Tree Containment with Polytomies

Mathias Weller

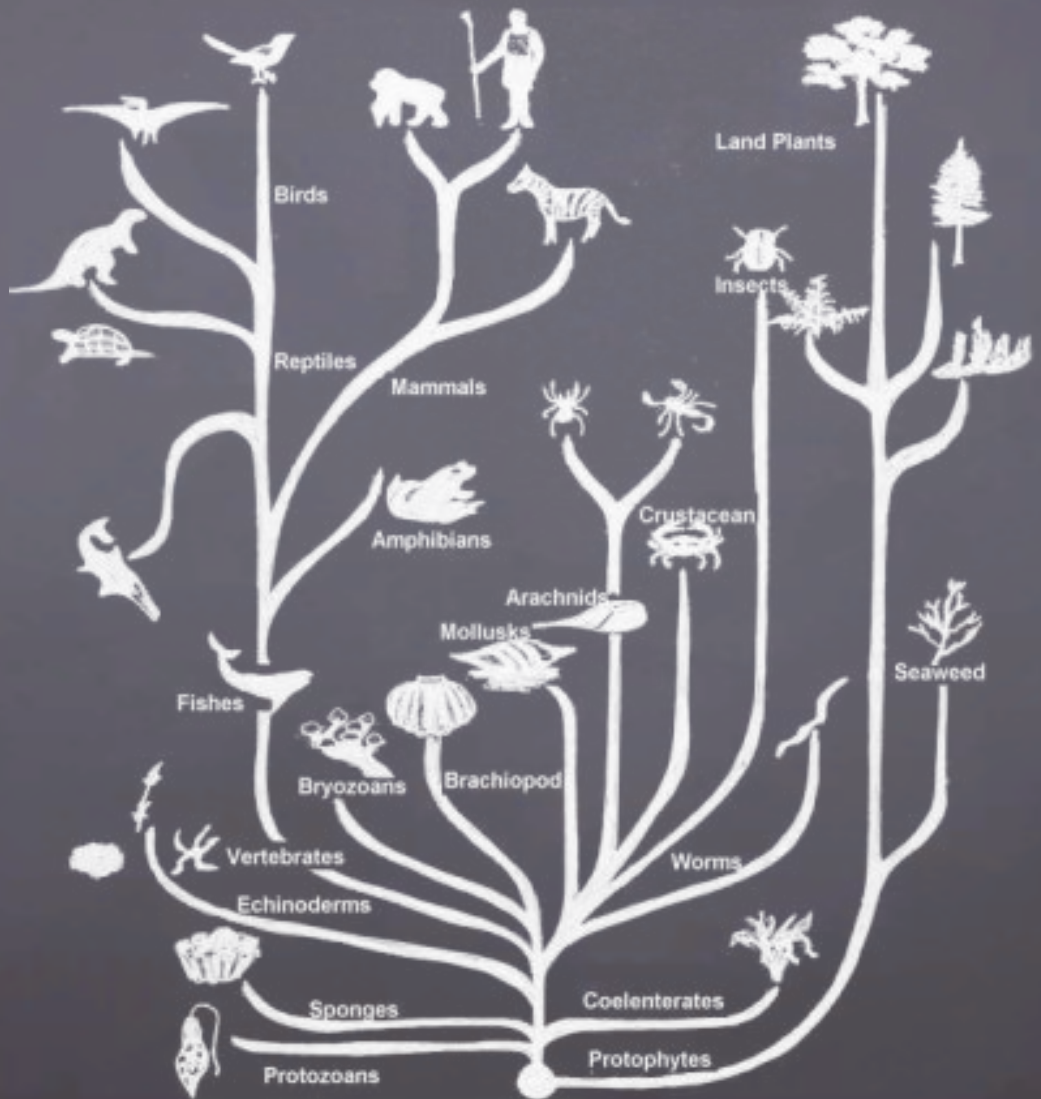
CNRS, LIGM, Paris EST, Marne-la-Vallée

Dagstuhl Seminar:
Algorithms & Complexity in Phylogenetics

Motivation: Tracing Evolution

Evolutionary Tree

= rooted trees with leaf labels ("trees" from now on)



Motivation: Tracing Evolution

Evolutionary Tree

= rooted trees with leaf labels ("trees" from now on)

But

reticulation events (hybridization, horiz. transfer, ...)

$\rightsquigarrow$ evolution is a network

Motivation: Tracing Evolution

Evolutionary Tree

= rooted trees with leaf labels ("trees" from now on)

But

reticulation events (hybridization, horiz. transfer, ...)
$\rightsquigarrow$ evolution is a network

Problem

Given a network N and a tree T, is N compatible with T?

Motivation: Tracing Evolution

Evolutionary Tree

= rooted trees with leaf labels ("trees" from now on)

But

reticulation events (hybridization, horiz. transfer, ...)
$\rightsquigarrow$ evolution is a network

Tree Containment

Given a network N and a tree T, does N display T?

Motivation: Tracing Evolution

Evolutionary Tree

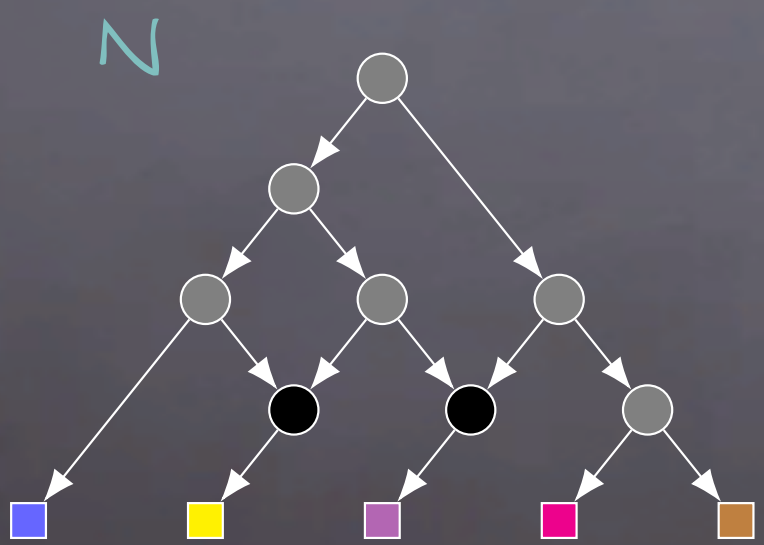
= rooted trees with leaf labels ("trees" from now on)

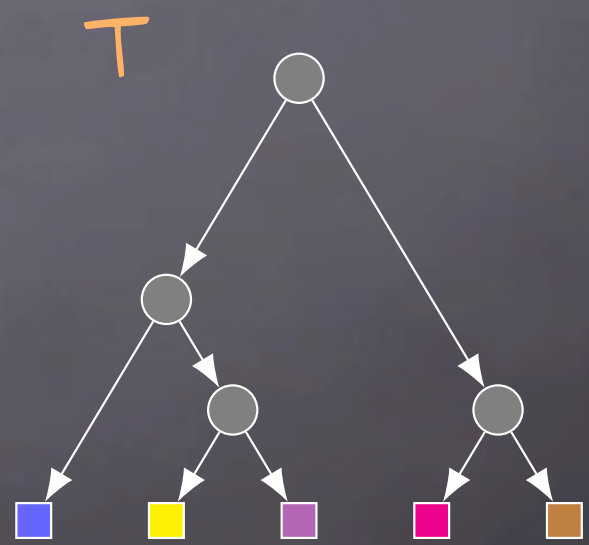
But

reticulation events (hybridization, horiz. transfer, ...)
$\rightsquigarrow$ evolution is a network

Tree Containment

Given a network N and a tree T, does N display T?





Motivation: Tracing Evolution

Evolutionary Tree

= rooted trees with leaf labels ("trees" from now on)

But

reticulation events (hybridization, horiz. transfer, ...)
$\rightsquigarrow$ evolution is a network

Tree Containment

Given a network N and a tree T, does N display T?

Polytomies: Hard & Soft

Polytomies: Hard & Soft

Polytomies: Hard ≠ Soft

Hard Polytomies

special events in evolution causing high fan-out

⇝ should be represented as-is

(display = contains subdivision)

Polytomies: Hard ≠ Soft

Hard Polytomies

special events in evolution causing high fan-out

⇝ should be represented as-is

(display = contains subdivision)

Polytomies: Hard ≠ Soft

Hard Polytomies

special events in evolution causing high fan-out

⇝ should be represented as-is

(display = contains subdivision)

Polytomies: Hard & Soft

Hard Polytomies

special events in evolution causing high fan-out

⇝ should be represented as-is

(display = contains subdivision)

Soft Polytomies

noisy/incomplete data

⇝ more liberal representation

(display = "Binary resolutions" display)

Polytomies: Hard ≠ Soft

Hard Polytomies

special events in evolution causing high fan-out

↝ should be represented as-is

(display = contains subdivision)

Soft Polytomies

noisy/incomplete data

↝ more liberal representation

(display = "binary resolutions" display)

= can be contracted to N (or T)

Polytomies: Hard ≠ Soft

Hard Polytomies

special events in evolution causing high fan-out

⇝ should be represented as-is

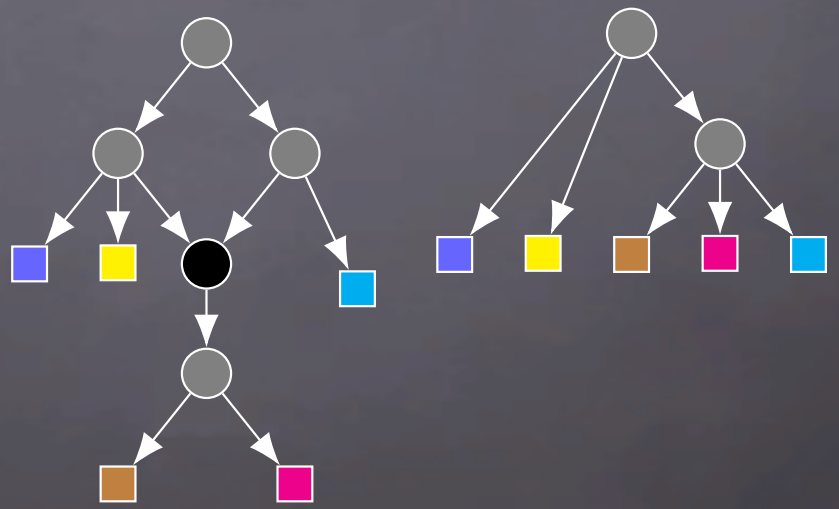
(display = contains subdivision)

Soft Polytomies

noisy/incomplete data

⇝ more liberal representation

(display = "Binary resolutions" display)



Polytomies: Hard ≠ Soft

Hard Polytomies

special events in evolution causing high fan-out

⇝ should be represented as-is

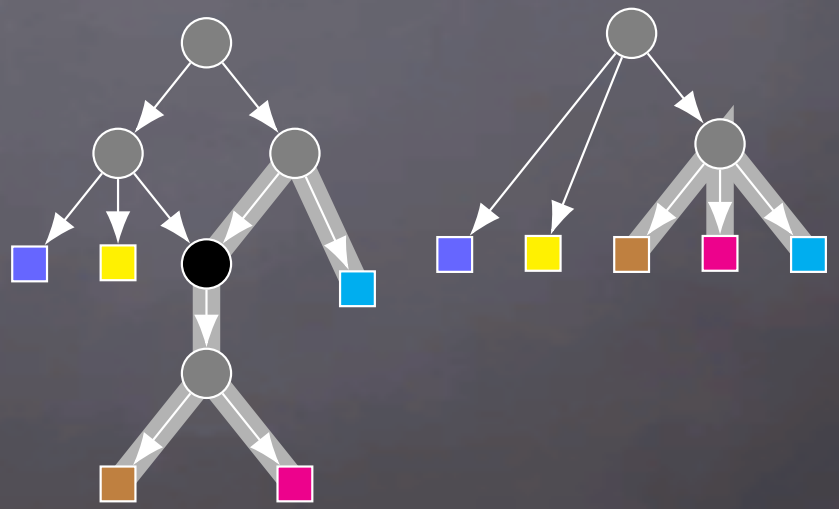
(display = contains subdivision)

Soft Polytomies

noisy/incomplete data

⇝ more liberal representation

(display = "Binary resolutions" display)



Polytomies: Hard ≠ Soft

Hard Polytomies

special events in evolution causing high fan-out

⇝ should be represented as-is

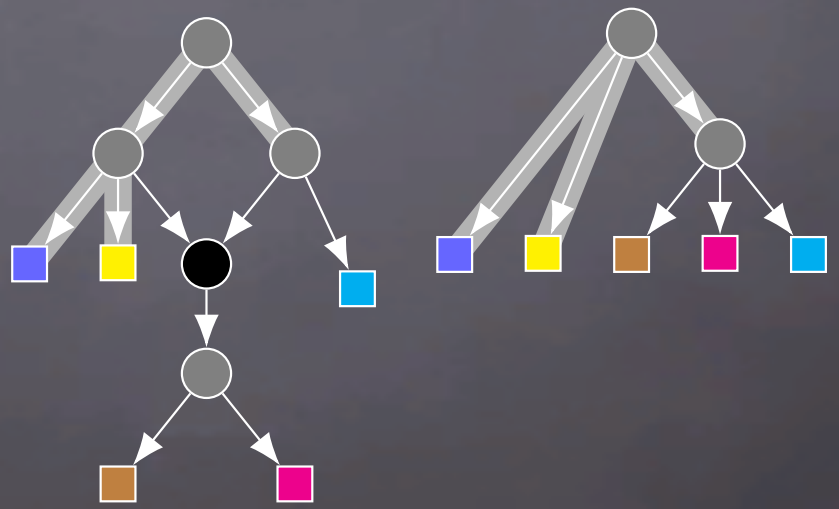
(display = contains subdivision)

Soft Polytomies

noisy/incomplete data

⇝ more liberal representation

(display = "Binary resolutions" display)



Polytomies: Hard ≠ Soft

Hard Polytomies

special events in evolution causing high fan-out

⇝ should be represented as-is

(display = contains subdivision)

Soft Polytomies

noisy/incomplete data

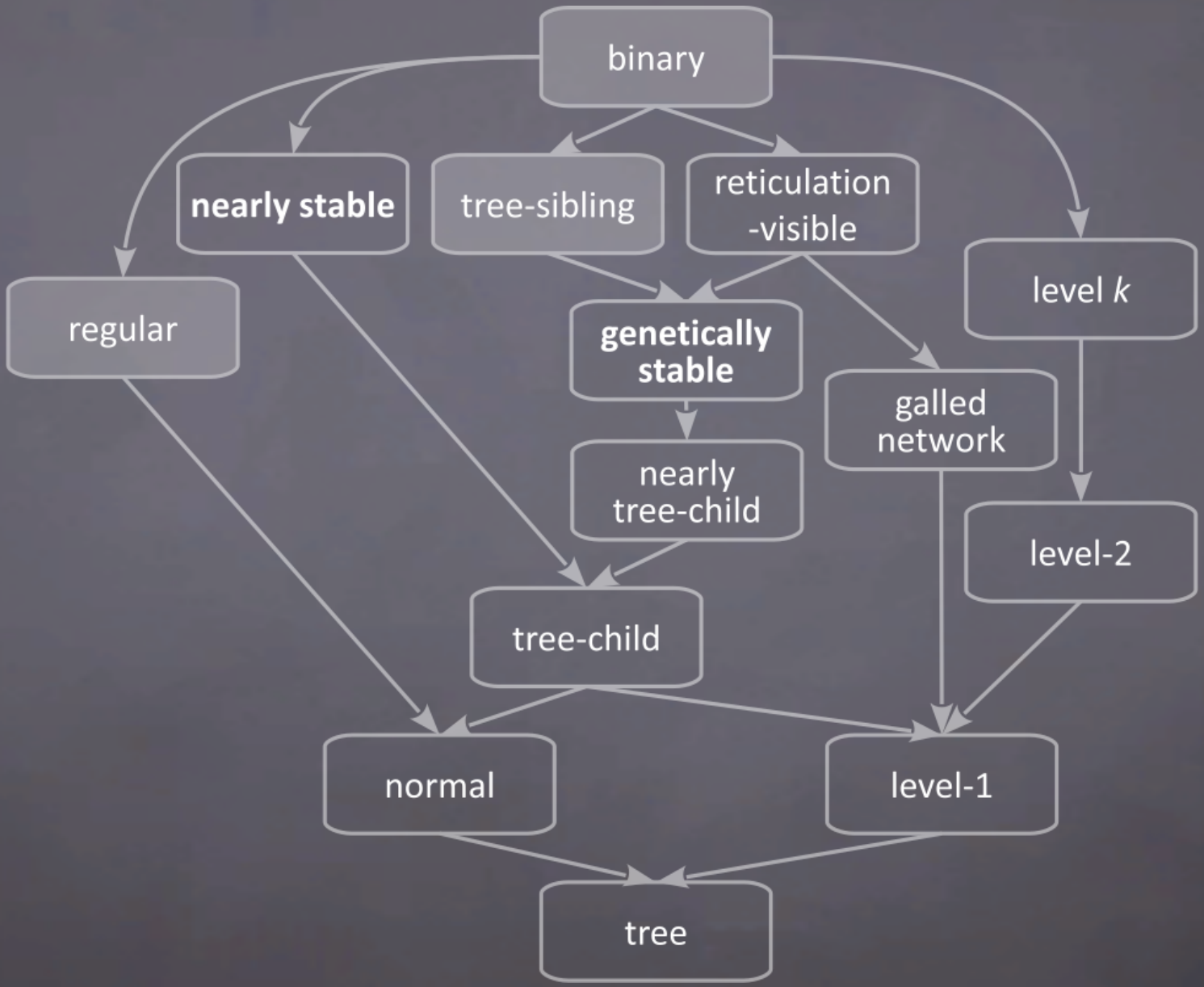
⇝ more liberal representation

(display = "binary resolutions" display)

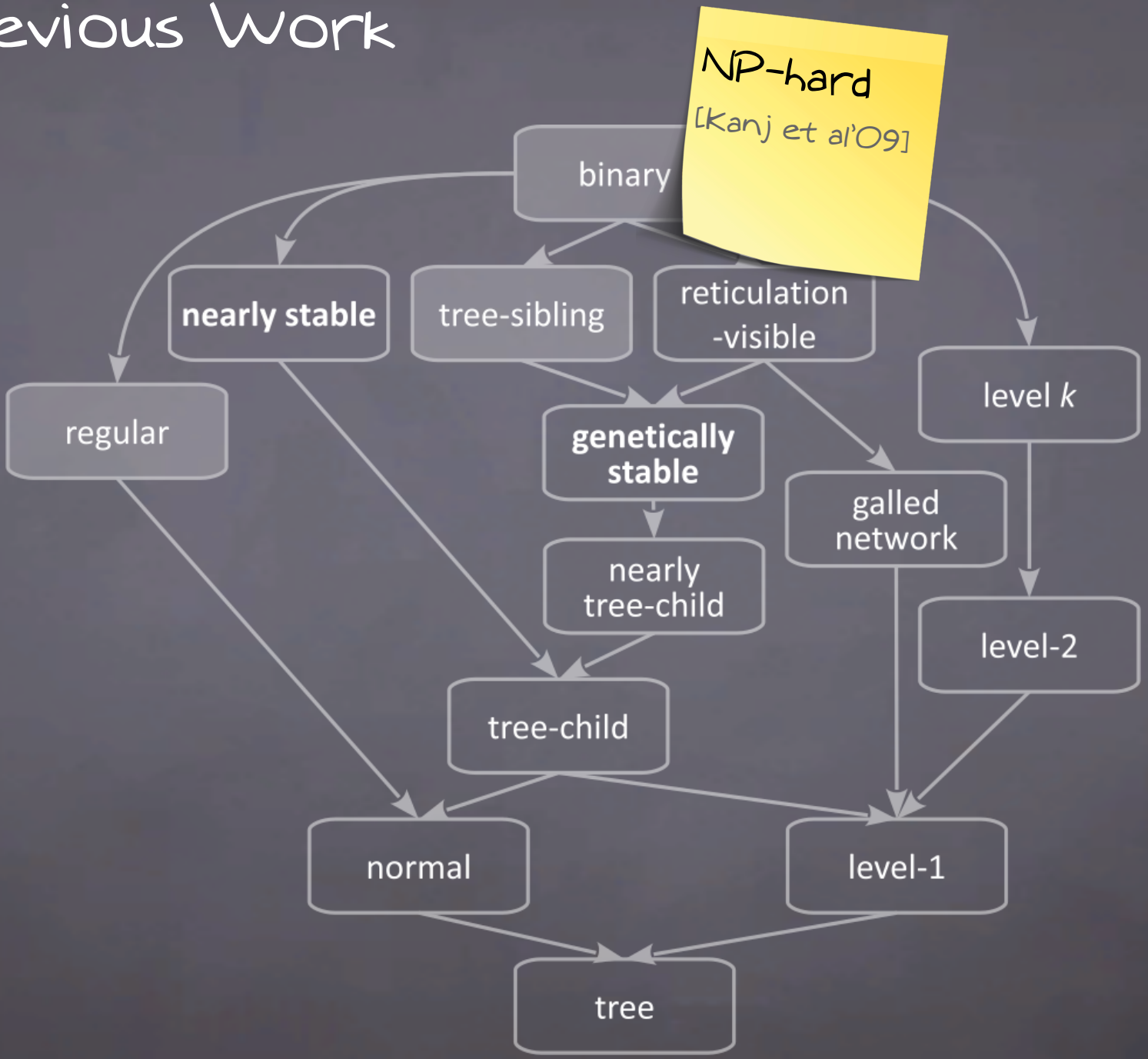
Remark

For binary $N \neq T$, they coincide!

Previous Work



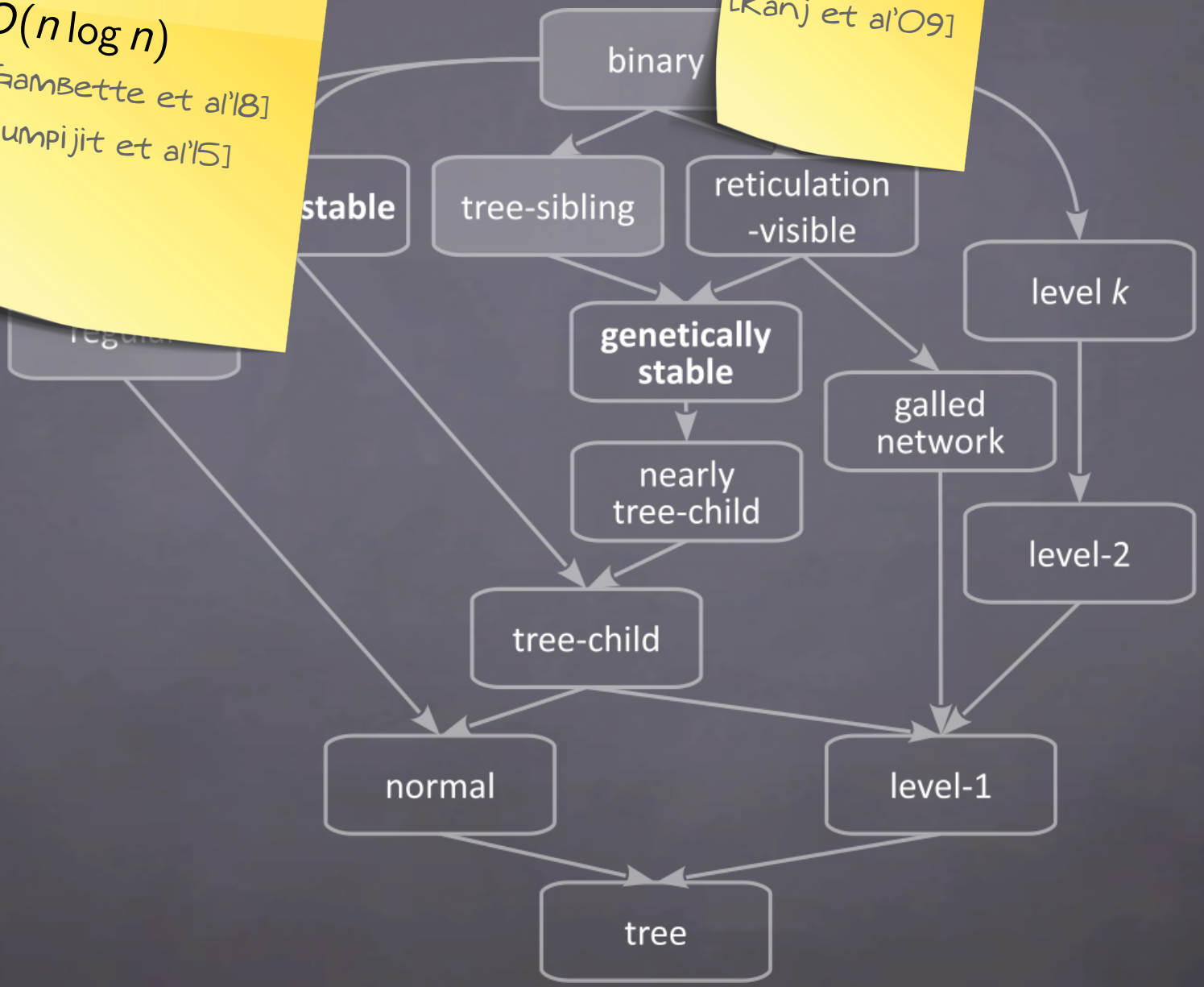
Previous Work



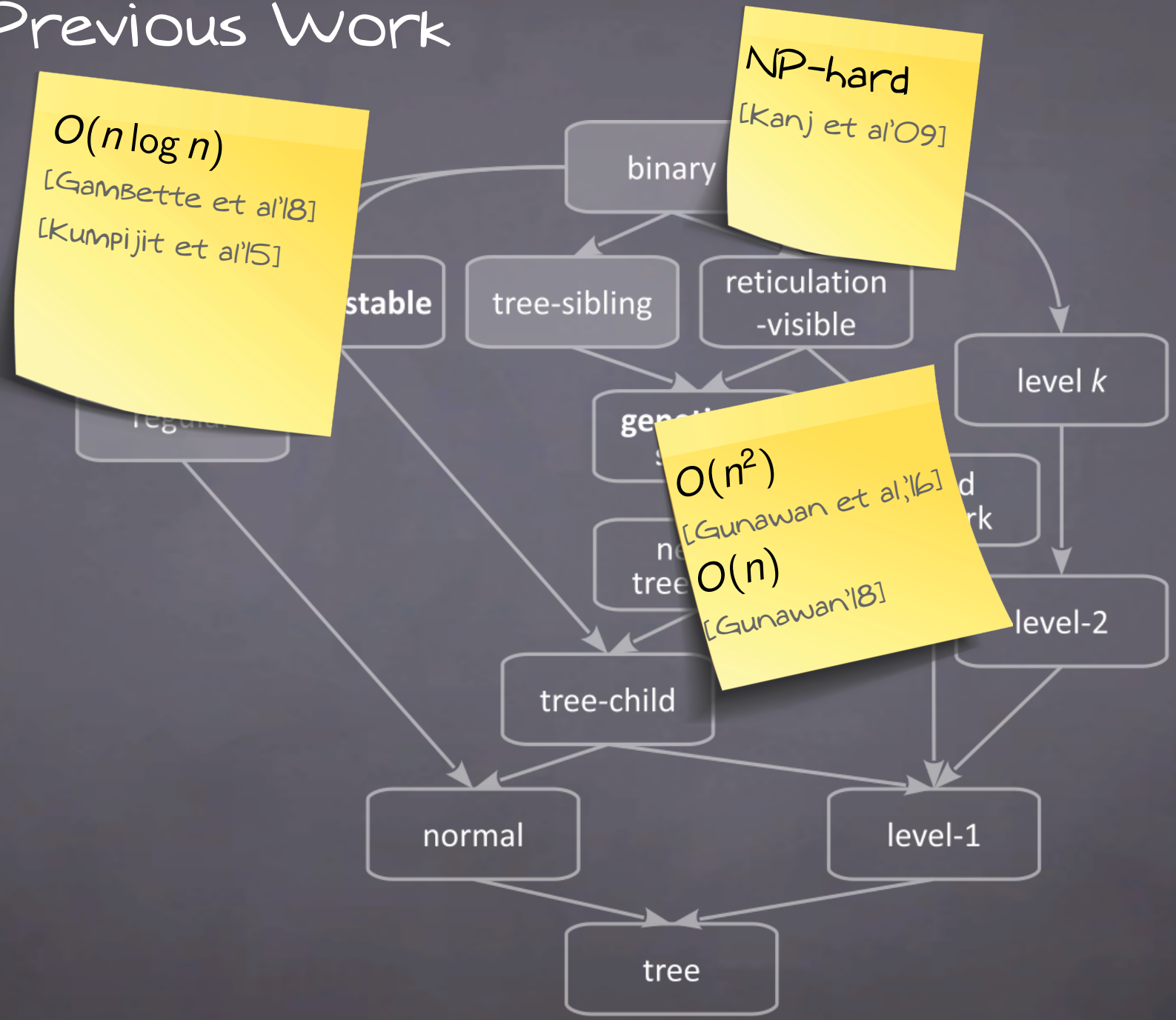
Previous Work

$O(n \log n)$
[Gambette et al'18]
[Kumpijit et al'15]

