioned into layers $L_1, \ldots, Lk$ such that **vertices in the same layer have the same neighbors** in $G - (C_1 \cup Y_1 \cup C_2 \cup Y_2)$.

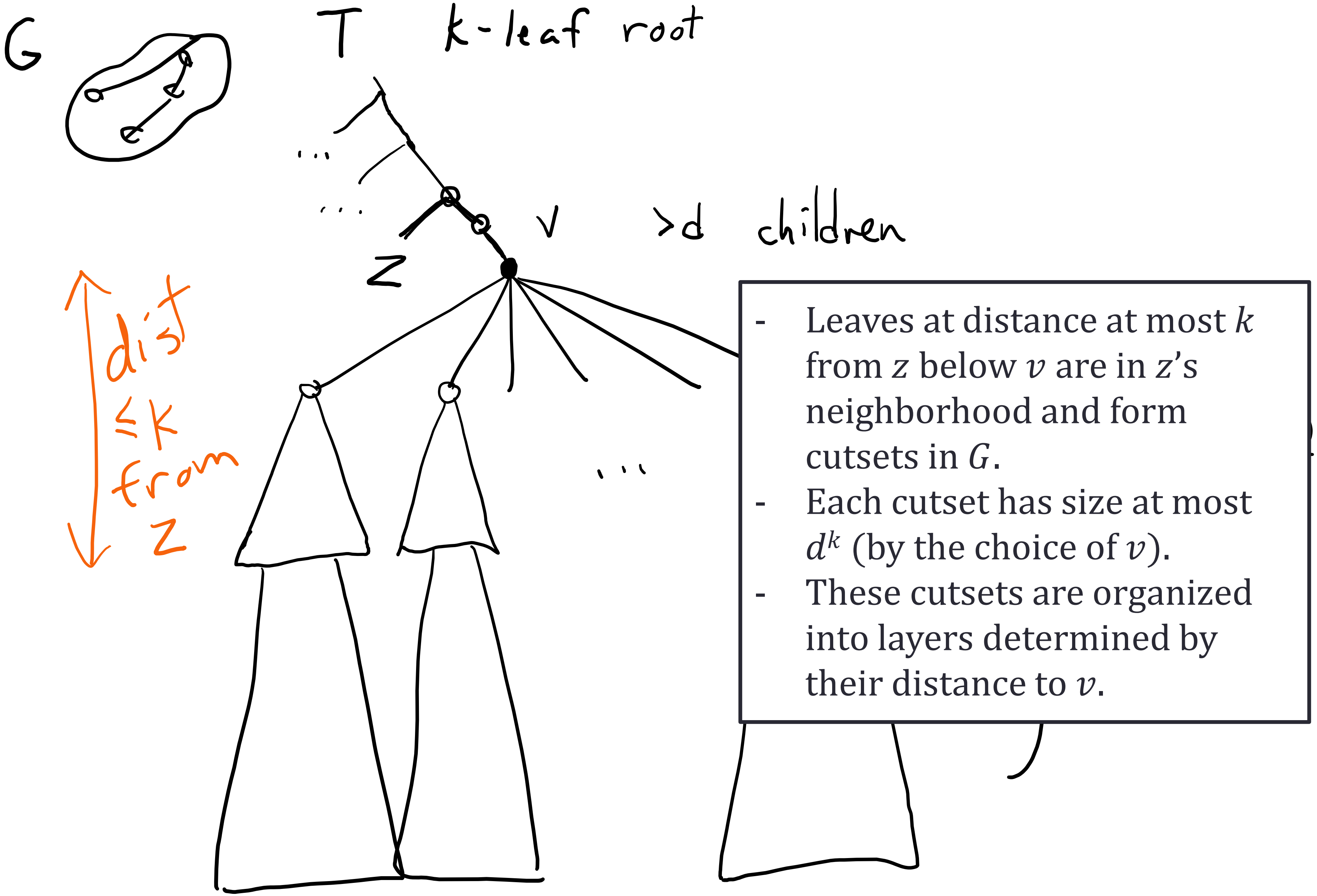
**Lemma**
If $G$ has a $k$-leaf root of maximum degree $> d$, then there exist disjoint $C_1 \cup Y_1, \ldots, Cd \cup Yd$ pairwise-similar subsets that use the same $z$. Also, each $C_i$ has size $\leq dk$.



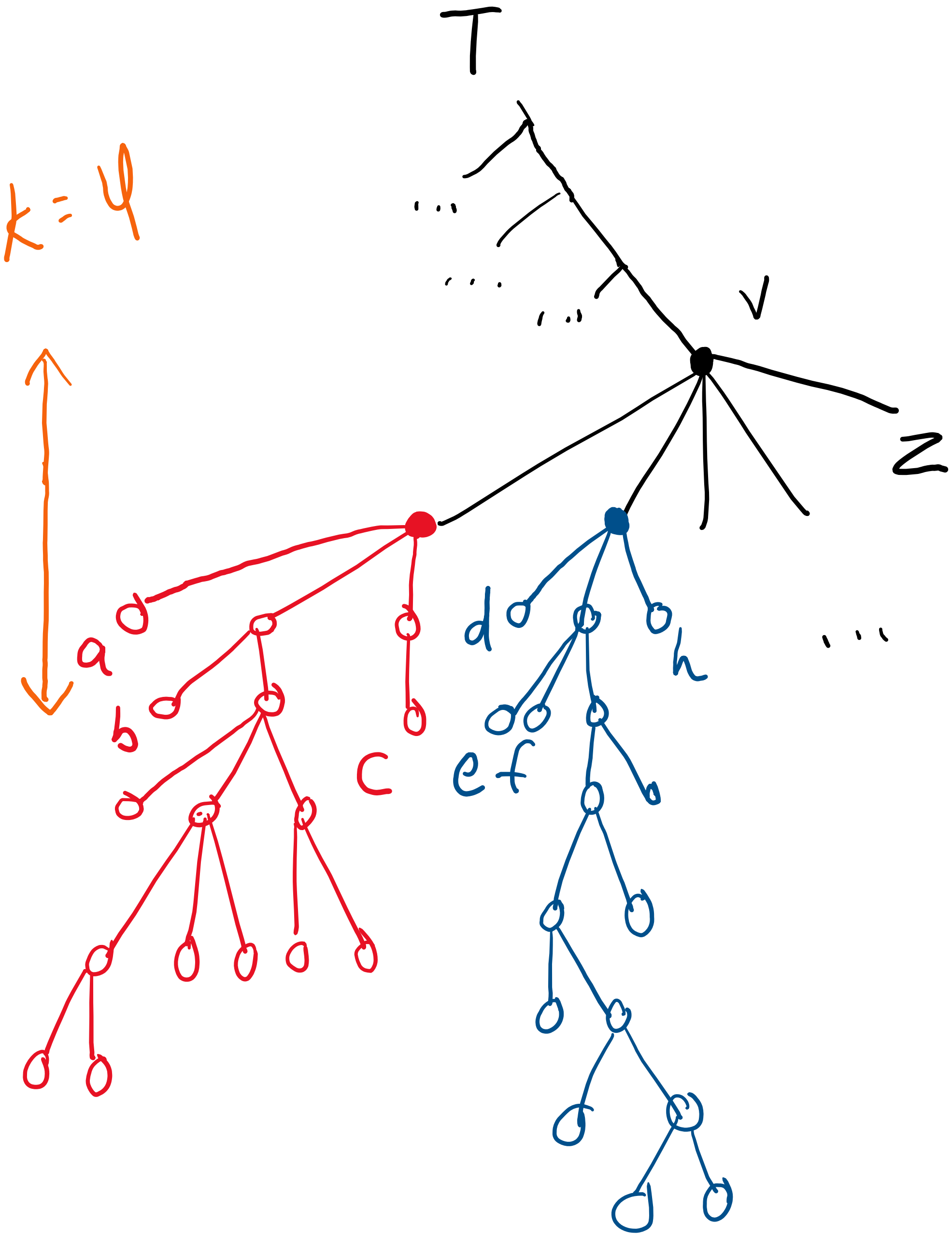

Let $v$ be a lowest node of degree $> d$. Let $z$ be the leaf closest to $v$. Choose $d$ children of $v$ that are not ancestors of $z$.

Let $v$ be a lowest node of degree $> d$. Let $z$ be the leaf closest to $v$. Choose $d$ children of $v$ that are not ancestors of $z$.

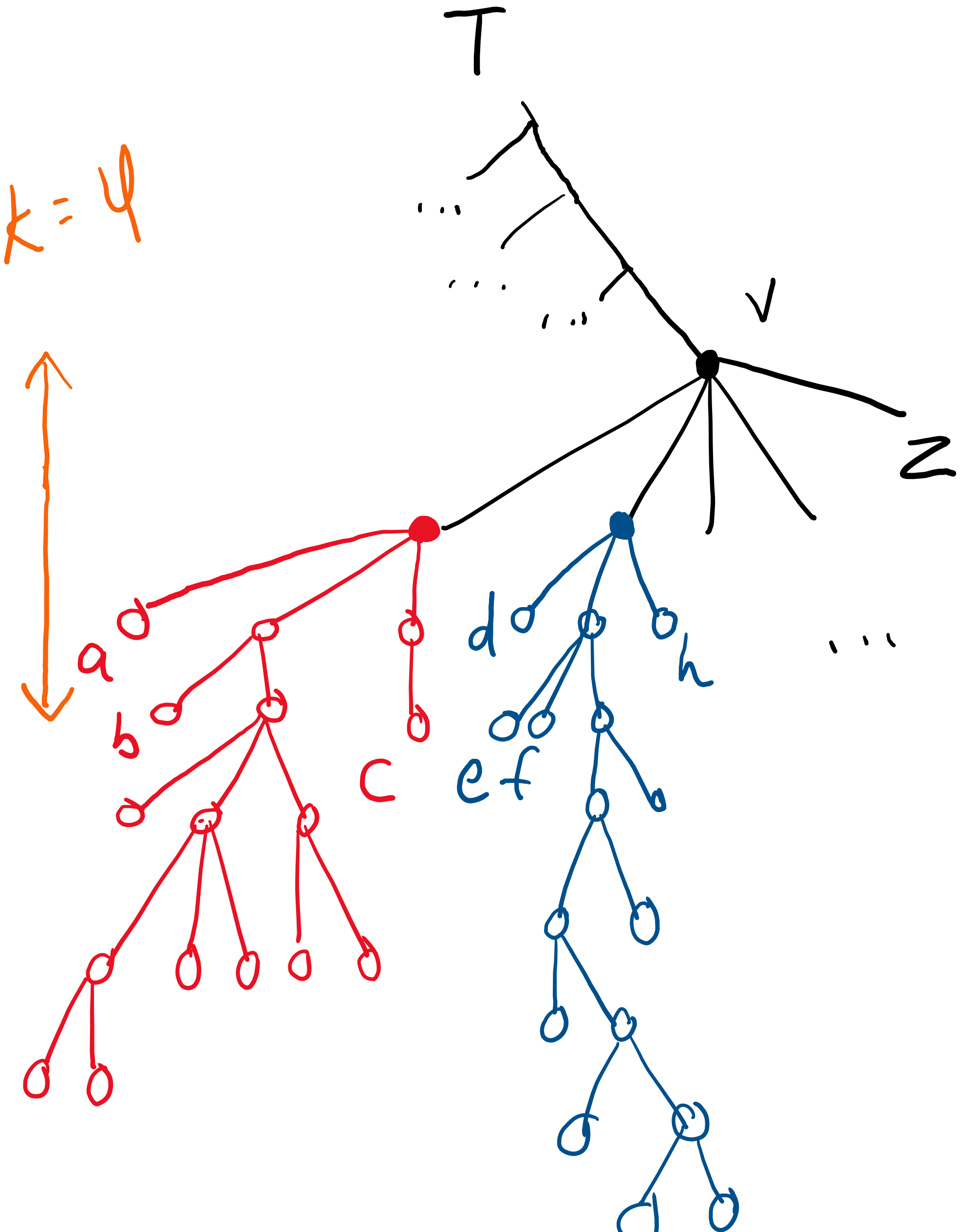




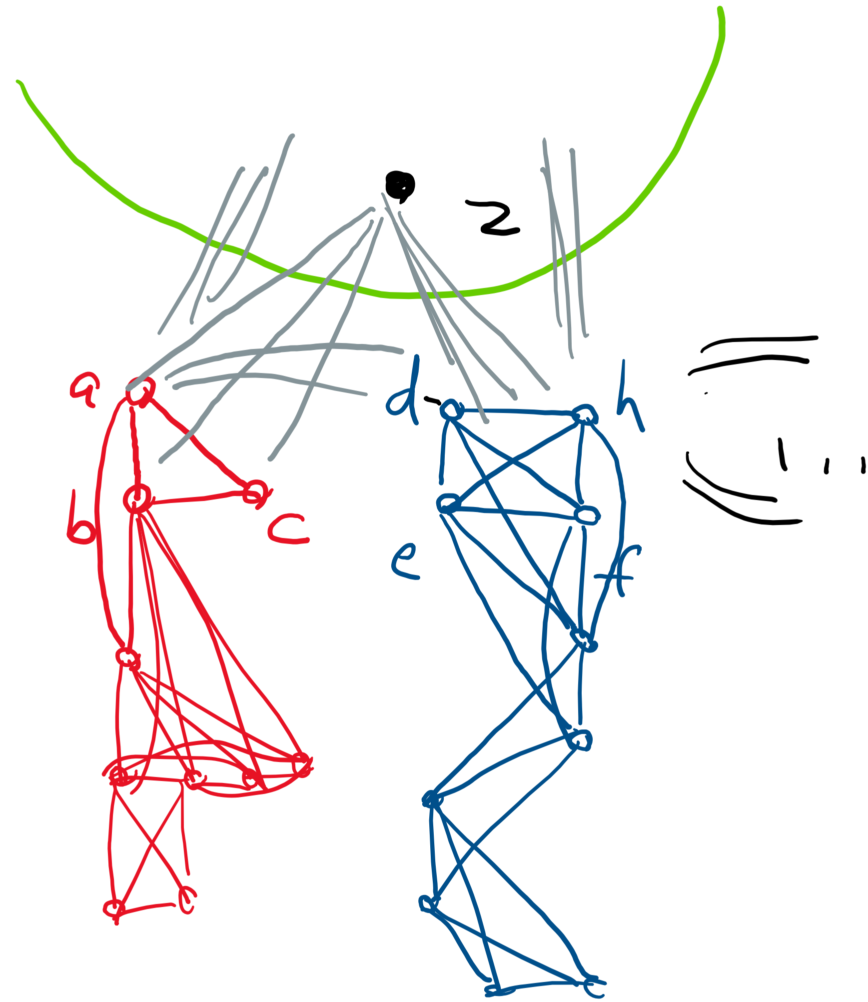

- Leaves at distance at most $k$ from $z$ below $v$ are in $z$'s neighborhood and form cutsets in $G$.
- Each cutset has size at most $d^k$ (by the choice of $v$).
- These cutsets are organized into layers determined by their distance to $v$.

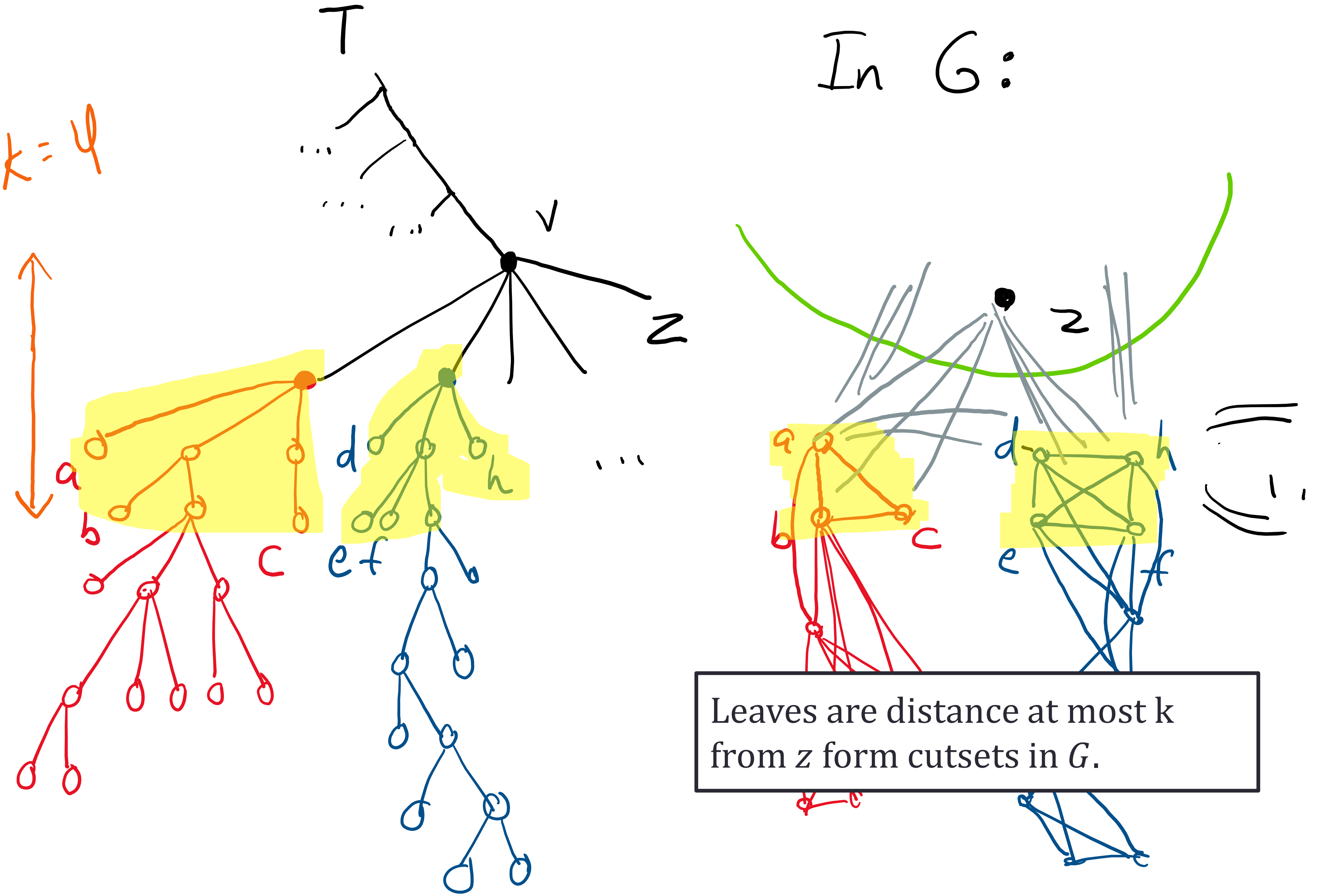



- Leaves at distance at most $k$ from $z$ below $v$ are in $z$'s neighborhood and form cutsets in $G$.
- Each cutset has size at most $d^k$ (by the choice of $v$).
- These cutsets are organized into layers determined by their distance to $v$.

