Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Lehrstuhl für
INFORMATIK I
Algorithmen & Komplexität

Institut für Informatik

Exact Algorithms

Sommer Term 2020

Lecture 11 Tree Decomposition

Based on: [Parameterized Algorithms: §7.2, 7.3.1]

(slides by J. Spoerhase, Th. van Dijk, S. Chaplick, and A. Wolff)

Alexander Wolff

Lehrstuhl für Informatik I

(Weighted) Independent Set

Given: graph G, weight function $\omega\colon V \to \mathbb{N}$
Question: What is the maximum weight of a set $S \subseteq V$ where no pair in S is adjacent in G?

(Weighted) Independent Set

Given: graph G, weight function $\omega\colon V \to \mathbb{N}$
Question: What is the maximum weight of a set $S \subseteq V$ where no pair in S is adjacent in G?

Thm. Independent Set is NP-complete.

(Weighted) Independent Set

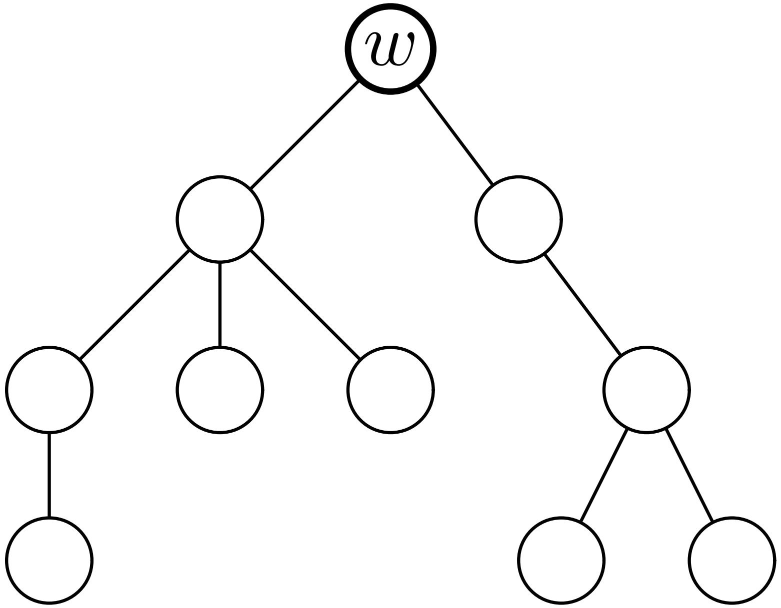
Given: graph G, weight function $\omega\colon V \to \mathbb{N}$
Question: What is the maximum weight of a set $S \subseteq V$ where no pair in S is adjacent in G?

Thm. Independent Set is NP-complete.

Thm. On trees, Independent Set can be solved in linear time.

Independent Sets in Trees

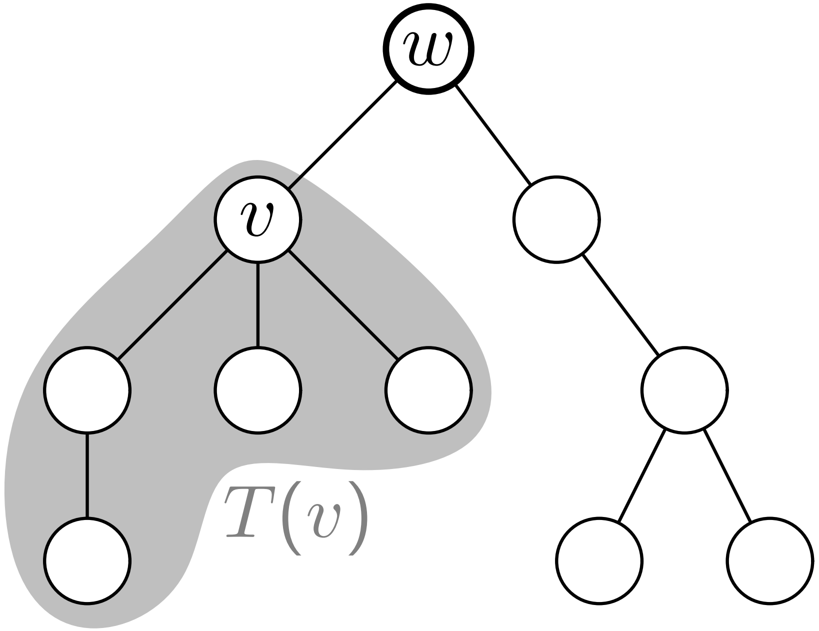
Choose an arbitrary root w.