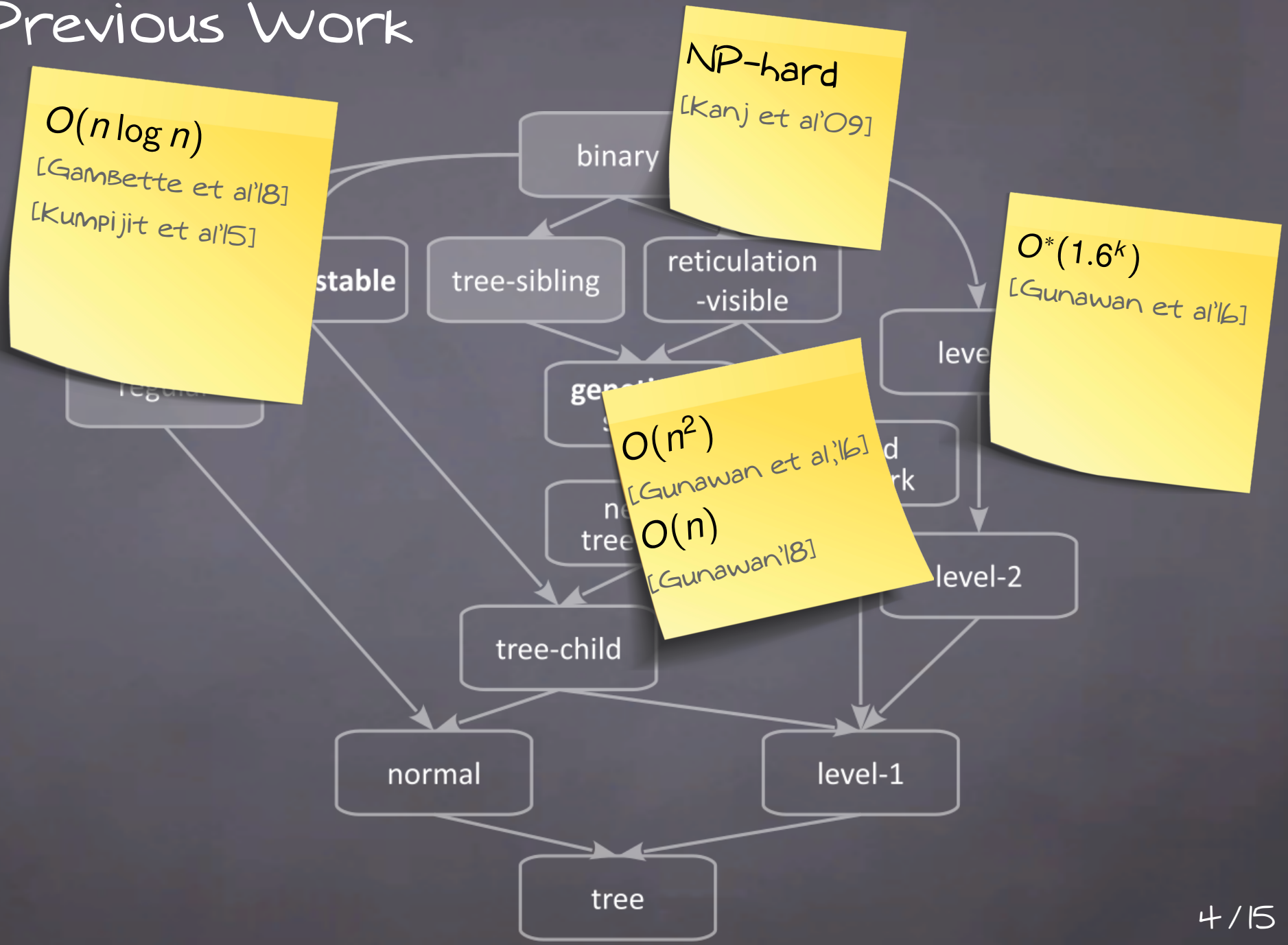
NP-hard
[Kanj et al'09]



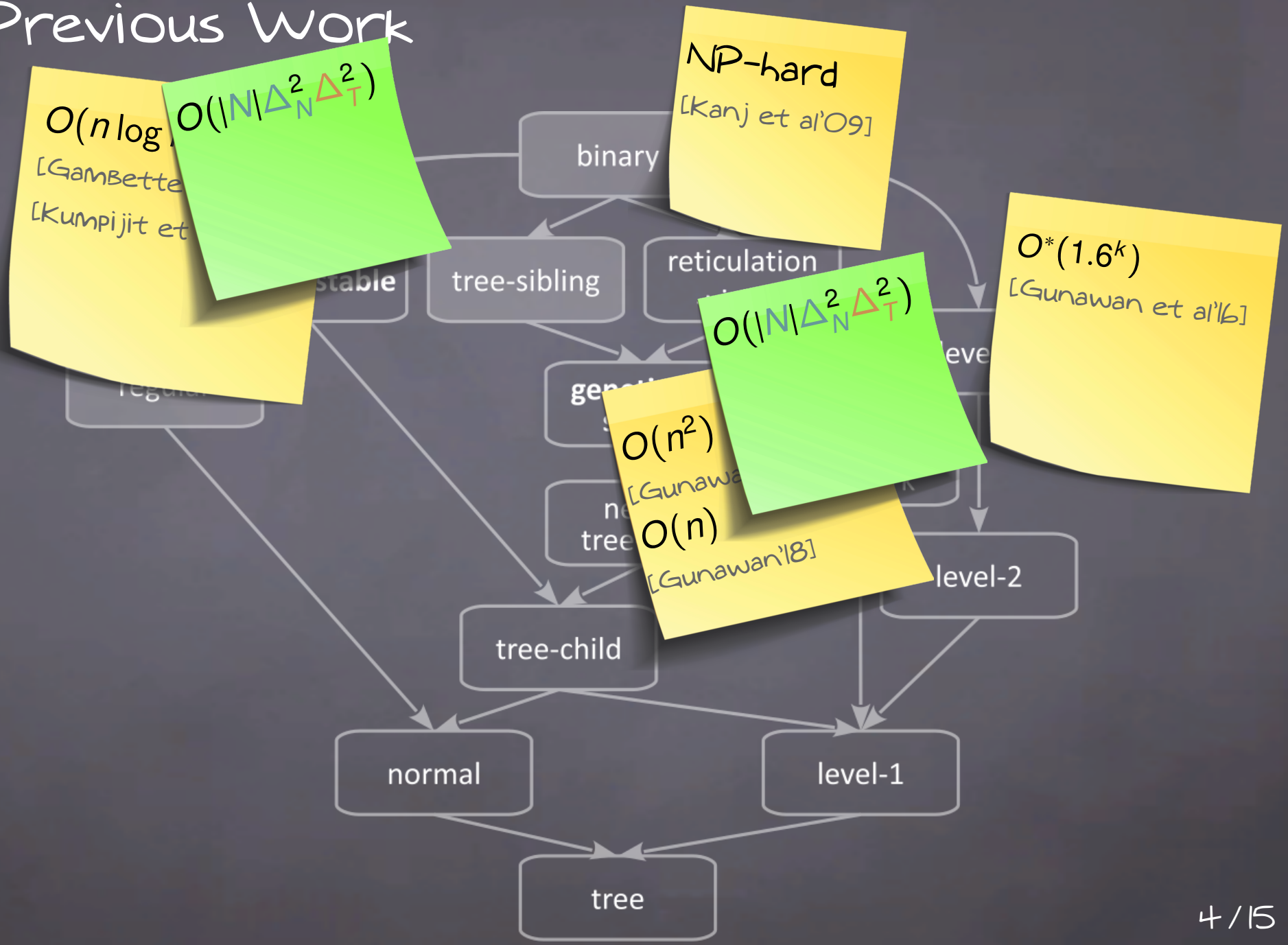
Previous Work



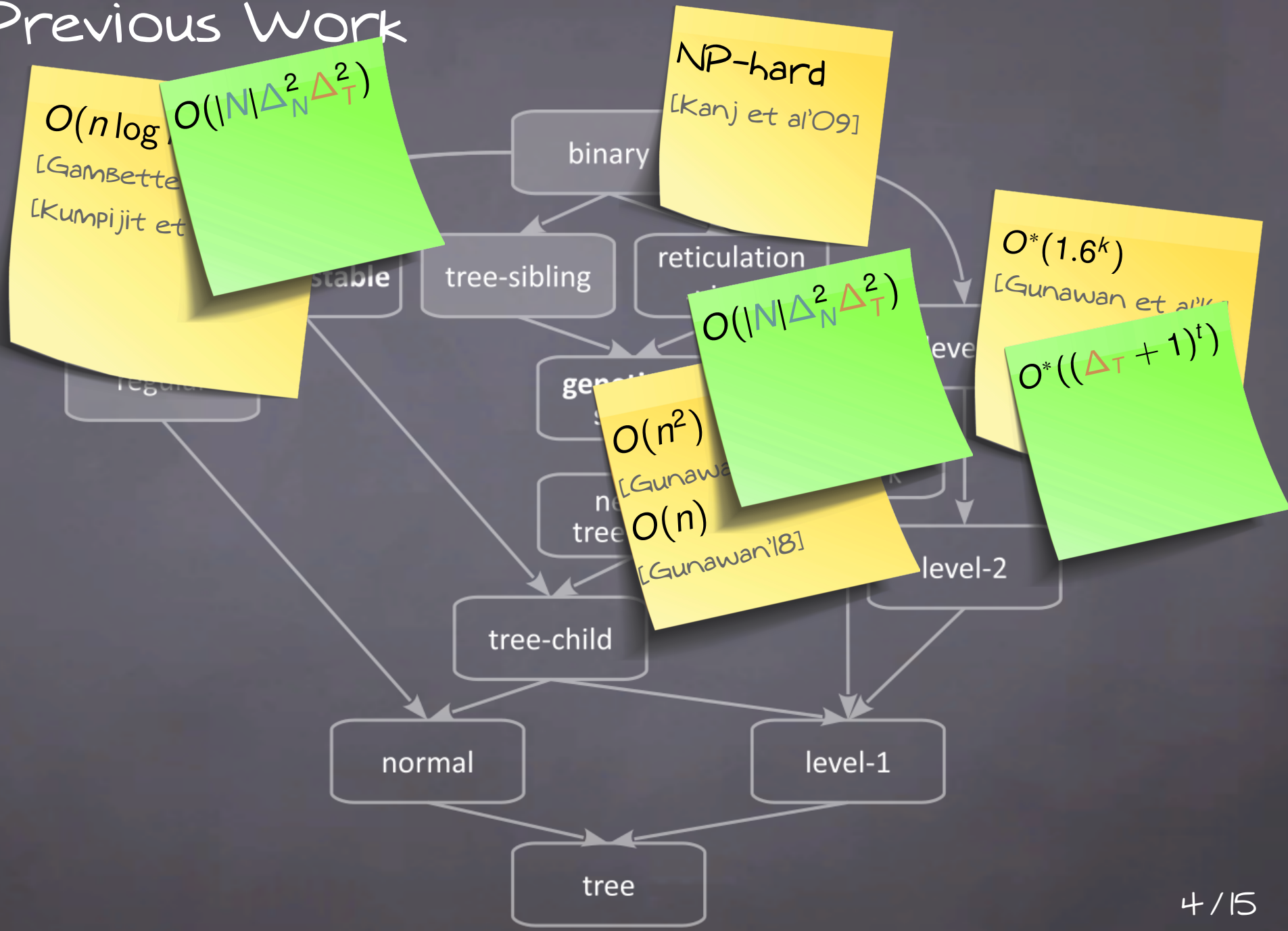
Previous Work



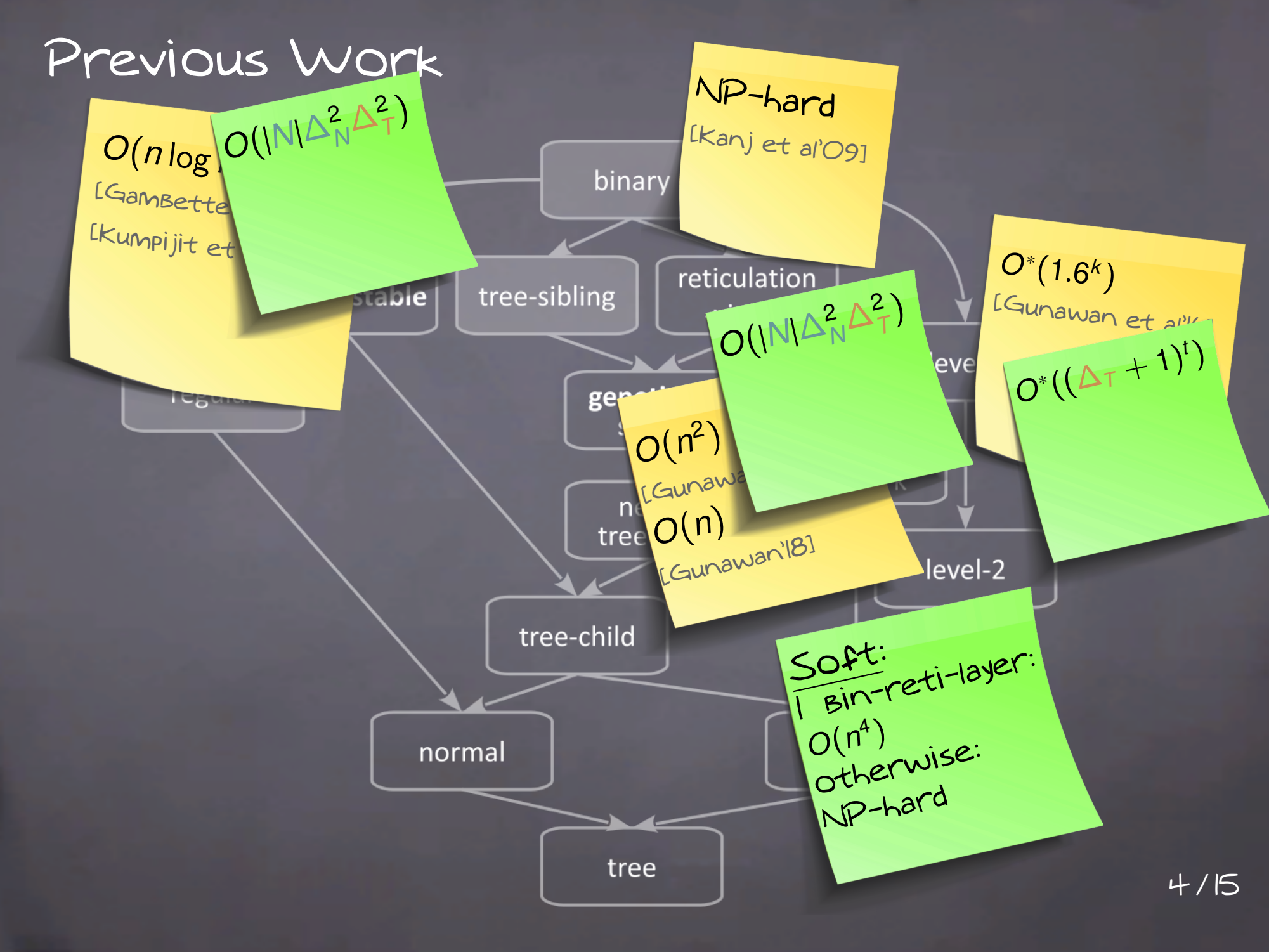
Previous Work



Previous Work



Previous Work



k-labelled Trees

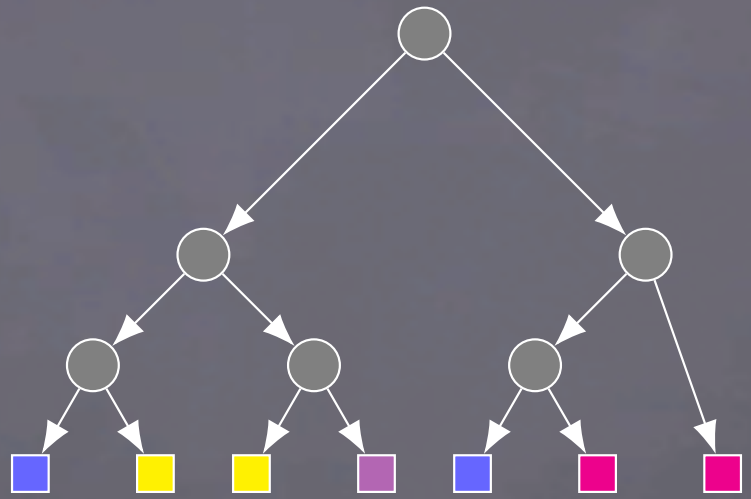
Important "Special Case"

N is a tree, but labels may occur multiple times

k-labelled Trees

Important "Special Case"

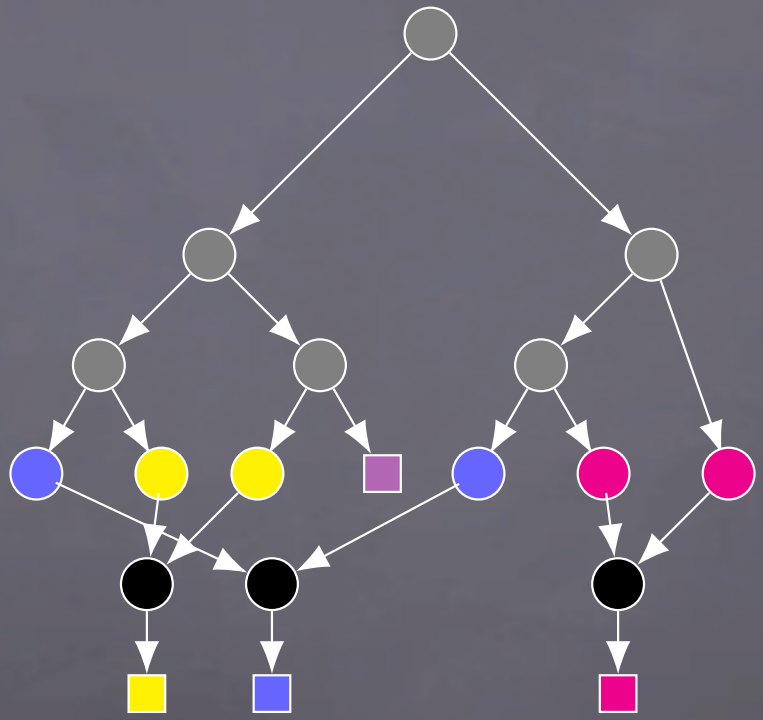
N is a tree, but labels may occur multiple times



k-labelled Trees

Important "Special Case"

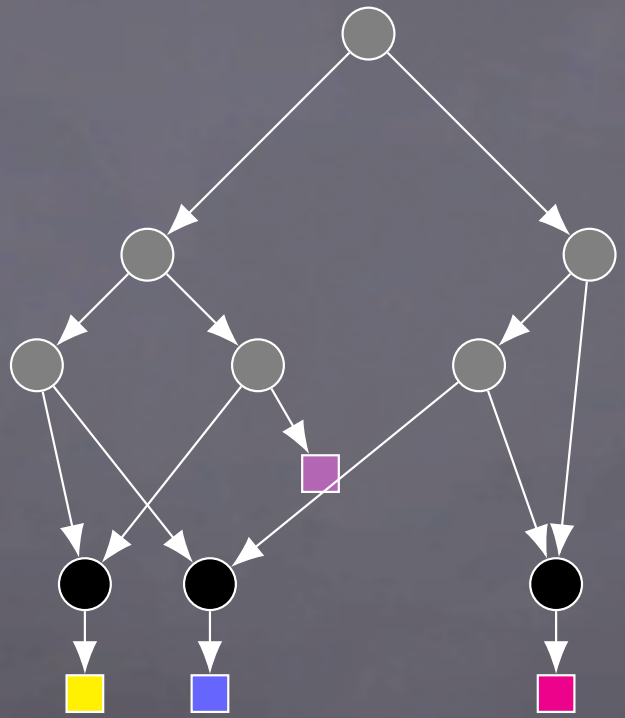
N is a tree, but labels may occur multiple times



k-labelled Trees

Important "Special Case"

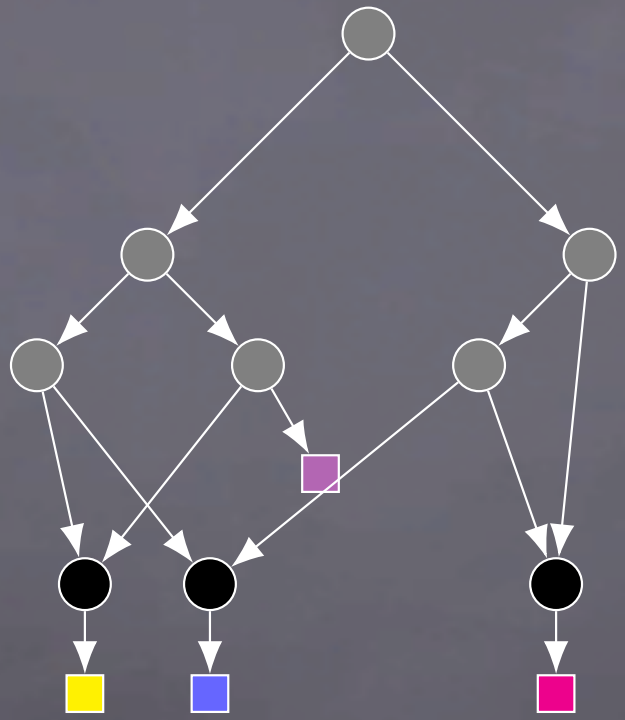
N is a tree, but labels may occur multiple times



k-labelled Trees

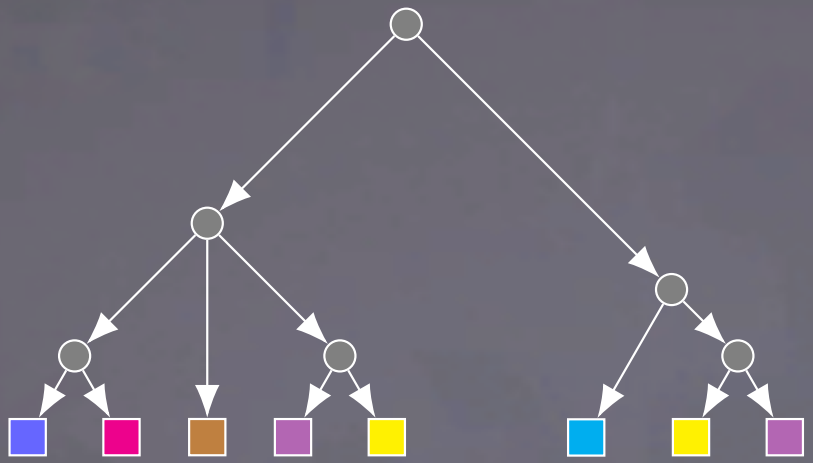
Important "Special Case"

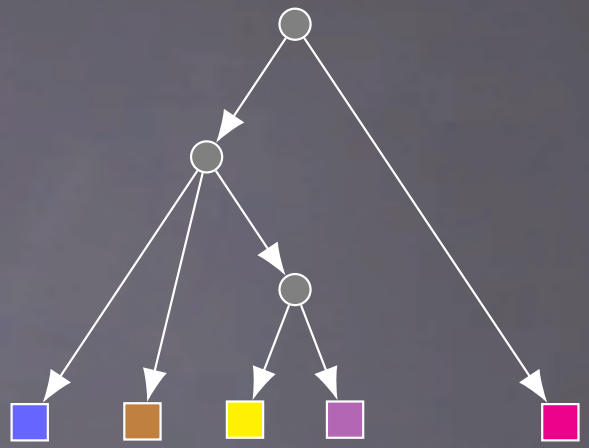
N is a tree, but labels may occur multiple times



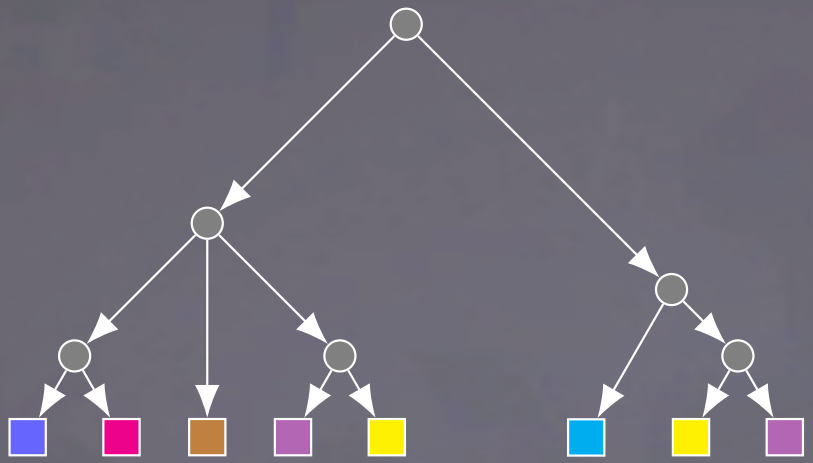
Hard Polytomies $\rightsquigarrow O(|N| \cdot k^2 \Delta_T{}^2)$
Soft Polytomies $\rightsquigarrow O(|N|^4)$ for 2 occurrences, NP-hard for 3

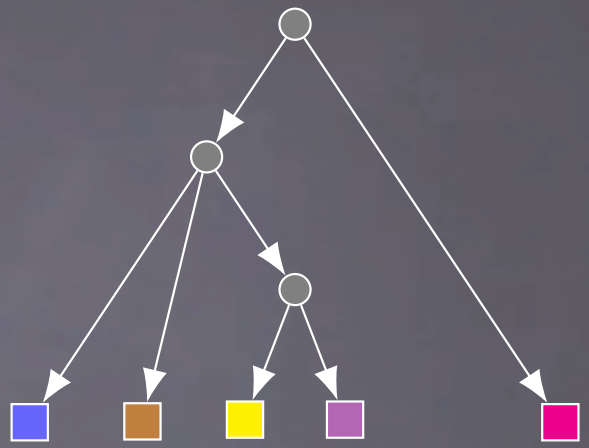
k-labeled Trees





k-labeled Trees

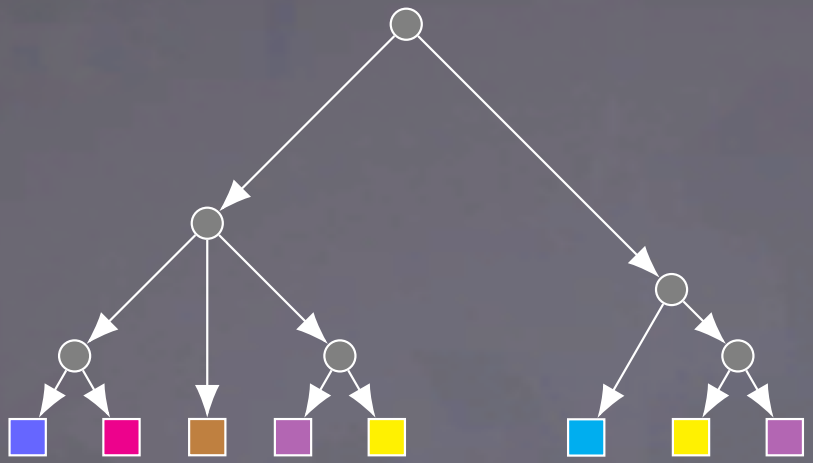


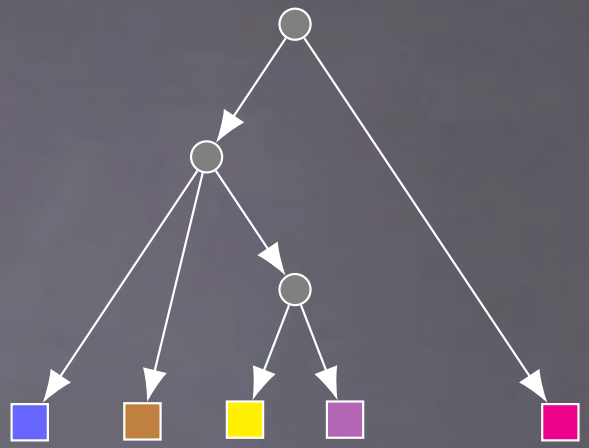


Definition

$v \in T$: $M(v)$ = lowest vertices u s.t. N_u displays T_v.