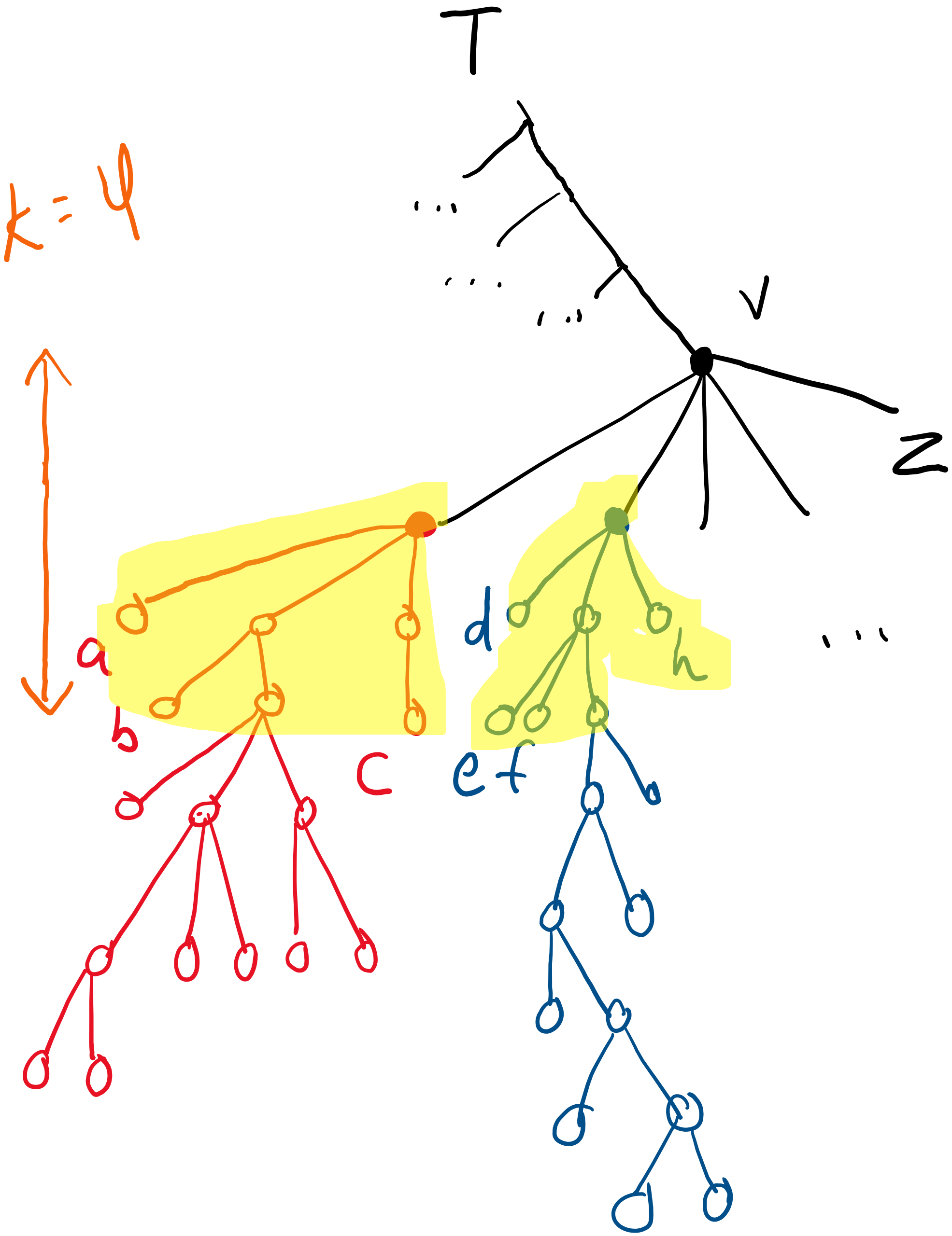


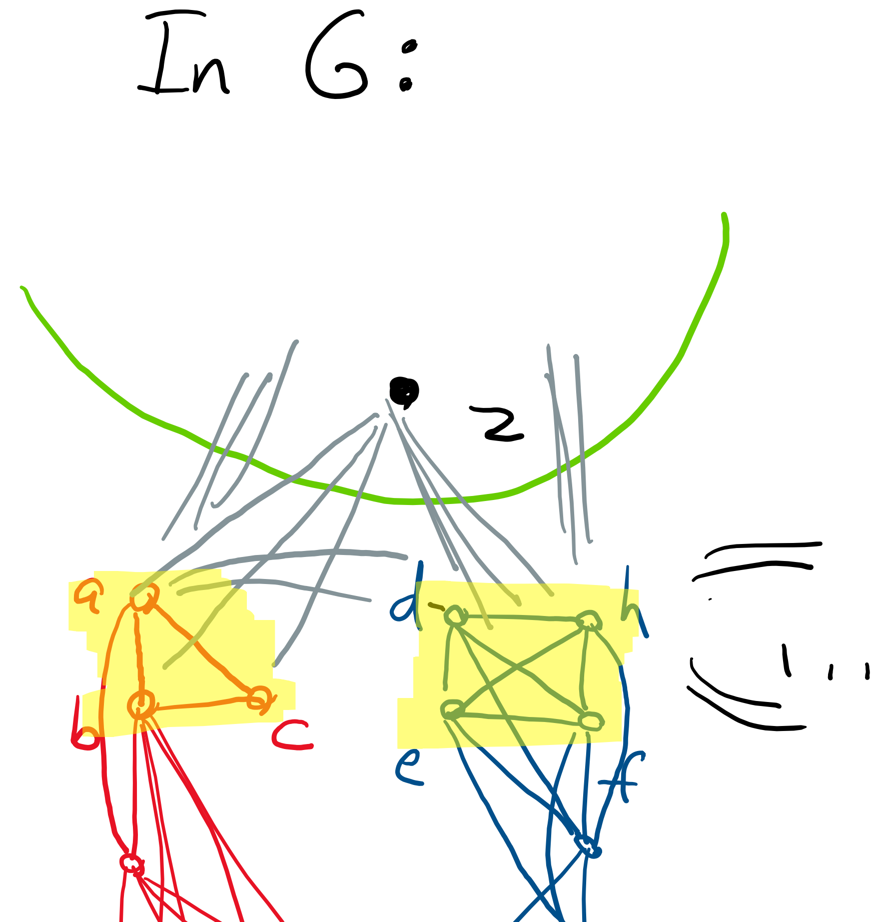

In G:



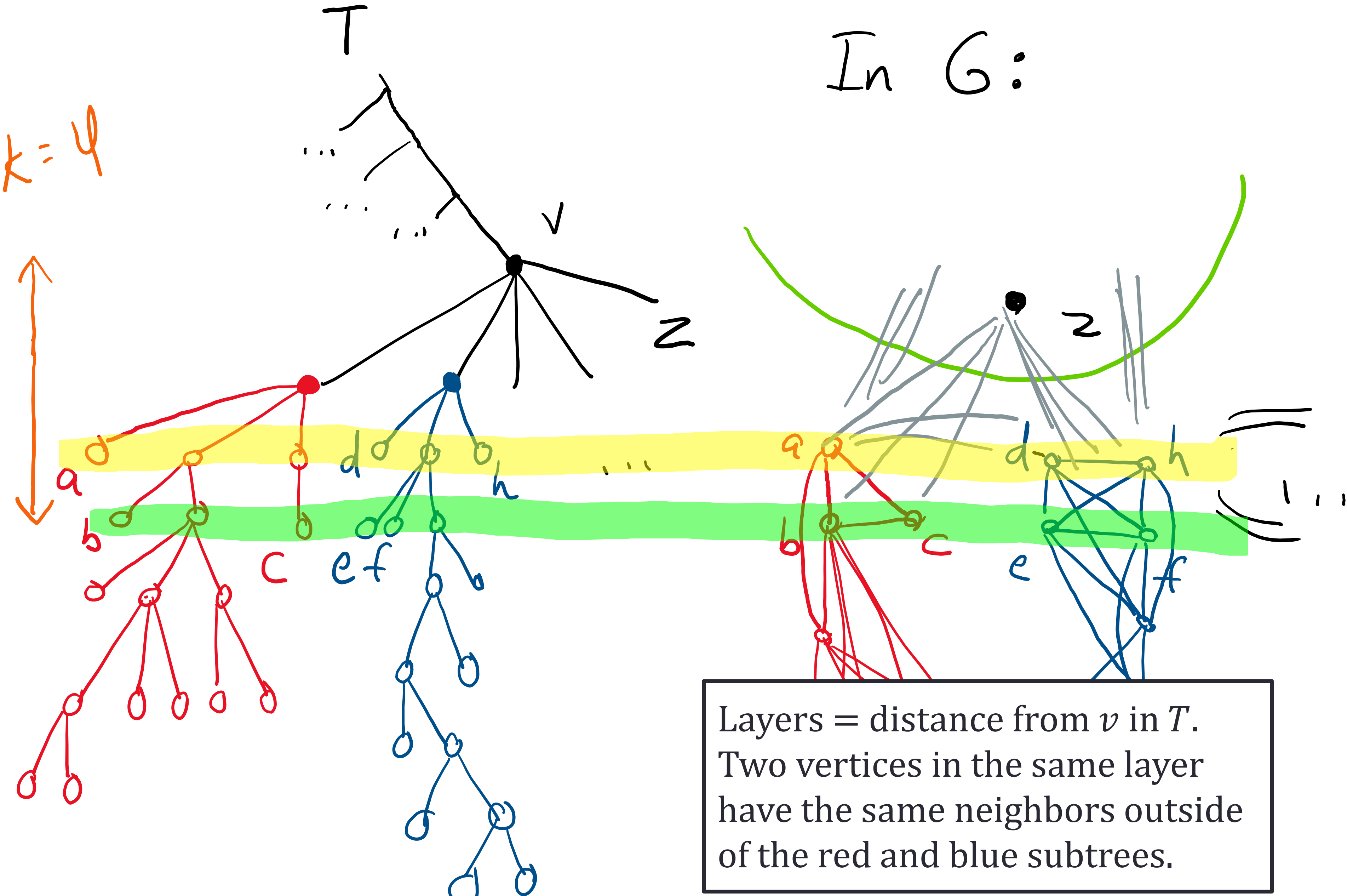



Leaves are distance at most k from $z$ form cutsets in $G$.



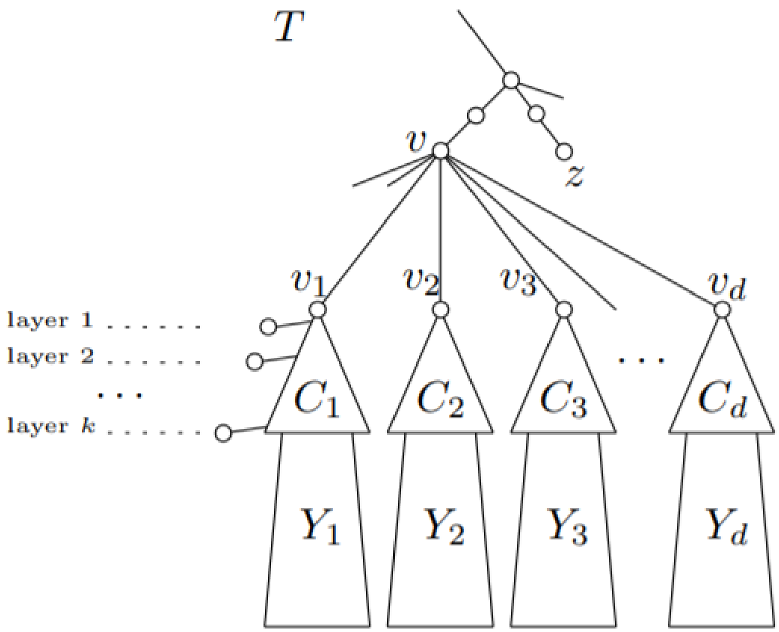



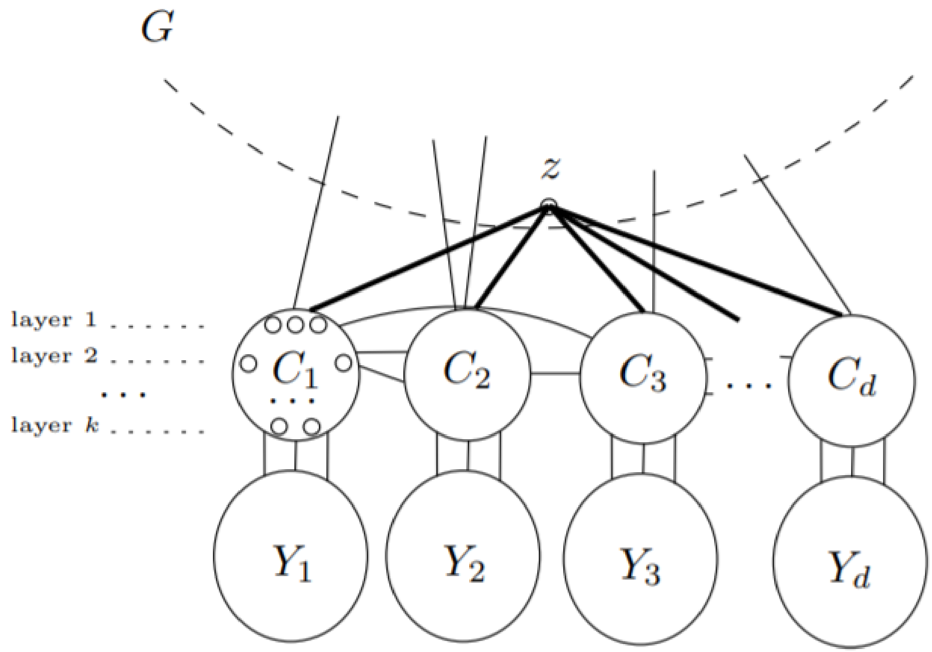

Each cutset has size at most $d^k$ because they are in a subtree of degree at most $d$.




Layers = distance from $v$ in $T$.
Two vertices in the same layer have the same neighbors outside of the red and blue subtrees.

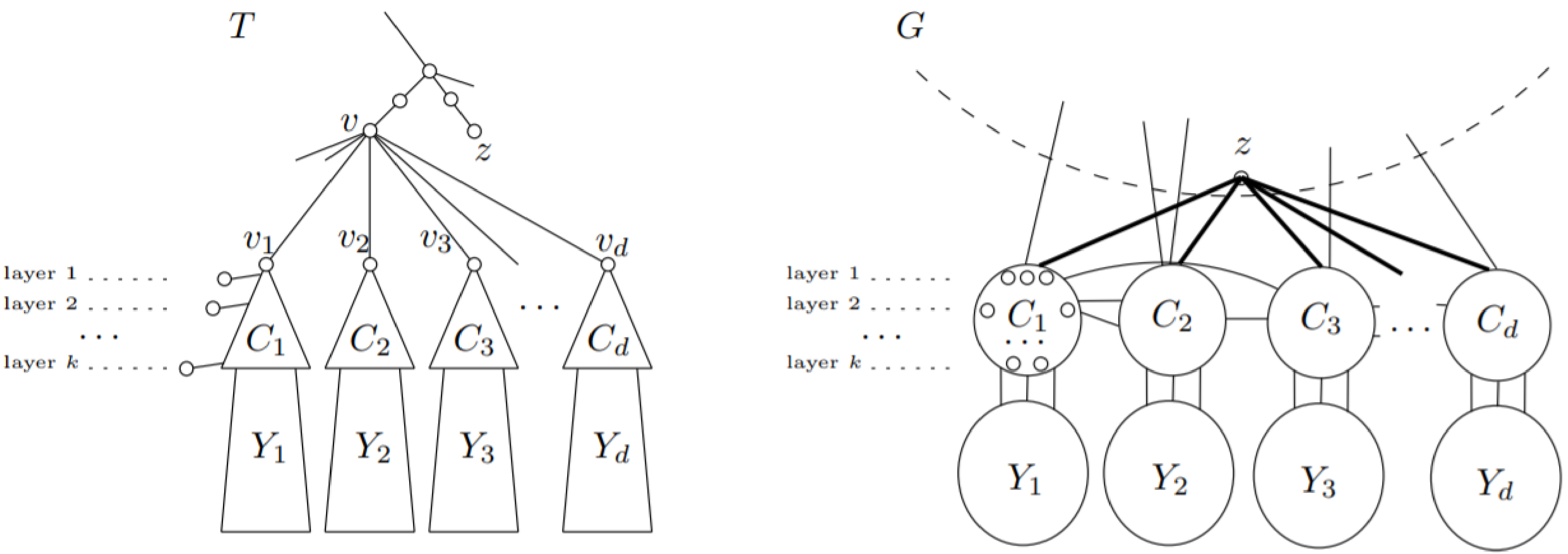
**Lemma**
If $G$ has a $k$-leaf root of maximum degree $> d$, then there exist disjoint $C_1 \cup\ Y_1, \dots, Cd \cup Yd$ pairwise-similar subsets that use the same $z$. Also, each $C_i$ has size $\leq dk$.



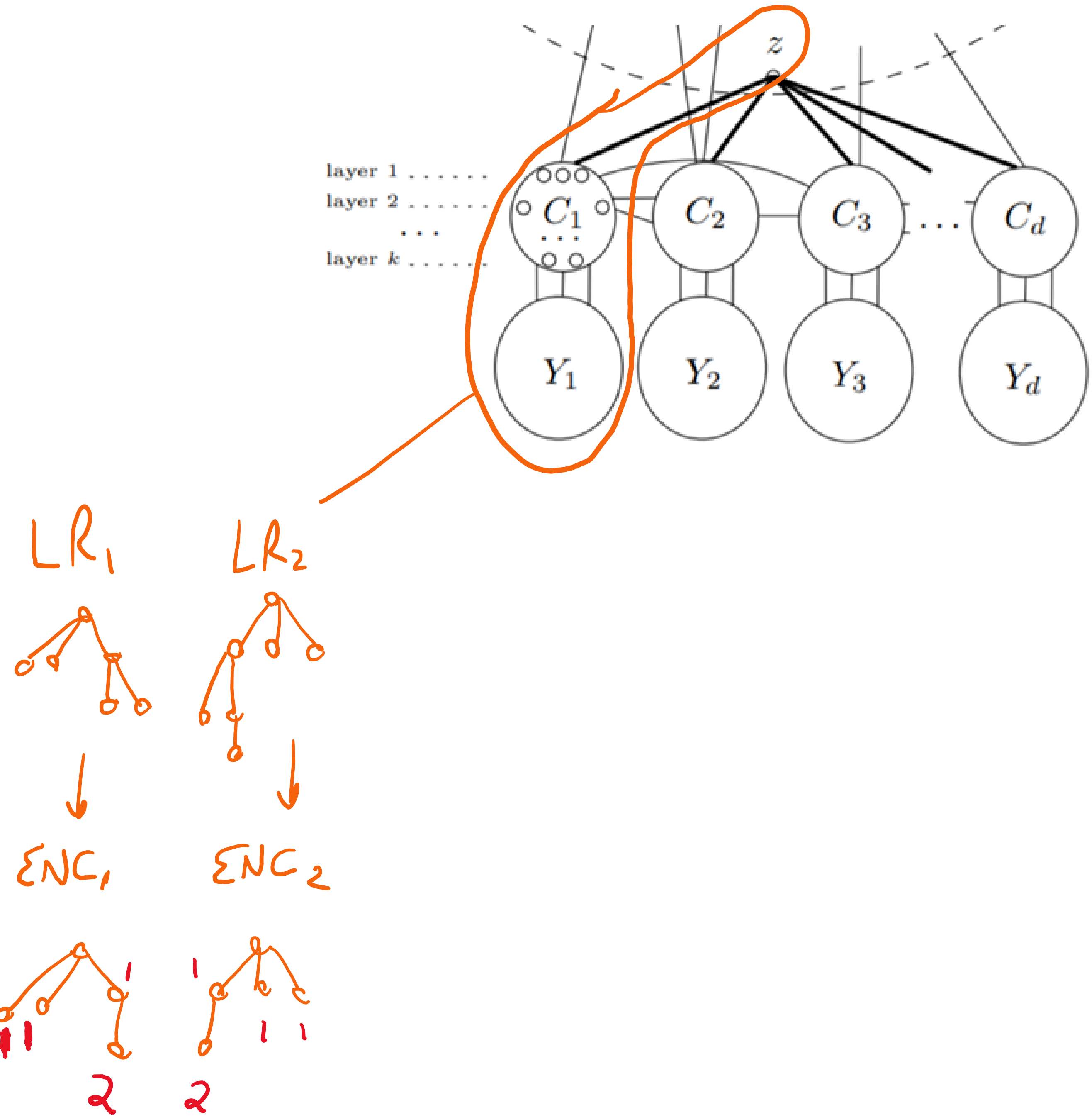



**Lemma**
If $G$ has a $k$-leaf root of maximum degree $> d$, then there exist disjoint $C_1 \cup Y_1, \dots, Cd \cup Yd$ pairwise-similar subsets that use the same $z$. Also, each $C_i$ has size $\leq dk$.

- So we can find many subsets with the same neighborhood structure.
- Next : find those that have the "same" $k$-leaf roots.