Independent Sets in Trees

Choose an arbitrary root w.

Let $T(v) :=$ subtree rooted at v

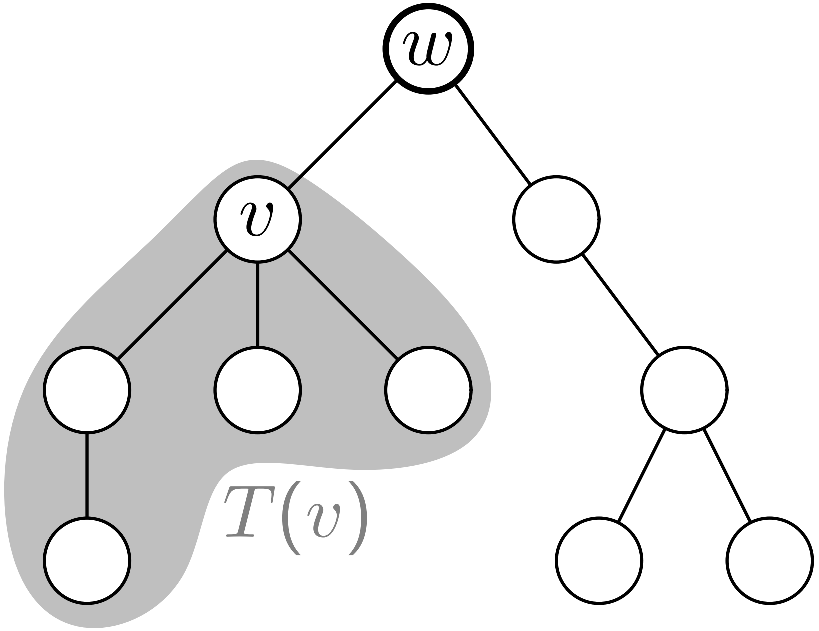


Independent Sets in Trees

Choose an arbitrary root w.

Let $T(v)$:= subtree rooted at v

Let $A(v)$:= maximum weight of an independent set S in $T(v)$



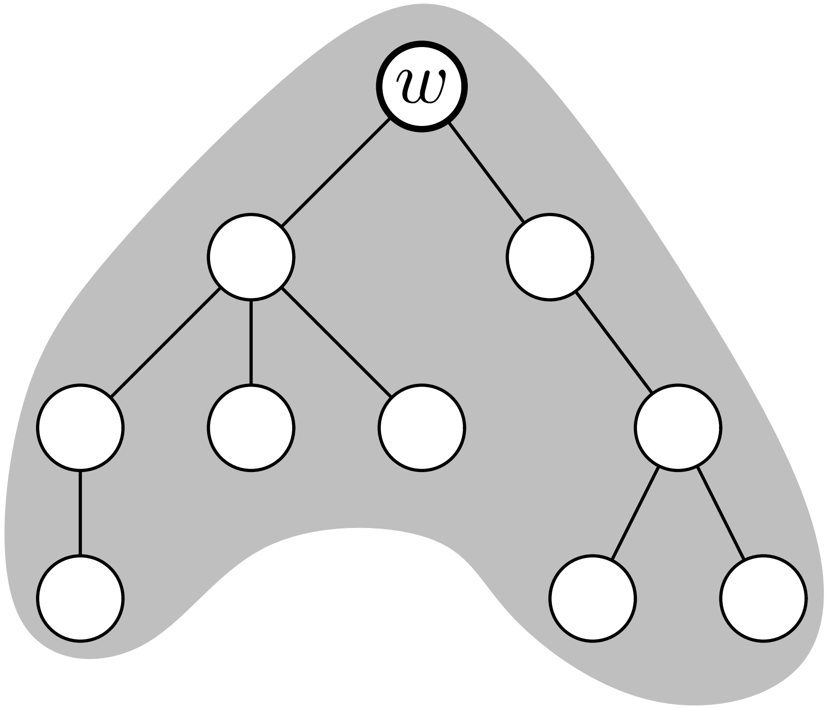
Independent Sets in Trees

Choose an arbitrary root w.