k-labeled Trees

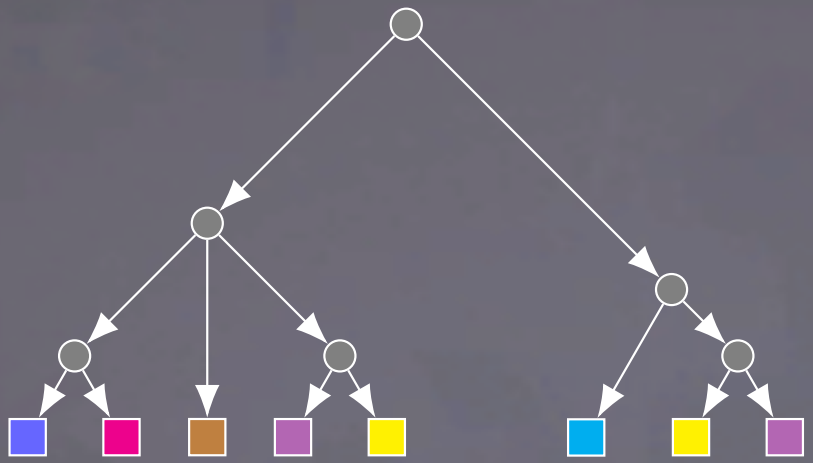


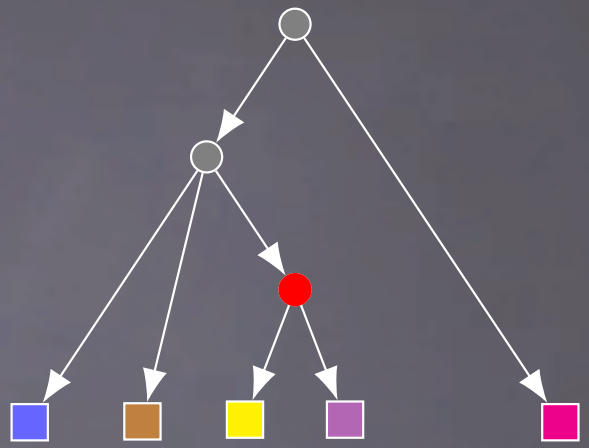


Definition

$v \in T$: $M(v) =$ lowest vertices u s.t. N_u displays T_v. $\rightsquigarrow |M(v)| \leq k$

k-labeled Trees



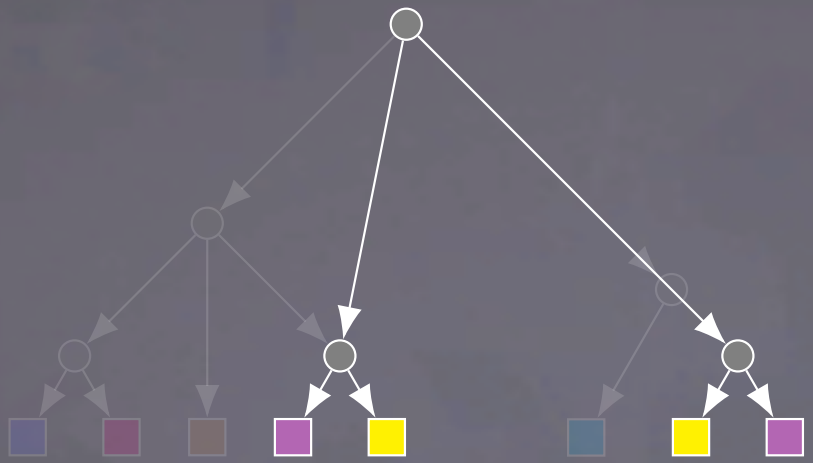


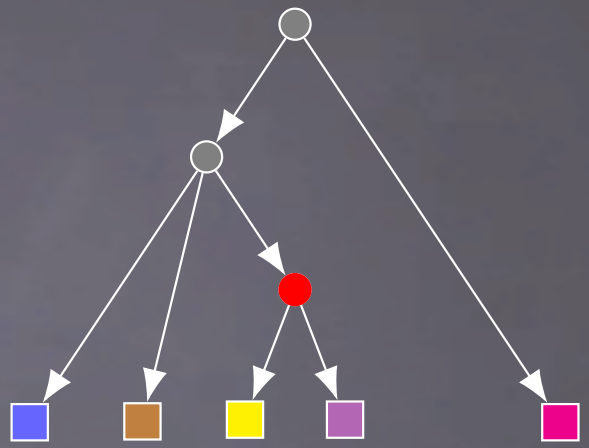
Definition

$v \in T$: $M(v) =$ lowest vertices u s.t. N_u displays T_v. $\rightsquigarrow |M(v)| \leq k$

for each v bottom-up:
1. Build $Z := \bigcup_i M(v_i) \lessdot N|_Z$
2. for each u in $N|_Z$:
   (a) Build bip. graph
   (b) Max. bip. matching

k-labeled Trees





Definition

$v \in T$: $M(v)$ = lowest vertices u s.t. N_u displays T_v. $\rightsquigarrow |M(v)| \leq k$

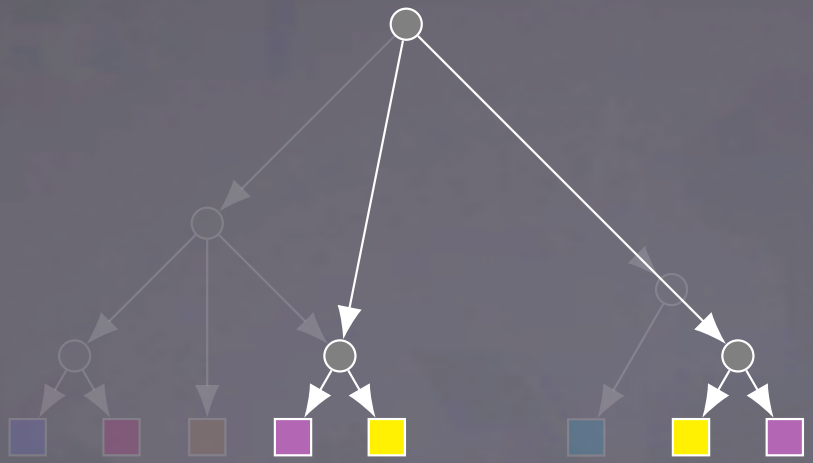
for each v bottom-up:
1. Build $Z := \bigcup_i M(v_i) \Leftarrow N|_Z$
2. for each u in $N|_Z$:
   (a) Build bip. graph
   (b) Max. bip. matching

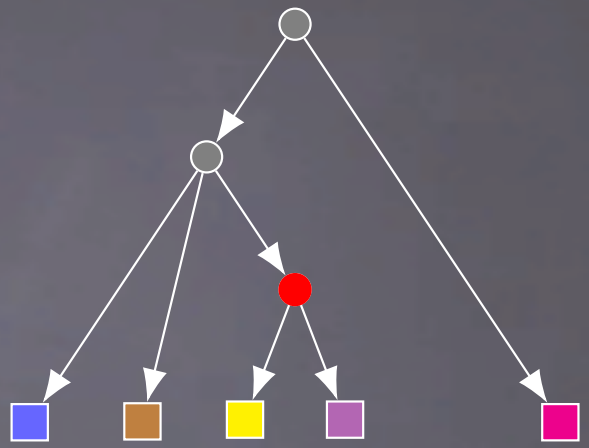
edge $u_j v_i$

$\Leftrightarrow$

$\exists x \leq u_j$ in $M(v_i)$

k-labeled Trees

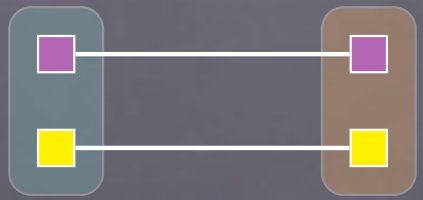




Definition

$v \in T$: $M(v)$ = lowest vertices u s.t. N_u displays T_v. $\rightsquigarrow |M(v)| \leq k$

for each v bottom-up:
1. Build $Z := \bigcup_i M(v_i) \Downarrow N|_Z$
2. for each u in $N|_Z$:
   (a) Build bip. graph
   (b) Max. bip. matching

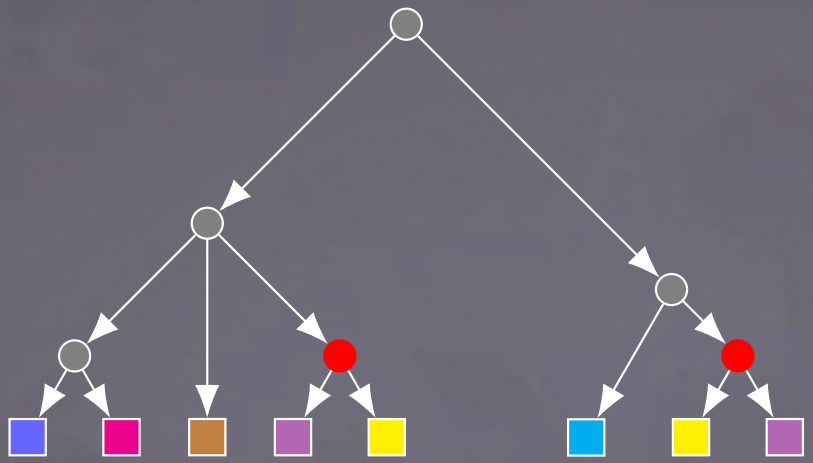


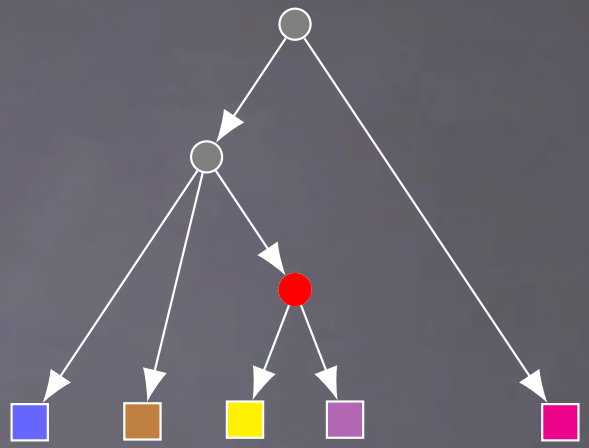
edge $u_j v_i$

$\Leftrightarrow$

$\exists x \leq u_j$ in $M(v_i)$

k-labeled Trees





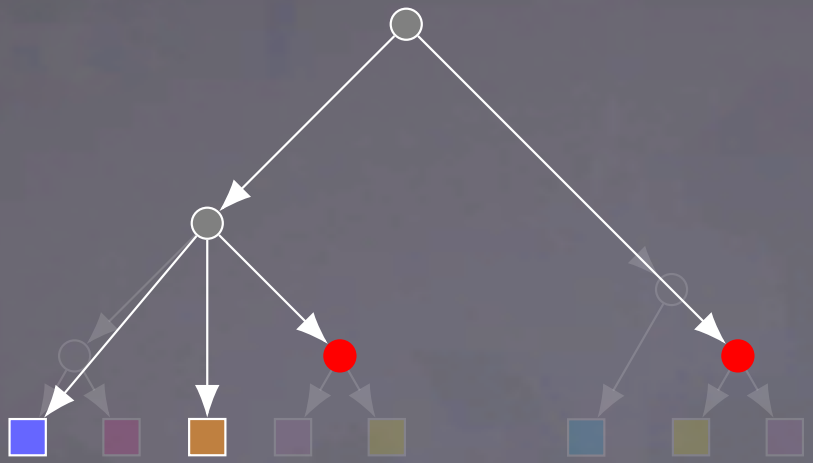
Definition

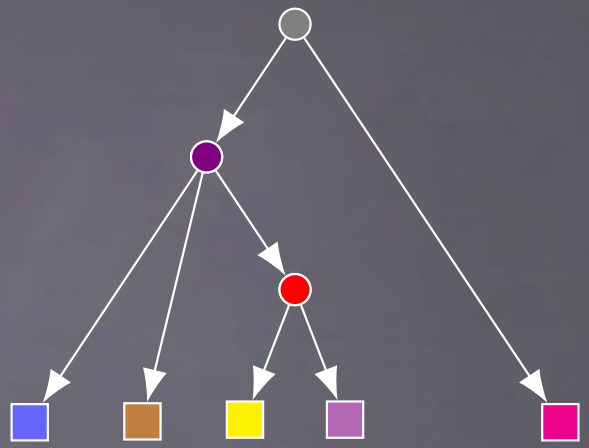
$v \in T$: $M(v)$ = lowest vertices u s.t. N_u displays T_v. $\rightsquigarrow |M(v)| \leq k$

for each v bottom-up:
1. Build $Z := \bigcup_i M(v_i) \lessdot N|_Z$
2. for each u in $N|_Z$:
   (a) Build bip. graph
   (b) Max. bip. matching

edge $u_j v_i$
$\Leftrightarrow$
$\exists x \leq u_j$ in $M(v_i)$

k-labeled Trees





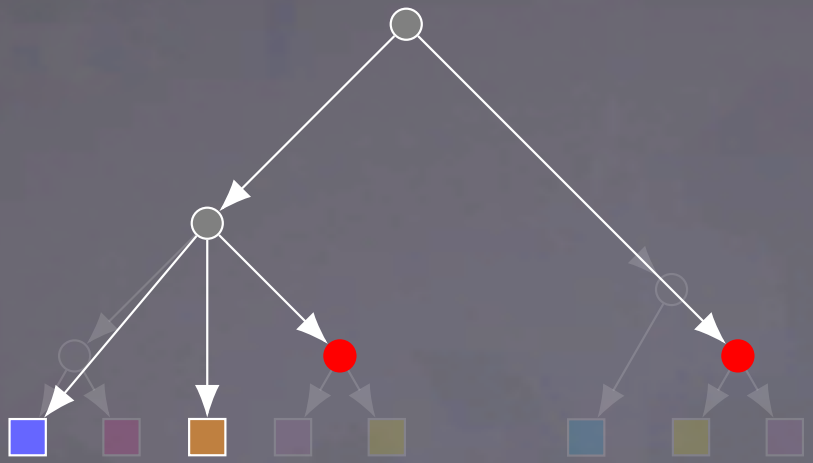
Definition

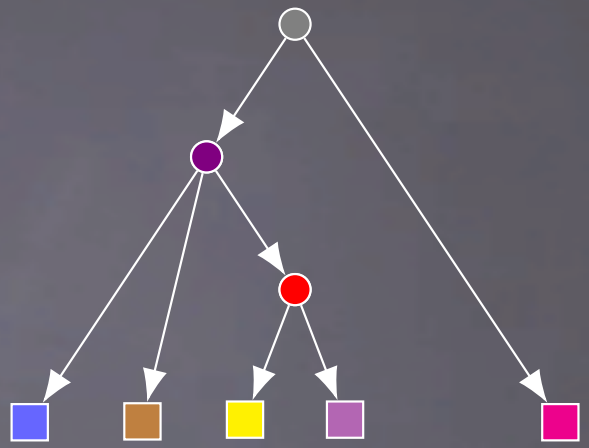
$v \in T$: $M(v)$ = lowest vertices u s.t. N_u displays T_v. $\rightsquigarrow |M(v)| \leq k$

for each v bottom-up:
1. Build $Z := \bigcup_i M(v_i) \downarrow N|_Z$
2. for each u in $N|_Z$:
   (a) Build bip. graph
   (b) Max. bip. matching

edge $u_j v_i$
$\Leftrightarrow$
$\exists x \leq u_j$ in $M(v_i)$

k-labeled Trees

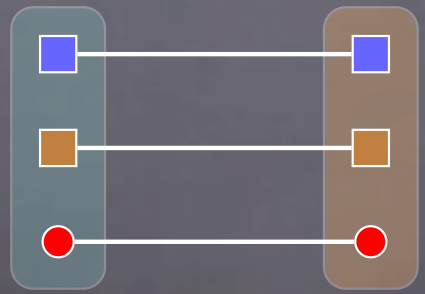




Definition

$v \in T$: $M(v)$ = lowest vertices u s.t. N_u displays T_v. $\rightsquigarrow |M(v)| \leq k$

for each v bottom-up:
1. Build $Z := \bigcup_i M(v_i) \Downarrow N|_Z$
2. for each u in $N|_Z$:
   (a) Build Bip. Graph
   (b) Max. Bip. Matching

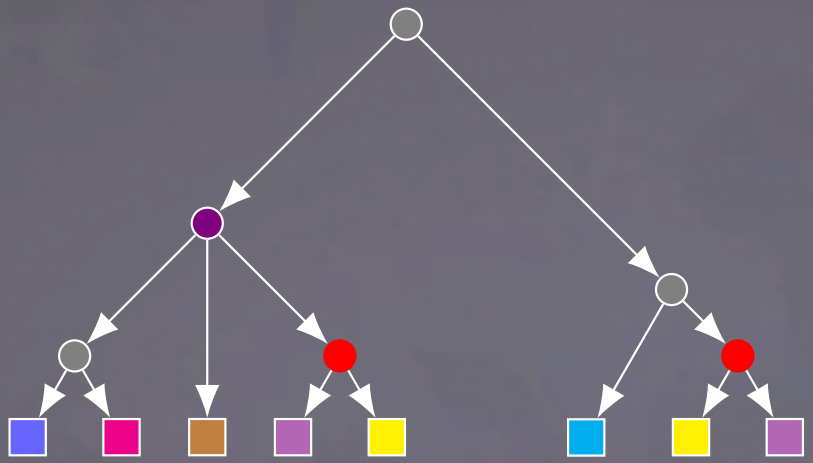


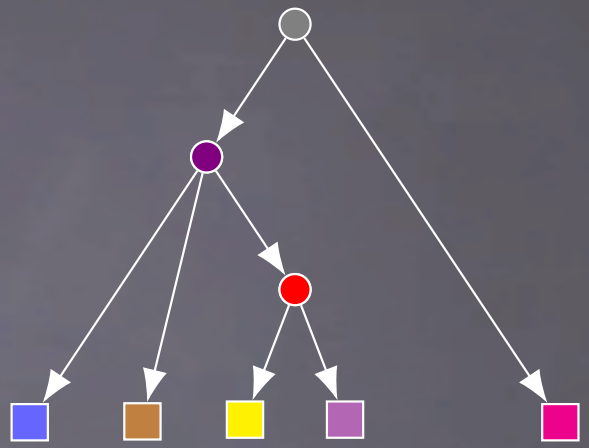
edge $u_j v_i$

$\Leftrightarrow$

$\exists x \leq u_j$ in $M(v_i)$

k-labeled Trees





Definition

$v \in T$: $M(v)$ = lowest vertices u s.t. N_u displays T_v. $\rightsquigarrow |M(v)| \leq k$

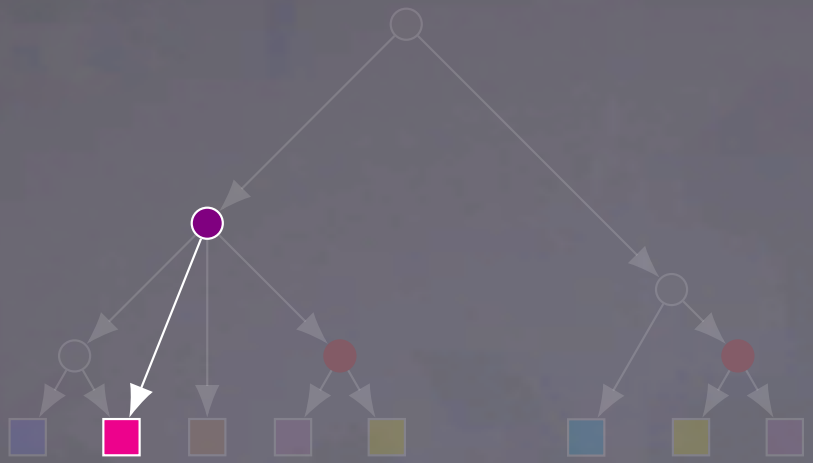
for each v bottom-up:
1. Build $Z := \bigcup_i M(v_i) \Leftarrow N|_Z$
2. for each u in $N|_Z$:
   (a) Build bip. graph
   (b) Max. bip. matching

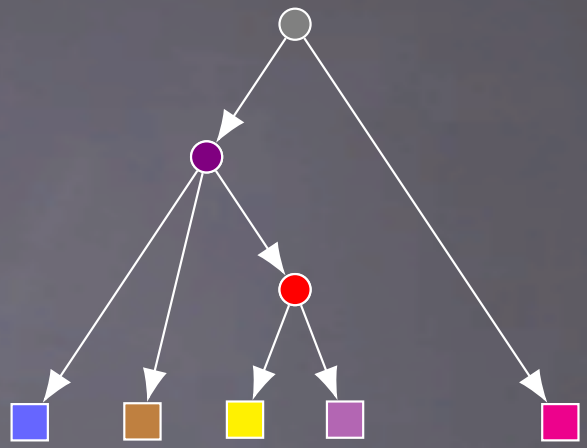
edge $u_j v_i$

$\Leftrightarrow$

$\exists x \leq u_j$ in $M(v_i)$

k-labeled Trees





Definition

$v \in T$: $M(v)$ = lowest vertices u s.t. N_u displays T_v. $\rightsquigarrow |M(v)| \leq k$

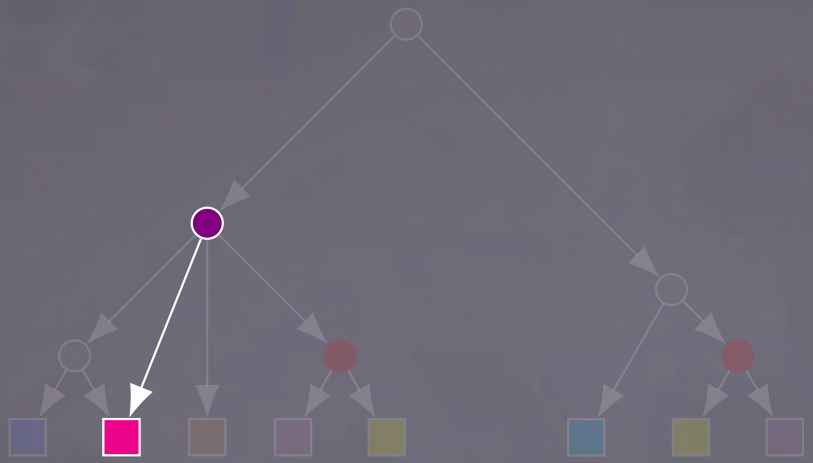
for each v bottom-up:
1. Build $Z := \bigcup_i M(v_i) \Leftarrow N|_Z$
2. for each u in $N|_Z$:
   (a) Build Bip. Graph
   (b) Max. Bip. Matching

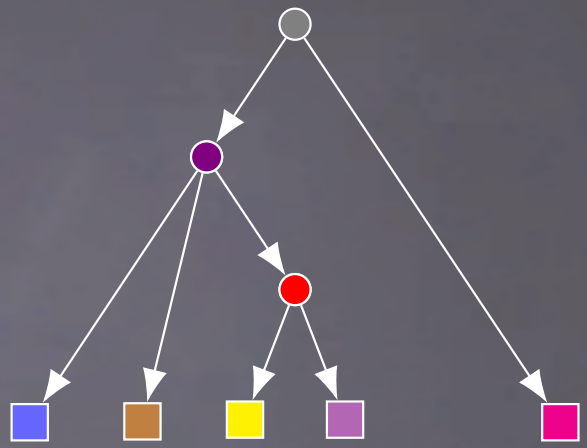
edge $u_j v_i$

$\Leftrightarrow$

$\exists x \leq u_j$ in $M(v_i)$

k-labeled Trees





Definition

$v \in T$: $M(v)$ = lowest vertices u s.t. N_u displays T_v. $\rightsquigarrow |M(v)| \leq k$

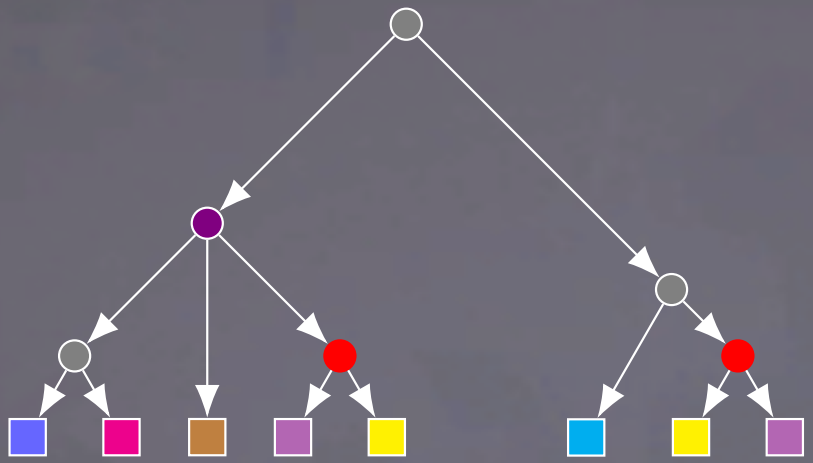
for each v bottom-up:
1. Build $Z := \bigcup_i M(v_i) \Downarrow N|_Z$
2. for each u in $N|_Z$:
   (a) Build bip. graph
   (b) Max. bip. matching

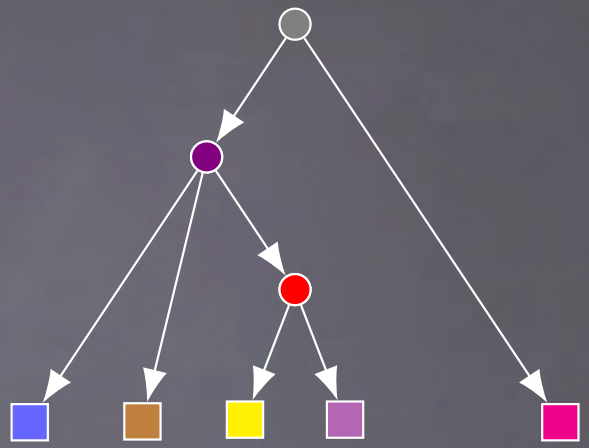
edge $u_j v_i$

$\Leftrightarrow$

$\exists x \leq u_j$ in $M(v_i)$

k-labeled Trees





Definition

$v \in T$: $M(v)$ = lowest vertices u s.t. N_u displays T_v. $\rightsquigarrow |M(v)| \leq k$

for each v bottom-up:
1. Build $Z := \bigcup_i M(v_i) \notin N|_Z$
2. for each u in $N|_Z$:
   (a) Build bip. graph
   (b) max. bip. matching

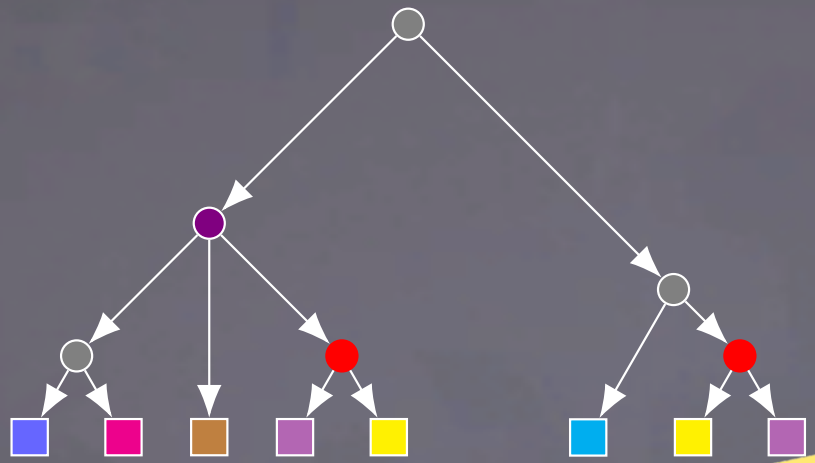
$\rightsquigarrow O(|N| \cdot (k\Delta_T)^2)$ time

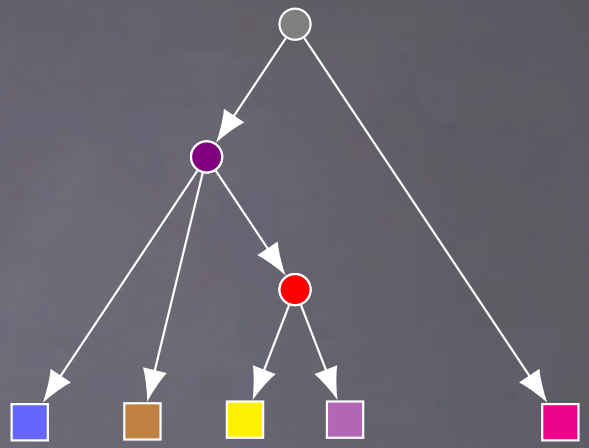
edge $u_j v_i$

$\Leftrightarrow$

$\exists x \leq u_j$ in $M(v_i)$

k-labeled Trees





Definition

$v \in T$: $M(v)$ = lowest ys T_v. $\rightsquigarrow |M(v)| \le k$

for each v bottom-up:
1. Build $Z := \bigcup_i M(v_i) \notin N|_Z$
2. for each u in $N|_Z$:
   (a) Build Bip. Graph
   (b) Max. Bip. Matching

$\rightsquigarrow O(|N| \cdot (k\Delta_T)^2)$ time

Theorem
We get highest v s.t. N displays T_v in $O(|N| \cdot (k\Delta_T)^2)$ time.

Note: these T_v are leaf-disjoint

edge $u_j v_i$
$\Leftrightarrow$
$\exists x \le u_j$ in $M(v_i)$

For Networks...