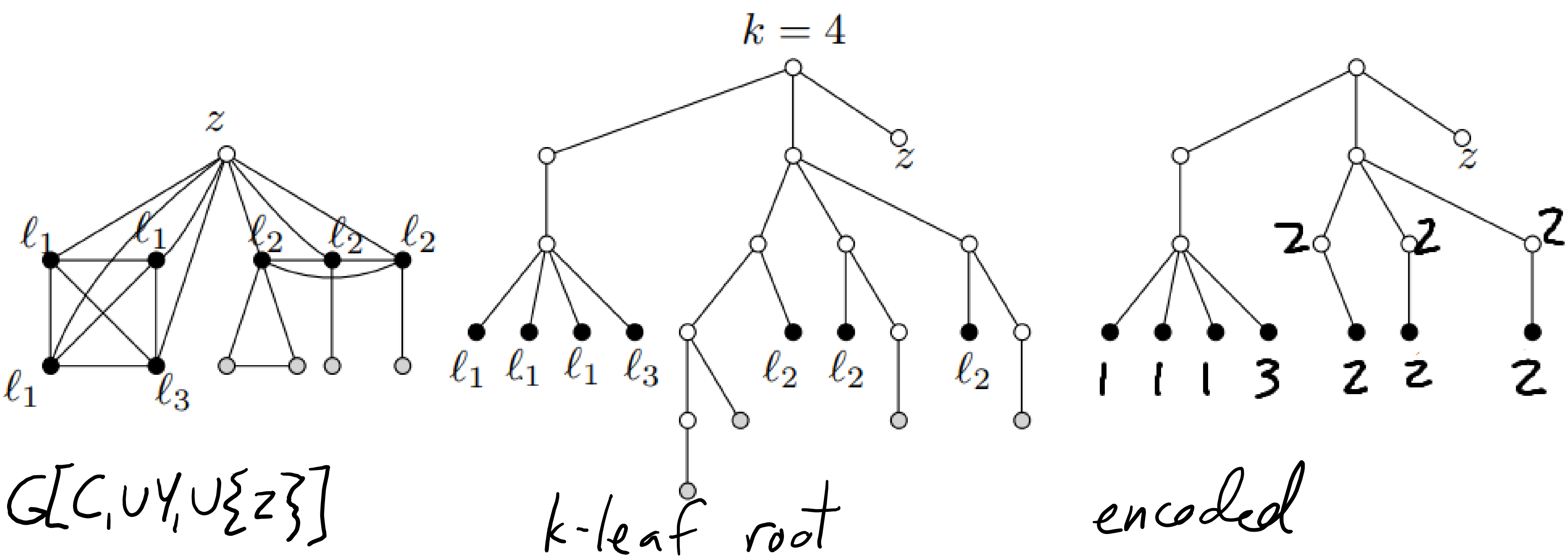


Step 2 : similar sets that have the same encoded $k$-leaf roots


z
layer 1
layer 2
layer k
$C_1$
$C_2$
$C_3$
$C_d$
$Y_1$
$Y_2$
$Y_3$
$Y_d$
$LR_1$
$LR_2$
$ENC_1$
$ENC_2$

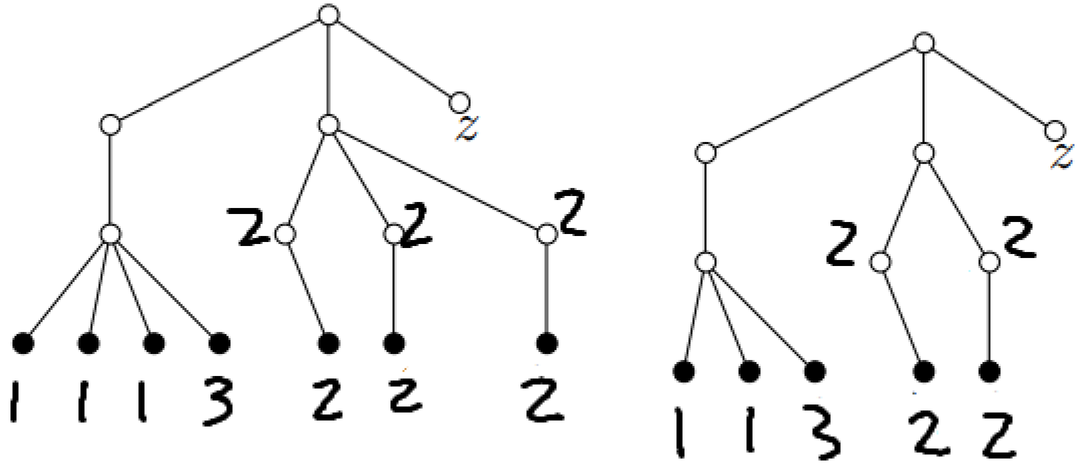
# Similar sets with the same leaf roots

- Let $C_1 \cup Y_1$ be a set of vertices organized into layers $L_1, \dots, Lk$.
- Let $T_1$ be a $k$-leaf root of $G[C_1 \cup Y_1 \cup \{z\}]$. The **layer-encoding** of $T_1$ is obtained by
  - restricting $T_1$ to $C_1$ and $z$, and their ancestors
  - replacing each leaf of $C_1$ by its layer number.
  - labeling internal nodes by the distance to its closest $Y_1$ leaf
  - also…



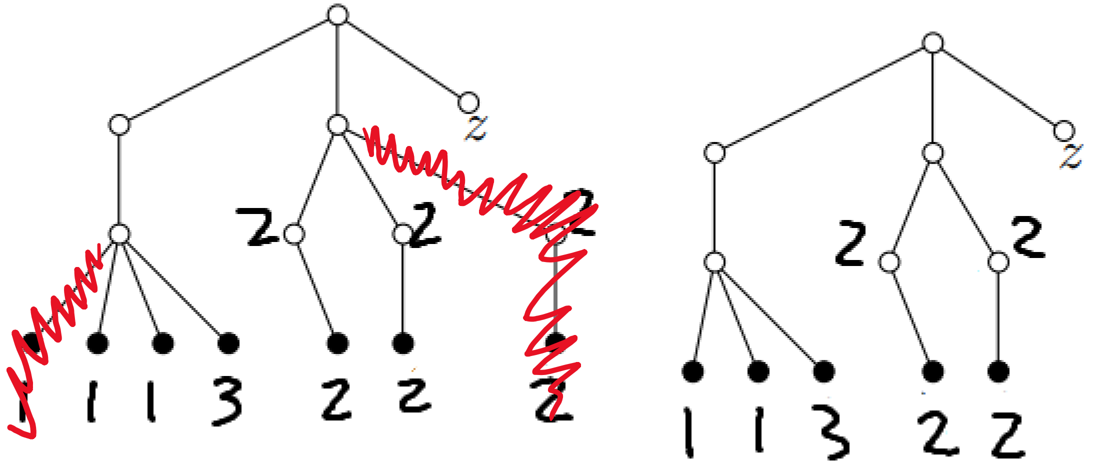
# Similar sets with the same leaf roots

- also…for each node $u$ that has at least 3 identical child subtrees, we remove one of these subtrees (they are redundant for our purposes).

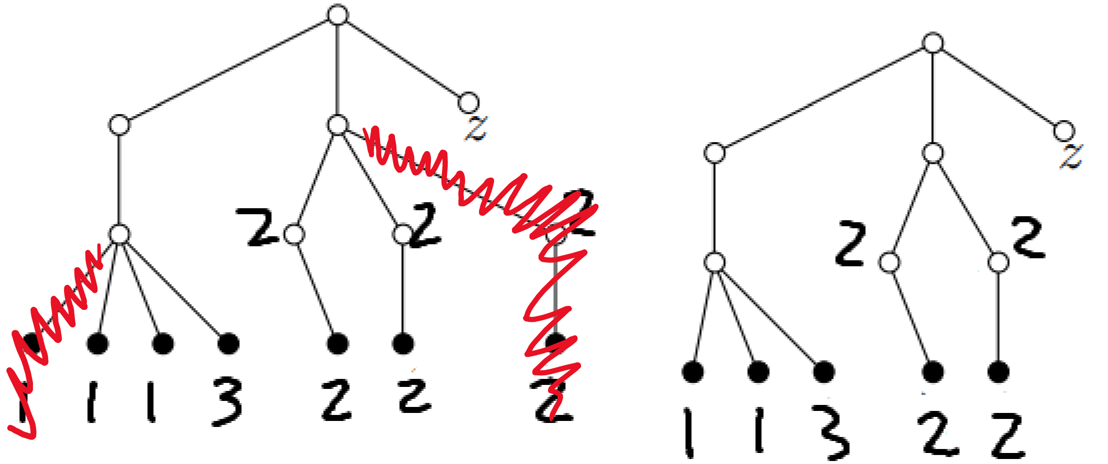


# Similar sets with the same leaf roots

- also…for each node $u$ that has at least 3 identical child subtrees, we remove one of these subtrees (they are redundant for our purposes).



**Lemma**
The number of possible layer-encoded $k$-leaf roots is at most $s(k)$, a function that depends only on $k$.



**Lemma**
The number of possible layer-encoded $k$-leaf roots is at most $s(k)$, a function that depends only on $k$.

Proof idea.

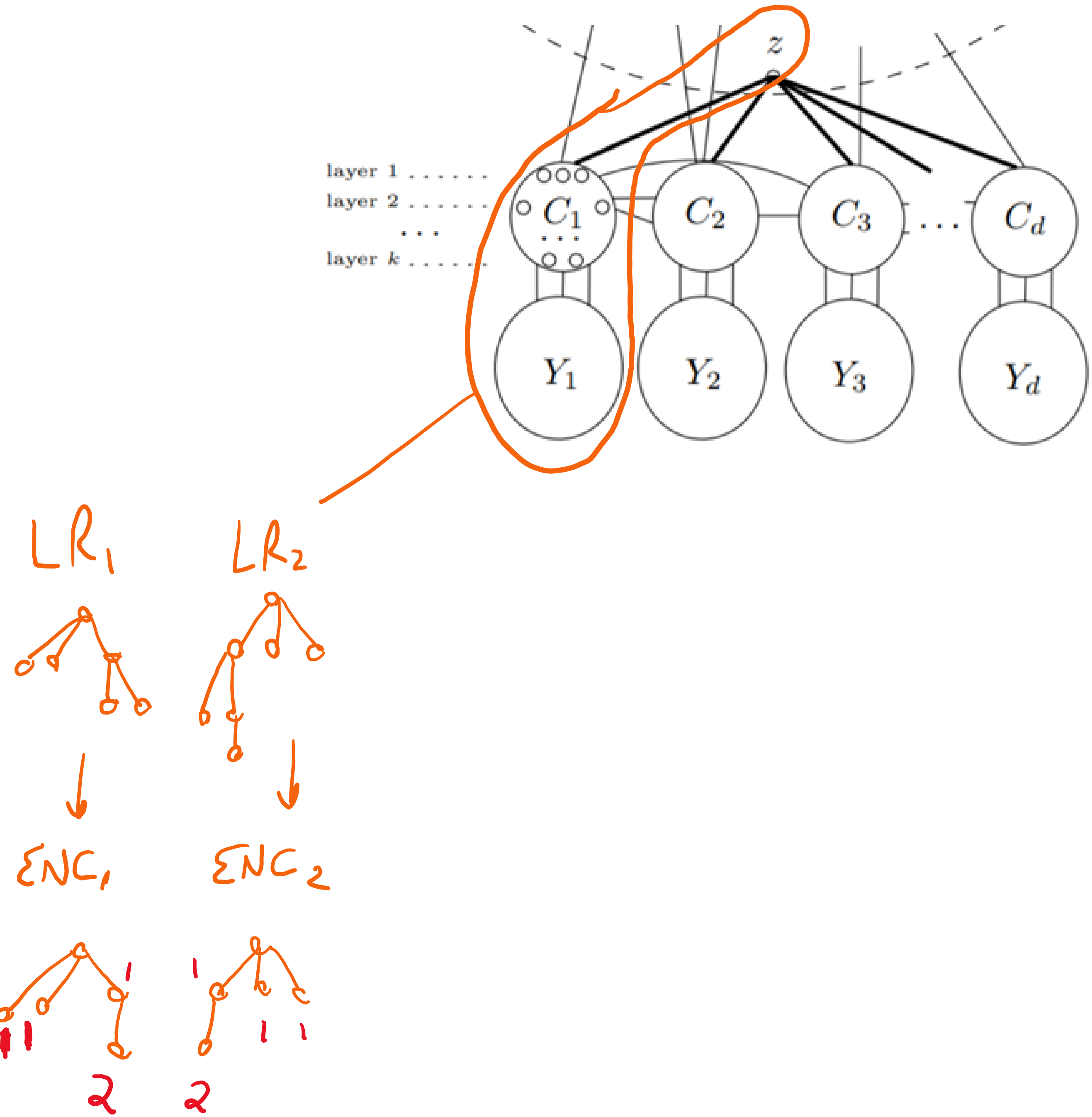
Layer-encoded k-leaf roots have height at most k.

Possible layer-encoded k-leaf roots:

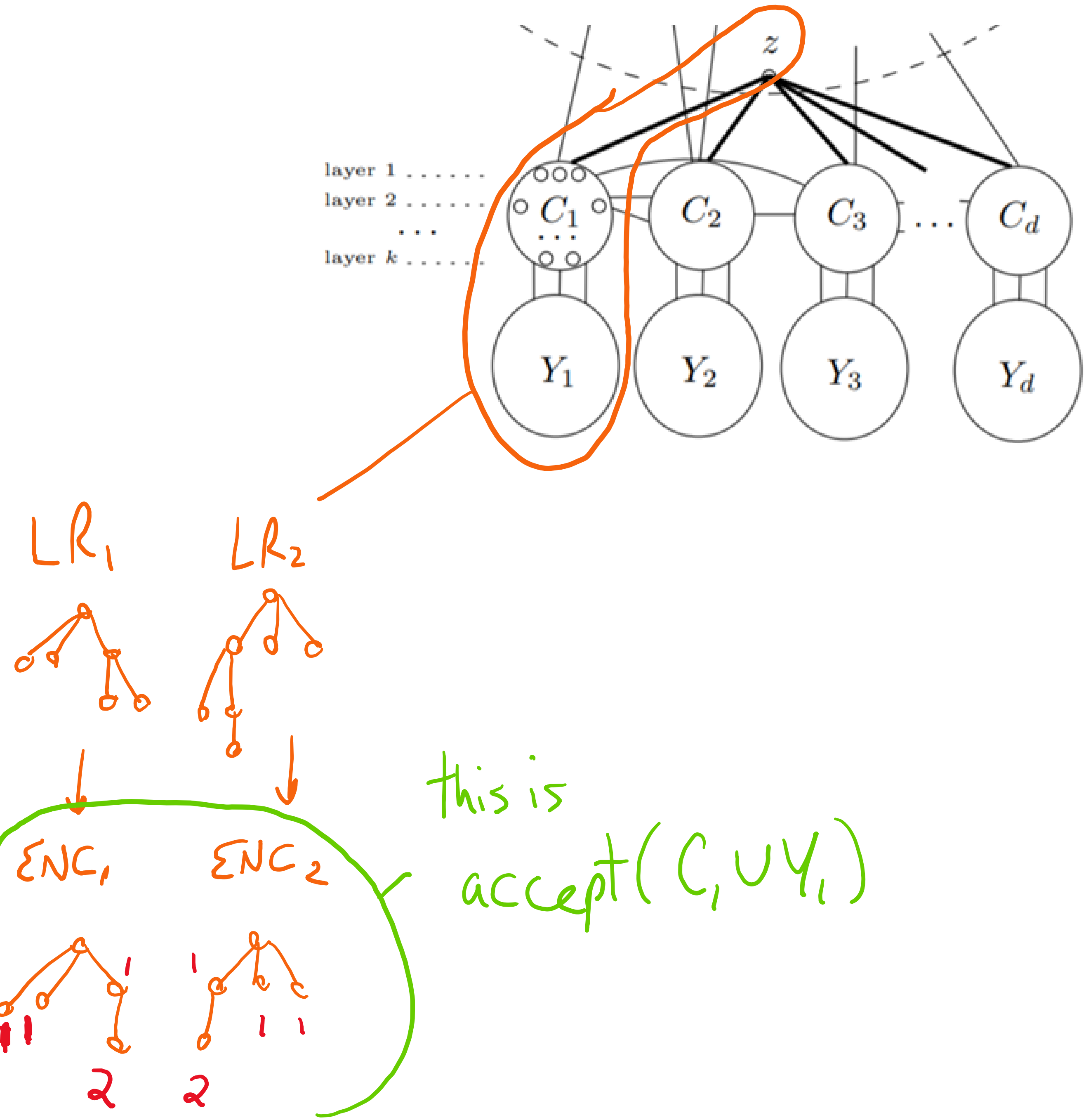
- of height 1 : $k$ (number of layer numbers)
- of height 2 : $k3^k$ ($k$ values for internal node, 0, 1 or 2 children of each type of height 1)
- of height 3 : $k3^{k3^k}$
- …
- of height $k$ : $k3^{k3^{k3^{\cdot^{\cdot^{\cdot^{k3^k}}}}}}$ } $k$ times

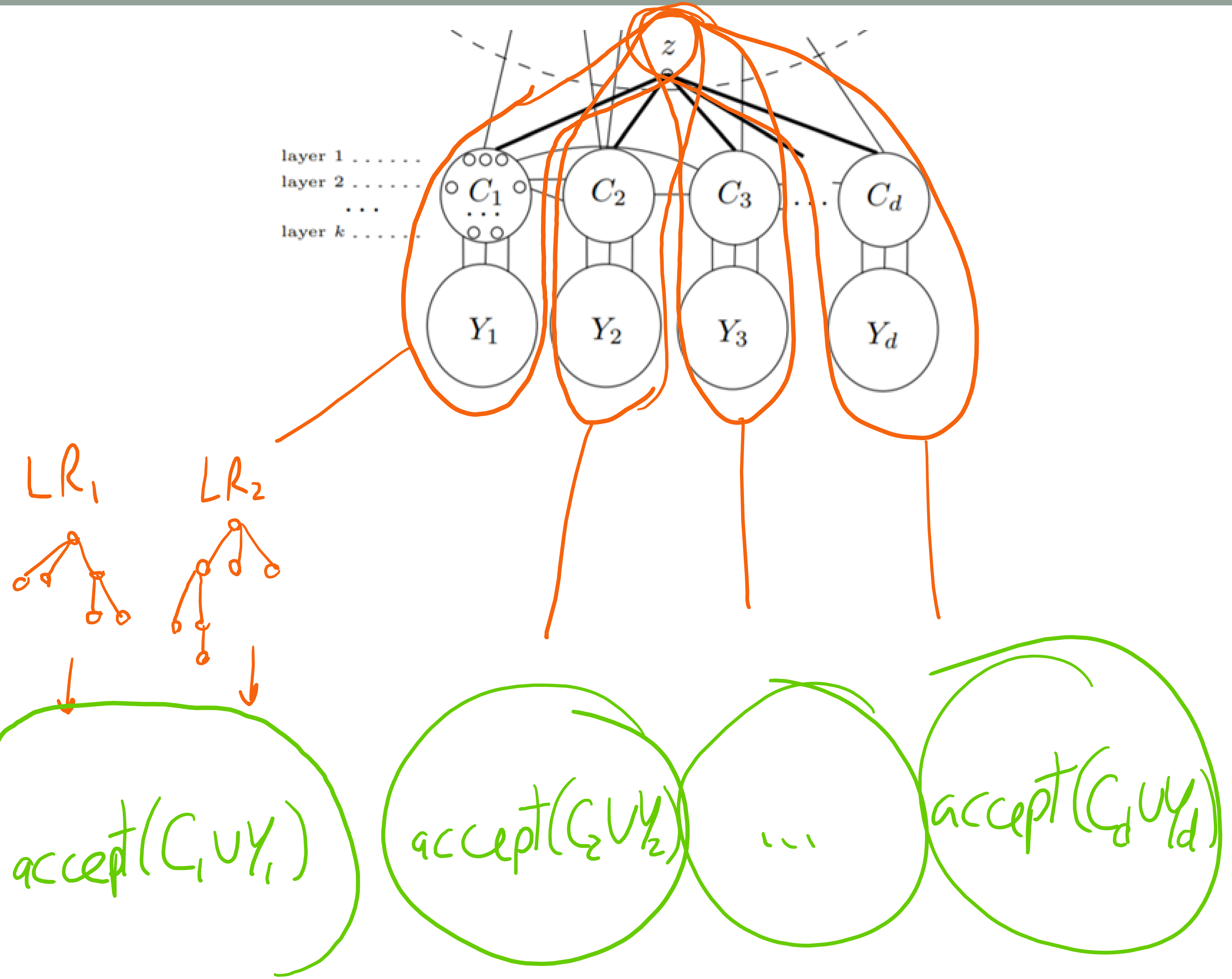
- For $C_i \cup Yi$, let $\boldsymbol{accept(C_i \cup Yi)}$ be the set of layer-encoded $k$-leaf roots of $G[C_i \cup Yi \cup \{z\}]$.
- We say that similar subsets $C_1 \cup Y_1, \ldots, C_d \cup Yd$ are **homogeneous** if all accept sets are equal, i.e.

$$\boldsymbol{accept(C_1 \cup Y_1)} = \ldots = \boldsymbol{accept(Cd \cup Yd)}.$$



z
layer 1
layer 2
layer k
$C_1$
$C_2$
$C_3$
$C_d$
$Y_1$
$Y_2$
$Y_3$
$Y_d$
LR1
LR2
ENC1
ENC2


z
layer 1
layer 2
layer k
$C_1$
$C_2$
$C_3$
$C_d$
$Y_1$
$Y_2$
$Y_3$
$Y_d$
$LR_1$
$LR_2$
$ENC_1$
$ENC_2$
this is
accept$(C_1 \cup Y_1)$



z
layer 1 . . . . . . .
layer 2 . . . . . . .
. . .
layer k . . . . . . .
C1
C2
C3
. . .
Cd
Y1
Y2
Y3
Yd
LR1
LR2
accept(C1 ∪ Y1)
accept(C2 ∪ Y2)
...
accept(Cd ∪ Yd)

- For $C_i \cup Yi$, let $\boldsymbol{accept(Ci \cup Yi)}$ be the set of layer-encoded $k$-leaf roots of $G[Ci \cup Yi \cup \{z\}]$.
- We say that similar subsets $C_1 \cup Y_1, \ldots, C_d \cup Yd$ are **homogeneous** if all accept sets are equal, i.e.

$$\boldsymbol{accept(C_1 \cup Y_1) = \ldots = accept(Cd \cup Yd)}.$$

**Lemma**

If G has a k-leaf root of maximum degree d $> 3s(k)\, 2^{s(k)}$, then $G$ contains $3s(k)$ similar and **homogeneous** subsets $C_1 \cup Y_1, \ldots, C_{3s(k)} \cup Y_{3s(k)}$. They all use the same $z$ and $|Ci| \leq dk$ for each $i$.