Let $T(v) :=$ subtree rooted at v

Let $A(v) :=$ maximum weight of an independent set S in $T(v)$

$A(w) =$ solution



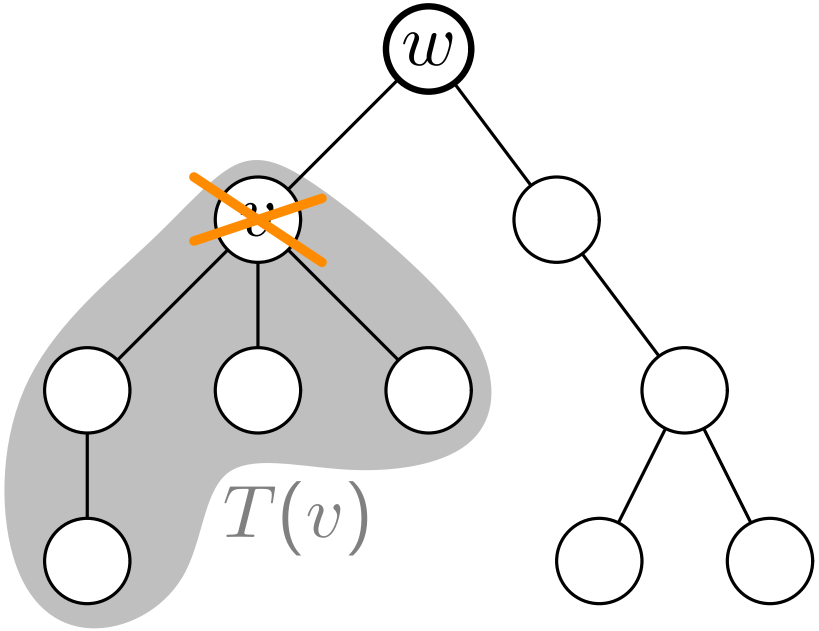
Independent Sets in Trees

Choose an arbitrary root w.

Let $T(v)$:= subtree rooted at v

Let $A(v)$:= maximum weight of an independent set S in $T(v)$

Let $B(v)$:= maximum weight of an independent set S in $T(v)$ *where* $v \notin S$



Independent Sets in Trees

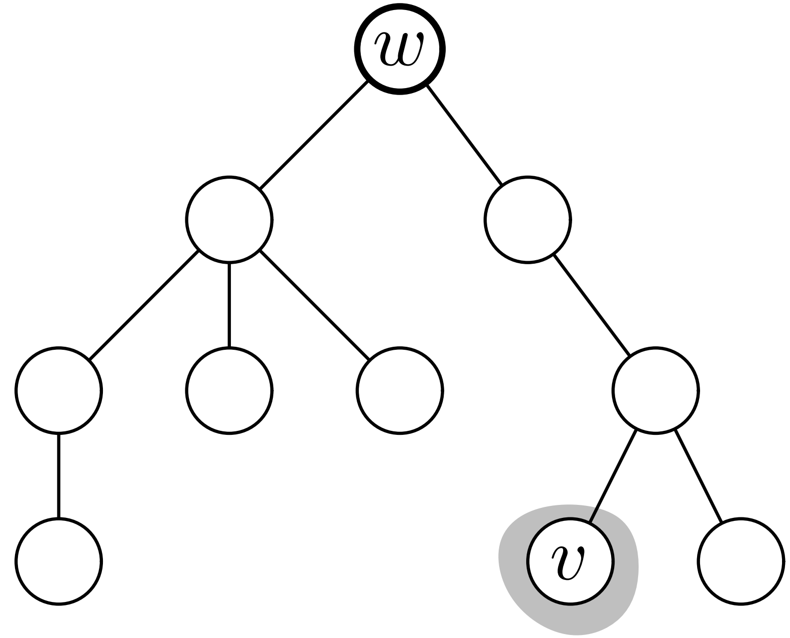
Choose an arbitrary root w.