Observation

[Gunawan et al., '16]

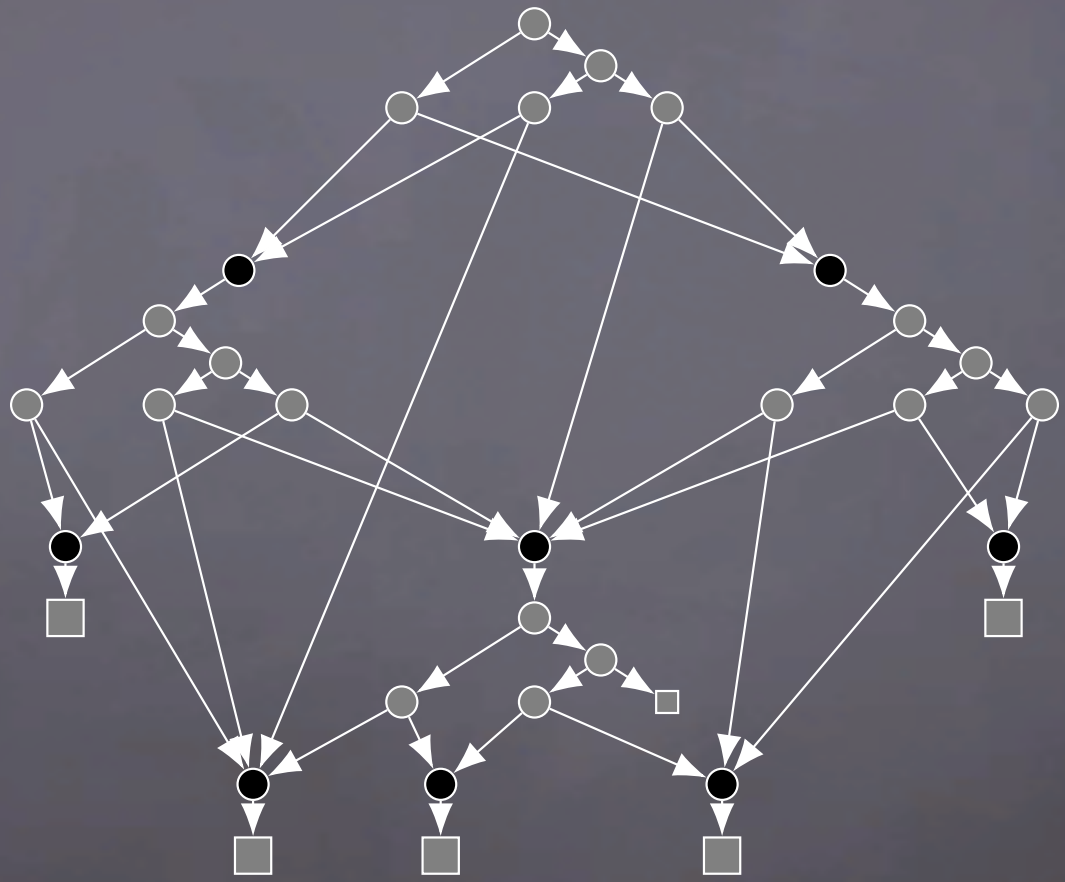
$N - R$ = forest $\rightsquigarrow$ N = DAG of tree components + R

For Networks...

Observation

[Gunawan et al., '16]

$N - R$ = forest $\rightsquigarrow$ N = DAG of tree components + R

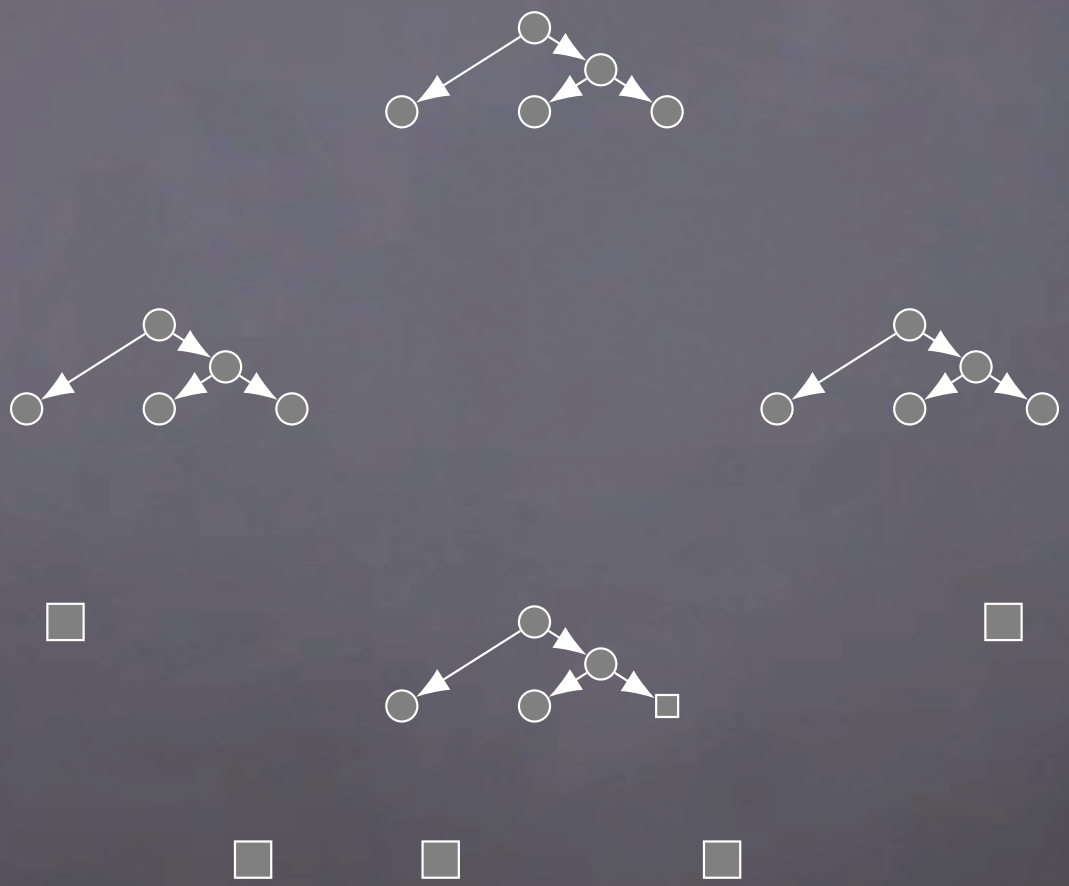


For Networks...

Observation

[Gunawan et al., '16]

$N - R$ = forest $\rightsquigarrow$ N = DAG of tree components + R

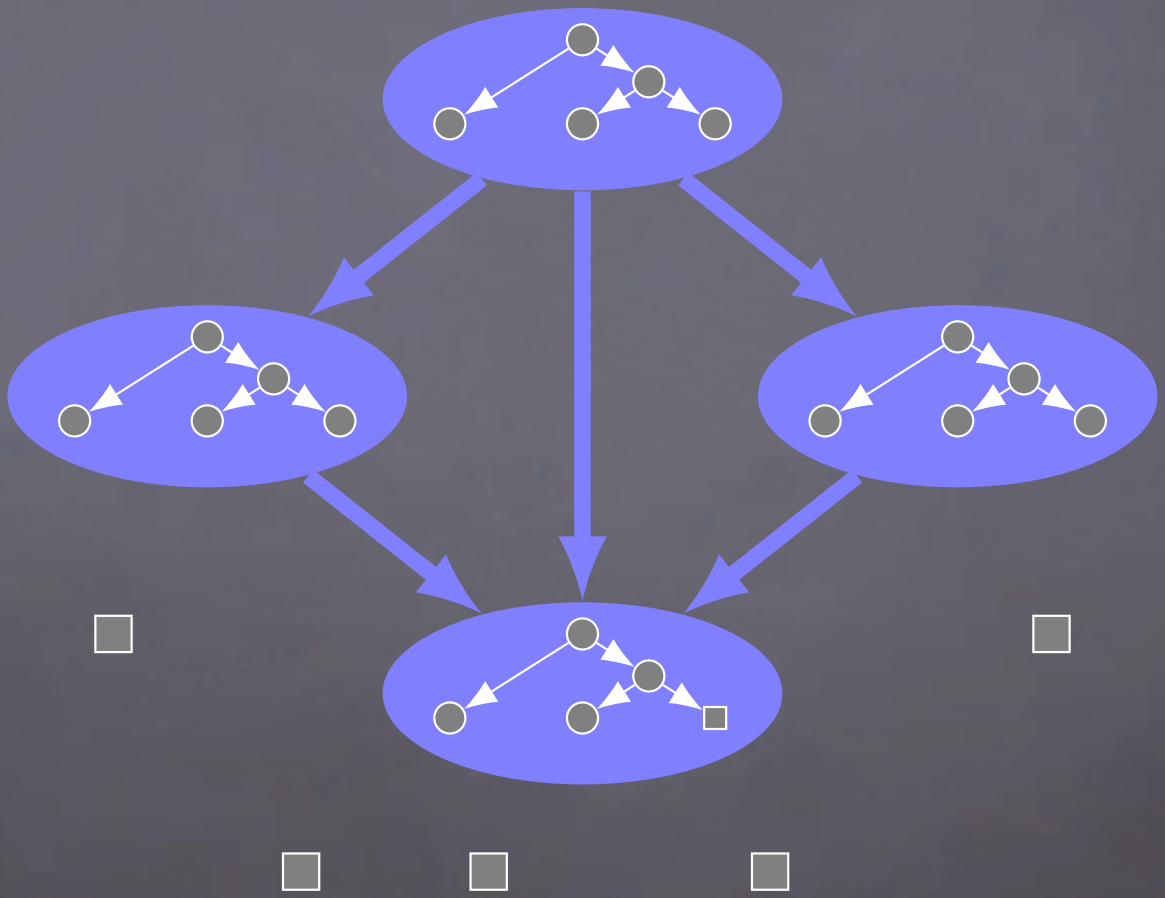


For Networks...

Observation

[Gunawan et al., '16]

$N - R$ = forest $\rightsquigarrow$ N = DAG of tree components + R

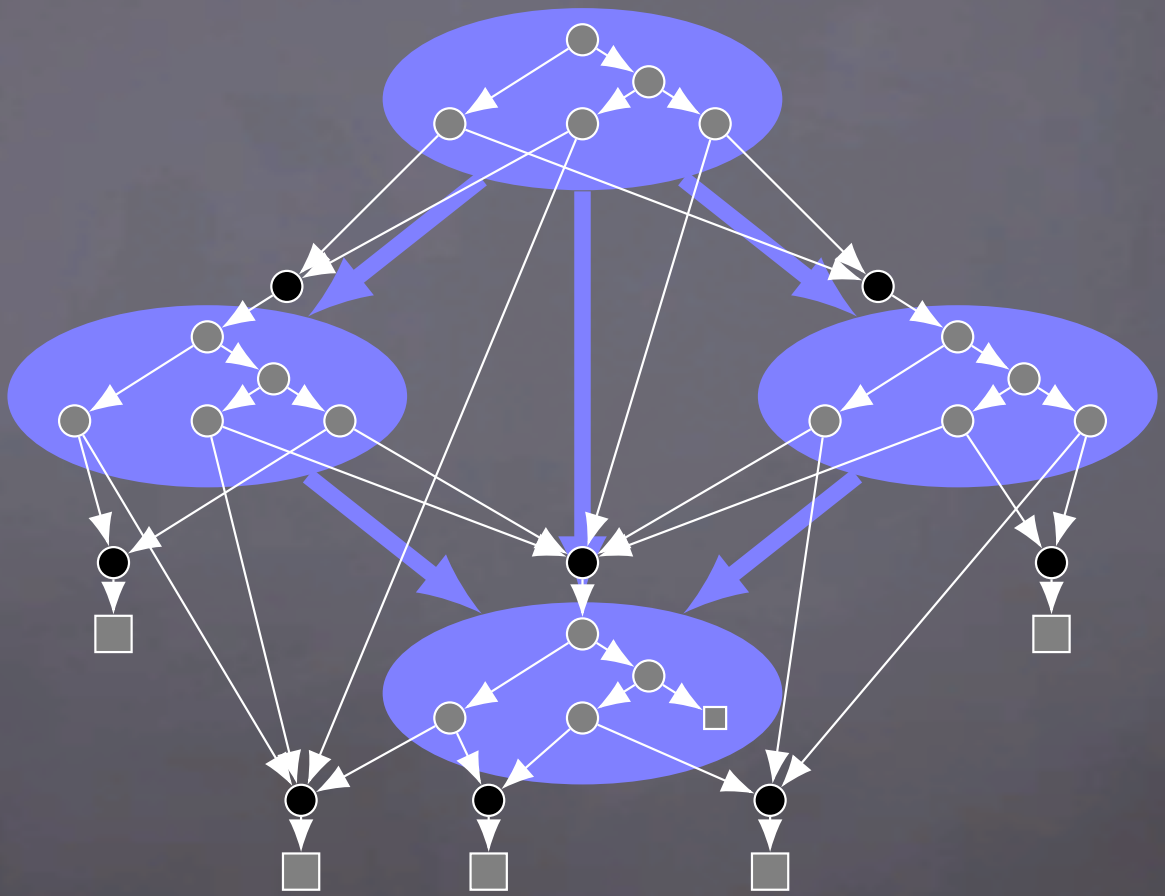


For Networks...

Observation

[Gunawan et al., '16]

$N - R$ = forest $\rightsquigarrow$ N = DAG of tree components + R

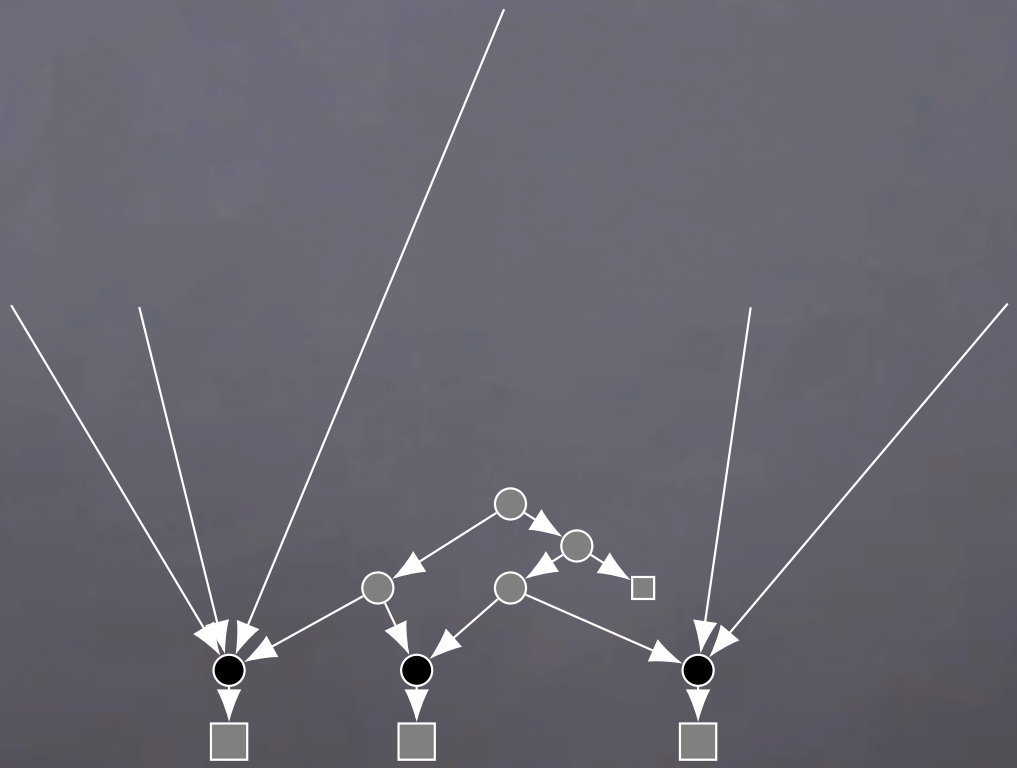


For Networks...

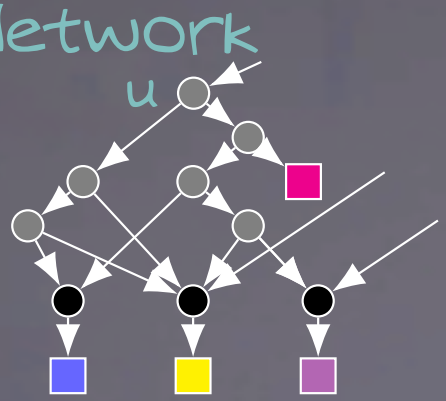
Observation

[Gunawan et al., '16]

$N - R$ = forest $\rightsquigarrow$ N = DAG of tree components + R

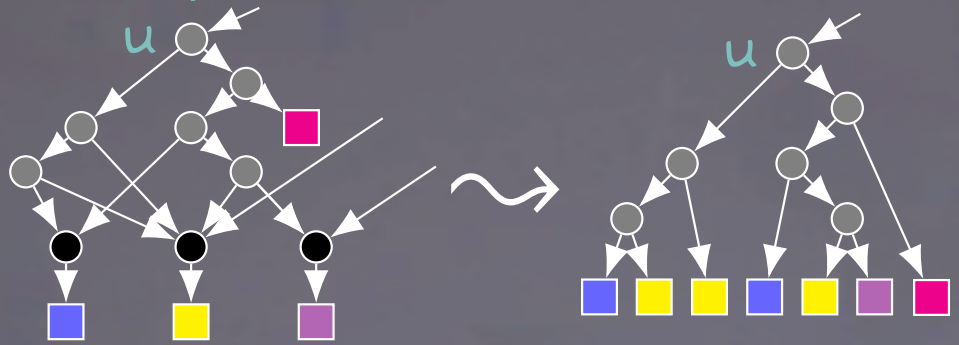


For Networks...



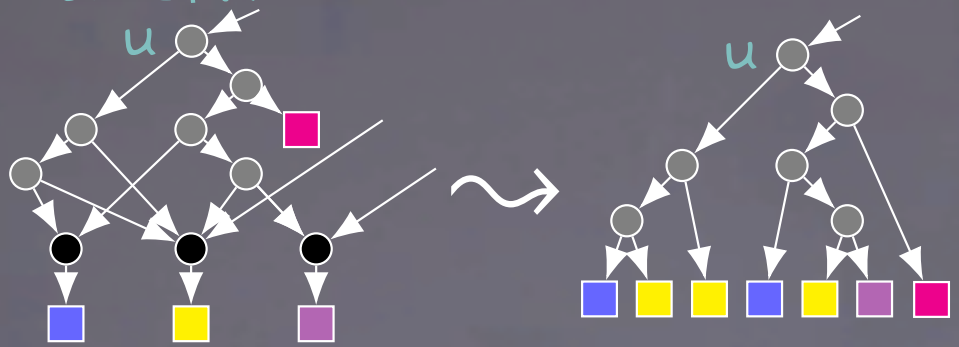
For Networks...

Network

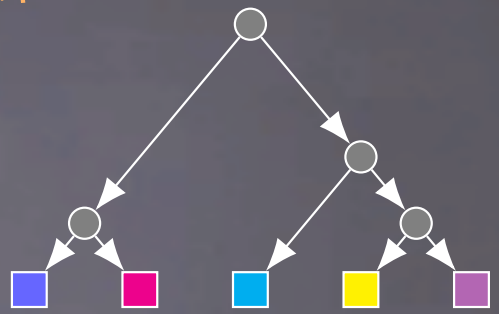


For Networks...

Network

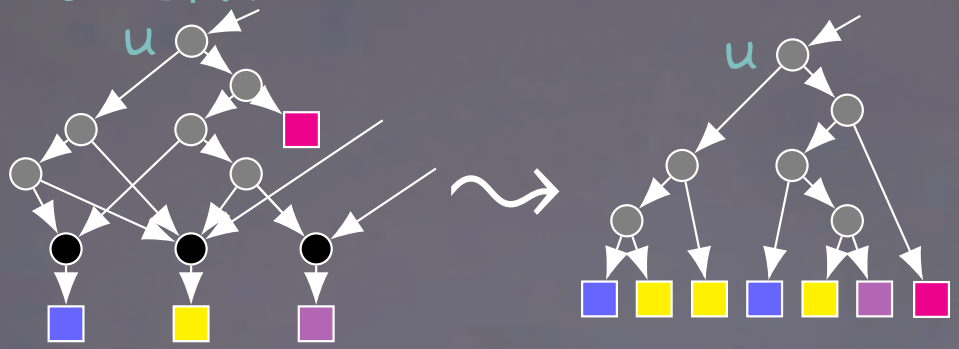


Tree

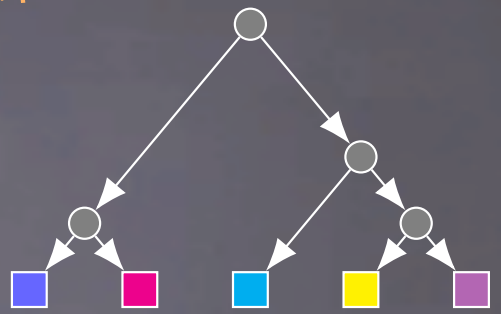


For Networks...

Network



Tree

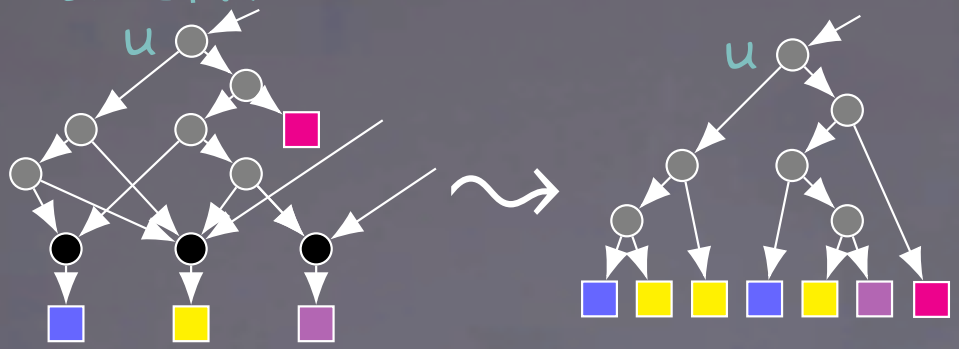


Recall:

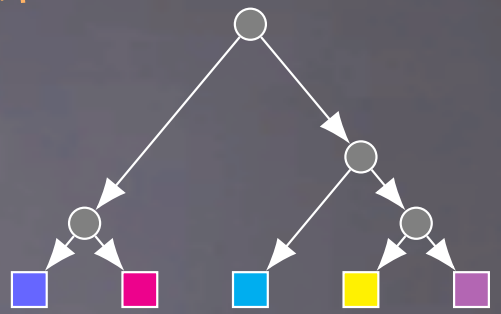
Can get highest T_v displayed by N_u in $O(|N_u| \cdot (\Delta_N \Delta_T)^2)$

For Networks...

Network



Tree



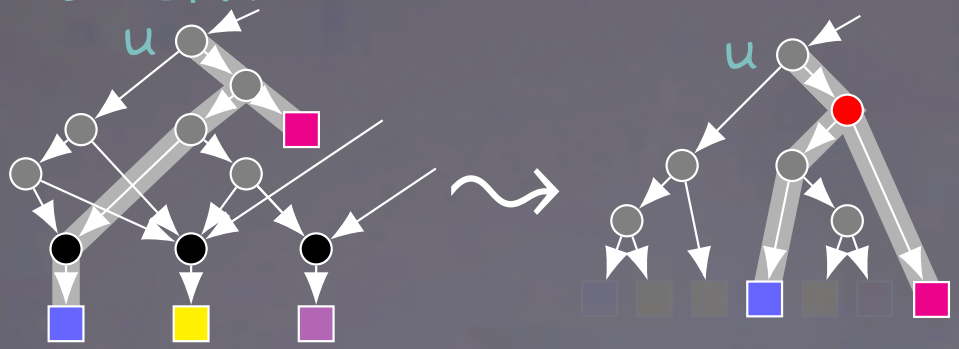
Recall:

Can get highest T_v displayed by N_u in $O(|N_u| \cdot (\Delta_N \Delta_T)^2)$

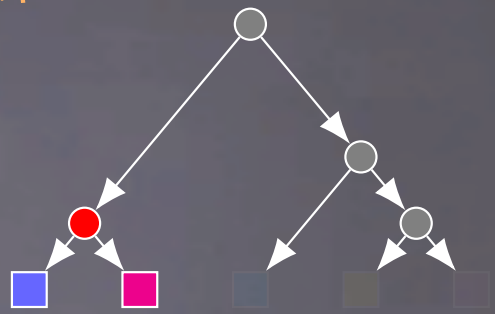
Problem
Which T_v to choose?

For Networks...

Network



Tree



Recall:

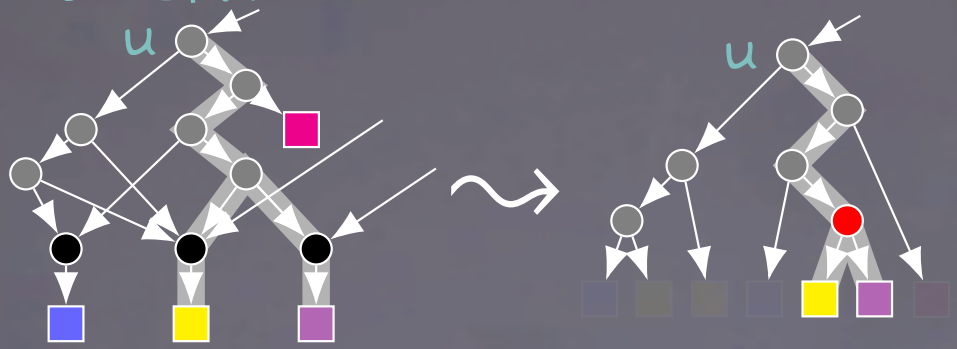
Can get highest T_v displayed by N_u in $O(|N_u| \cdot (\Delta_N \Delta_T)^2)$

Problem

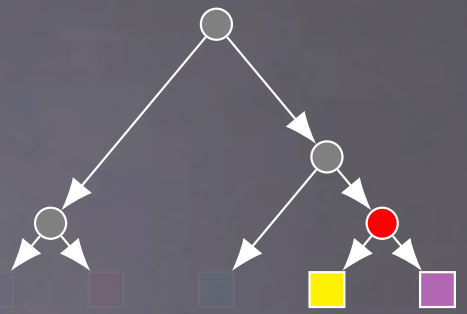
Which T_v to choose?

For Networks...

Network



Tree



Recall:

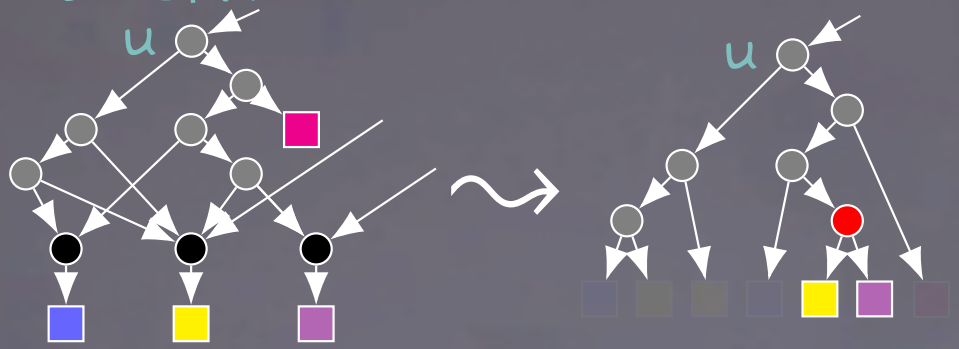
Can get highest T_v displayed by N_u in $O(|N_u| \cdot (\Delta_N \Delta_T)^2)$

Problem

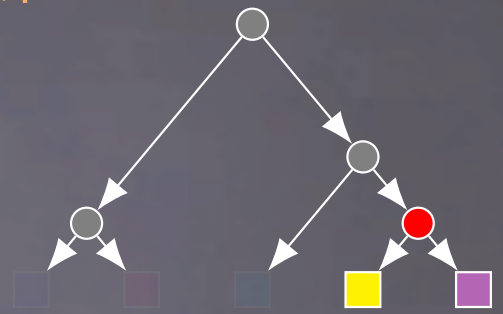
Which T_v to choose?

For Networks...