- For $C_i \cup Yi$, let ***accept*** $(\boldsymbol{Ci} \cup \boldsymbol{Yi})$ be the set of layer-encoded $k$-leaf roots of $G[Ci \cup Yi \cup \{z\}]$.
- We say that similar subsets $C_1 \cup Y_1, \ldots, C_d \cup Yd$ are **homogeneous** if all accept sets are equal, i.e.

  $\boldsymbol{accept(C_1 \cup Y_1) = \ldots = accept(Cd \cup Yd)}$.

**Lemma**

If G has a k-leaf root of maximum degree d $> 3s(k)\,2^{s(k)}$, then $G$ contains $3s(k)$ similar and **homogeneous** subsets $C_1 \cup Y_1, \ldots, C_{3s(k)} \cup Y_{3s(k)}$. They all use the same $z$ and $|Ci| \leq dk$ for each $i$.

Pigeonhole argument. There are $2^{s(k)}$ possible accept sets. If $d > 3s(k)\,2^{s(k)}$, we find $d$ similar subsets and at least $3s(k)$ of them have the same accept set.

# Step 3 : pruning one homogeneous subset and embedding its k-leaf root

- Recall the thing that I'm trying to do.

**Theorem**

There is $f$ such that if $G$ admits a $k$-leaf root of max degree $d > f(k)$, then $G$ contains a subset $C$ of vertices such that **$G$ is a $k$-leaf power if and only if $G - C$ is a $k$-leaf power**.

Moreover, $C$ can be found in time $O(n^{f(k)})$ if it exists.

- Recall the thing that I'm trying to do.

> **Theorem**
> There is $f$ such that if $G$ admits a $k$-leaf root of max degree $d > f(k)$, then $G$ contains a subset $C$ of vertices such that **$G$ is a $k$-leaf power if and only if $G - C$ is a $k$-leaf power**.
> Moreover, $C$ can be found in time $O(n^{f(k)})$ if it exists.

Let $C_1 \cup Y_1, \ldots, C_{3s(k)} \cup Y_{3s(k)}$ be a large enough number of similar + homogeneous sets.

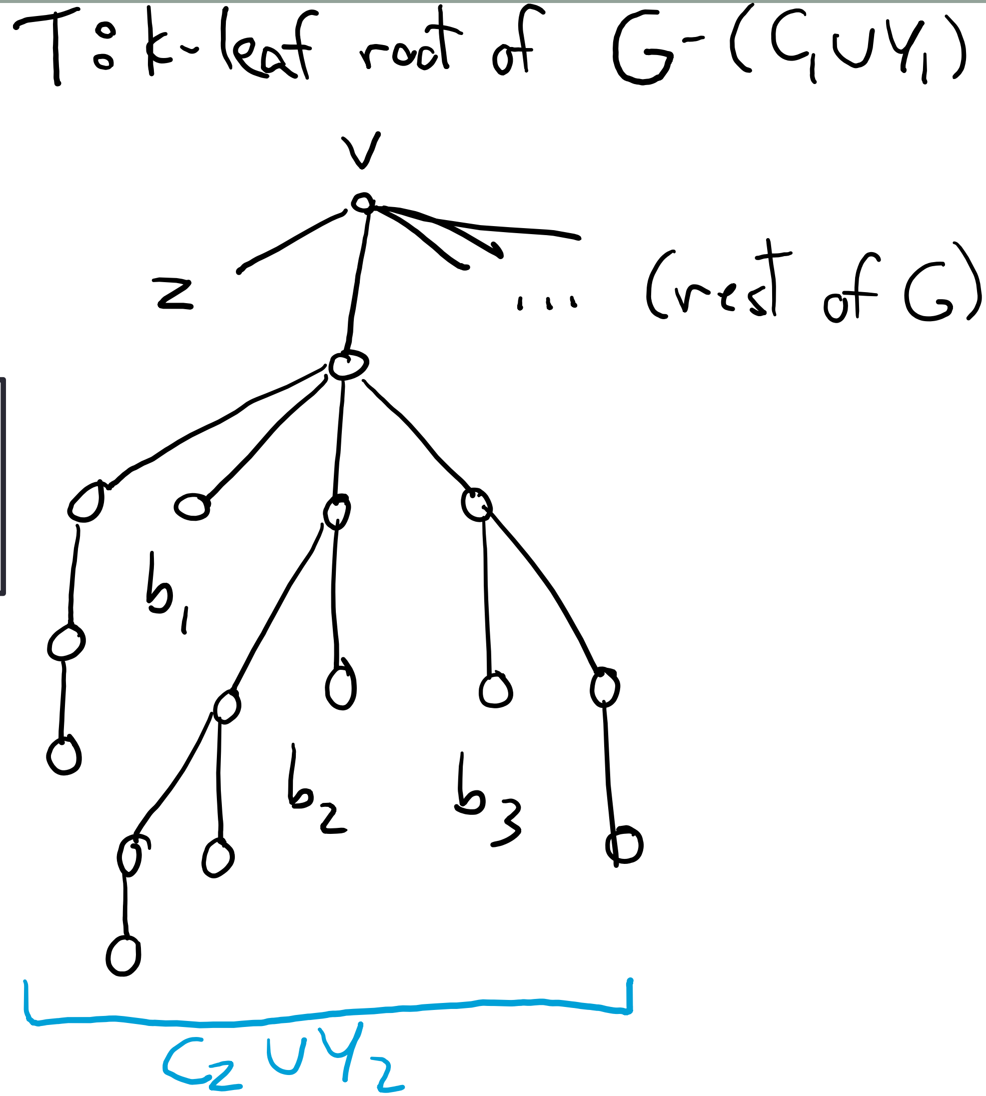
Consider $G - (C_1 \cup Y_1)$.

$\Rightarrow$ If $G$ is a $k$-leaf power, then $G - (C_1 \cup Y_1)$ is a $k$-leaf power.

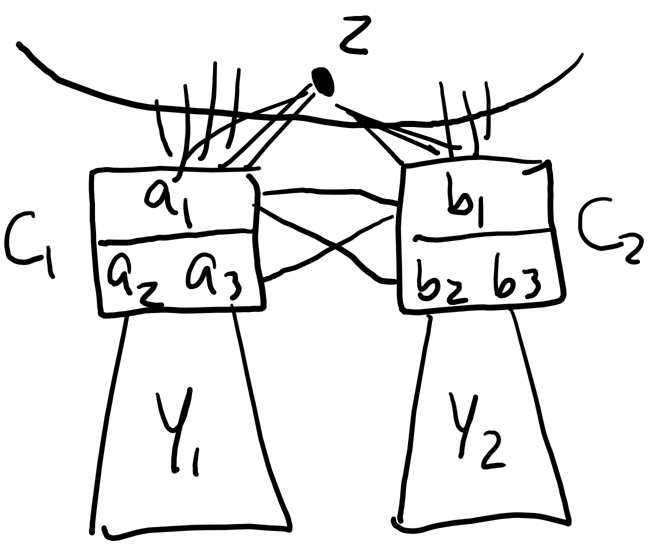
$\Leftarrow$ Assume that $G - (C_1 \cup Y_1)$ is a $k$-leaf power.

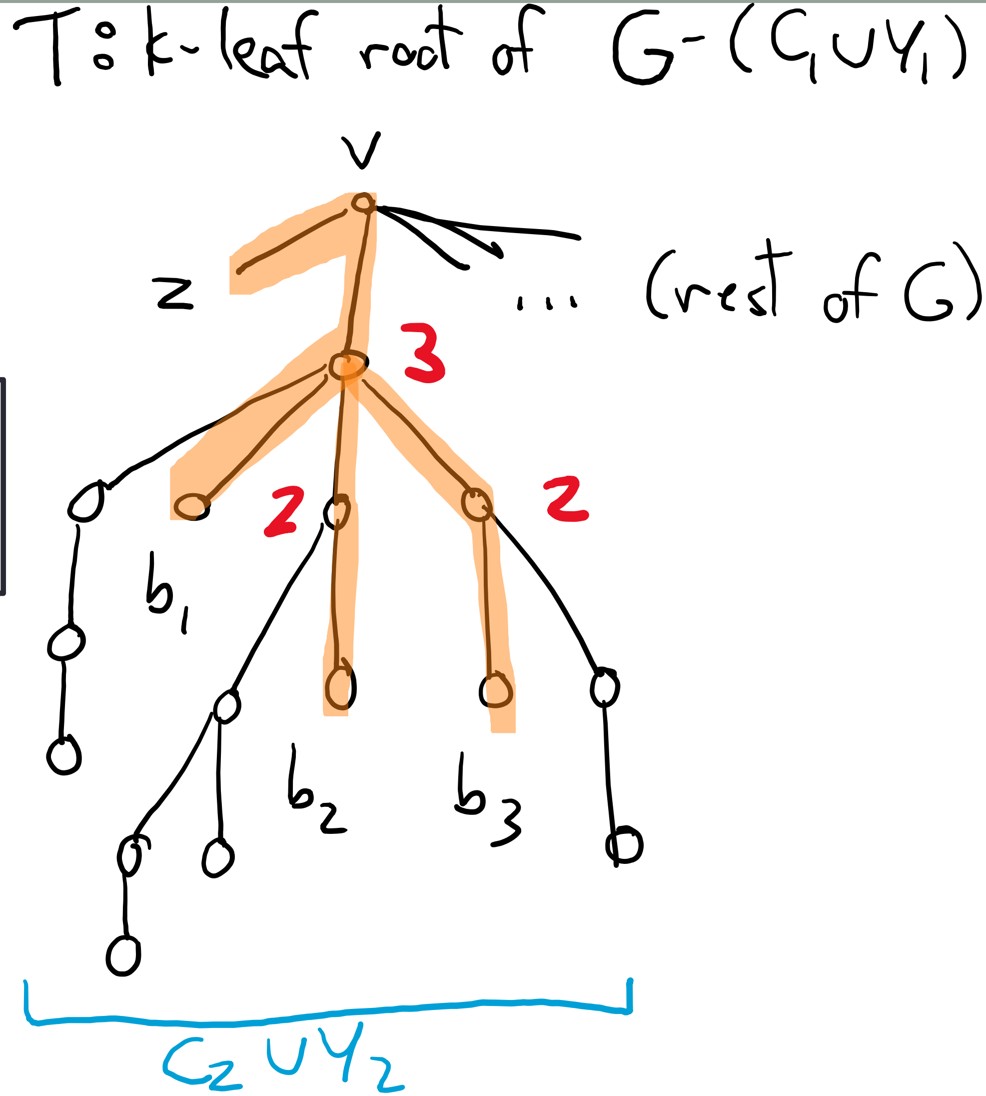
GOAL : argue that $G$ is a $k$-leaf power.

Start with a $k$-leaf root $T$ of $G - (C_1 \cup Y_1)$. Somehow, add $C_1 \cup Y_1$ into it while satisfying distance requirements.

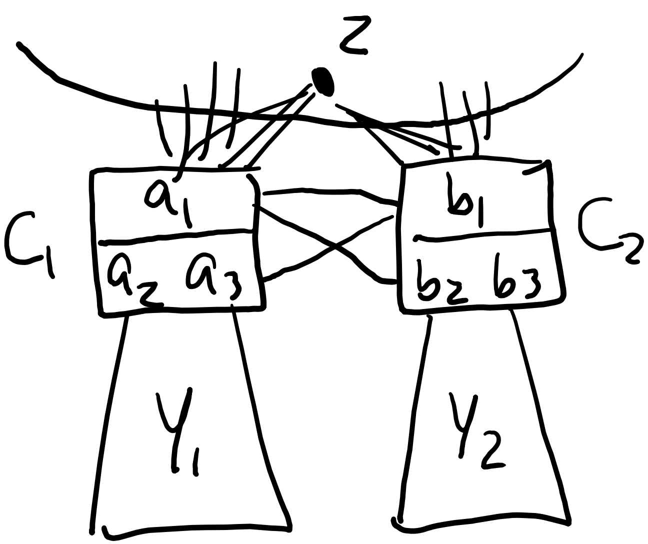



*Attempt 1 : embed using* $C_2$ *and* $Y_2$.
$C_1 \cup Y_1$ and $C_2 \cup Y_2$ have the same **accept** sets.



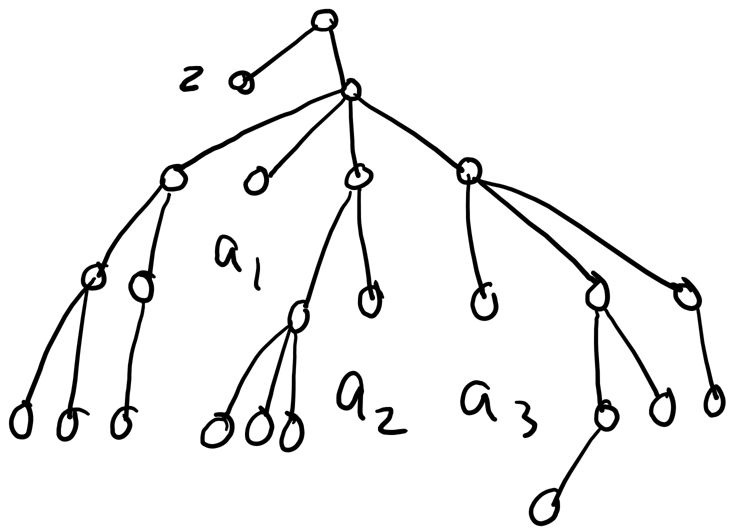


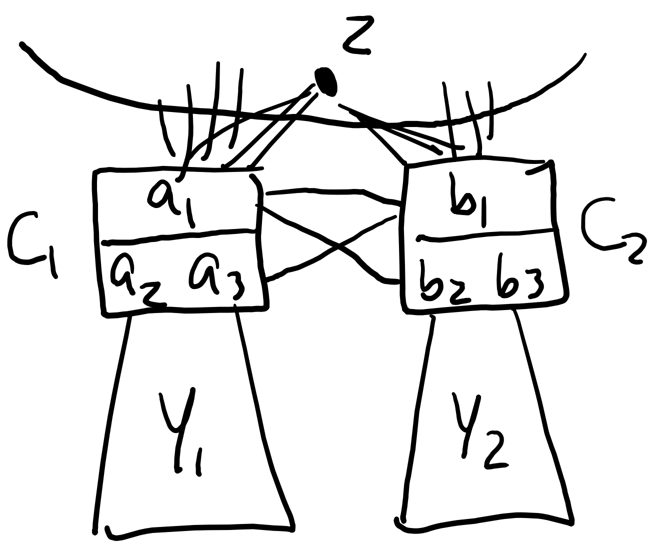

*Attempt 1 : embed using $C_2$ and $Y_2$.*
$C_1 \cup Y_1$ and $C_2 \cup Y_2$ have the same **accept** sets.