Let $T(v) :=$ subtree rooted at v

Let $A(v) :=$ maximum weight of an independent set S in $T(v)$

Let $B(v) :=$ maximum weight of an independent set S in $T(v)$ *where* $v \notin S$

– If v is a leaf: $B(v) =$



Independent Sets in Trees

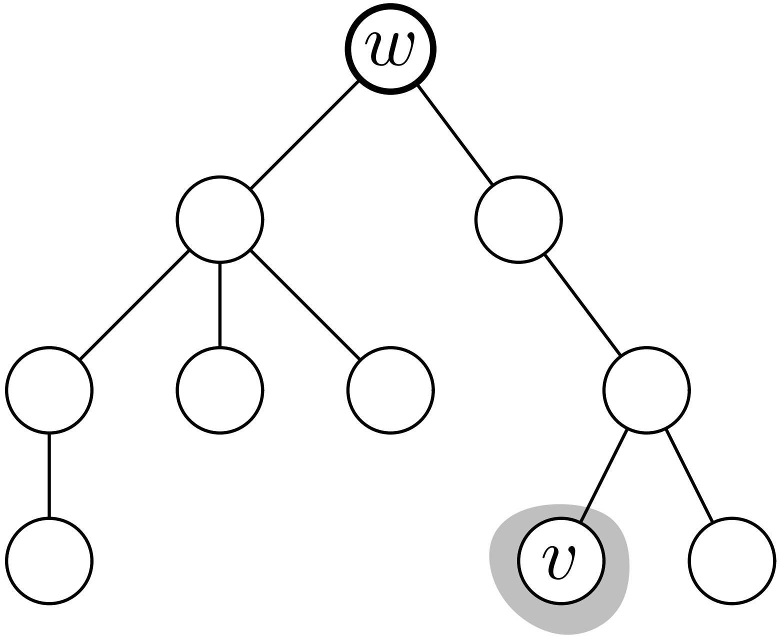
Choose an arbitrary root w.

Let $T(v)$:= subtree rooted at v

Let $A(v)$:= maximum weight of an independent set S in $T(v)$

Let $B(v)$:= maximum weight of an independent set S in $T(v)$ *where* $v \notin S$

– If v is a leaf: $B(v) = 0$ and $A(v) =$



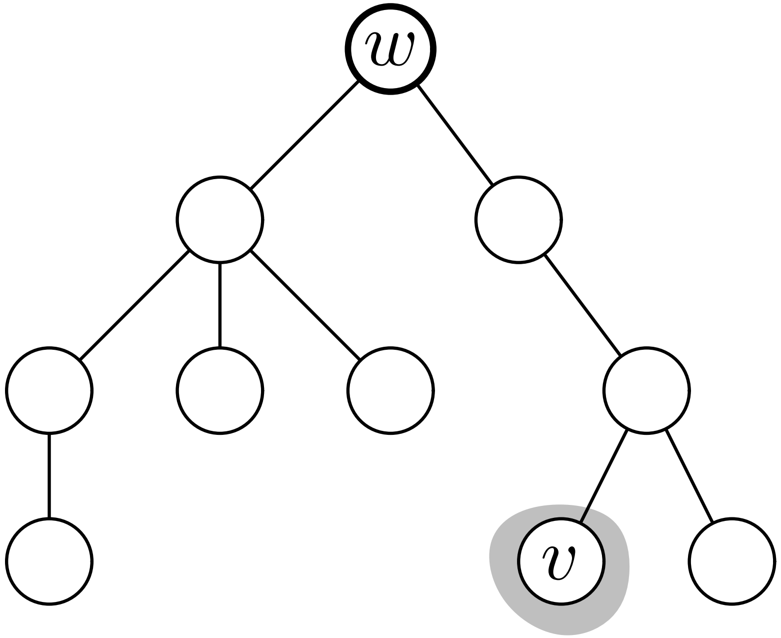
Independent Sets in Trees

Choose an arbitrary root w.

Let $T(v) :=$ subtree rooted at v

Let $A(v) :=$ maximum weight of an independent set S in $T(v)$

Let $B(v) :=$ maximum weight of an independent set S in $T(v)$ *where* $v \notin S$