Network



Tree



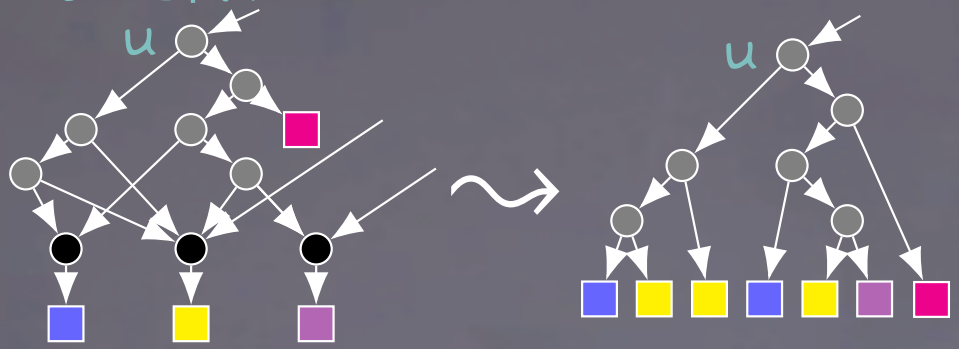
Recall:

Can get highest T_v displayed by N_u in $O(|N_u| \cdot (\Delta_N \Delta_T)^2)$

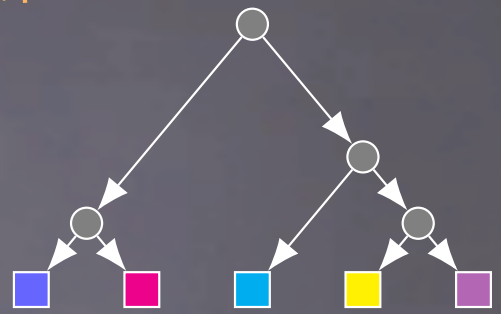
Choose the T_v containing the leaves that are accessible only via u!

For Networks...

Network



Tree



Recall:

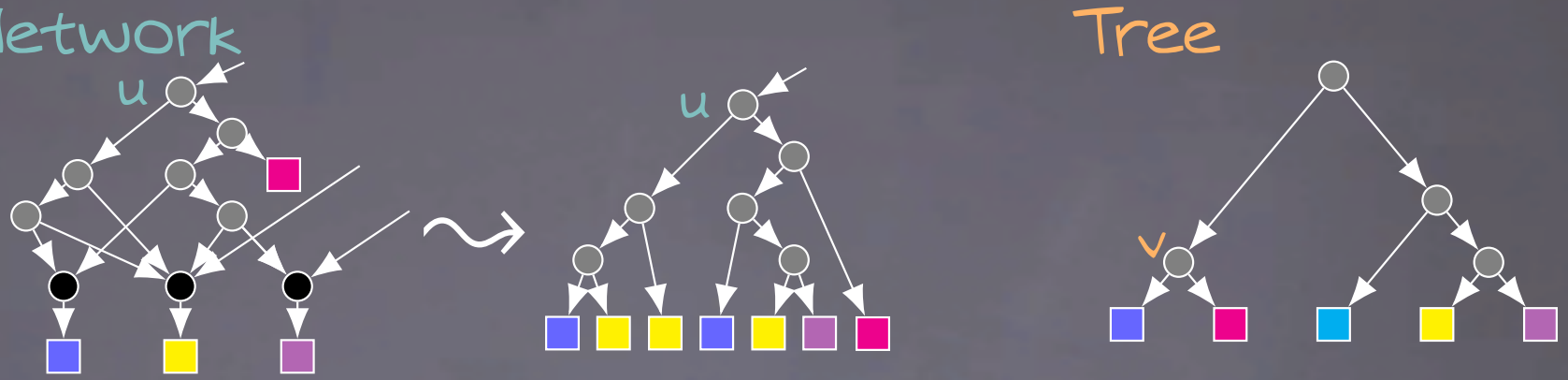
Can get highest T_v displayed by N_u in $O(|N_u| \cdot (\Delta_N \Delta_T)^2)$

Definition

u is visible on ■ $\Leftrightarrow$ all root-■-paths contain u

Choose the T_v containing the leaves that are accessible only via u!

For Networks...



Recall:

Can get highest T_v displayed by N_u in $O(|N_u| \cdot (\Delta_N \Delta_T)^2)$

Definition

u is visible on ■ $\Leftrightarrow$ all root-■-paths contain u

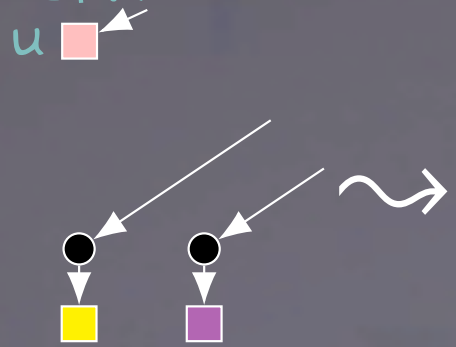
Reduction Rule

Root u of lowest tree-component N_u in N is visible on ■ $\rightsquigarrow$ collapse $N_u \notin$ the highest T_v with ■ $\in T_v$ that N_u displays.

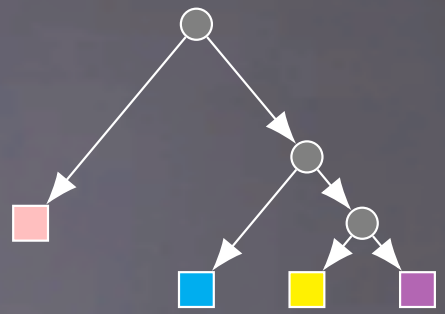
Choose the T_v containing the leaves that are accessible only via u!

For Networks...

Network



Tree



Recall:

Can get highest T_v displayed by N_u in $O(|N_u| \cdot (\Delta_N \Delta_T)^2)$

Definition

u is visible on ■ $\Leftrightarrow$ all root-■-paths contain u

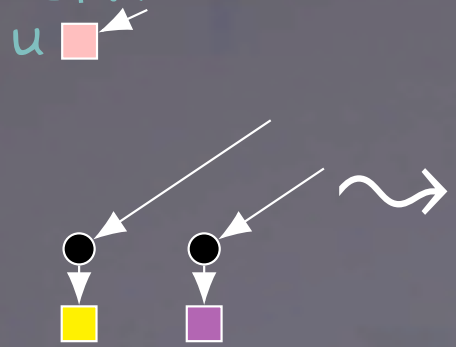
Reduction Rule

Root u of lowest tree-component N_u in N is visible on ■ $\rightsquigarrow$ collapse $N_u \notin$ the highest T_v with ■ $\in T_v$ that N_u displays.

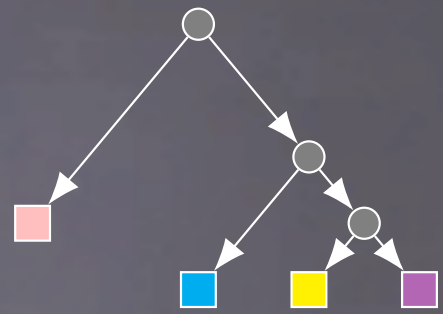
Choose the T_v containing the leaves that are accessible only via u!

For Networks...

Network



Tree



Recall:

Can get highest T_v displayed by N_u in $O(|N_u| \cdot (\Delta_N \Delta_T)^2)$

Definition

u is [illegible] all root-■-paths contain u

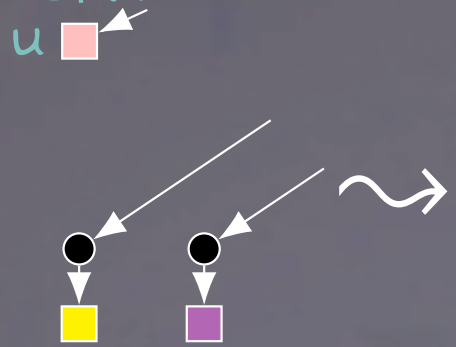
ree-component N_u
collapse N_u ⇜ the
that N_u displays.

Theorem
all tree-component-roots are visible
⇝ solve TC in $O(|N|(c\Delta_N\Delta_T)^2)$
where c = longest reticulation chain

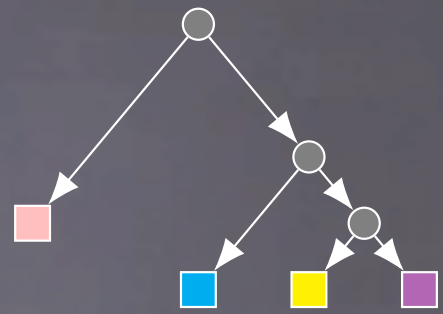
Choose the T_v containing the leaves that are accessible only via u!

For Networks...

Network



Tree



Recall:

Can get highest T_v displayed by N_u in $O(|N_u| \cdot (\Delta_N \Delta_T)^2)$

Definition

u is … a … u

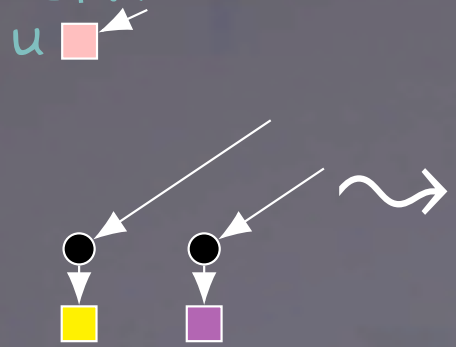
Theorem
all tree-component-roots are visible
$\rightsquigarrow$ solve TC in $O(|N|(c\Delta_N\Delta_T)^2)$
where c = longest reticulation chain

Corollary
solve Bin. TC in $O(|N|)$ on
(a) reticulation visible
(all reti. visible)
$\not\Leftarrow$
(b) nearly-stable
(invisible $\Rightarrow$ visible parents)

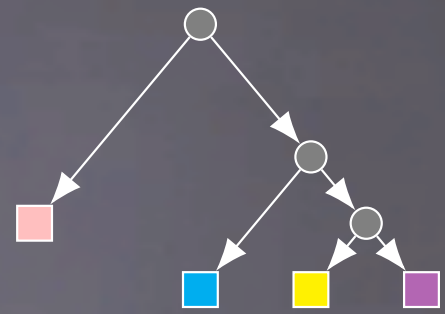
Choose the T_v containing the leaves that are accessible only via u!

For Networks...

Network



Tree



Recall:

Can get highest T_v displayed by N_u in $O(|N_u| \cdot (\Delta_N$

Definition

u is a u

Theorem
all tree-component-roots are visible
$\rightsquigarrow$ solve TC in $O(|N|(c\Delta_N\Delta_T)^2)$
where c = longest reticulation chain

Corollary
solve Bin. TC in $O(|N|)$ on
(a) reticulation visible
(all reti. visible)
$\not\Leftarrow$
(B) nearly-stable
(invisible $\Rightarrow$ visible parents)

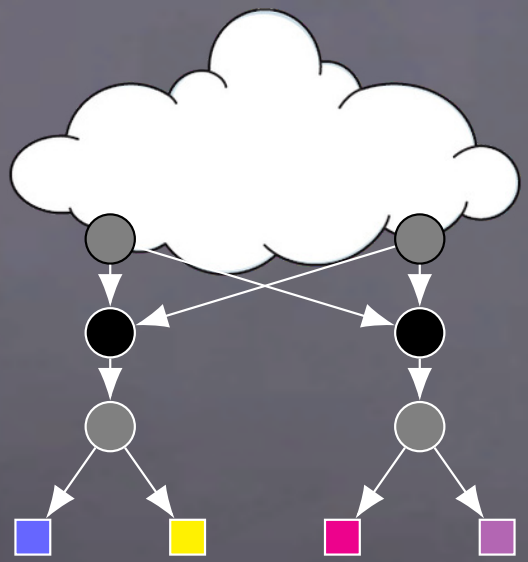
Corollary
TC can be solved in poly time on networks N in which all tree-vertices with reticulation parents are visible.

General Networks

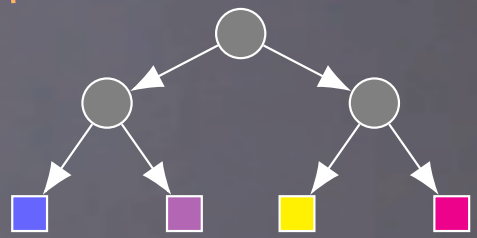
Idea: $u \in M(v)$

$\Leftrightarrow$ N_u displays T_v

Network



Tree



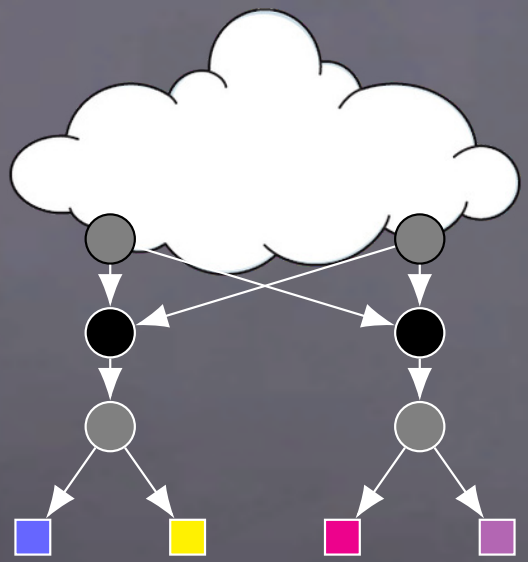
General Networks

Idea: $u \in M(v)$

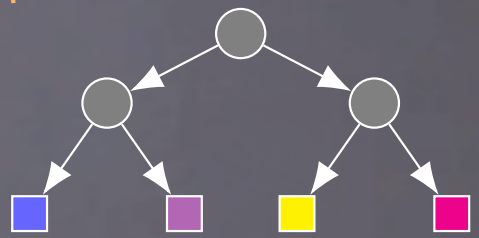
$\Leftrightarrow$ N_u displays T_v

Problem:
Avoid using tree-components multiple times!

Network



Tree



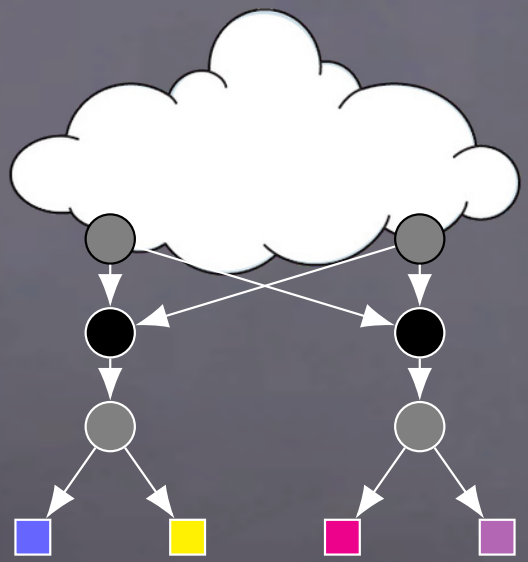
General Networks

Idea: $u \in M(v, R)$

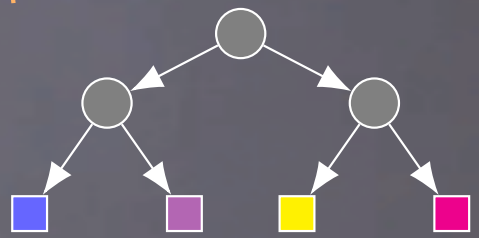
$\Leftrightarrow$ N_u displays T_v using invisible component roots R

Problem:
Avoid using tree-components multiple times!

Network



Tree



General Networks

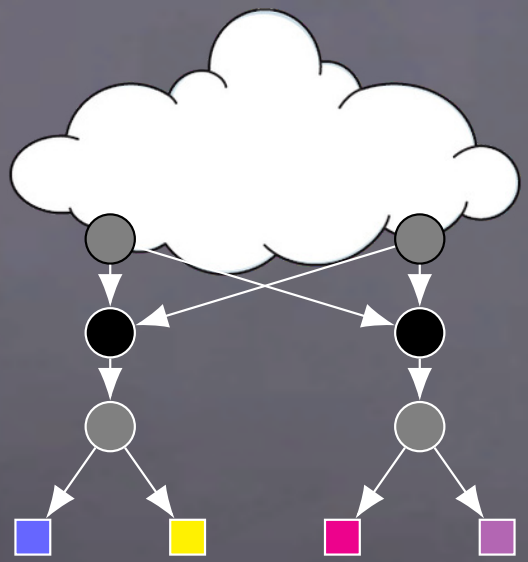
Idea: $u \in M(v, R)$

$\Leftrightarrow$ N_u displays T_v using invisible component roots R

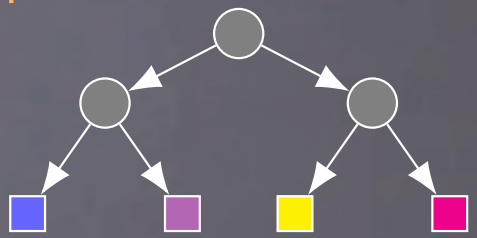
$\Leftrightarrow$ R can be partitoned into $R_1 \uplus R_2 \uplus \ldots \uplus R_{\deg(v)}$ s.t. bipartite graph with edge set $\{u_i v_j \mid u_i \in M(v_j, R_j)\}$ has matching of size $\deg(v)$

Problem: Avoid using tree-components multiple times!

Network



Tree



General Networks

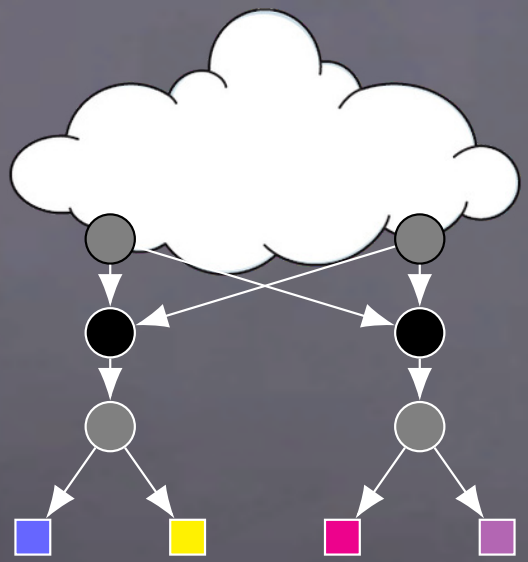
Idea: $u \in M(v, R)$

$\Leftrightarrow$ N_u displays T_v using invisible component roots R

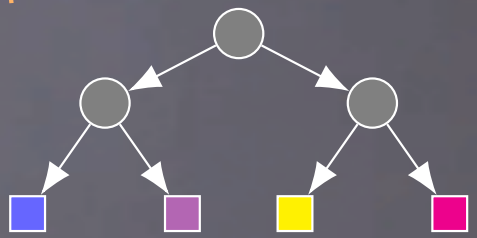
$\Leftrightarrow$ R can be partitoned into $R_1 \uplus R_2 \uplus \ldots \uplus R_{\deg(v)}$ s.t. bipartite graph with edge set $\{u_i v_j \mid u_i \in M(v_j, R_j)\}$ has matching of size $\deg(v)$

Problem: Avoid using tree-components multiple times!

Network



Tree



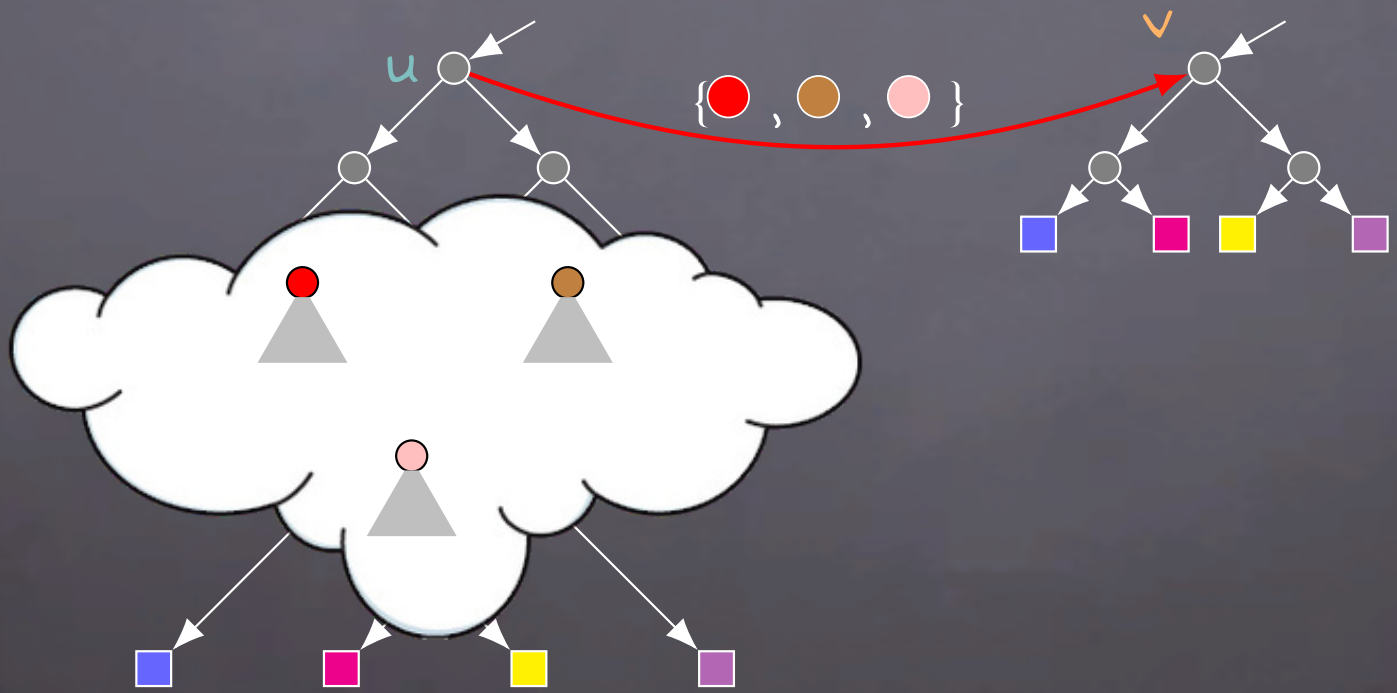
General Networks

Idea: $u \in M(v, R)$

$\Leftrightarrow$ N_u displays T_v using invisible component roots R

$\Leftrightarrow$ R can be partitoned into $R_1 \uplus R_2 \uplus \ldots \uplus R_{\deg(v)}$ s.t. bipartite graph with edge set $\{u_i v_j \mid u_i \in M(v_j, R_j)\}$ has matching of size $\deg(v)$

Problem:
Avoid using tree-components multiple times!



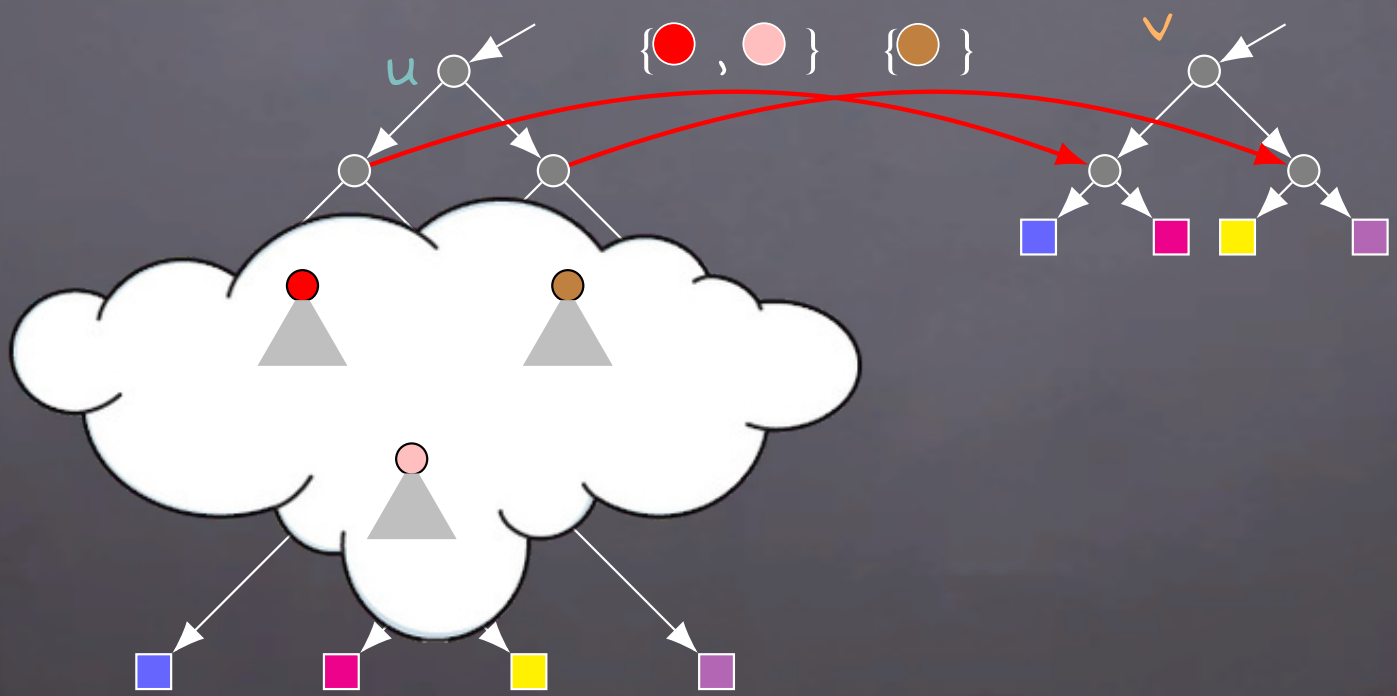
General Networks

Idea: $u \in M(v, R)$

$\Leftrightarrow$ N_u displays T_v using invisible component roots R

$\Leftrightarrow$ R can be partitoned into $R_1 \uplus R_2 \uplus \ldots \uplus R_{\deg(v)}$ s.t. bipartite graph with edge set $\{u_i v_j \mid u_i \in M(v_j, R_j)\}$ has matching of size $\deg(v)$

Problem:
Avoid using tree-components multiple times!



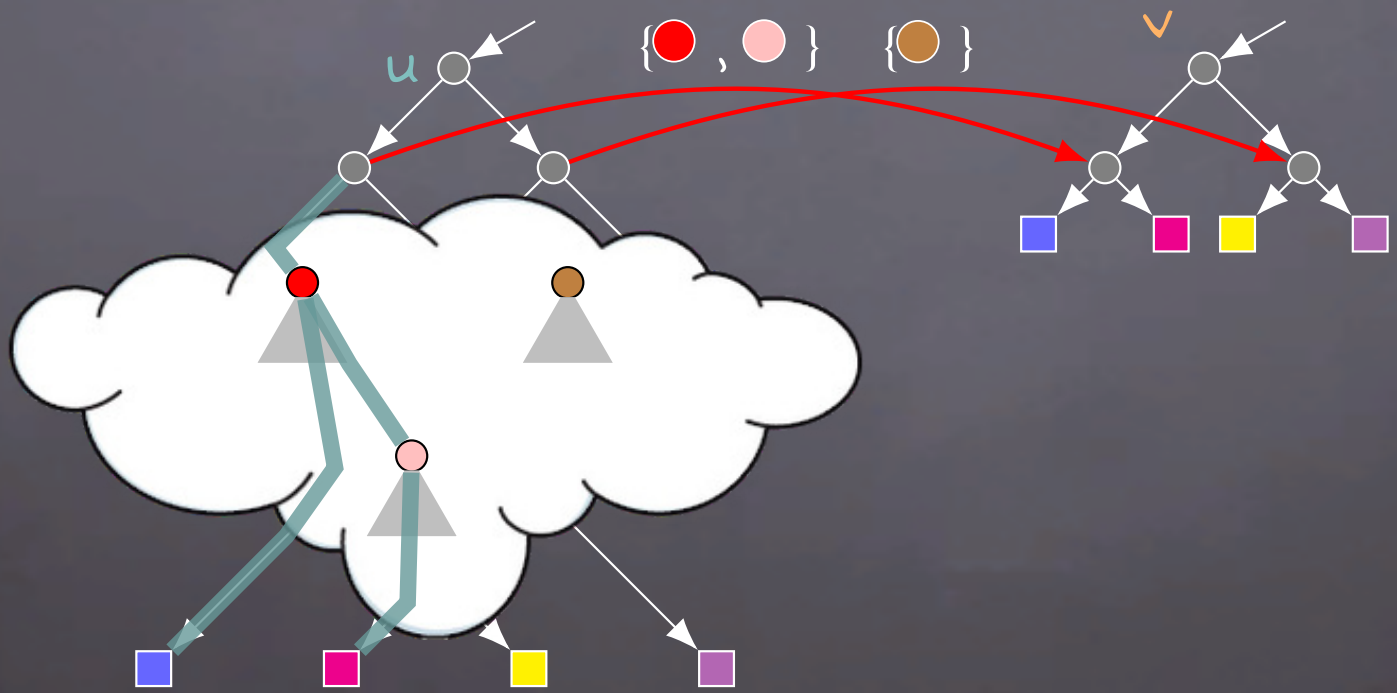
General Networks

Idea: $u \in M(v, R)$

$\Leftrightarrow$ N_u displays T_v using invisible component roots R

$\Leftrightarrow$ R can be partitoned into $R_1 \uplus R_2 \uplus \ldots \uplus R_{\deg(v)}$ s.t. bipartite graph with edge set $\{u_i v_j \mid u_i \in M(v_j, R_j)\}$ has matching of size $\deg(v)$

Problem:
Avoid using tree-components multiple times!