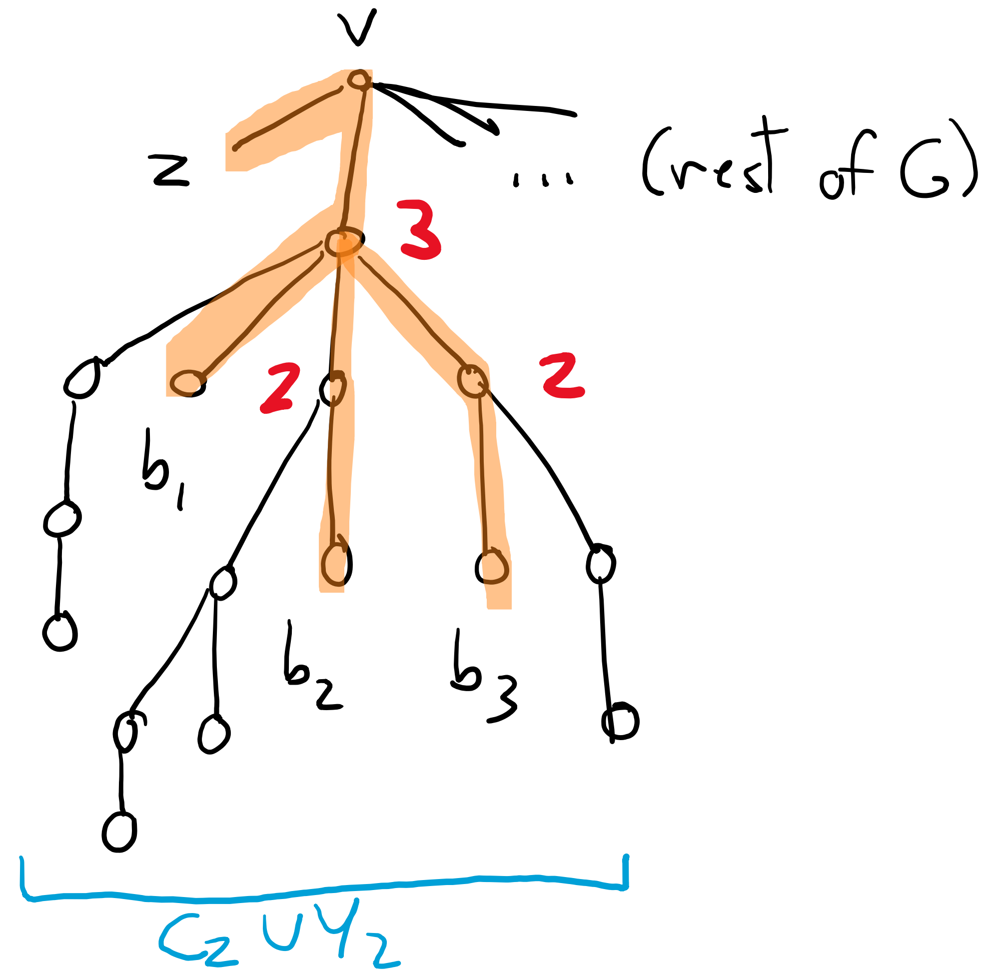

Highlighted = layer-encoding of T restricted to $C_2 \cup Y_2 \cup \{z\}$

$T_1$ : k-leaf root of $C_1 \cup Y_1 \cup \{z\}$

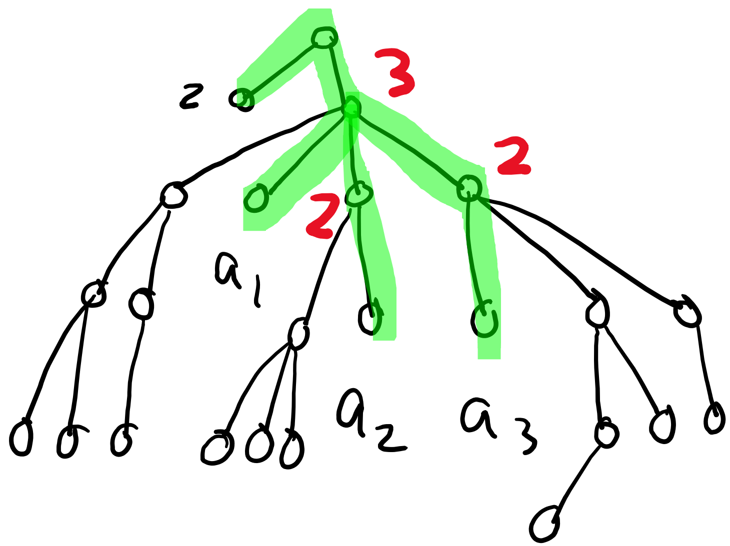


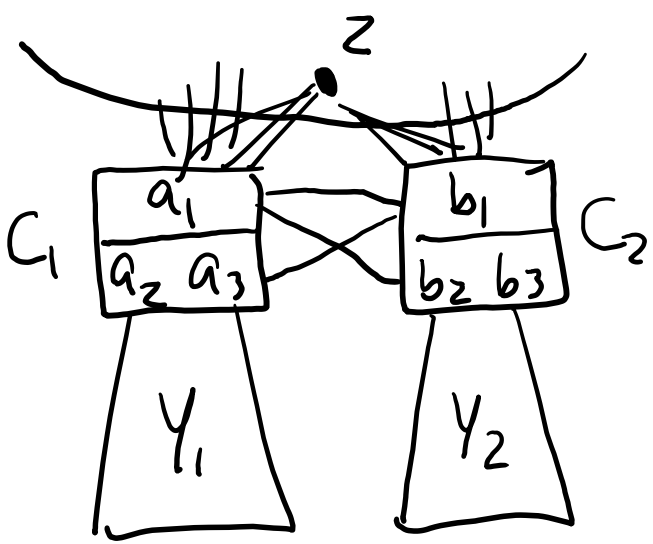




T : k-leaf root of $G - (C_1 \cup Y_1)$

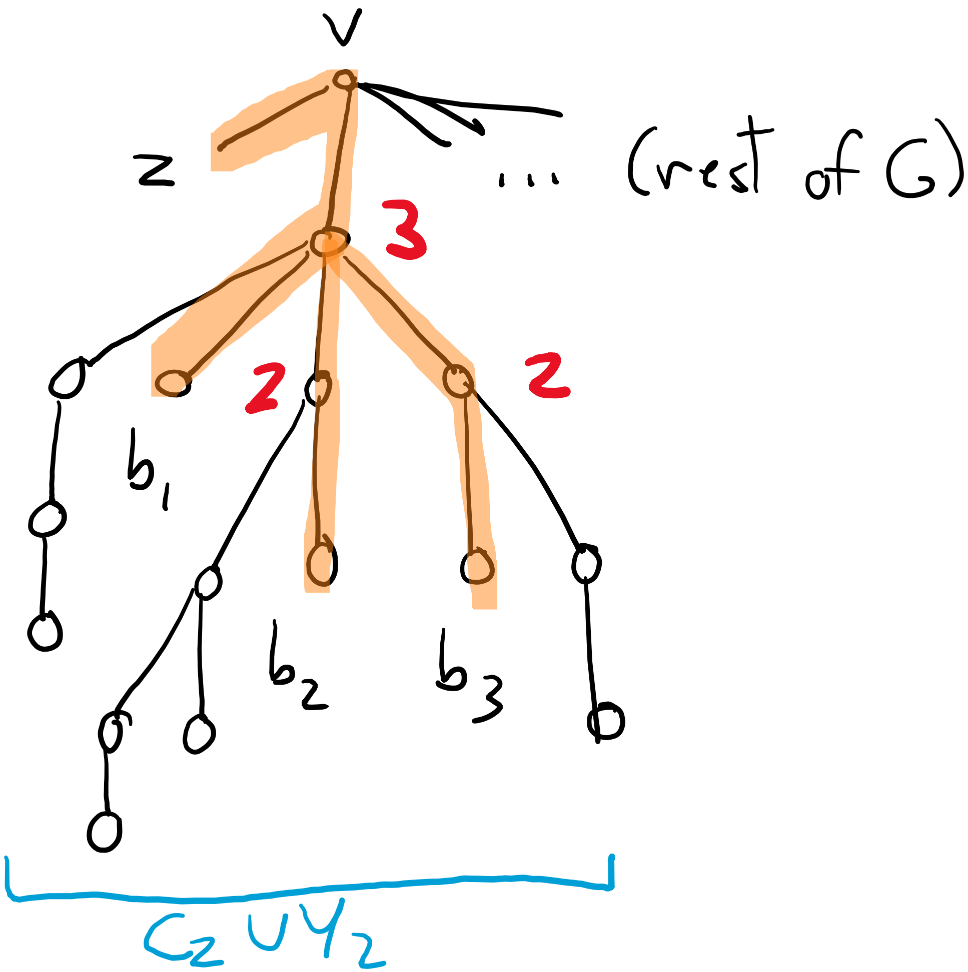


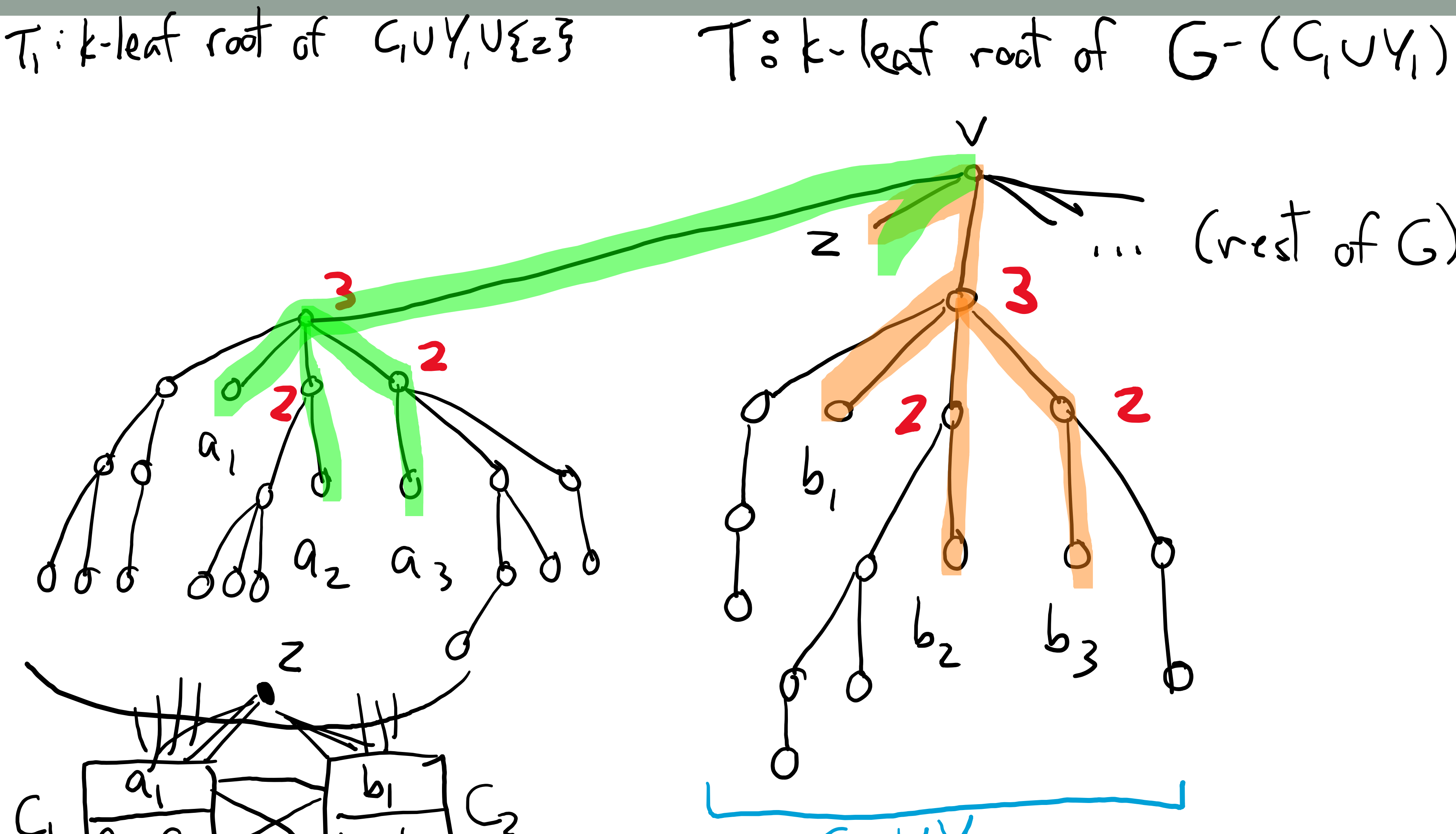
$T_1$ : k-leaf root of $C_1 \cup Y_1 \cup \{z\}$



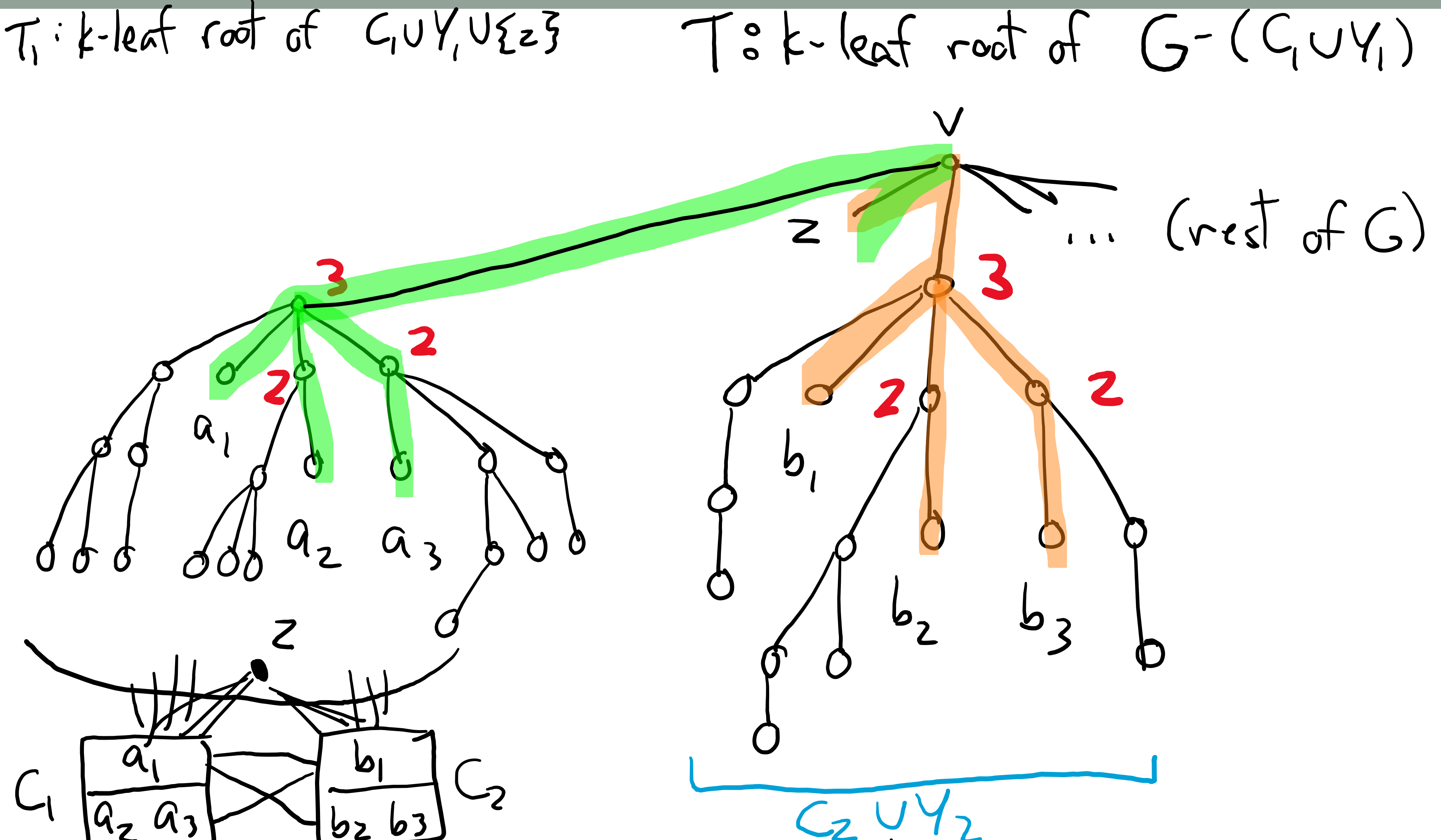




T : k-leaf root of $G-(C_1 \cup Y_1)$






$a_1$ and $b_1$ have the same neighbors in 'rest of $G$', and their distance to the 'rest of $G$' leaves is the same. Thus $a_1$ has the correct distances to 'rest of $G$'.
Same with $a_2/a_3$ and $b_2/b_3$.
The $Y_1$ leaves have the same distances as the $Y_2$ leaves, all is good.