– If v is a leaf: $B(v) = 0$ and $A(v) = \omega(v)$

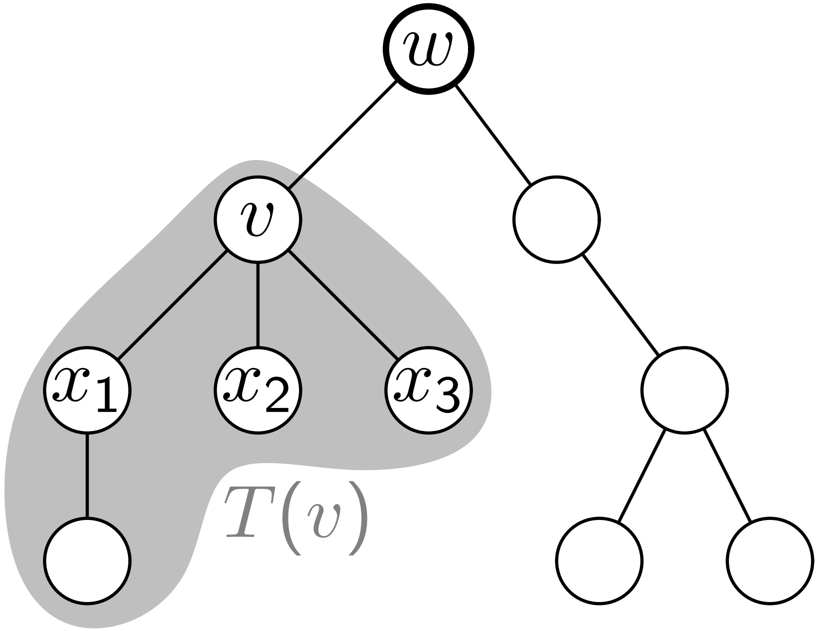
Independent Sets in Trees

Choose an arbitrary root w.

Let $T(v) :=$ subtree rooted at v

Let $A(v) :=$ maximum weight of an independent set S in $T(v)$

Let $B(v) :=$ maximum weight of an independent set S in $T(v)$ *where* $v \notin S$



– If v is a leaf: $B(v) = 0$ and $A(v) = \omega(v)$

– If v has children $x_1, \ldots, x_r$:

$B(v) =$

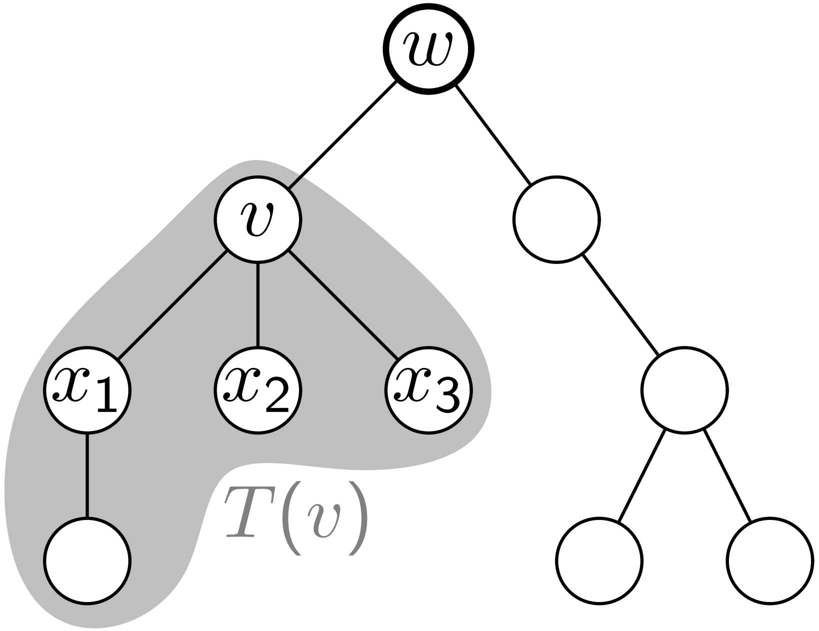
Independent Sets in Trees

Choose an arbitrary root w.

Let $T(v) :=$ subtree rooted at v

Let $A(v) :=$ maximum weight of an independent set S in $T(v)$

Let $B(v) :=$ maximum weight of an independent set S in $T(v)$ *where* $v \notin S$



– If v is a leaf: $B(v) = 0$ and $A(v) = \omega(v)$

– If v has children $x_1, \ldots, x_r$:

$B(v) = \sum_{i=1}^{r} A(x_i)$;