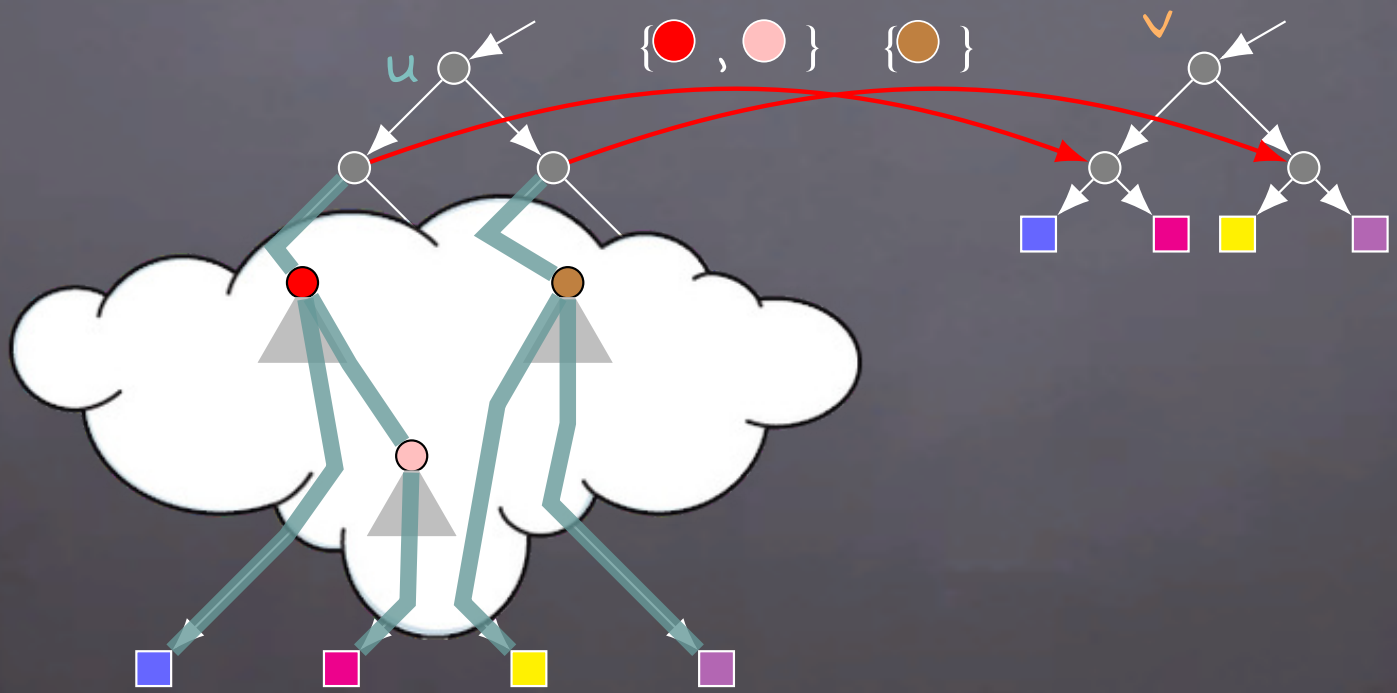
General Networks

Idea: $u \in M(v, R)$

$\Leftrightarrow$ N_u displays T_v using invisible component roots R

$\Leftrightarrow$ R can be partitoned into $R_1 \uplus R_2 \uplus \ldots \uplus R_{\deg(v)}$ s.t. bipartite graph with edge set $\{u_i v_j \mid u_i \in M(v_j, R_j)\}$ has matching of size $\deg(v)$

Problem:
Avoid using tree-components multiple times!



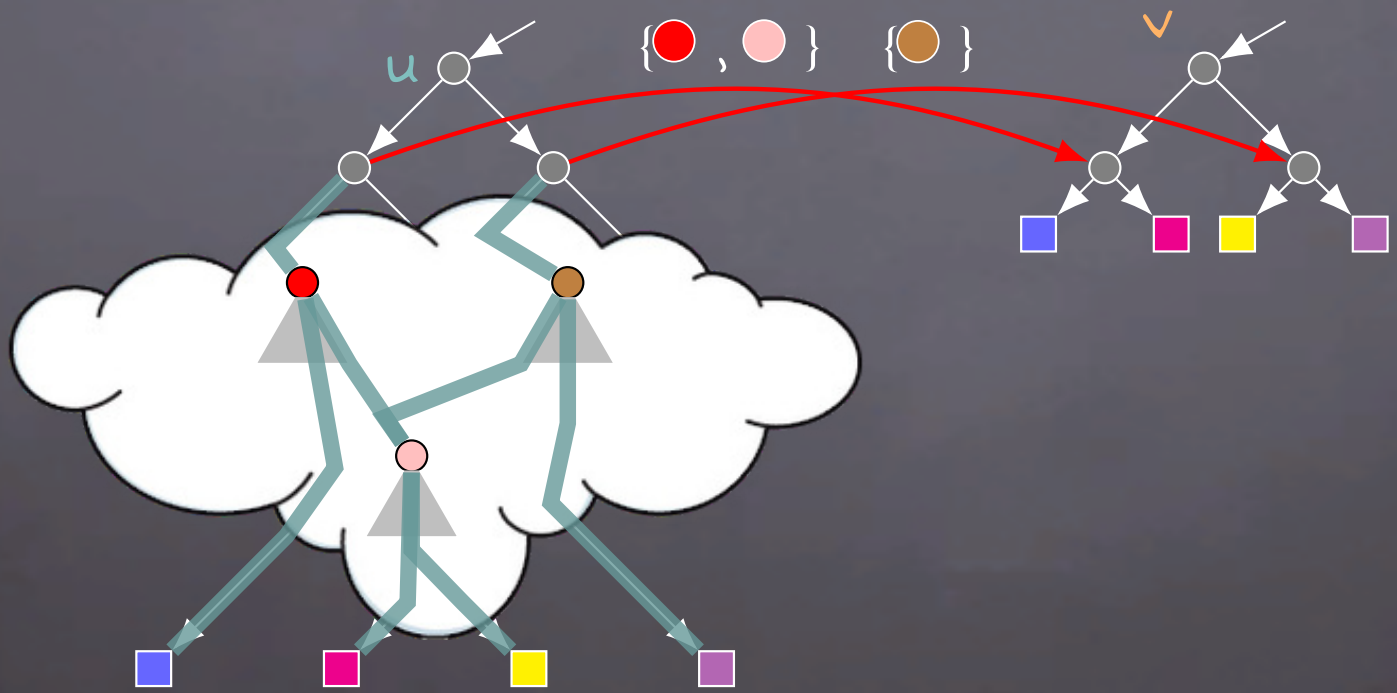
General Networks

Idea: $u \in M(v, R)$

$\Leftrightarrow$ N_u displays T_v using invisible component roots R

$\Leftrightarrow$ R can be partitoned into $R_1 \uplus R_2 \uplus \ldots \uplus R_{\deg(v)}$ s.t. bipartite graph with edge set $\{u_i v_j \mid u_i \in M(v_j, R_j)\}$ has matching of size $\deg(v)$

Problem: Avoid using tree-components multiple times!



General Networks

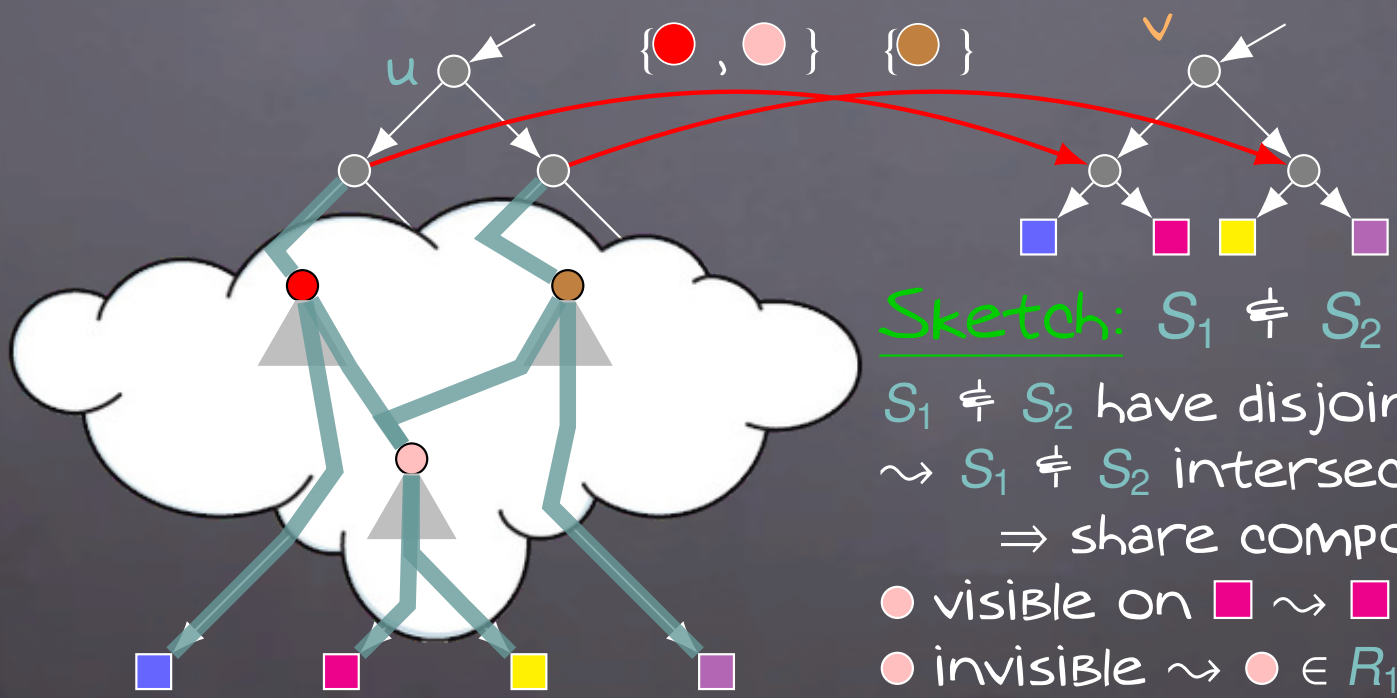
Idea: $u \in M(v, R)$

$\Leftrightarrow$ N_u displays T_v using invisible component roots R
$\Leftrightarrow$ R can be partitoned into $R_1 \uplus R_2 \uplus \ldots \uplus R_{\deg(v)}$ s.t. bipartite graph with edge set $\{u_i v_j \mid u_i \in M(v_j, R_j)\}$ has matching of size $\deg(v)$

Problem: Avoid using tree-components multiple times!

Definition

Subdivision S of T_v is nice $\Leftrightarrow$
for each $u \in S$, S contains all leaves that u is visible on.



Sketch: $S_1 \lneq S_2$ disjoint

$S_1 \lneq S_2$ have disjoint leaf-sets
$\rightsquigarrow$ $S_1 \lneq S_2$ intersect
$\Rightarrow$ share component-root ●
● visible on ■ $\rightsquigarrow$ ■ $\in S_1 \lneq S_2$ ↯
● invisible $\rightsquigarrow$ ● $\in R_1 \lneq R_2$ ↯

General Networks

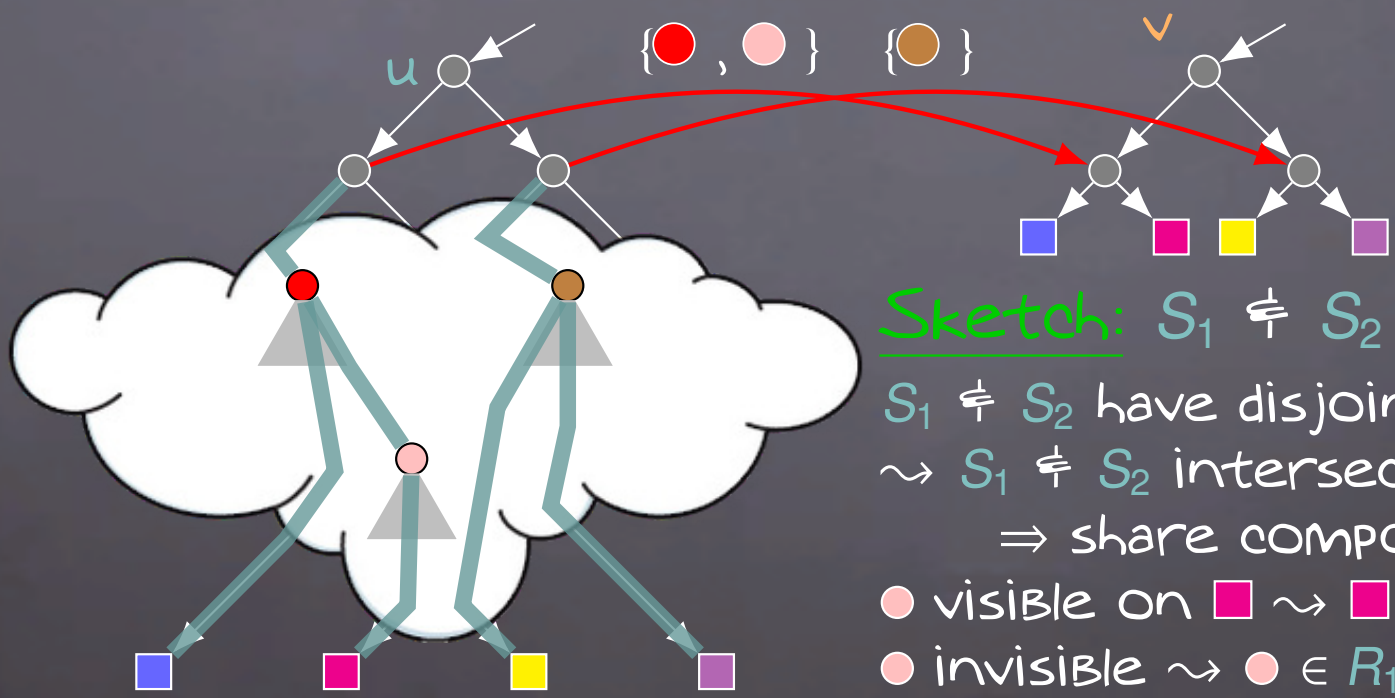
Idea: $u \in M(v, R)$

$\Leftrightarrow$ N_u displays T_v using invisible component roots R
$\Leftrightarrow$ R can be partitoned into $R_1 \uplus R_2 \uplus \ldots \uplus R_{\deg(v)}$ s.t. bipartite graph with edge set $\{u_i v_j \mid u_i \in M(v_j, R_j)\}$ has matching of size $\deg(v)$

Problem: Avoid using tree-components multiple times!

Definition

Subdivision S of T_v is nice $\Leftrightarrow$
for each $u \in S$, S contains all leaves that u is visible on.



Sketch: $S_1 \not\leftrightarrows S_2$ disjoint

$S_1 \not\leftrightarrows S_2$ have disjoint leaf-sets
$\rightsquigarrow$ $S_1 \not\leftrightarrows S_2$ intersect
$\Rightarrow$ share component-root ●
● visible on ■ $\rightsquigarrow$ ■ $\in S_1 \not\leftrightarrows S_2$ ↯
● invisible $\rightsquigarrow$ ● $\in R_1 \not\leftrightarrows R_2$ ↯

General Networks

Idea: $u \in M(v, R)$

$\Leftrightarrow$ N_u displays T_v using invisible component roots R
$\Leftrightarrow$ R can be partitoned into $R_1 \uplus R_2 \uplus \ldots \uplus R_{\deg(v)}$ s.t. bipartite graph with edge set $\{u_i v_j \mid u_i \in M(v_j, R_j)\}$ has matching of size $\deg(v)$

Problem: Avoid using tree-components multiple times!

Definition

Subdivision S of T_v is nice $\Leftrightarrow$
for each $u \in S$, S contains all leaves that u is visible on.

Sketch: $S_1 \lessdot S_2$ disjoint

$S_1 \lessdot S_2$ have disjoint leaf-sets
$\rightsquigarrow$ $S_1 \lessdot S_2$ intersect
$\Rightarrow$ share component-root ●
● visible on ■ $\rightsquigarrow$ ■ $\in S_1 \lessdot S_2$ ↯
● invisible $\rightsquigarrow$ ● $\in R_1 \lessdot R_2$ ↯


Part II:
Soft
Poly-
tomies

Recall:
Bin. resolution of N displays
Bin. resolution of T

Hardness For MUL-Trees

2-Union Independent Set

Input: collection $G = (V, E_1)$ of disjoint P_3s,
collection $H = (V, E_2)$ of ℓ disjoint cliques

Question: Does $G + H$ have a size-ℓ independent set?

Hardness For MUL-Trees

k-SAT:
≤ k-cliques

2-Union Independent Set

Input: collection $G = (V, E_1)$ of disjoint P_3s,
collection $H = (V, E_2)$ of ℓ disjoint cliques

Question: Does $G + H$ have a size-ℓ independent set?

Hardness For MUL-Trees

k-SAT:
≤ k-cliques

2-Union Independent Set

Input: collection $G = (V, E_1)$ of disjoint P_3s,
ℓ-coloring of vertices

Question: Does G have a size-ℓ colorful independent set?

Hardness For MUL-Trees

k-SAT:
each color
$\leq k$ x

2-Union Independent Set

Input: collection $G = (V, E_1)$ of disjoint P_3s,
ℓ-coloring of vertices

Question: Does G have a size-ℓ colorful independent set?

Hardness For MUL-Trees

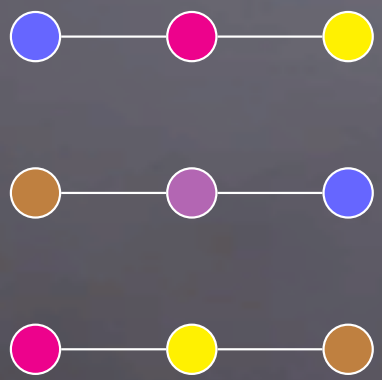
k-SAT:
each color
$\leq k$x

2-Union Independent Set

Input: collection $G = (V, E_1)$ of disjoint P_3s,
ℓ-coloring of vertices

Question: Does G have a size-ℓ colorful independent set?

Example



Hardness For MUL-Trees

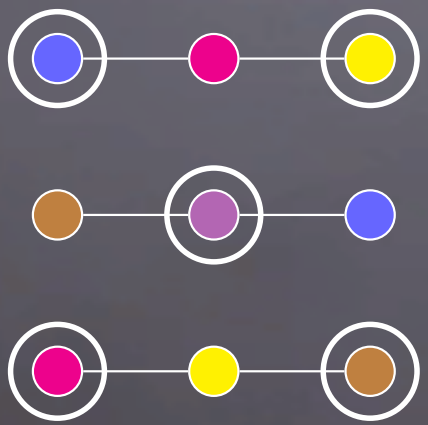
k-SAT:
each color
$\leq k$ x

2-Union Independent Set

Input: collection $G = (V, E_1)$ of disjoint P_3s,
ℓ-coloring of vertices

Question: Does G have a size-ℓ colorful independent set?

Example



Hardness For MUL-Trees

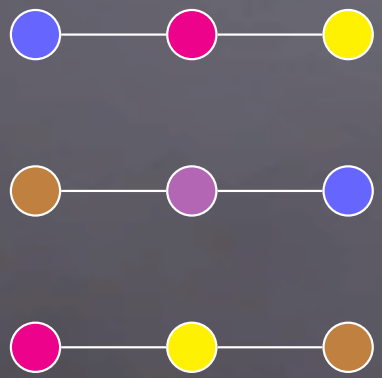
k-SAT:
each color
$\leq k$ x

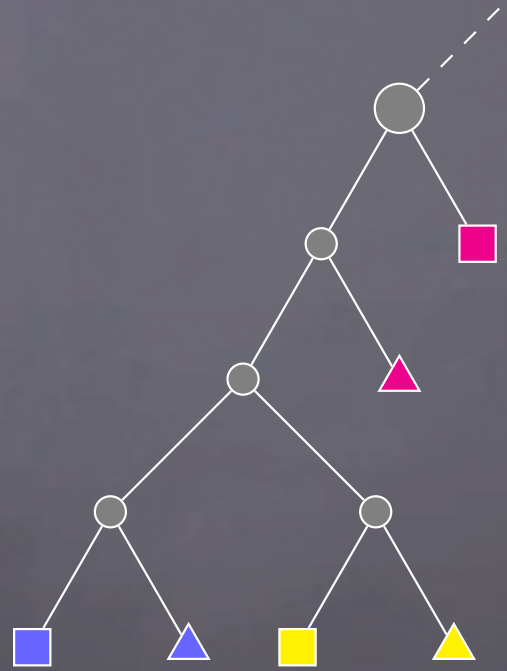
2-Union Independent Set

Input: collection $G = (V, E_1)$ of disjoint P_3s,
ℓ-coloring of vertices

Question: Does G have a size-ℓ colorful independent set?

Example





Hardness For MUL-Trees

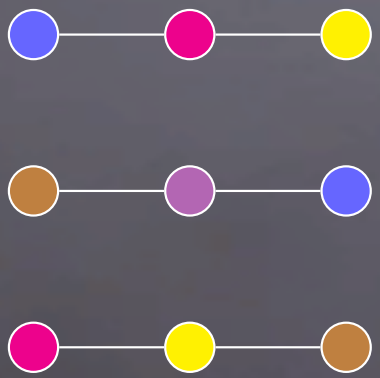
k-SAT:
each color
$\leq k$ x

2-Union Independent Set

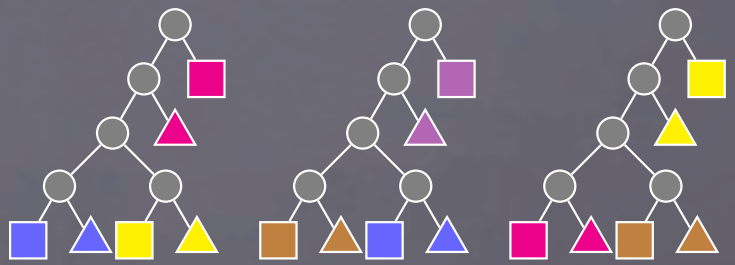
Input: collection $G = (V, E_1)$ of disjoint P_3s,
ℓ-coloring of vertices

Question: Does G have a size-ℓ colorful independent set?

Example



N



Hardness For MUL-Trees

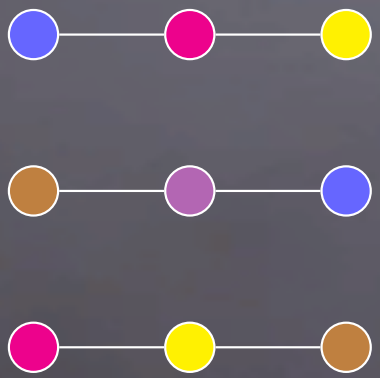
k-SAT:
each color
$\leq k$ x

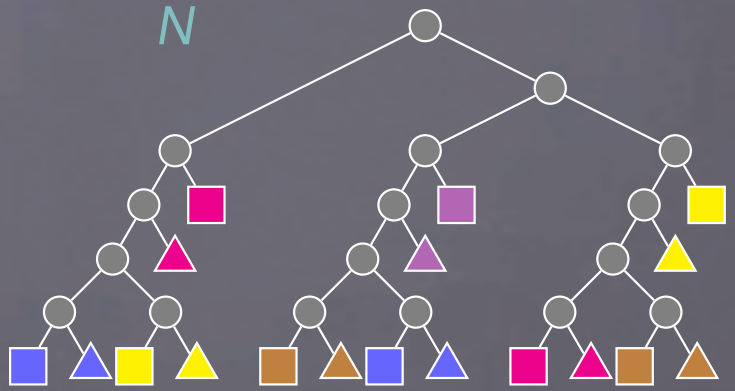
2-Union Independent Set

Input: collection $G = (V, E_1)$ of disjoint P_3s,
ℓ-coloring of vertices

Question: Does G have a size-ℓ colorful independent set?

Example





Hardness For MUL-Trees

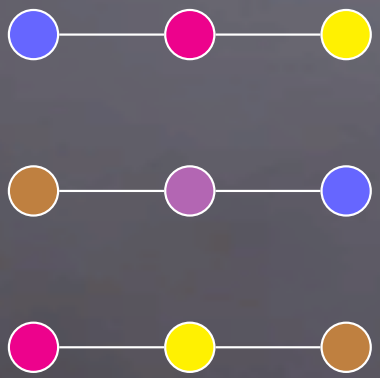
k-SAT:
each color
$\leq k$x

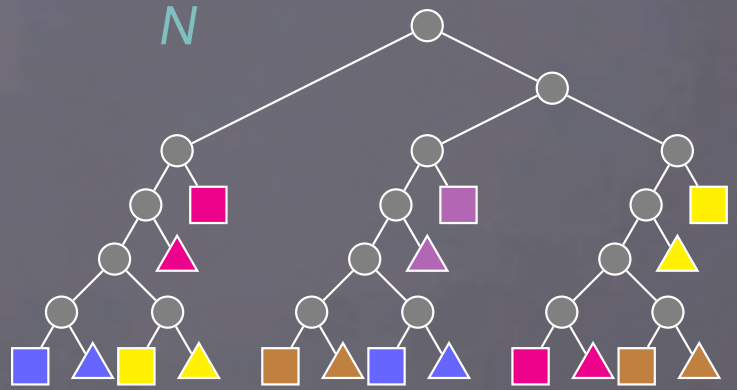
2-Union Independent Set

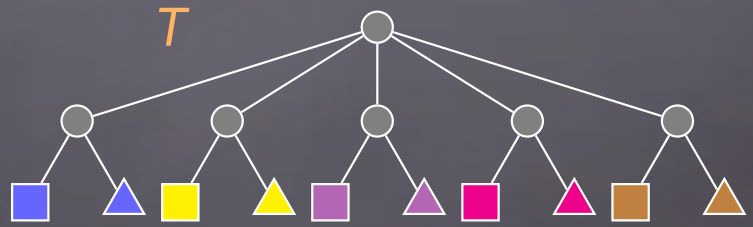
Input: collection $G = (V, E_1)$ of disjoint P_3s,
ℓ-coloring of vertices

Question: Does G have a size-ℓ colorful independent set?

Example







Hardness For MUL-Trees

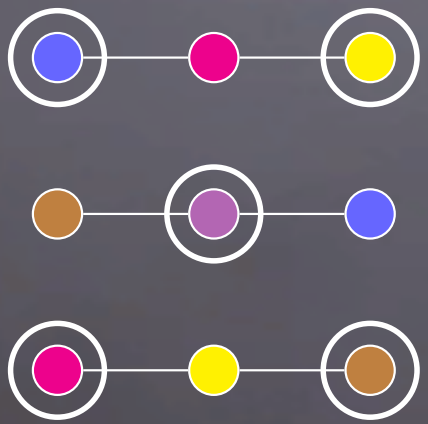
k-SAT:
each color
$\leq k$x

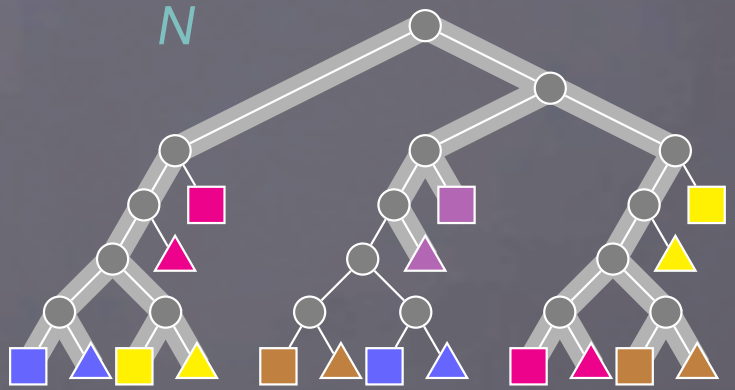
2-Union Independent Set

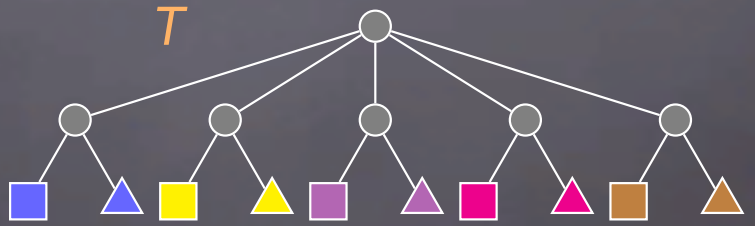
Input: collection $G = (V, E_1)$ of disjoint P_3s,
ℓ-coloring of vertices

Question: Does G have a size-ℓ colorful independent set?