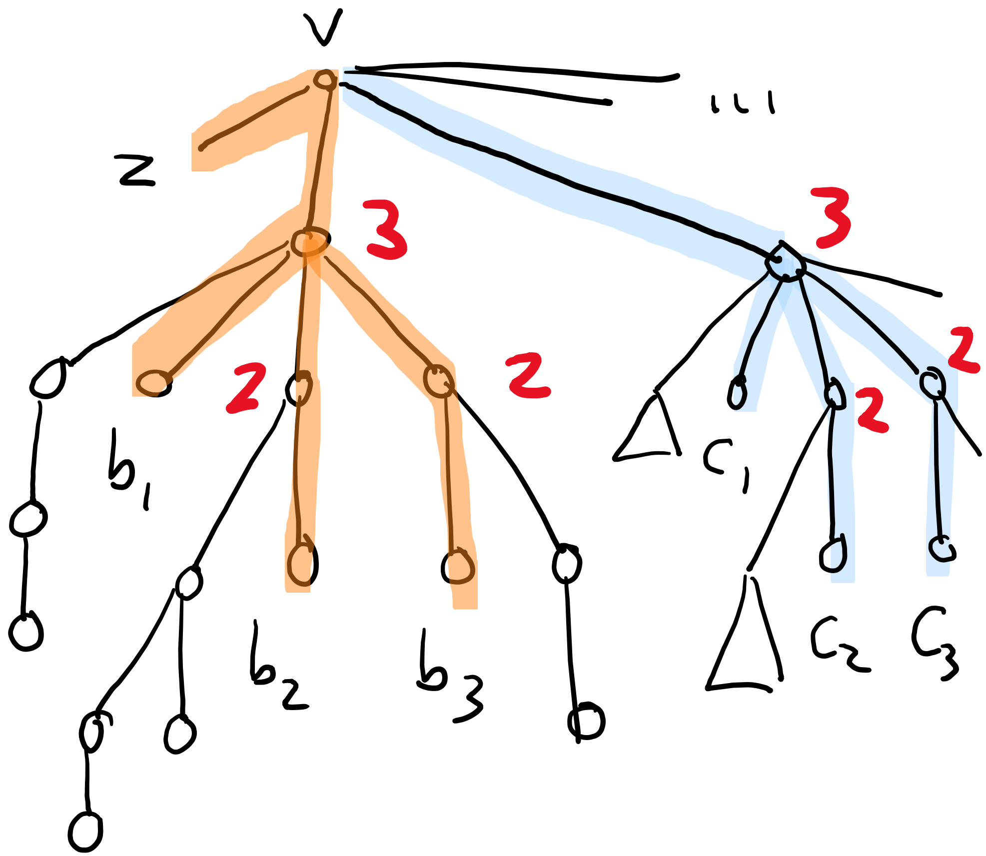

**PROBLEM** : are the distances relationships ok between members of $C_1$ and $C_2$?

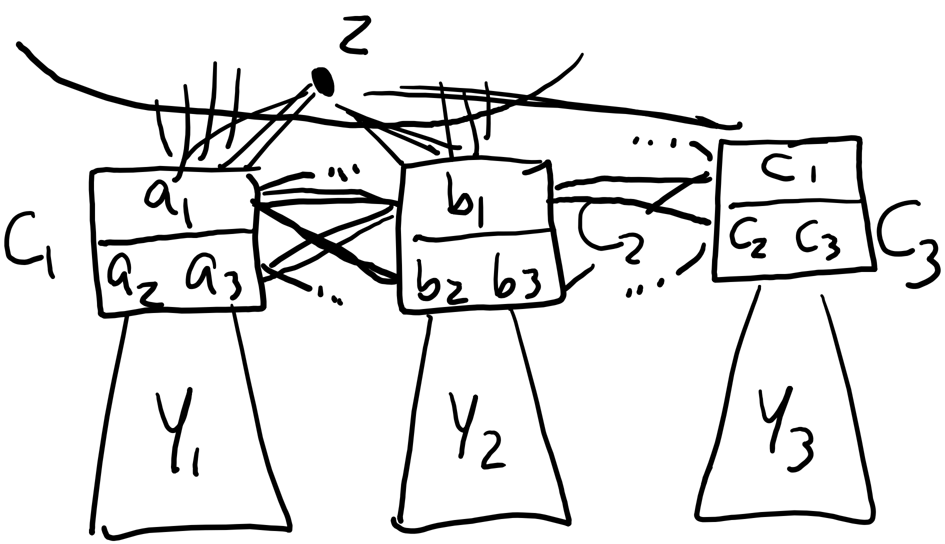
**No way to guarantee it!**

**Idea** : consider another similar homogeneous set $C_3 \cup Y_3$.

$T_1$: k-leaf root of $C_1 \cup Y_1 \cup \{z\}$

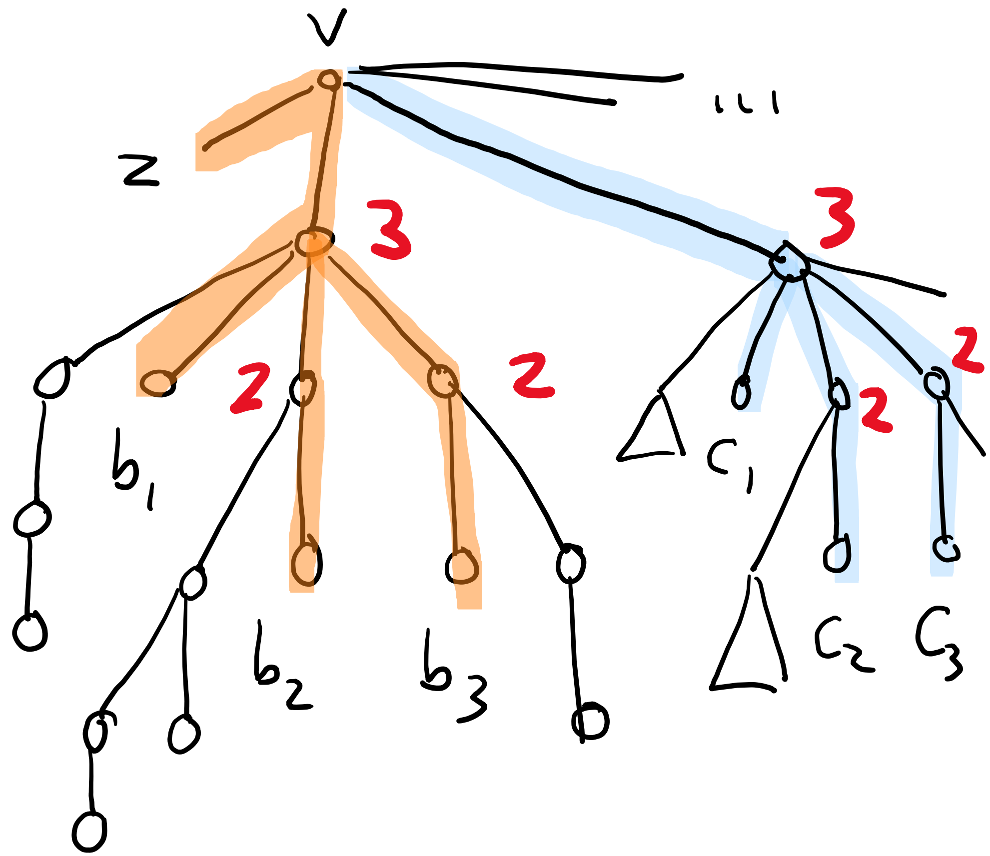
T: k-leaf root of $G-(C_1 \cup Y_1)$



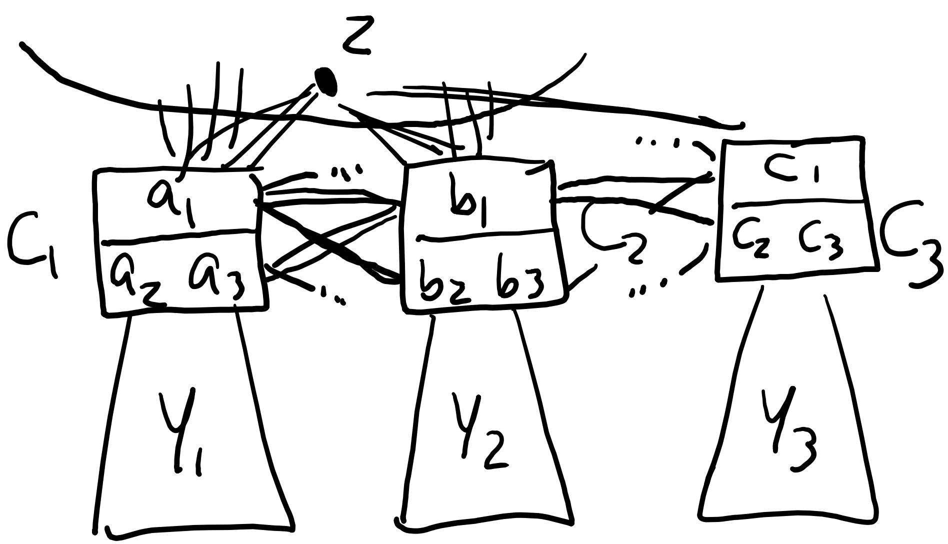



$T_1$: $k$-leaf root of $C_1 \cup Y_1 \cup \{z\}$

$T$: $k$-leaf root of $G - (C_1 \cup Y_1)$

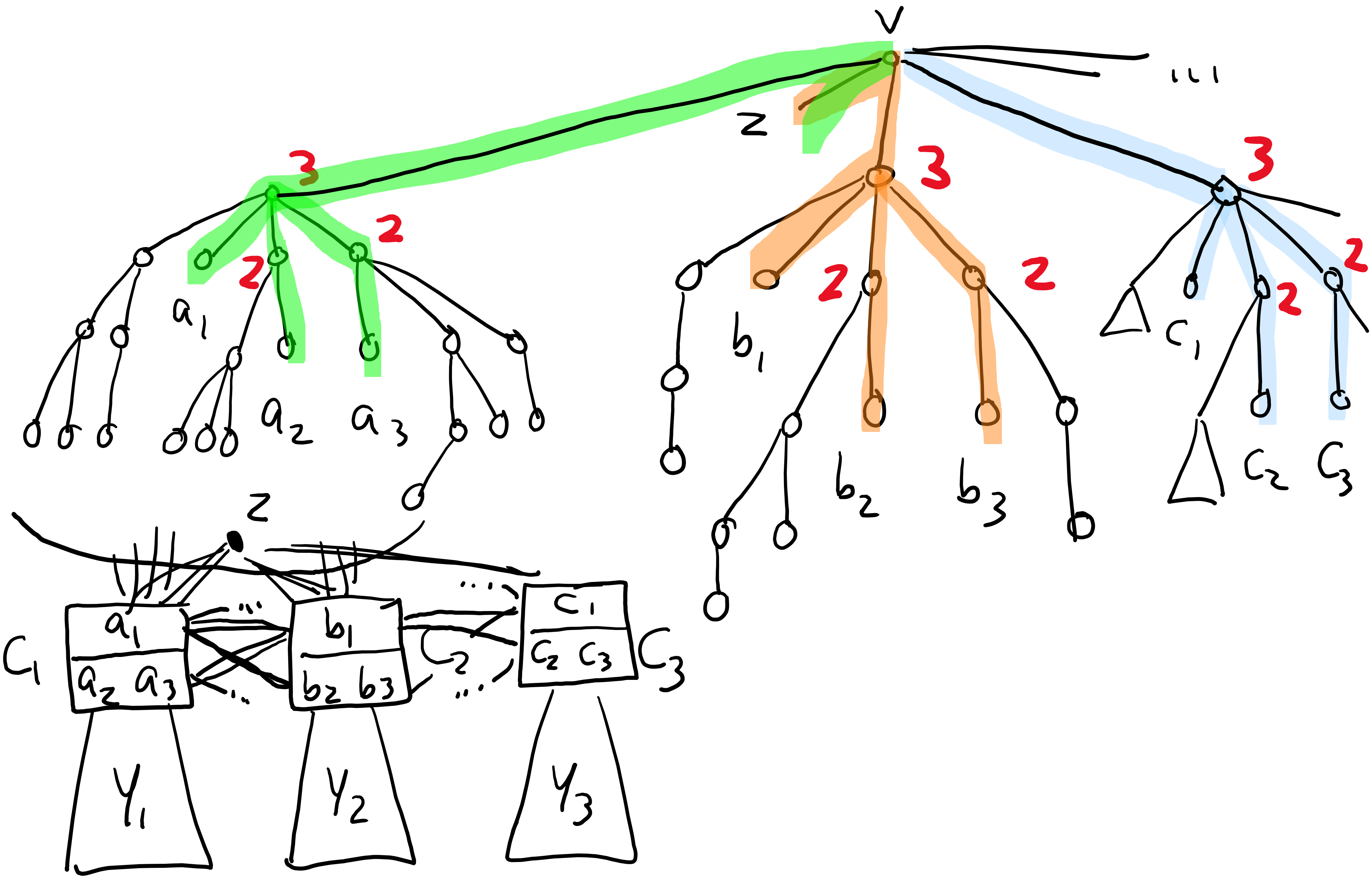



Because we have $3s(k)$ homogeneous subsets, two of them must be displayed with the same encoding in $T$.

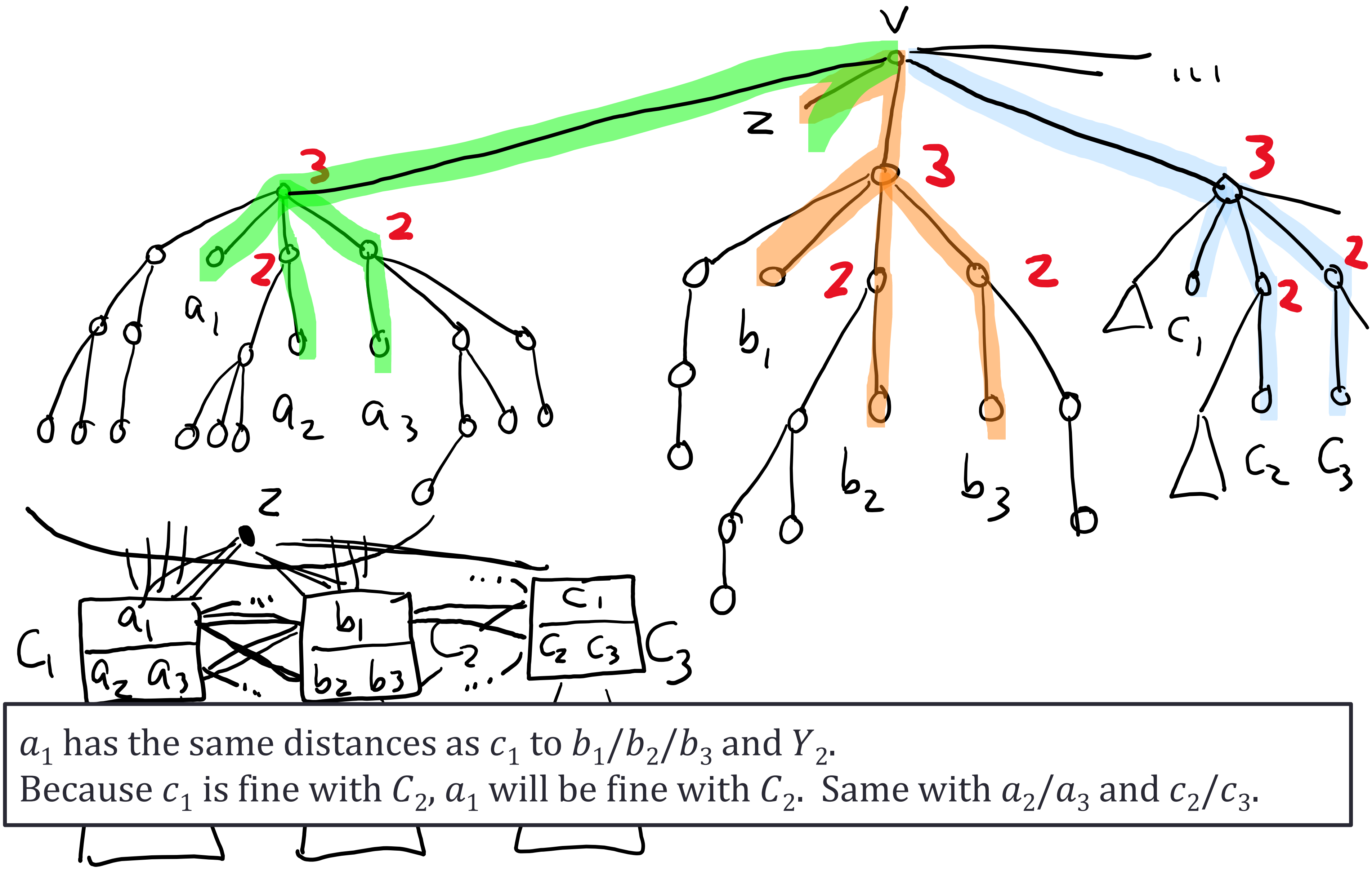


$T_1$: k-leaf root of $C_1 \cup Y_1 \cup \{z\}$

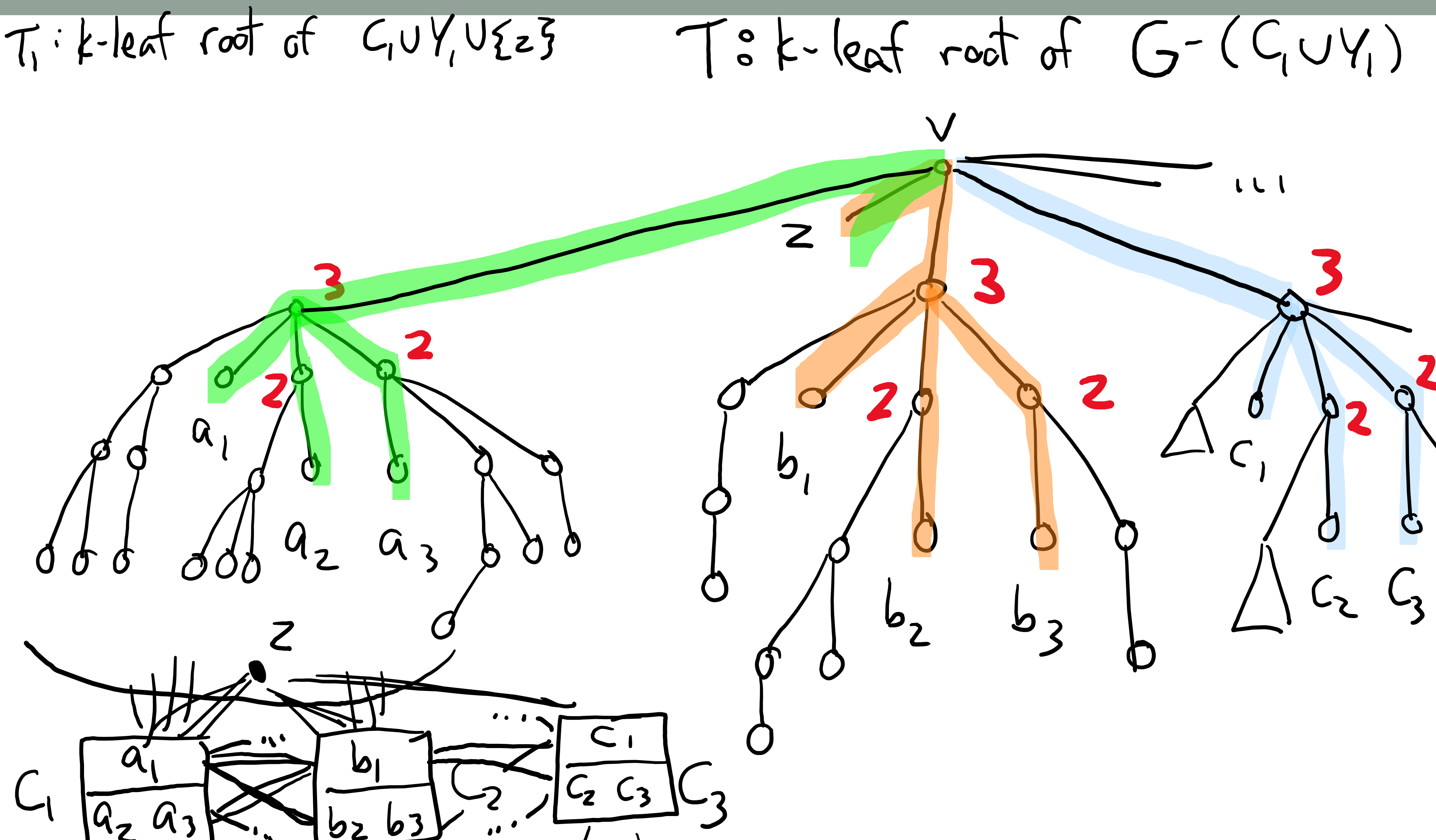
$T$: k-leaf root of $G-(C_1 \cup Y_1)$



$T_1$: k-leaf root of $C_1 \cup Y_1 \cup \{z\}$ T: k-leaf root of $G-(C_1 \cup Y_1)$



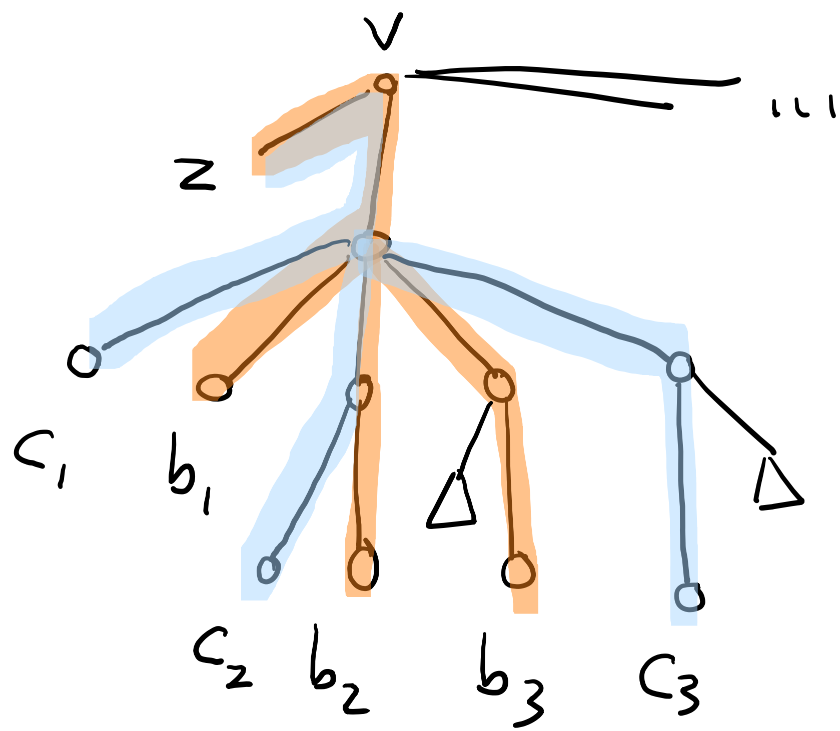

$a_1$ has the same distances as $c_1$ to $b_1/b_2/b_3$ and $Y_2$.
Because $c_1$ is fine with $C_2$, $a_1$ will be fine with $C_2$. Same with $a_2/a_3$ and $c_2/c_3$.

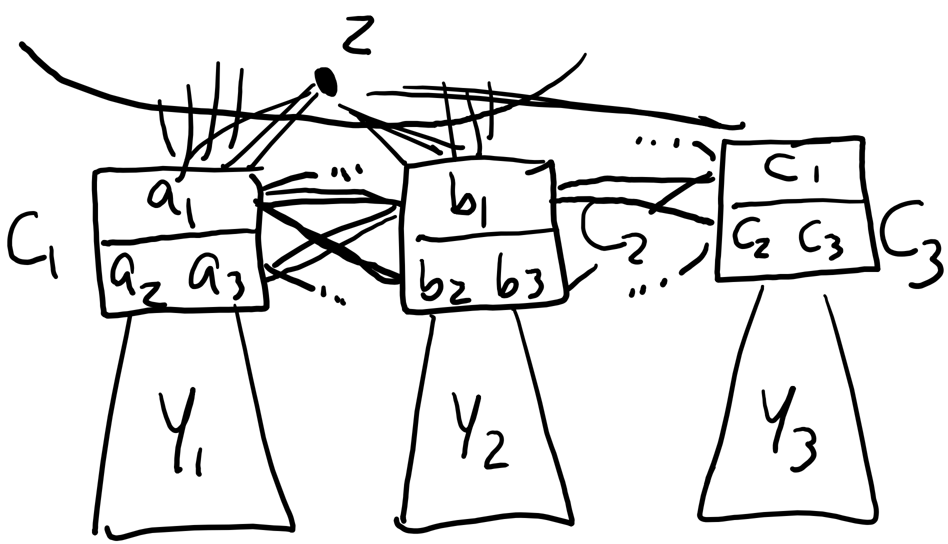



**PROBLEM** : no guarantee that in $T$ the $C_2$ and $C_3$ subtrees are well-separated like that.

$T_1$: k-leaf root of $C_1 \cup Y_1 \cup \{z\}$

$T$: k-leaf root of $G - (C_1 \cup Y_1)$

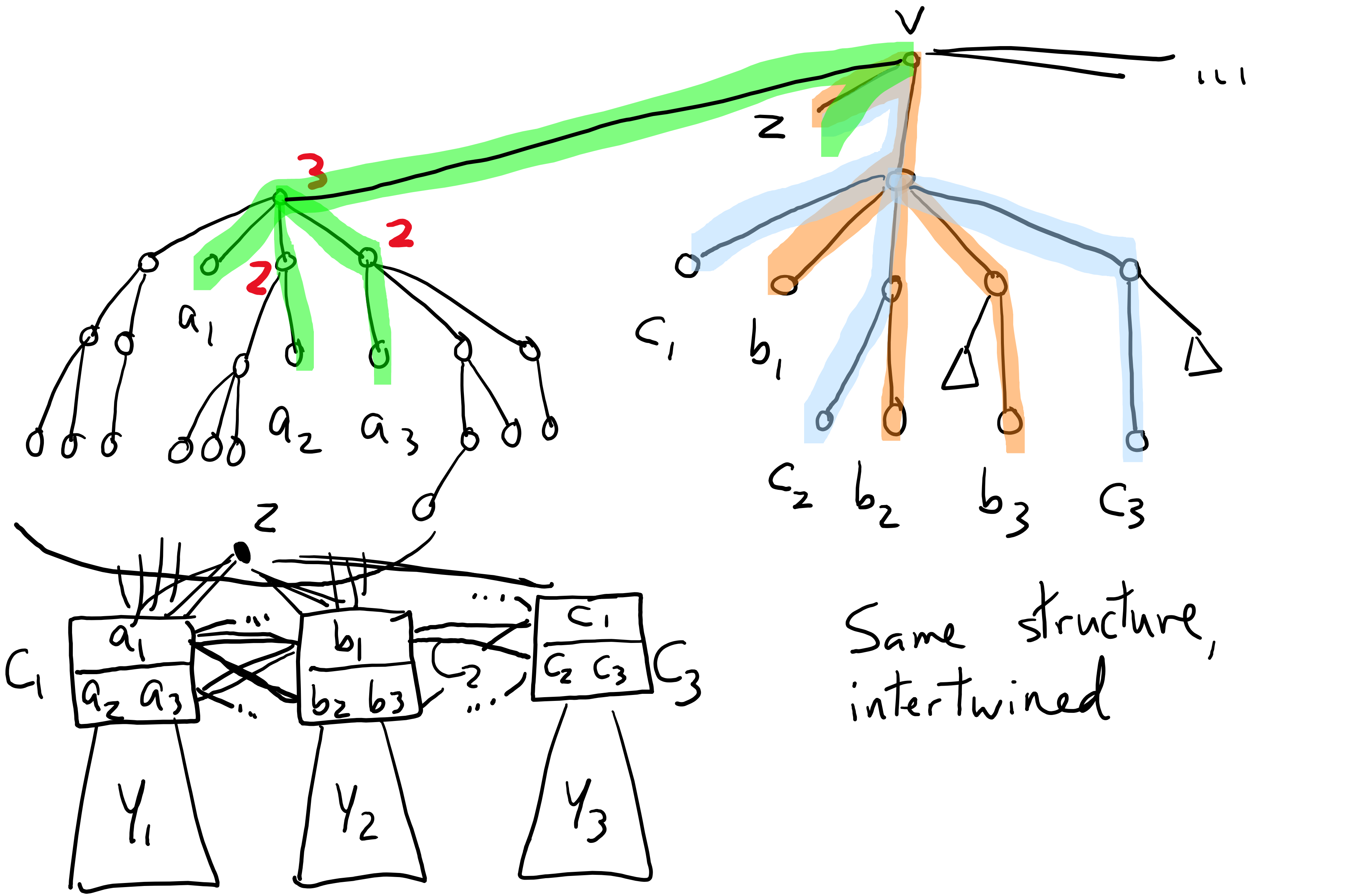


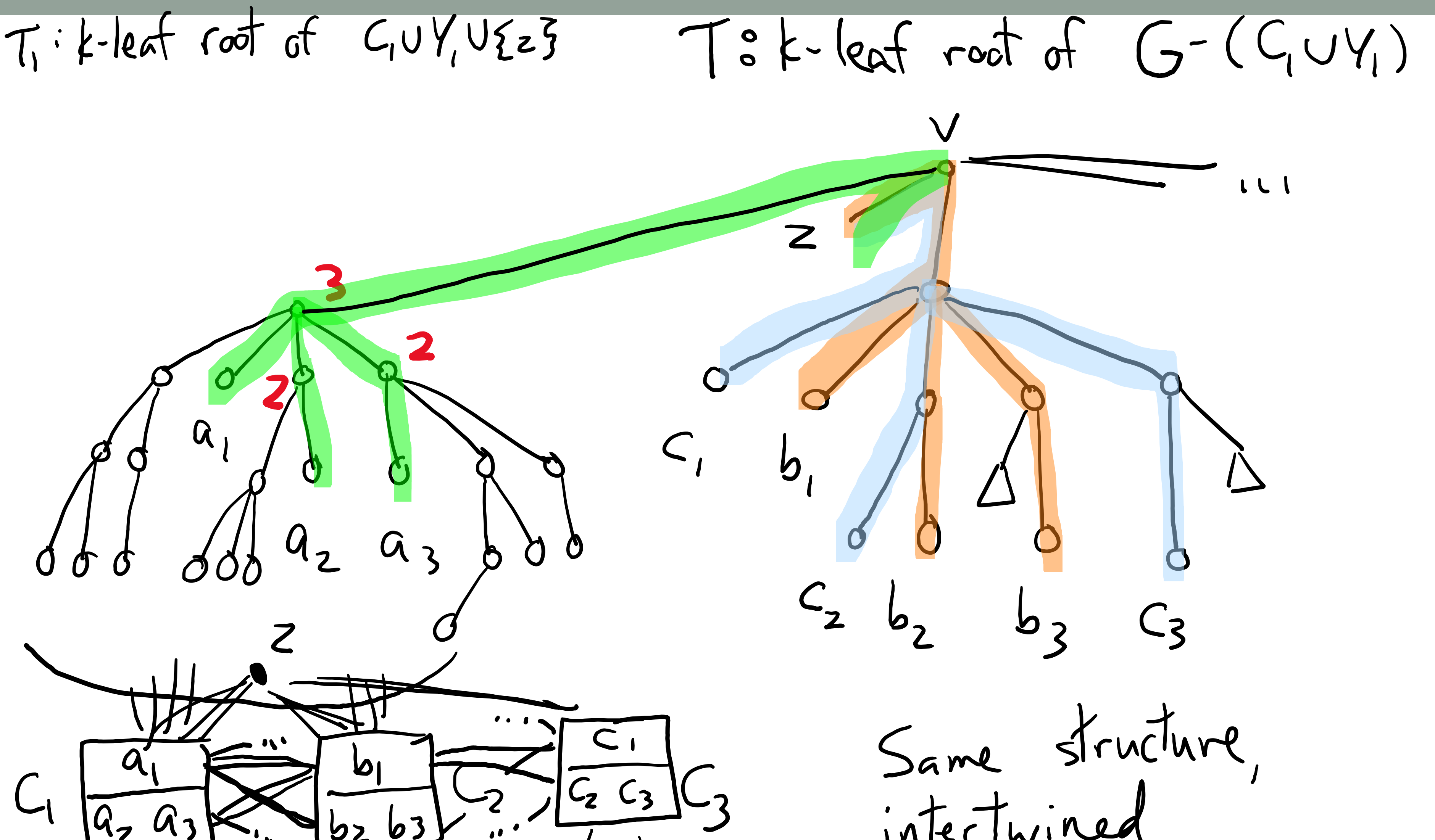

Same structure, intertwined

$T_1$: k-leaf root of $C_1 \cup Y_1 \cup \{z\}$ 

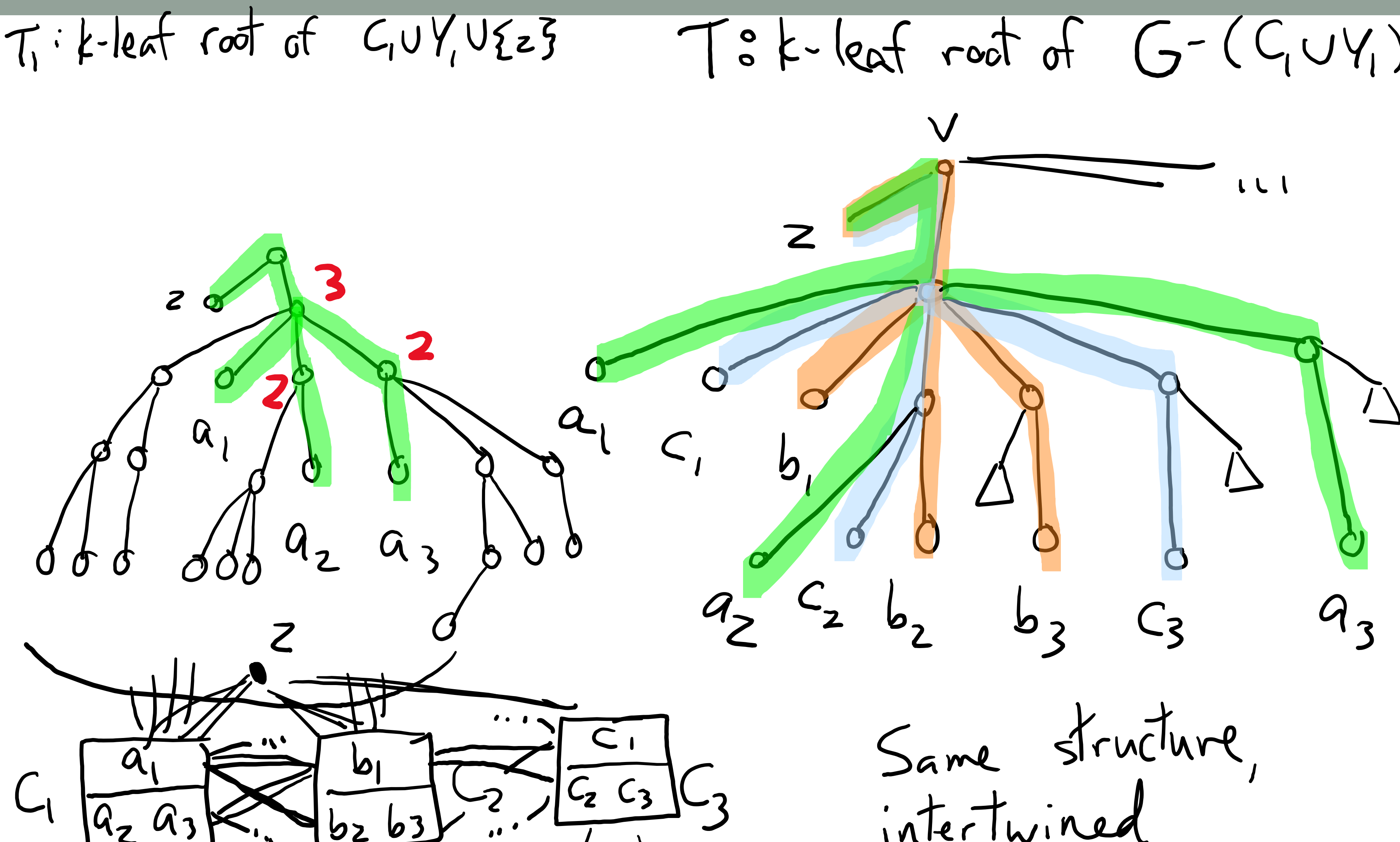
$T$: k-leaf root of $G-(C_1 \cup Y_1)$




Same structure, intertwined




**PROBLEM** : previous argument does not work. $a_1$ and $c_1$ don't have the same distances to $b_1/b_2/b_3$.




**SOLUTION** : embed $T_1$ into $T$ by "imitating" the structure of the two other subtrees. When **orange** and **blue** share a common edge, make embedded **green** share that common edge.

$insert(r(T_1^*), r(R))$ //initial call

**Function** *insert*$(t, r)$
    $//t \in V(T_1^*)$ is the node of $T_1^*$ we are inserting
    $//r \in V(R)$ is the node of $R$ we are inserting on
    **foreach** *child* $u \in ch_{T_1^*}(t) \setminus ch_{T_1}(t)$ **do**
        Insert the $T_1^*(u)$ subtree as a child of $r$
    **end**
    **foreach** *child* $u \in ch_{T_1}(t)$ **do**
        **if** $\exists w \in ch_{T_1}(t) \setminus \{u\}$ *such that* $sig_{\ell_1}(\mathcal{T}_1(w)) = sig_{\ell_1}(\mathcal{T}_1(u))$ **then**
            Insert the $T_1^*(u)$ subtree as a child of $r$
        **else**
            Let $u_2 \in ch_{T_2}(r)$ such that $sig_{\ell_2}(\mathcal{T}_2(u_2)) = sig_{\ell_1}(\mathcal{T}_1(u))$
            Let $u_3 \in ch_{T_3}(r)$ such that $sig_{\ell_3}(\mathcal{T}_3(u_3)) = sig_{\ell_1}(\mathcal{T}_1(u))$
            **if** $u_2 \neq u_3$ **then**
                Insert the $T_1^*(u)$ subtree as a child of $r$
            **else**
                **if** $u_2 \neq z$ **then**
                    Recursively call $insert(u, u_2)$
    **end**
**end**

# Bottomline

- If $G - C_1 \cup\ Y_1$ is a $k$-leaf power, then we can find enough similar + homogeneous subsets. With that, we can:
  1) find a $k$-leaf root $T$ of $G - C_1 \cup\ Y_1$
  2) find $C_2$ and $C_3$ such that their restrictions in $T$ yields the same layer-encoding (need enough homogeneous subsets to guarantee it).
  3) find a $k$-leaf root $T_1$ of $G[C_1 \cup\ Y_1\ \cup \{z\}]$ with that same encoding.
  4) embed $T_1$ into $T$ based on $C_2$ and $C_3$.
  5) all distance relationships will be the same as either $C_2$ or $C_3$ => all is good => $T$ is a $k$-leaf root of $G$.
     - That part requires more work than I showed…

# Step 4 : making an algorithm out of this

```
Function isLeafPower(G, k)
    d ← 3|S(k,3k)|2^{|S(k,3k)|};
    if G has maximum degree at most d^k then
        Check if G is a k-leaf power and return the result;
    foreach collection C = {C_1, ..., C_l} of disjoint subsets of V(G), with l = 3|S(k,3k)| and with each
      |C_i| ≤ d^k do
        Let G' = G − ⋃_{i∈[l]} C_i;
        Let X = {X_1, ..., X_t} be the connected components of G';
        Let z ∈ V(G') such that ⋃_{i∈[l]} C_i ⊆ N_G(z);
        if z does not exist then
            continue to the next C;
        Let X_z ∈ X such that z ∈ X_z;
        if some X_j ∈ X \ {X_z} has neighbors in two distinct C_i, C_j then
            continue to the next C;
        For i ∈ [l], let Y_i be the union of every X_j ∈ X \ X_z such that N_G(X_j) ⊆ C_i};
        if ∃i ∈ [l], G[C_i ∪ Y_i ∪ {z}] has maximum degree above d^k then
            continue to the next C;
        foreach set of layering functions L = {ℓ_1, ..., ℓ_l} do
            if S = (C, Y = {Y_1, ..., Y_d}, z, L) is a similar structure then
                foreach i ∈ [l] do
                    Compute accept(S, C_i);
                end
                if all the accept(S, C_i) are equal and non-empty then
                    return isLeafPower(G − (C_1 ∪ Y_1), k) ;
        end
    end
    return "Not a k-leaf power";
end
```

**Algorithm 2:** Deciding if a graph is a $k$-leaf power.

**Function** *isLeafPower*$(G, k)$

$d \leftarrow 3|S(k,3k)|2^{|S(k,3k)|}$;

**if** *$G$ has maximum degree at most $d^k$* **then**

Check if $G$ is a $k$-leaf power and return the result;

**foreach** *collection $\mathcal{C} = \{C_1, \ldots, C_l\}$ of disjoint subsets of $V(G)$, with $l = 3|S(k,3k)|$ and with each $|C_i| \leq d^k$* **do**

Let $G' = G - \bigcup_{i \in [l]} C_i$;

Let $X = \{X_1, \ldots, X_t\}$ be the connected components of $G'$;

Let $z \in V(G')$ such that $\bigcup_{i \in [l]} C_i \subseteq N_G(z)$;

**if** *$z$ does not exist* **then**

continue to the next $\mathcal{C}$;

Let $X_z \in X$ such that $z \in X_z$;

**if** *some $X_j \in X \setminus \{X_z\}$ has neighbors in two distinct $C_i, C_j$* **then**

continue to the next $\mathcal{C}$;

For $i \in [l]$, let $Y_i$ be the union of every $X_j \in X \setminus X_z$ such that $N_G(X_j) \subseteq C_i$};

**if** $\exists i \in [l], G[C_i \cup Y_i \cup \{z\}]$ *has maximum degree above $d^k$* **then**

continue to the next $\mathcal{C}$;

**foreach** *set of layering functions $\mathcal{L} = \{\ell_1, \ldots, \ell_l\}$* **do**

**if** *$\mathcal{S} = (\mathcal{C}, \mathcal{Y} = \{Y_1, \ldots, Y_d\}, z, \mathcal{L})$ is a similar structure* **then**

**foreach** $i \in [l]$ **do**

Compute *accept*$(\mathcal{S}, C_i)$;

**end**

**if** *all the accept$(\mathcal{S}, C_i)$ are equal and non-empty* **then**

return *isLeafPower*$(G - (C_1 \cup Y_1), k)$ ;

**end**

**end**

return "Not a $k$-leaf power";

**end**

**Algorithm 2:** Deciding if a graph is a $k$-leaf power.

**Function** *isLeafPower*$(G, k)$

  $d \leftarrow 3|S(k,3k)|2^{|S(k,3k)|}$;

  **if** *$G$ has maximum degree at most $d^k$* **then**

    Check if $G$ is a $k$-leaf power and return the result;

  **foreach** *collection $\mathcal{C} = \{C_1, \dots, C_l\}$ of disjoint subsets of $V(G)$, with $l = 3|S(k,3k)|$ and with each $|C_i| \leq d^k$* **do**

    Let $G' = G - \bigcup_{i \in [l]} C_i$;

    Let $X = \{X_1, \dots, X_t\}$ be the connected components of $G'$;

    Let $z \in V(G')$ such that $\bigcup_{i \in [l]} C_i \subseteq N_G(z)$;

    **if** *$z$ does not exist* **then**

      continue to the next $\mathcal{C}$;

    Let $X_z \in X$ such that $z \in X_z$;

    **if** *some $X_j \in X \setminus \{X_z\}$ has neighbors in two distinct $C_i, C_j$* **then**

      continue to the next $\mathcal{C}$;

    For $i \in [l]$, let $Y_i$ be the union of every $X_j \in X \setminus X_z$ such that $N_G(X_j) \subseteq C_i\}$;

    **if** *$\exists i \in [l], G[C_i \cup Y_i \cup \{z\}]$ has maximum degree above $d^k$* **then**

      continue to the next $\mathcal{C}$;

    **foreach** *set of layering functions $\mathcal{L} = \{\ell_1, \dots, \ell_l\}$* **do**

      **if** *$\mathcal{S} = (\mathcal{C}, \mathcal{Y} = \{Y_1, \dots, Y_d\}, z, \mathcal{L})$ is a similar structure* **then**

        **foreach** $i \in [l]$ **do**

          Compute *accept*$(\mathcal{S}, C_i)$;

        **end**

        **if** *all the accept$(\mathcal{S}, C_i)$ are equal and non-empty* **then**

          return *isLeafPower*$(G - (C_1 \cup Y_1), k)$ ;

    **end**

  **end**

  return "Not a $k$-leaf power";

**end**

**Algorithm 2:** Deciding if a graph is a $k$-leaf power.

- Computing $\boldsymbol{accept}(\boldsymbol{C_i} \cup \boldsymbol{Yi})$
- Recall that $G[Ci \cup Yi \cup \{z\}]$ has maximum degree at most $d^k$, where here $d$ is that power tower function.
- Also, $G[Ci \cup Yi \cup \{z\}]$ is chordal (assuming it is a $k$-leaf power).
- Hence, $G[Ci \cup Yi \cup \{z\}]$ has treewidth at most $d^k$.
- The list of layer-encoded $k$-leaf roots can be computed using dynamic programming on the tree decomposition.
  - See paper...

# What's next?

# What's next?

**Open problem 1**

Can the ridiculous $n^{f(k)}$ complexity be improved? Or is the power tower behavior necessary in the exponent?

**Open problem 2**

Is $k$-leaf power recognition FPT in $k$? i.e. $f(k) * poly(n)$ algorithm?

**Open problem 3**

Can leaf powers be recognized in polynomial time? Techniques from here usable? (probably not)

**Other questions**

- Techniques applicable to other tree-definable graph classes? (e.g. PCGs)
- Graph-theoretical characterization of $k$-leaf powers?
  - ad hoc analysis for low degree, higher degree = redundancy

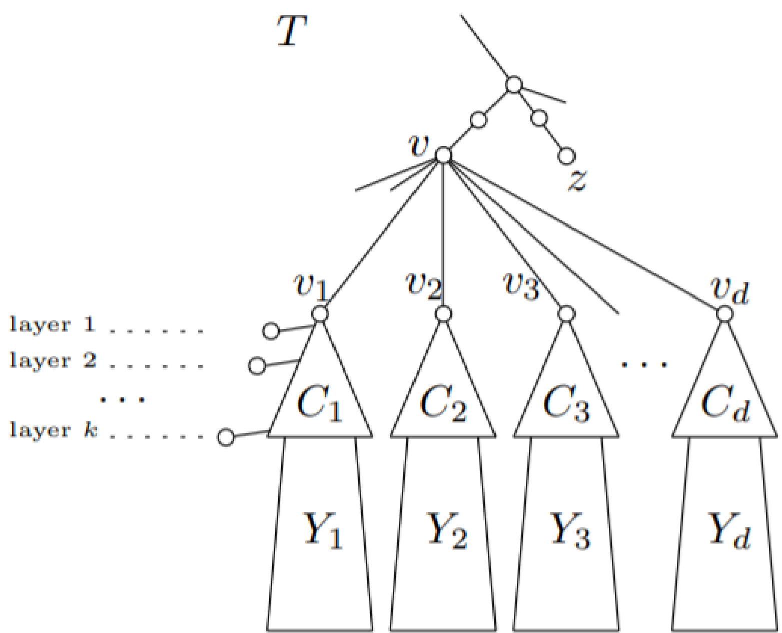
**Theorem**
There is $f$ such that if $G$ admits a $k$-leaf root of max degree $d > f(k)$, then $G$ contains a subset $C$ of vertices such that ***$G$* is a *$k$*-leaf power if and only if *$G - C$* is a *$k$*-leaf power**.
Moreover, $C$ can be found in time $O(n^{f(k)})$ if it exists.