Example







Hardness For MUL-Trees

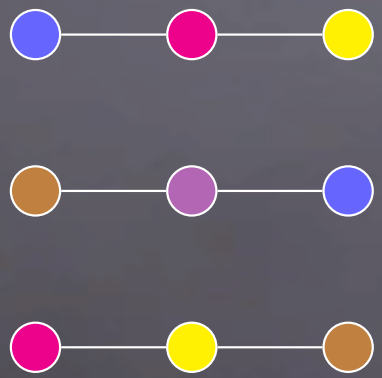
k-SAT:
each color
$\leq k$ x

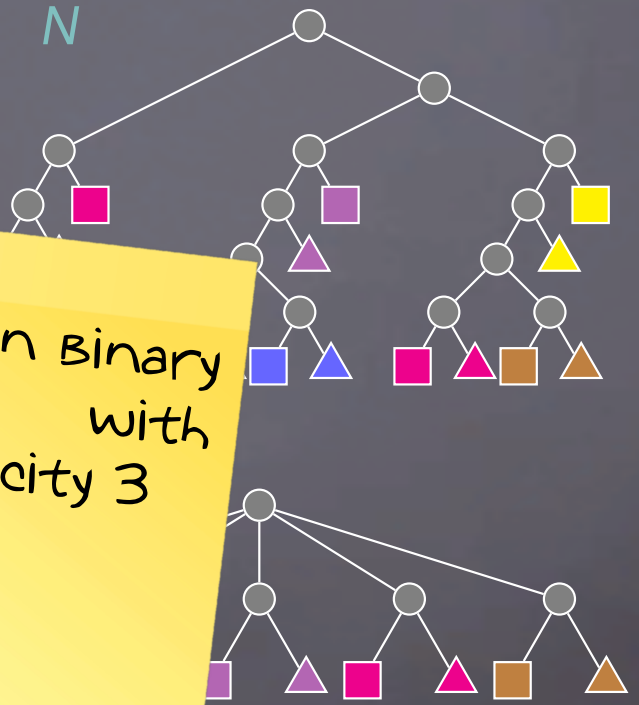
2-Union Independent Set

Input: collection $G = (V, E_1)$ of disjoint P_3s,
ℓ-coloring of vertices

Question: Does G have a size-ℓ colorful independent set?

Example





Theorem
Soft TC on binary mul-trees with label-multiplicity 3
$\geq$
3-SAT

Hardness For MUL-Trees

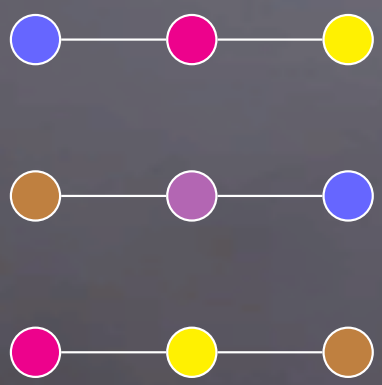
k-SAT:
each color
$\leq k$ times

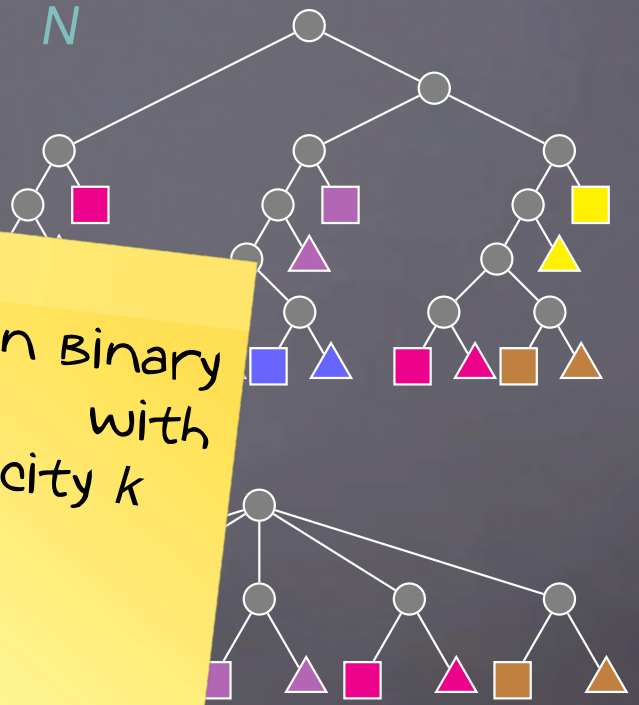
2-Union Independent Set

Input: collection $G = (V, E_1)$ of disjoint P_3s,
ℓ-coloring of vertices

Question: Does G have a size-ℓ colorful independent set?

Example





Theorem
Soft TC on binary mul-trees with label-multiplicity k
$\geq$
k-SAT

Hardness For MUL-Trees

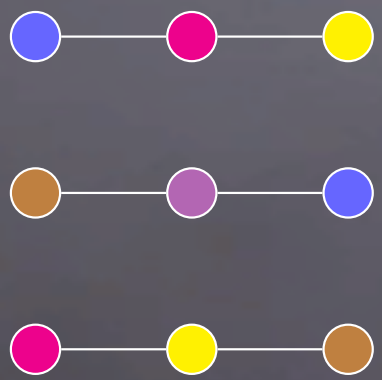
k-SAT:
each color
$\leq k$ x

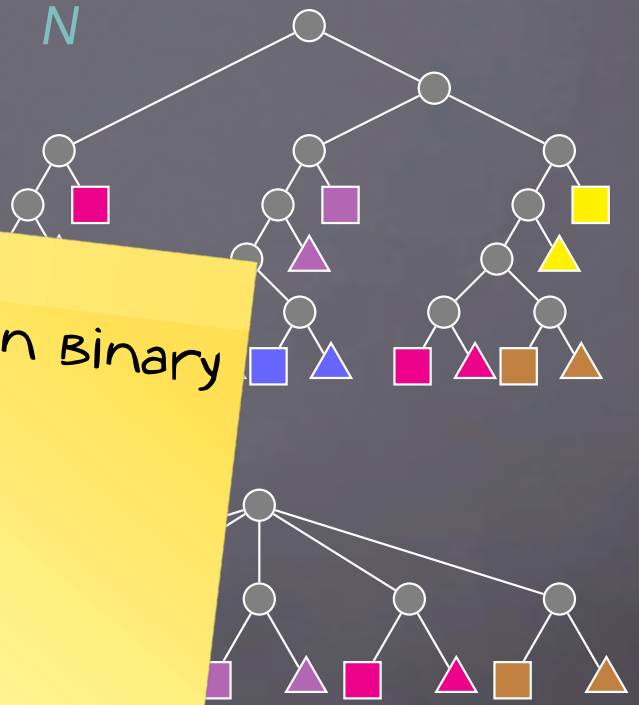
2-Union Independent Set

Input: collection $G = (V, E_1)$ of disjoint P_3s,
ℓ-coloring of vertices

Question: Does G have a size-ℓ colorful independent set?

Example





Theorem
Soft TC on Binary mul-trees

$\geq$

CNF-SAT

The Polynomial Case: 2-Labeled Trees

Observation

"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v))$

The Polynomial Case: 2-Labeled Trees

Observation

"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \neq \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$

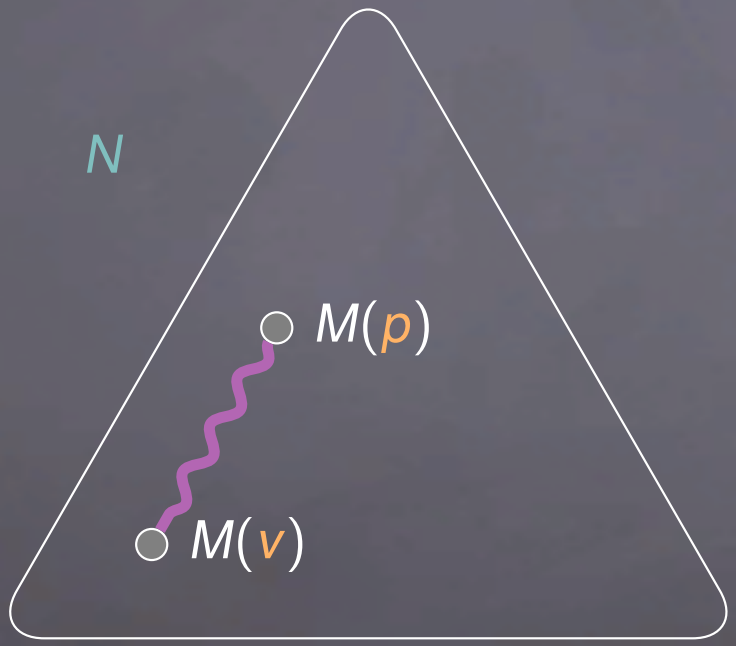
The Polynomial Case: 2-Labeled Trees

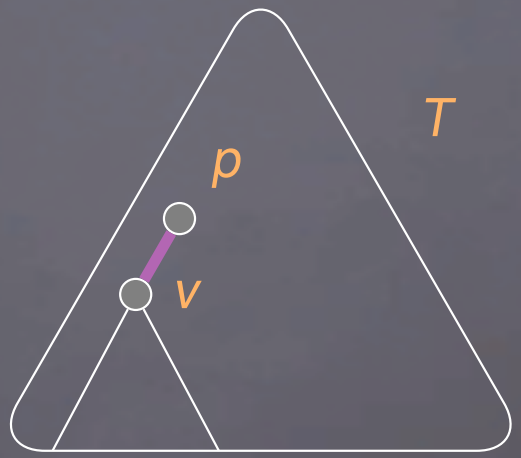
Observation

"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \neq \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$





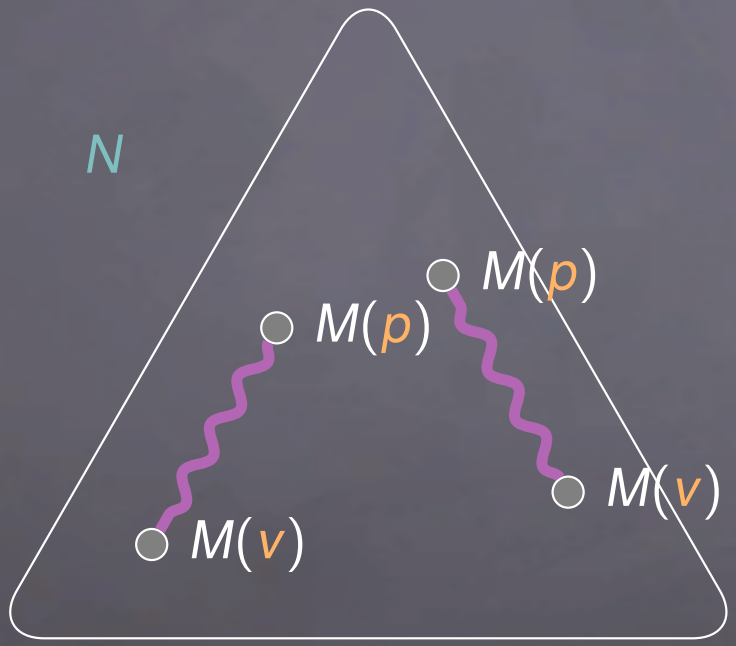
The Polynomial Case: 2-Labeled Trees

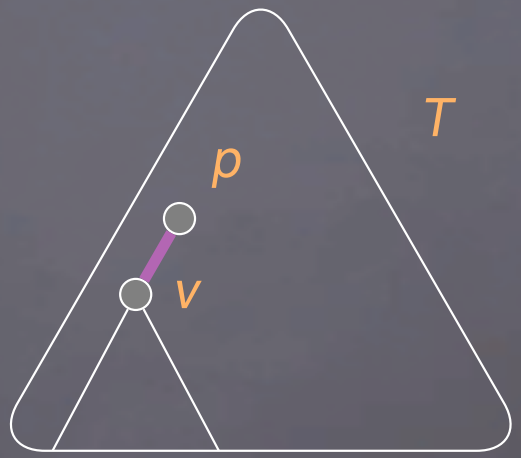
Observation

"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \neq \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$





The Polynomial Case: 2-Labeled Trees

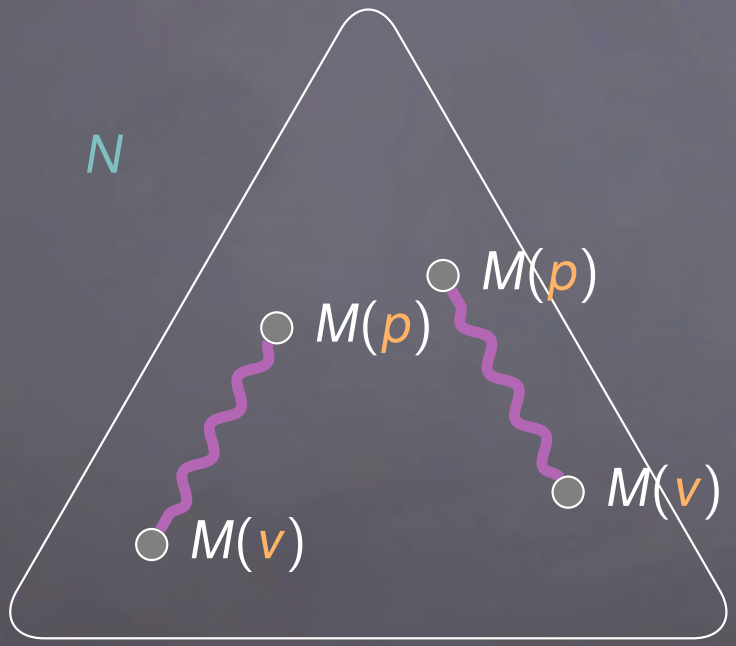
Observation

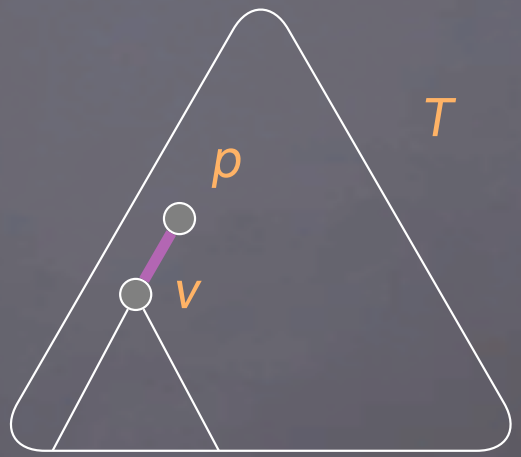
"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \notin \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$





The Polynomial Case: 2-Labeled Trees

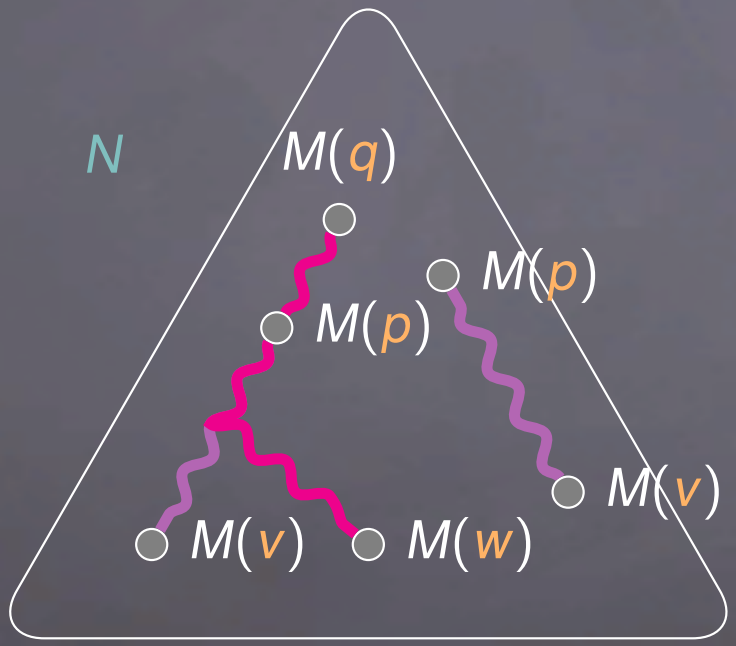
Observation

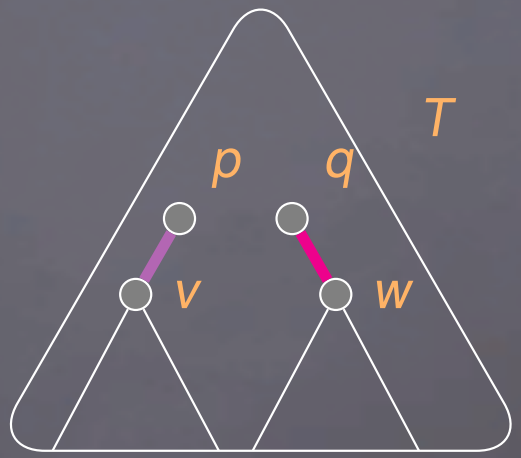
"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \notin \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$





The Polynomial Case: 2-Labeled Trees

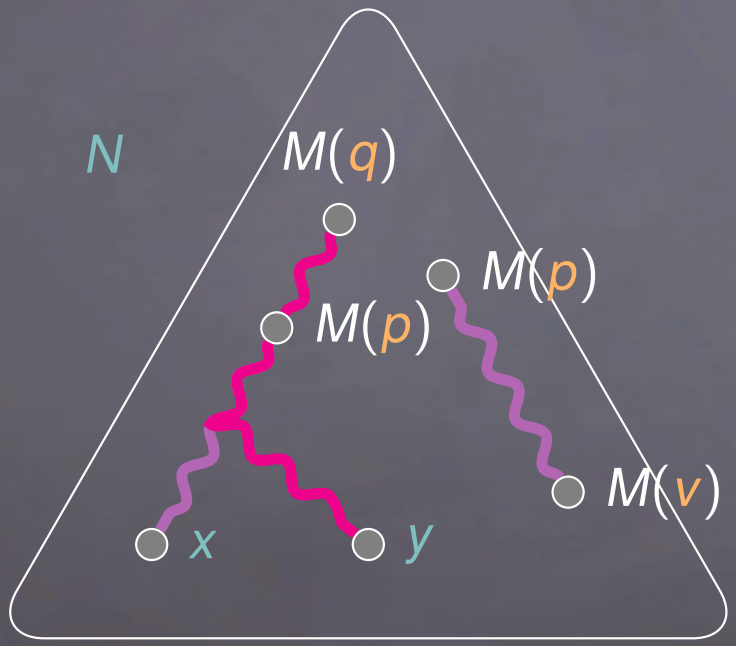
Observation

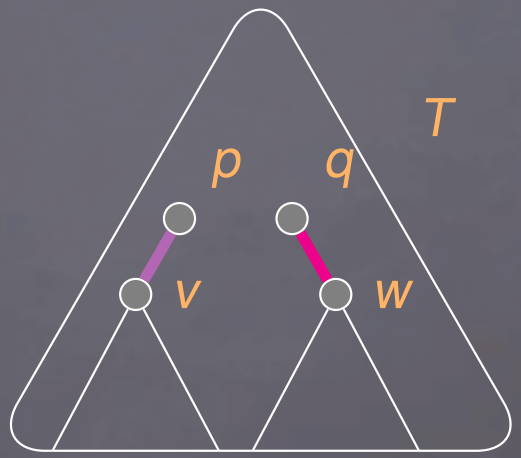
"solution" $S \rightsquigarrow$ map "$\to_S$" with $v \to_S \mathrm{LCA}_S(\mathcal{L}(v)) \notin \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$





Lemma

asc. paths $p_{vx} \notin p_{wy}$ overlap $\notin v \to_S x, w \to_S y \Rightarrow v, w$ siblings in T

The Polynomial Case: 2-Labeled Trees

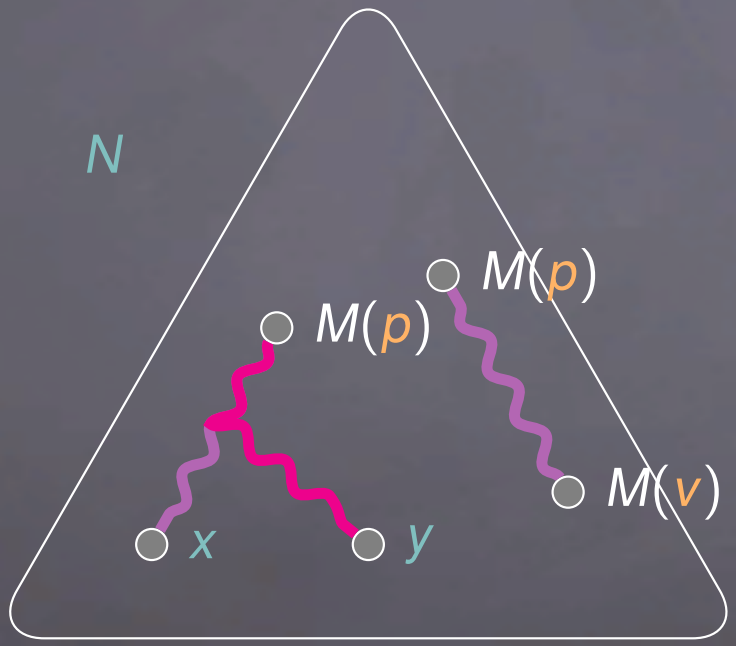
Observation

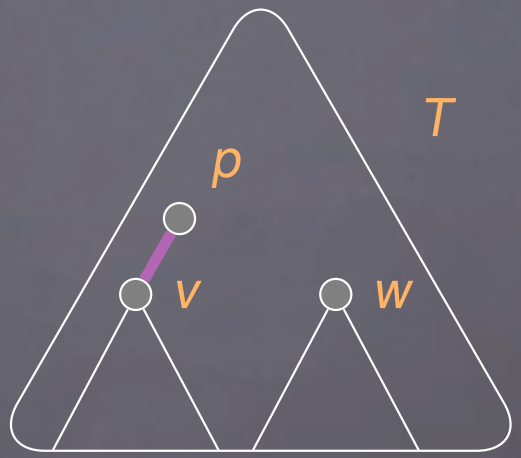
"solution" $S \rightsquigarrow$ map "$\to_S$" with $v \to_S \mathrm{LCA}_S(\mathcal{L}(v)) \notin \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$





Lemma

asc. paths $p_{vx} \notin p_{wy}$ overlap $\notin v \to_S x, w \to_S y \Rightarrow v, w$ siblings in T

The Polynomial Case: 2-Labeled Trees

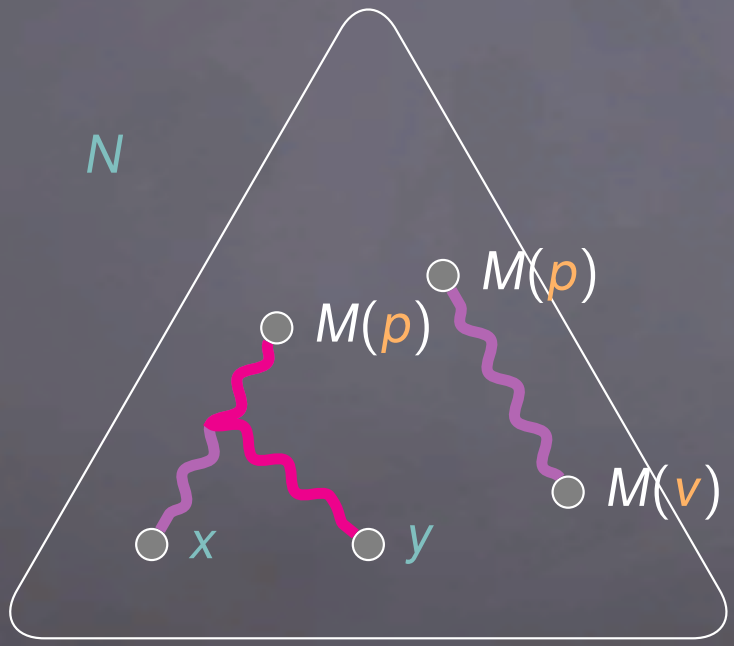
Observation

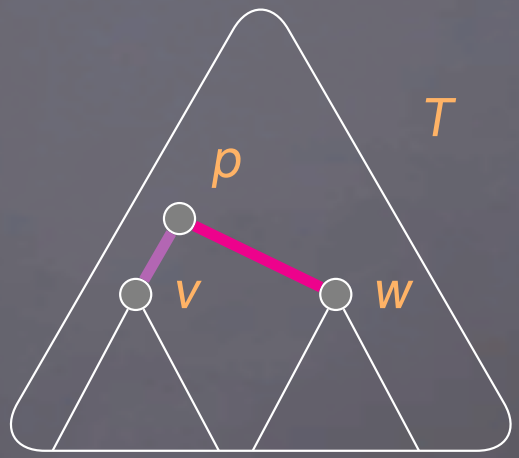
"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \neq \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$





Lemma

asc. paths $p_{vx} \not\parallel p_{wy}$ overlap $\not\parallel$ v, w not siblings in $T \Rightarrow v \not\mapsto_S x$ or $w \not\mapsto_S y$

The Polynomial Case: 2-Labeled Trees

Observation

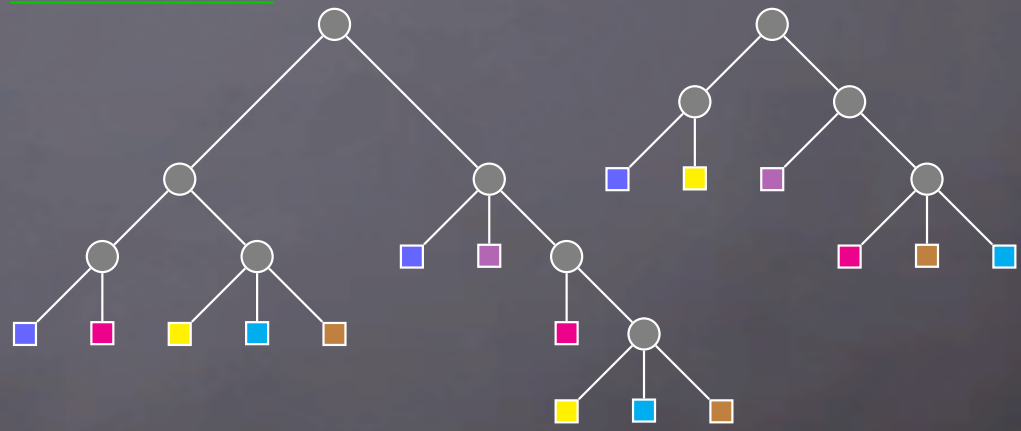
"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \notin \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$

Example



Lemma

asc. paths $p_{vx} \notin p_{wy}$ overlap $\notin v, w$ not siblings in $T \Rightarrow v \not\rightarrow_S x$ or $w \not\rightarrow_S y$

The Polynomial Case: 2-Labeled Trees

Observation

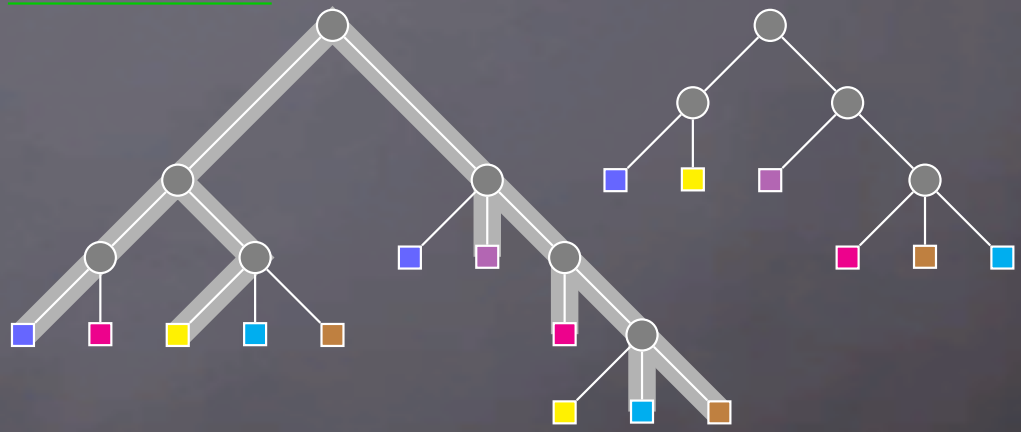
"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \notin \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$

Example



Lemma

asc. paths $p_{vx} \notin p_{wy}$ overlap $\notin v, w$ not siblings in $T \Rightarrow v \not\rightarrow_S x$ or $w \not\rightarrow_S y$

The Polynomial Case: 2-Labeled Trees

Observation

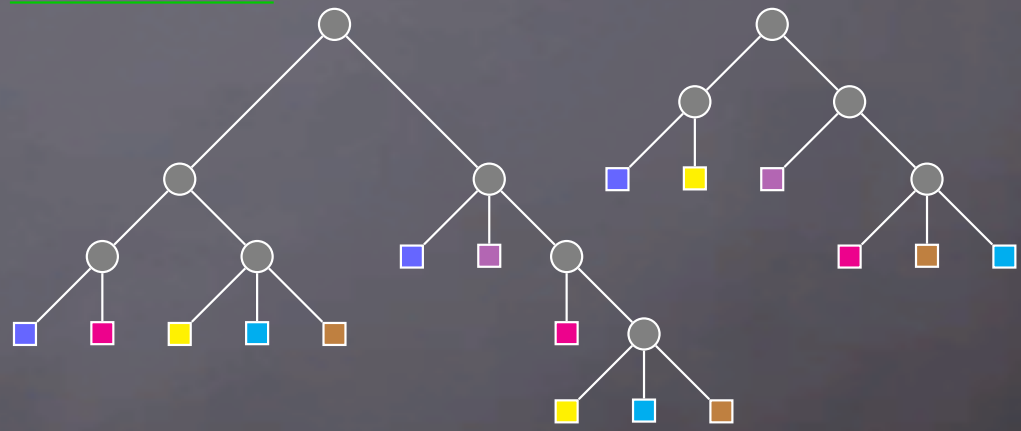
"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \notin \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$