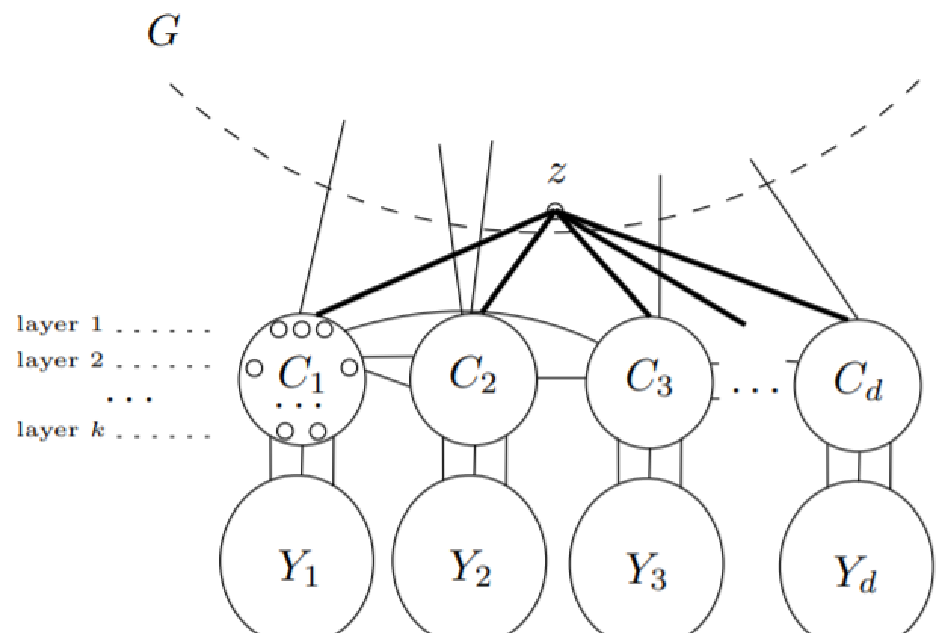
This is proved as follows:

1. Show that if a k-leaf root has degree $> d$, one can find subsets C1 U Y1, …, Cd U Yd, such that Ci cuts Yi from the rest of G.
2. Moreover, C1 U C2 U … U Cd can be partitioned into layers that have the same neighborhood in G – (C1 U Y1 U … U Cd U Yd).
3. Moreover again, G[C1 U Y1] admits the same set of encoded k-leaf roots as some G[Ci U Yi] (to be defined).
4. Find a k-leaf root T of G – (C1 U Y1). If none exists, we are done. Otherwise, look at how Ci U Yi is organized in T. By (3), C1 U Y1 allows the same k-leaf root organization. We embed C1 U Y1 into T by mimicking C2 U Y2. By (2), this works.

This is proved as follows:

1. Show that if a k-leaf root has degree $> d$, one can find subsets C1 U Y1, …, Cd U Yd, such that Ci cuts Yi from the rest of G.
2. Moreover, C1 U C2 U … U Cd can be partitioned into layers that have the same neighborhood in G – (C1 U Y1 U … U Cd U Yd).
3. If d is large, some G[Ci U Yi] and G[Cj U Yj] admit the same set of encoded k-leaf roots (to be defined).
4. Find a k-leaf root T of G – (Ci U Yi). Look at how Cj U Yj is organized in T. By (3), Ci U Yi allows the same k-leaf root organization. We embed Ci U Yi into T by mimicking Cj U Yj. By (2), this works.

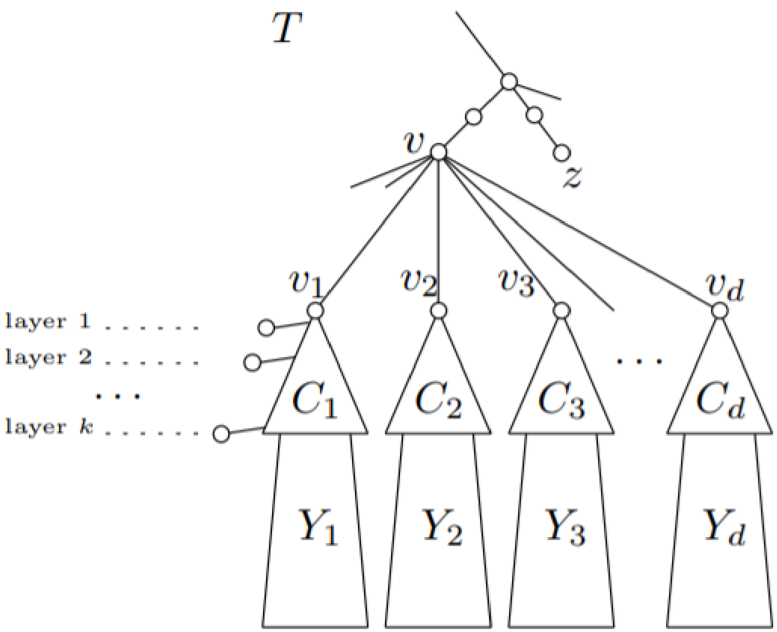


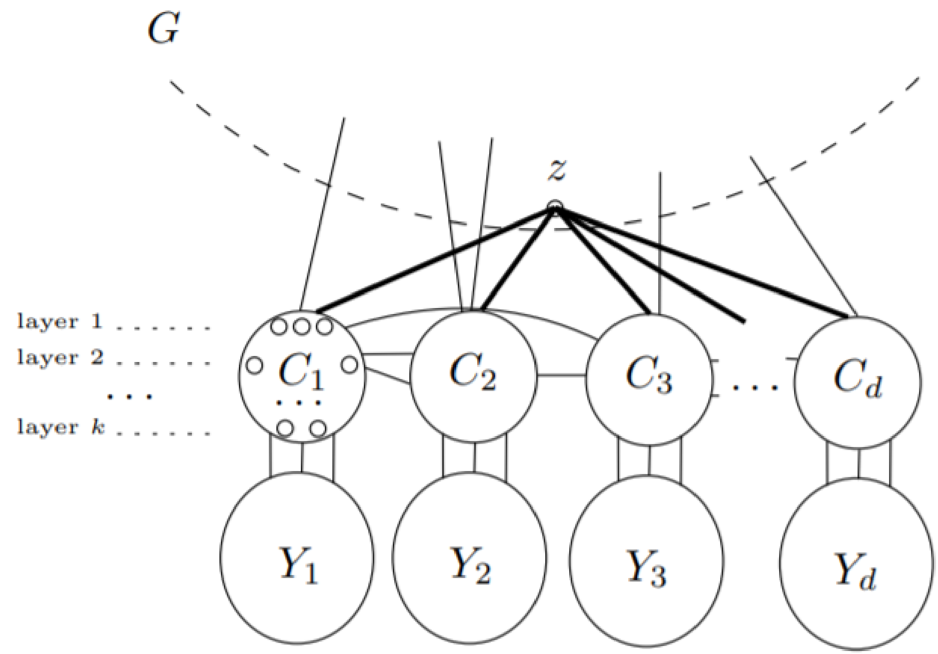



# $k$-leaf roots with high degree

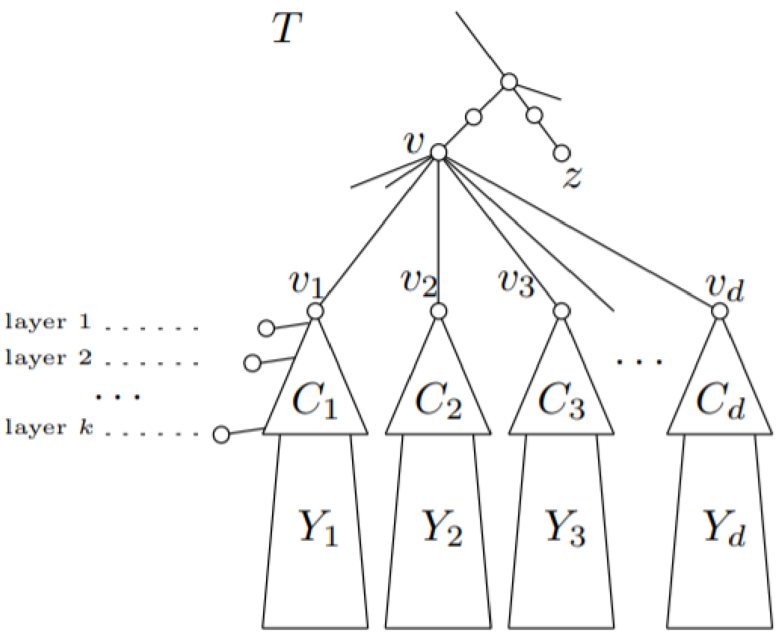
**Theorem**
There is $f$ such that if $G$ admits a $k$-leaf root of max degree $d > f(k)$, then $G$ contains a subset $C$ of vertices such that **G is a *k*-leaf power if and only if $G - C$ is a *k*-leaf power**.
Moreover, $C$ can be found in time $O(n^{f(k)})$ if it exists.

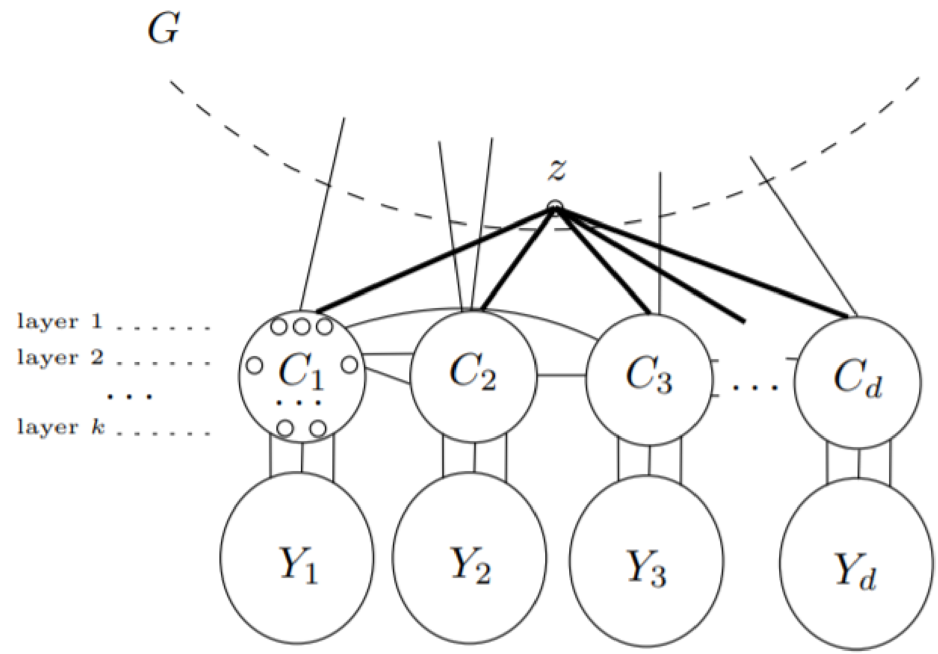
- T = leaf root of G
- v = lowest max of degree >d
- z = closest leaf to v
- Ci = subtrees at distance <= k from v
- Layer j = leaves at distance j from v



- Of course, we don't have $T$. Still, by brute-force we can find the $C_i$'s and $Y_i$'s that satisfy the cutset, size and layering properties. This is feasible since the $C_i$'s have bounded size.



**3.1 Similar structures** A *similar structure* of a graph $G$ is a tuple $\mathcal{S} = (\mathcal{C}, \mathcal{Y}, z, \mathcal{L})$ where:

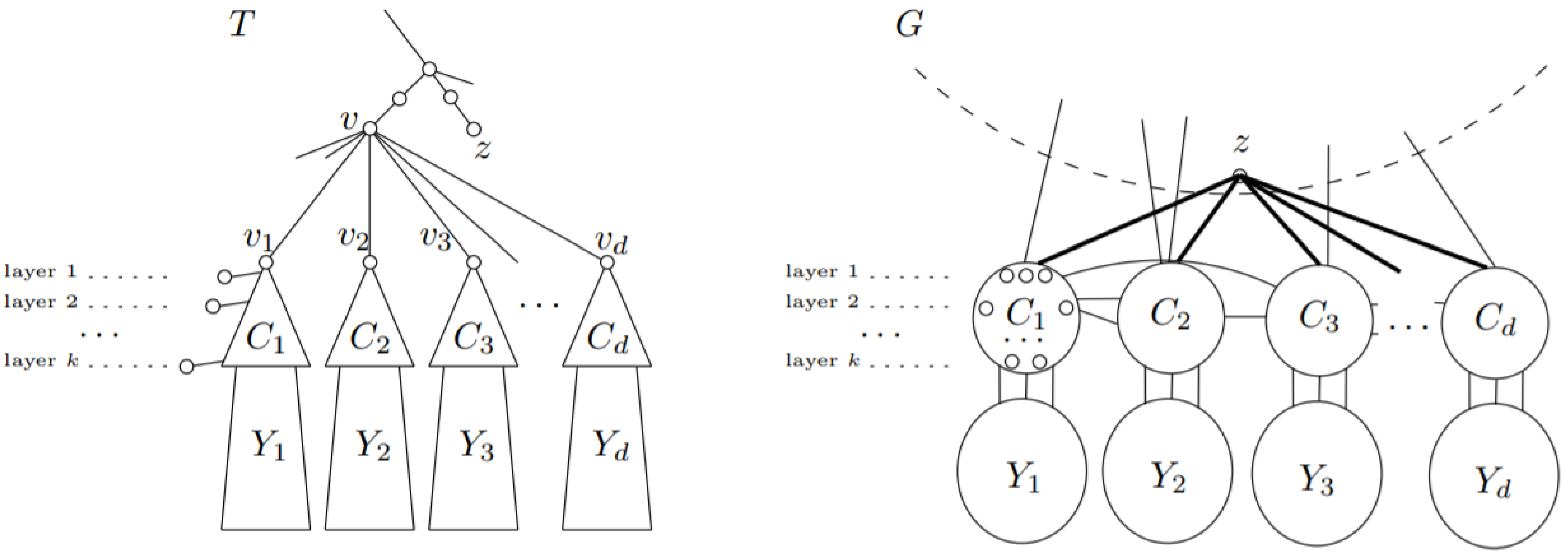
- $\mathcal{C} = \{C_1, \ldots, C_d\}$ is a collection of $d \geq 2$ pairwise disjoint, non-empty subsets of vertices of $G$;
- $\mathcal{Y} = \{Y_1, \ldots, Y_d\}$ is a collection of pairwise disjoint subsets of vertices of $G$, some of which are possibly empty. Also, $C_i \cap Y_j = \emptyset$ for any $i, j \in [d]$;
- $z \in V(G)$ and does not belong to any subset of $\mathcal{C}$ or $\mathcal{Y}$;
- $\mathcal{L} = \{\ell_1, \ldots, \ell_d\}$ is a set of functions where, for each $i \in [d]$, we have $\ell_i : C_i \cup \{z\} \to \{0, 1, \ldots, k\}$. The functions in $\mathcal{L}$ are called *layering functions.*

Additionally, $\mathcal{S}$ must satisfy several conditions. Let us denote $C^* = \bigcup_{i \in [d]} C_i$. Let $X = \{X_1, \ldots, X_t\}$ be the connected components of $G - C^*$. For each $i \in [d]$, denote $X^{(i)} = \{X_j \in X : N_G(X_j) \subseteq C_i\}$, i.e. the components that have neighbors only in $C_i$.

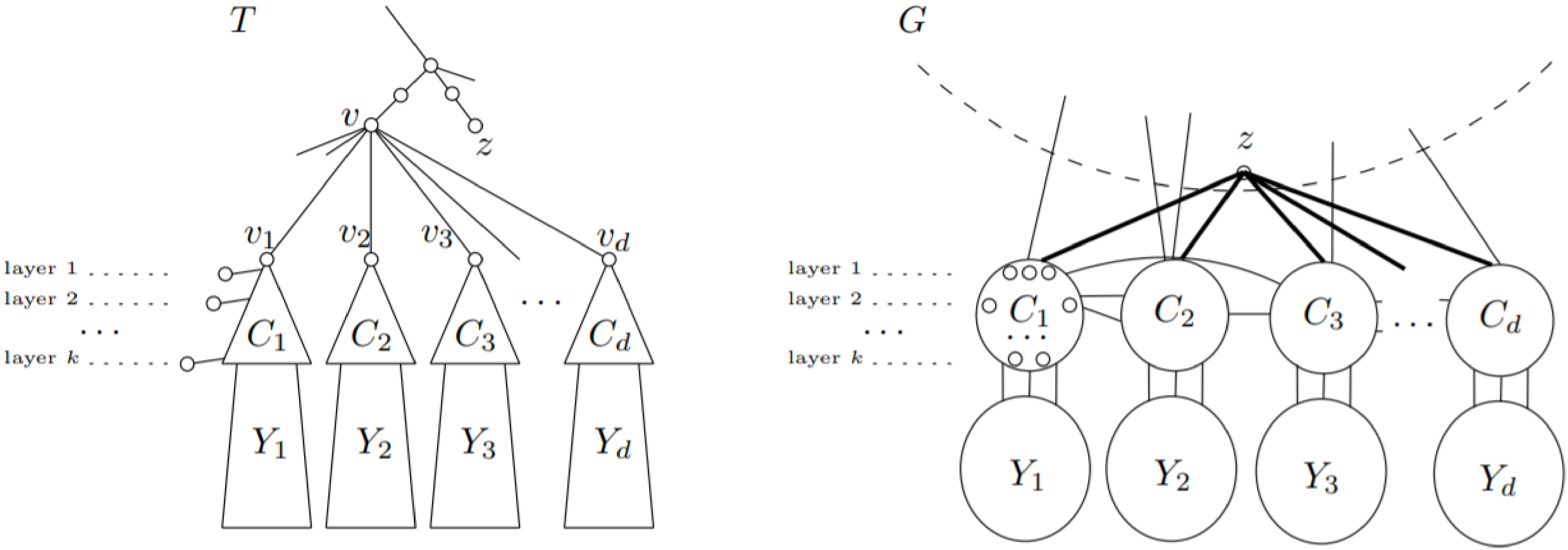
Then all the following conditions must hold:

1. for each $i \in [d]$, $Y_i = \bigcup_{X_j \in X^{(i)}} X_j$ ($Y_i = \emptyset$ is possible);
2. there is exactly one connected component $X_z \in X$ such that for all $i \in [d]$, $N_G(X_z) \cap C_i \neq \emptyset$. Moreover, $z \in X_z$ and $C^* \subseteq N_G(z)$;
3. for all $X_j \in X \setminus \{X_z\}$, $X_j \subseteq Y_i$ for some $i \in [d]$. In particular, $X_z$ is the only connected component of $G - C^*$ with neighbors in two or more $C_i$'s;
4. the layering functions $\mathcal{L}$ satisfy the following:
   (a) for each $i \in [d]$, $\ell_i(z) = 0$. Moreover, $\ell_i(x) > 0$ for any $x \in C_i$;
   (b) for any $i, j \in [d]$ and any $x \in C_i, y \in C_j$, $\ell_i(x) = \ell_j(y)$ implies $N_G(x) \setminus (C_i \cup Y_i \cup C_j \cup Y_j) = N_G(y) \setminus (C_i \cup Y_i \cup C_j \cup Y_j)$. Note that this includes the case $i = j$;
   (c) for any $i, j \in [d]$ and any $x \in C_i, y \in C_j$, $\ell_i(x) + \ell_j(y) \leq k$ implies $xy \in E(G)$. Note that this includes the case $i = j$.
   (d) for any *two distinct* $i, j \in [d]$ and any $x \in C_i, y \in C_j$, $\ell_i(x) + \ell_j(y) > k$ implies $xy \notin E(G)$. Note that this does *not* include the case $i = j$

- Of course, we don't have $T$. Still, by brute-force we can find the $C_i$'s and $Y_i$'s that satisfy the cutset, size and layering properties. This is feasible since the $C_i$'s have bounded size.



- Of course, we don't have $T$. Still, by brute-force we can find the $C_i$'s and $Y_i$'s that satisfy the cutset, size and layering properties. This is feasible since the $C_i$'s have bounded size.
- Look at the k-leaf roots of each G[Ci U Yi].
- WANT : two G[Ci U Yi] and G[Cj U Yj] that admit the same set of layer-encoded k-leaf roots.



- WANT : two G[Ci U Yi] and G[Cj U Yj] that admit the same set of layer-encoded k-leaf roots.