Example



Lemma

asc. paths $p_{vx} \notin p_{wy}$ overlap $\notin v, w$ not siblings in $T \Rightarrow v \not\rightarrow_S x$ or $w \not\rightarrow_S y$

The Polynomial Case: 2-Labeled Trees

Observation

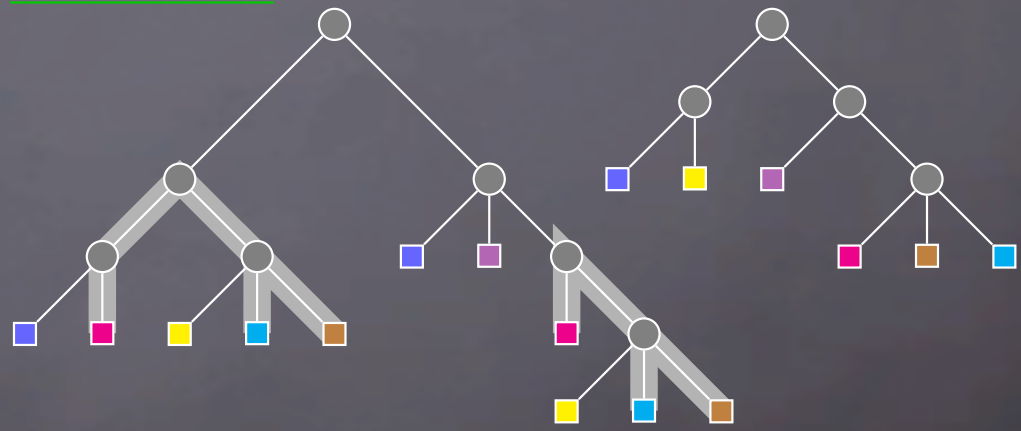
"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \notin \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$

Example



Lemma

asc. paths $p_{vx} \notin p_{wy}$ overlap $\notin v, w$ not siblings in $T \Rightarrow v \not\rightarrow_S x$ or $w \not\rightarrow_S y$

The Polynomial Case: 2-Labeled Trees

Observation

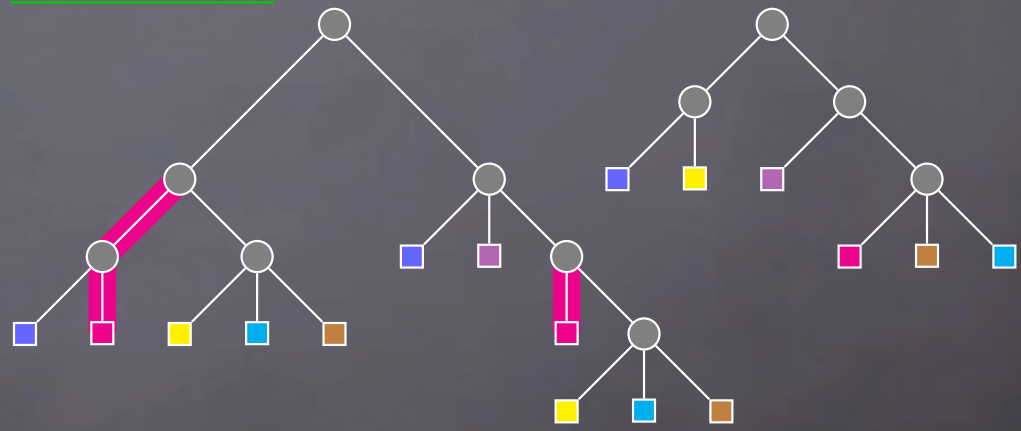
"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \notin \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$

Example



Lemma

asc. paths $p_{vx} \notin p_{wy}$ overlap $\notin v, w$ not siblings in $T \Rightarrow v \not\rightarrow_S x$ or $w \not\rightarrow_S y$

The Polynomial Case: 2-Labeled Trees

Observation

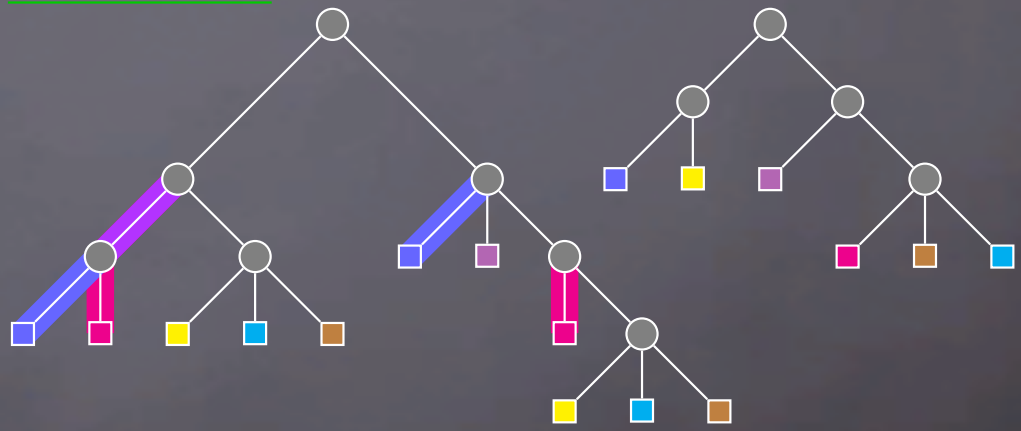
"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \notin \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$

Example



Lemma

asc. paths $p_{vx} \notin p_{wy}$ overlap $\notin v, w$ not siblings in $T \Rightarrow v \not\rightarrow_S x$ or $w \not\rightarrow_S y$

The Polynomial Case: 2-Labeled Trees

Observation

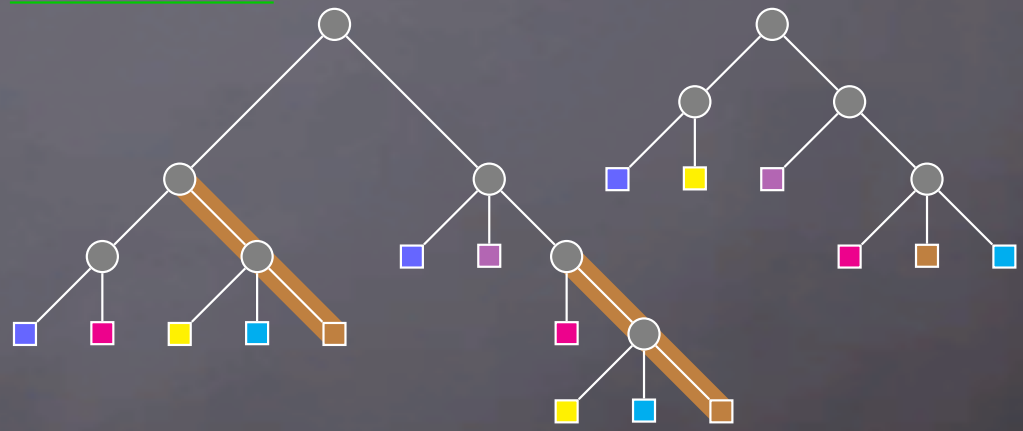
"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \notin \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$

Example



Lemma

asc. paths $p_{vx} \notin p_{wy}$ overlap $\notin v, w$ not siblings in $T \Rightarrow v \not\rightarrow_S x$ or $w \not\rightarrow_S y$

The Polynomial Case: 2-Labeled Trees

Observation

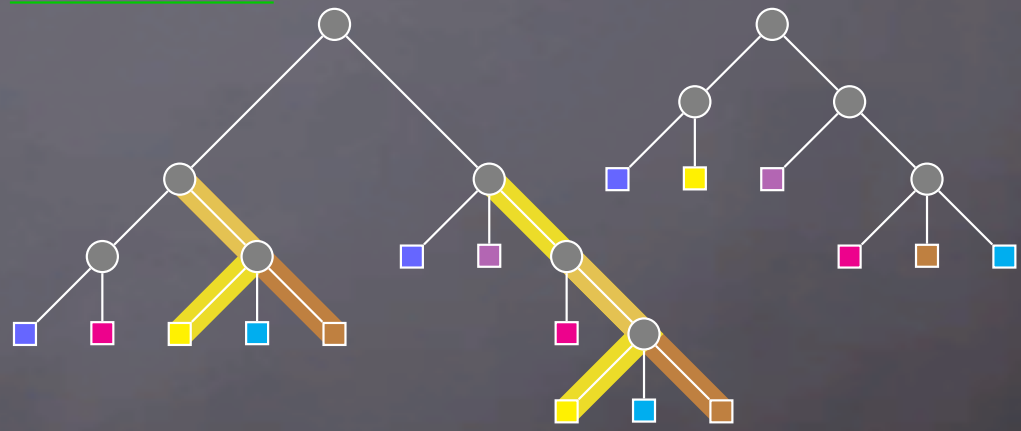
"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \notin \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$

Example



Lemma

asc. paths $p_{vx} \notin p_{wy}$ overlap $\notin$ v, w not siblings in $T \Rightarrow v \not\rightarrow_S x$ or $w \not\rightarrow_S y$

The Polynomial Case: 2-Labeled Trees

Observation

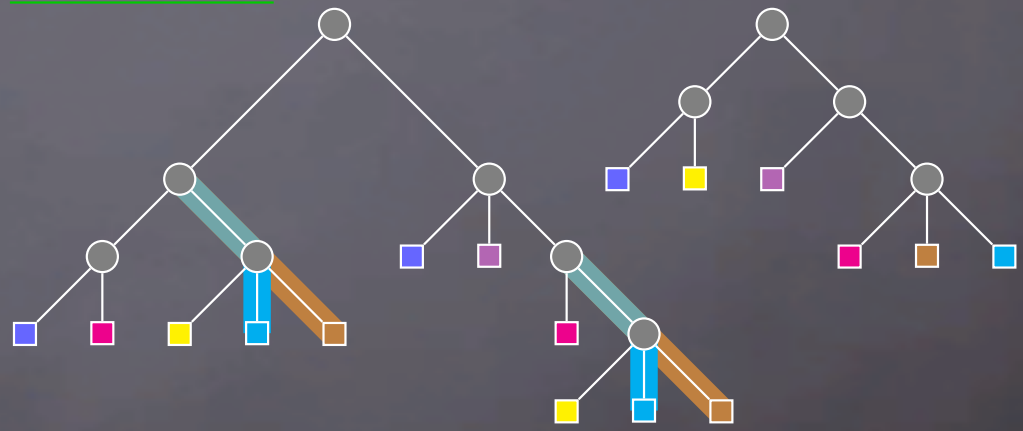
"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \notin \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$

Example



Lemma

asc. paths $p_{vx} \notin p_{wy}$ overlap $\notin$ v, w not siblings in $T \Rightarrow v \not\rightarrow_S x$ or $w \not\rightarrow_S y$

The Polynomial Case: 2-Labeled Trees

Observation

"solution" $S \rightsquigarrow$ map "$\rightarrow_S$" with $v \rightarrow_S \mathrm{LCA}_S(\mathcal{L}(v)) \notin \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

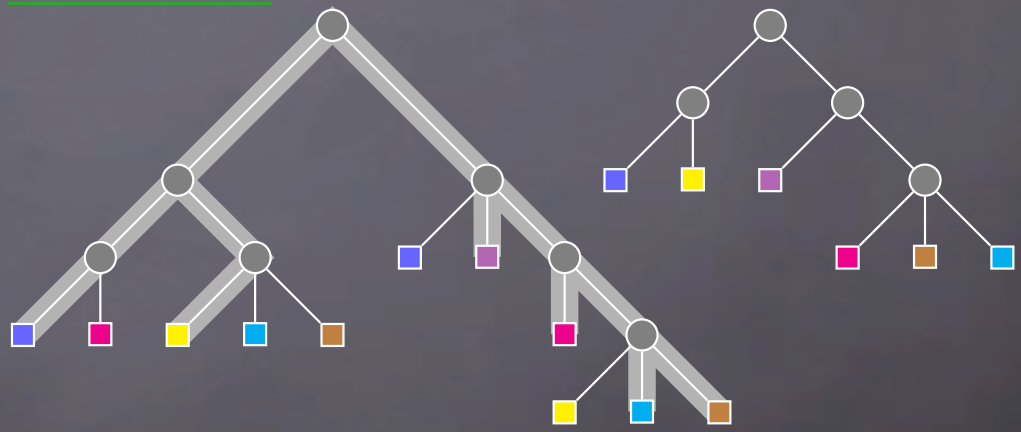
$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$

Observation

For each v, $\rightarrow_S$...

1. ...maps v only once
2. ...maps v above children
3. ...respects asc. paths

Example



Lemma

asc. paths $p_{vx} \notin p_{wy}$ overlap $\notin$ v, w not siblings in $T \Rightarrow v \not\rightarrow_S x$ or $w \not\rightarrow_S y$

The Polynomial Case: 2-Labeled Trees

Observation

"solution" $S \rightsquigarrow$ map "$\to_S$" with $v \to_S \mathrm{LCA}_S(\mathcal{L}(v)) \notin \mathrm{LCA}_S(\mathcal{L}(v)) \in M(v)$

Definition

ascending path p_{vx} = path to $x \in M(v)$ from any $M(p)$

$M(v) := \min\{u \mid N_u \text{ displays } T_v\}$

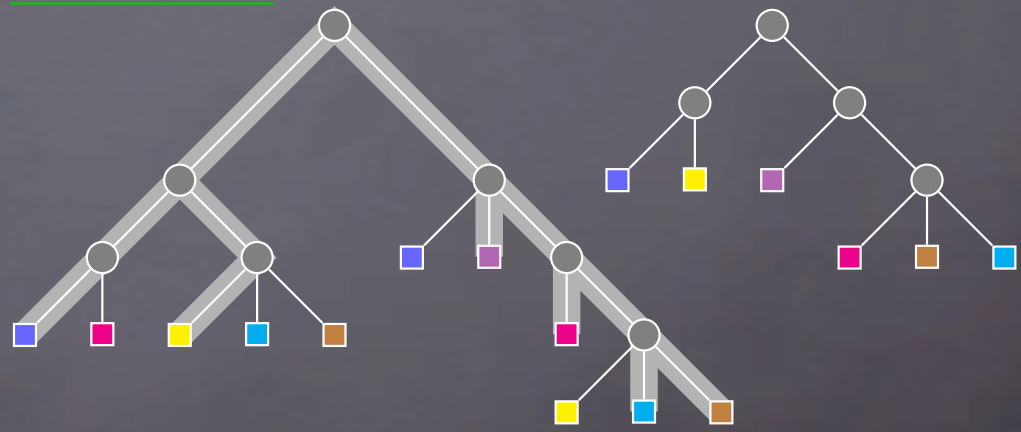
Observation

For each v, $\to_S$...

1. ...maps v only once
2. ...maps v above children
3. ...respects asc. paths

sufficient!

Example



Lemma

asc. paths $p_{vx} \notin p_{wy}$ overlap $\notin$ v, w not siblings in $T \Rightarrow v \not\to_S x$ or $w \not\to_S y$

The Polynomial Case: 2-Labeled Trees

Bottom-up dynamic programming: decide if N_u displays T_ρ

The Polynomial Case: 2-Labeled Trees

Bottom-up dynamic programming: decide if N_u displays T_ρ

For each $v < \rho$, we know $M(v)$

$|M(v)| \leq 2$

The Polynomial Case: 2-Labeled Trees

Bottom-up dynamic programming: decide if N_u displays T_ρ

Construction

construct 2-CNF $\varphi_{\rho \to u} := \bigwedge_{v \in T_\rho} \varphi_v$:

Variables:

For each $v < \rho$, we know $M(v)$

$|M(v)| \leq 2$

The Polynomial Case: 2-Labeled Trees

Bottom-up dynamic programming: decide if N_u displays T_ρ

Construction

construct 2-CNF $\varphi_{\rho \to u} := \bigwedge_{v \in T_\rho} \varphi_v$:

Variables:

- $x_{v \to y}$ for each $y \in M(v)$

For each $v < \rho$, we know $M(v)$

$|M(v)| \leq 2$

The Polynomial Case: 2-Labeled Trees

Bottom-up dynamic programming: decide if N_u displays T_ρ

Construction

construct 2-CNF $\varphi_{\rho \to u} := \bigwedge_{v \in T_\rho} \varphi_v$:

Variables:

- $x_{v \to y}$ for each $y \in M(v)$

Clauses:

For each $v < \rho$, we know $M(v)$

$|M(v)| \leq 2$

The Polynomial Case: 2-Labeled Trees

Bottom-up dynamic programming: decide if N_u displays T_ρ

Construction

construct 2-CNF $\varphi_{\rho \to u} := \bigwedge_{v \in T_\rho} \varphi_v$:

Variables:

- $x_{v \to y}$ for each $y \in M(v)$

Clauses:

1. $\bigoplus_{y \in M(v)} x_{v \to y}$ (v only maps once)

For each $v < \rho$, we know $M(v)$

$|M(v)| \leq 2$

The Polynomial Case: 2-Labeled Trees

Bottom-up dynamic programming: decide if N_u displays T_ρ

Construction

construct 2-CNF $\varphi_{\rho \to u} := \bigwedge_{v \in T_\rho} \varphi_v$:

Variables:

- $x_{v \to y}$ for each $y \in M(v)$

Clauses:

1. $\bigoplus_{y \in M(v)} x_{v \to y}$ (v only maps once)
2. parent p of v: $y \in M(v)$, $z \in M(p)$ w/ $y \not\leq z$:
   $x_{v \to y} \Rightarrow \neg x_{p \to z}$ (map parent above)

For each $v < \rho$, we know $M(v)$

$|M(v)| \leq 2$

The Polynomial Case: 2-Labeled Trees

Bottom-up dynamic programming: decide if N_u displays T_ρ

Construction

construct 2-CNF $\varphi_{\rho \to u} := \bigwedge_{v \in T_\rho} \varphi_v$:

Variables:

- $x_{v \to y}$ for each $y \in M(v)$

Clauses:

1. $\bigoplus_{y \in M(v)} x_{v \to y}$ (v only maps once)
2. parent p of v: $y \in M(v)$, $z \in M(p)$ w/ $y \not\leq z$:
   $x_{v \to y} \Rightarrow \neg x_{p \to z}$ (map parent above)
3. w ¬sibling of v: $y \in M(v)$, $z \in M(w)$ w/ overlapping asc. paths:
   $x_{v \to y} \Rightarrow \neg x_{w \to z}$ (ascending paths)

For each $v < \rho$, we know $M(v)$

$|M(v)| \leq 2$

The Polynomial Case: 2-Labeled Trees

Bottom-up dynamic programming: decide if N_u displays T_ρ

Construction

construct 2-CNF $\varphi_{\rho \to u} := \bigwedge_{v \in T_\rho} \varphi_v$:

Variables:

- $x_{v \to y}$ for each $y \in M(v)$

Clauses:

1. $\bigoplus_{y \in M(v)} x_{v \to y}$ (v only maps once)
2. parent p of v: $y \in M(v)$, $z \in M(p)$ w/ $y \not\leq z$:
   $x_{v \to y} \Rightarrow \neg x_{p \to z}$ (map parent above)
3. w ¬sibling of v: $y \in M(v)$, $z \in M(w)$ w/ overlapping asc. paths:
   $x_{v \to y} \Rightarrow \neg x_{w \to z}$ (ascending paths)

For each $v < \rho$, we know $M(v)$

$|M(v)| \leq 2$

Lemma

$\varphi_{\rho \to u}$ satisfyable $\Leftrightarrow$ N_u displays T_ρ

The Polynomial Case: 2-Labeled Trees

Bottom-up dynamic programming: decide if N_u displays T_ρ

Construction

construct 2-CNF $\varphi_{\rho \to u} := \bigwedge_{v \in T_\rho} \varphi_v$:

Variables:

- $x_{v \to y}$ for each $y \in M(v)$

Clauses:

1. $\bigoplus_{y \in M(v)} x_{v \to y}$ (v only maps once)
2. parent p of v: $y \in M(v)$, $z \in M(p)$ w/ $y \not\leq z$:
   $x_{v \to y} \Rightarrow \neg x_{p \to z}$ (map parent above)
3. w ¬sibling of v: $y \in M(v)$, $z \in M(w)$ w/ overlapping asc. paths:
   $x_{v \to y} \Rightarrow \neg x_{w \to z}$ (ascending paths)

For each $v < \rho$, we know $M(v)$

$|M(v)| \leq 2$

Lemma

$\varphi_{\rho \to u}$ satisfyable $\Leftrightarrow$ N_u displays T_ρ

$\varphi_{\rho \to u}$ can be constructed in $O(|N_u| \cdot |T_\rho|^2)$ time.

The Polynomial Case: 2-Labeled Trees

Bottom-up dynamic programming: decide if N_u displays T_ρ

Construction

construct 2-CNF $\varphi_{\rho \to u} := \bigwedge_{v \in T_\rho} \varphi_v$:

Variables:

- $x_{v \to y}$ for each $y \in M(v)$

Clauses:

1. $\bigoplus_{y \in M(v)} x_{v \to y}$ (v only maps once)
2. parent p of v: $y \in M(v)$, $z \in M(p)$ w/ $y \not\leq z$:
   $x_{v \to y} \Rightarrow \neg x_{p \to z}$ (map parent above)
3. w ¬sibling of v: $y \in M(v)$, $z \in M(w)$ w/ overlappi
   $x_{v \to y} \Rightarrow \neg x_{w \to z}$ (ascending paths)

For each $v < \rho$, we know $M(v)$

$|M(v)| \leq 2$

Theorem

Soft TC on 2-labeled trees in $O(|N|^2 |T|^2)$

Lemma

$\varphi_{\rho \to u}$ satisfyable $\Leftrightarrow$ N_u displays T_ρ

$\varphi_{\rho \to u}$ can be constructed in $O(|N_u| \cdot |T_\rho|^2)$ time.

Conclusion

What we Saw...

- hard TC in $O(|N|\Delta_N^2\Delta_T^2)$ on...
  - ...reticulation visible
  - ...nearly stable
  - ...more

Conclusion

What we Saw...

- hard TC in $O(|N|\Delta_N^2\Delta_T^2)$ on...
  - ...reticulation visible
  - ...nearly stable
  - ...more

Conclusion

What we Saw...

- hard TC in $O(|N|\Delta_N^2\Delta_T^2)$ on...
  - ...reticulation visible
  - ...nearly stable
  - ...more
- hard TC in $O^*((\Delta_T + 1)^t)$ time

Parameters

t = #invisible comp.-roots in N

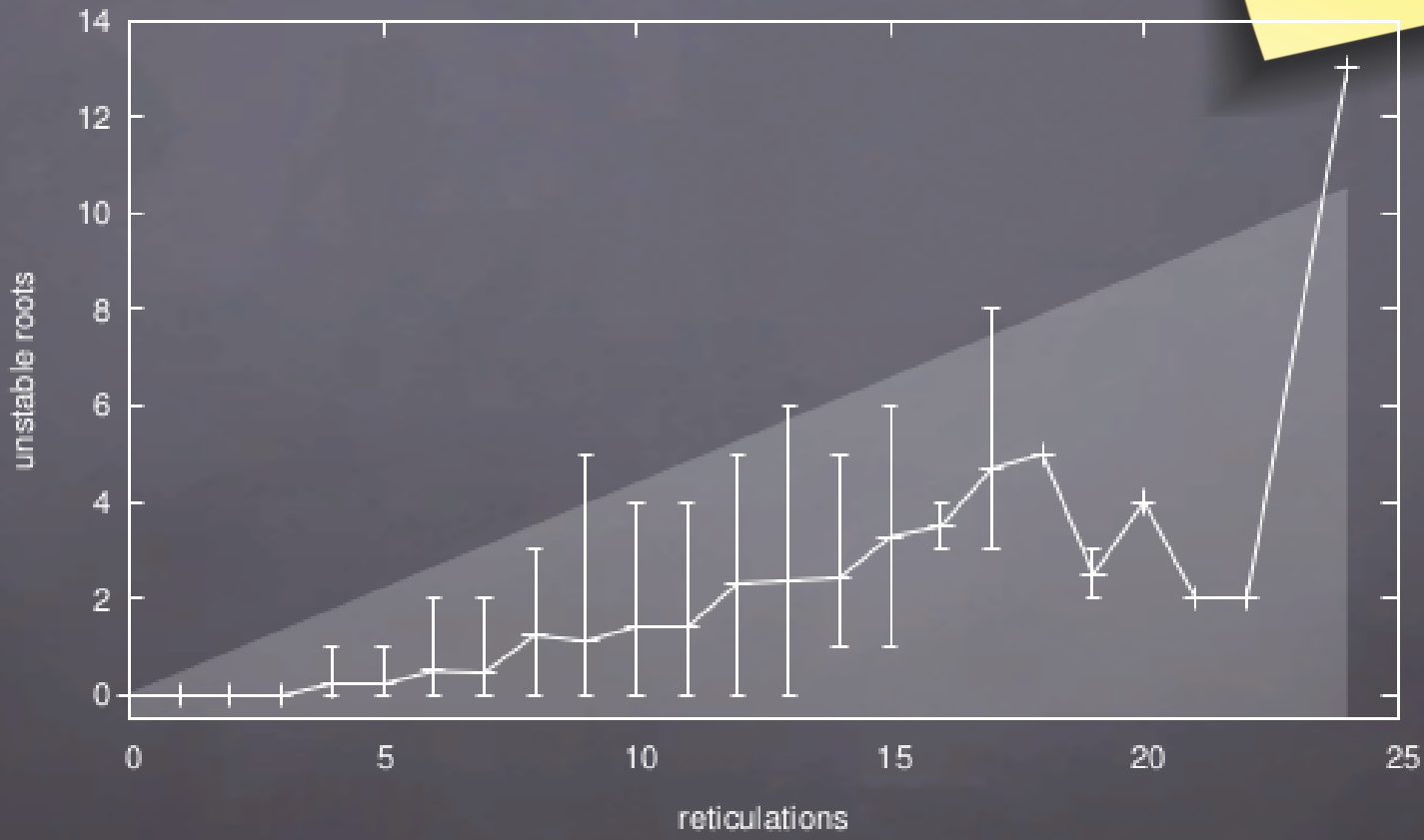
Conclusion

What we Saw...

- hard TC in $O(|N|\Delta_N^2\Delta_T^2)$ on...
  - ...reticulation visible
  - ...nearly stable
  - ...more
- hard TC in $O^*((\Delta_T + 1)^t)$ time

Parameters

t = #invisible comp.-roots in N



Conclusion

What we Saw...

- hard TC in $O(|N|\Delta_N^2\Delta_T^2)$ on...
  - ...reticulation visible
  - ...nearly stable
  - ...more
- hard TC in $O^*((\Delta_T + 1)^t)$ time
- soft TC somewhat more difficult than hard TC
  $\rightsquigarrow$ NP-hard in 3-labeled mul-trees
  $\rightsquigarrow$ NP-hard in reticulation visible with in-degree ≥ 3
- poly in 2-labelled mul-trees

Parameters

t = #invisible comp.-roots in N

Conclusion

What we Saw...

- hard TC in $O(|N|\Delta_N^2\Delta_T^2)$ on...
  - ...reticulation visible
  - ...nearly stable
  - ...more
- hard TC in $O^*((\Delta_T + 1)^t)$ time
- soft TC somewhat more difficult than hard TC
  $\rightsquigarrow$ NP-hard in 3-labeled mul-trees
  $\rightsquigarrow$ NP-hard in reticulation visible with in-degree ≥ 3
- poly in 2-labelled mul-trees

Parameters

t = #invisible comp.-roots in N

t^* = max. #invisible comp.-roots / block

Also in the Paper

- solve biconnected components "independently" ($t \to t^*$)
- no poly kernel for t^* (already for weaker "level")

Conclusion

What we Saw...

- hard TC in $O(|N|\Delta_N^2\Delta_T^2)$ on...
  - ...reticulation visible
  - ...nearly stable
  - ...more
- hard TC in $O^*((\Delta_T+1)^t)$ time
- soft TC somewhat more difficult than hard TC
  $\rightsquigarrow$ NP-hard in 3-labeled mul-trees
  $\rightsquigarrow$ NP-hard in reticulation visible with in-degree ≥ 3
- poly in 2-labelled mul-trees

Parameters

t = #invisible comp.-roots in N

t^* = max. #invisible comp.-roots / block

Also in the Paper

- solve biconnected components "independently" ($t \to t^*$)
- no poly kernel for t^* (already for weaker "level")

Ongoing Work

- parameterized cpx for soft TC
- other types of networks (tree-based, ...)
- parameterize more by t^*
- soft TC $\neq$ Cluster Containment
- mix soft $\neq$ hard polytomies


The End