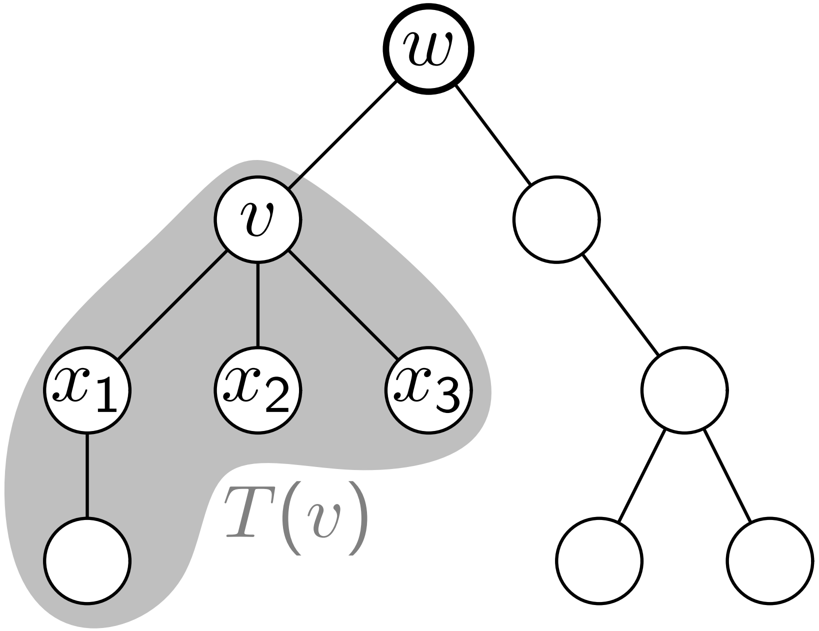
Independent Sets in Trees

Choose an arbitrary root w.

Let $T(v)$:= subtree rooted at v

Let $A(v)$:= maximum weight of an independent set S in $T(v)$

Let $B(v)$:= maximum weight of an independent set S in $T(v)$ *where* $v \notin S$



– If v is a leaf: $B(v) = 0$ and $A(v) = \omega(v)$

– If v has children $x_1, \ldots, x_r$:

$B(v) = \sum_{i=1}^{r} A(x_i)$; $A(v) =$

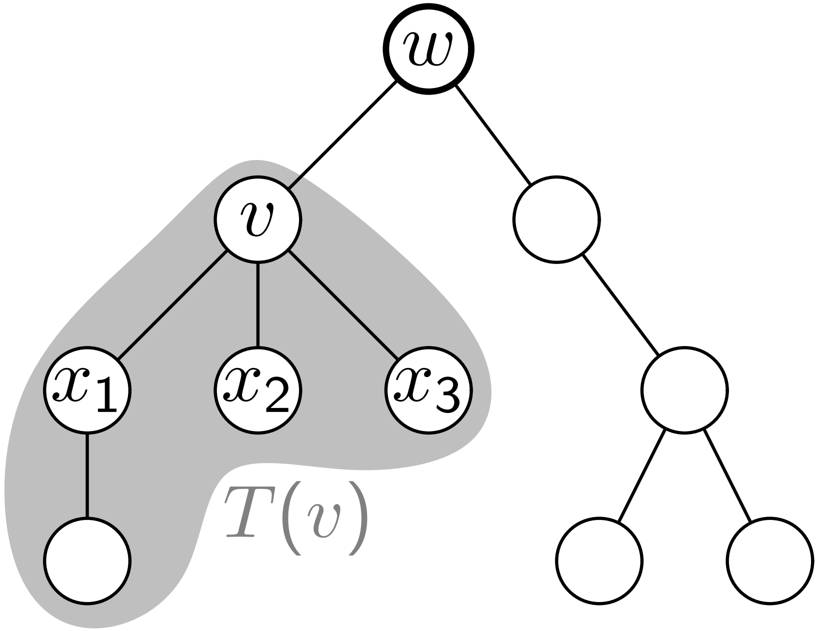
Independent Sets in Trees

Choose an arbitrary root w.

Let $T(v)$:= subtree rooted at v

Let $A(v)$:= maximum weight of an independent set S in $T(v)$

Let $B(v)$:= maximum weight of an independent set S in $T(v)$ *where* $v \notin S$



– If v is a leaf: $B(v) = 0$ and $A(v) = \omega(v)$

– If v has children $x_1, \ldots, x_r$:

$B(v) = \sum_{i=1}^{r} A(x_i)$; $A(v) = \max\{B(v),$

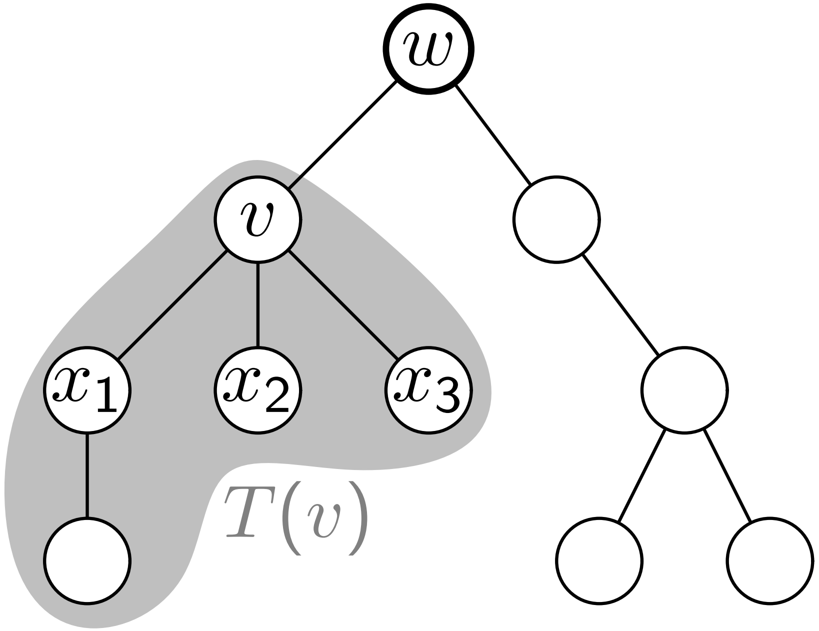
Independent Sets in Trees

Choose an arbitrary root w.

Let $T(v)$:= subtree rooted at v

Let $A(v)$:= maximum weight of an independent set S in $T(v)$

Let $B(v)$:= maximum weight of an independent set S in $T(v)$ *where* $v \notin S$



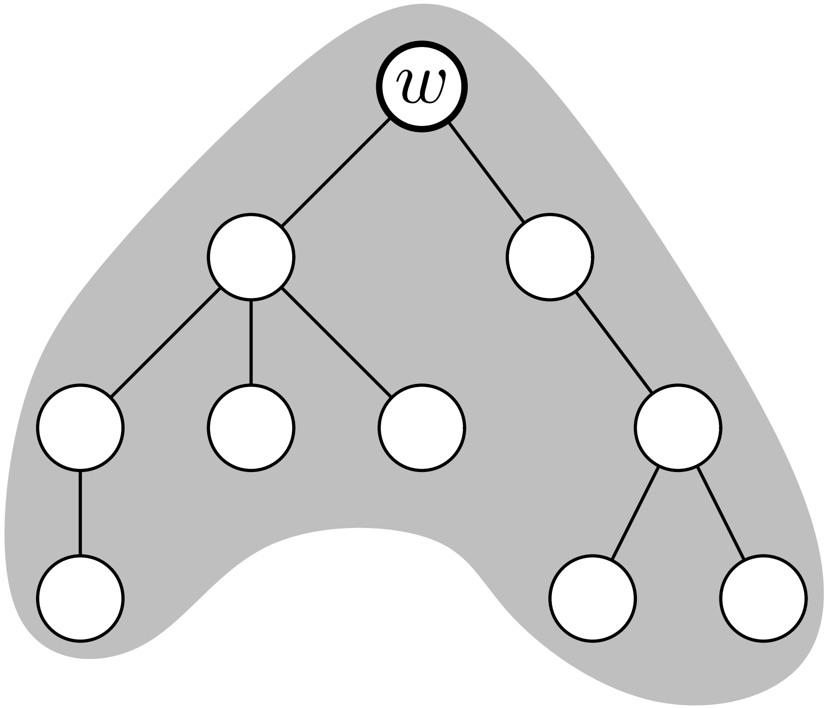
– If v is a leaf: $B(v) = 0$ and $A(v) = \omega(v)$

– If v has children $x_1, \ldots, x_r$:

$B(v) = \sum_{i=1}^{r} A(x_i)$; $A(v) = \max\{B(v), \omega(v) + \sum_{i=1}^{r} B(x_i)\}$

Independent Sets in Trees

$A(w) =$ solution



Choose an arbitrary root w.

Let $T(v) :=$ subtree rooted at v

Let $A(v) :=$ maximum weight of an independent set S in $T(v)$

Let $B(v) :=$ maximum weight of an independent set S in $T(v)$ *where* $v \notin S$

– If v is a leaf: $B(v) = 0$ and $A(v) = \omega(v)$

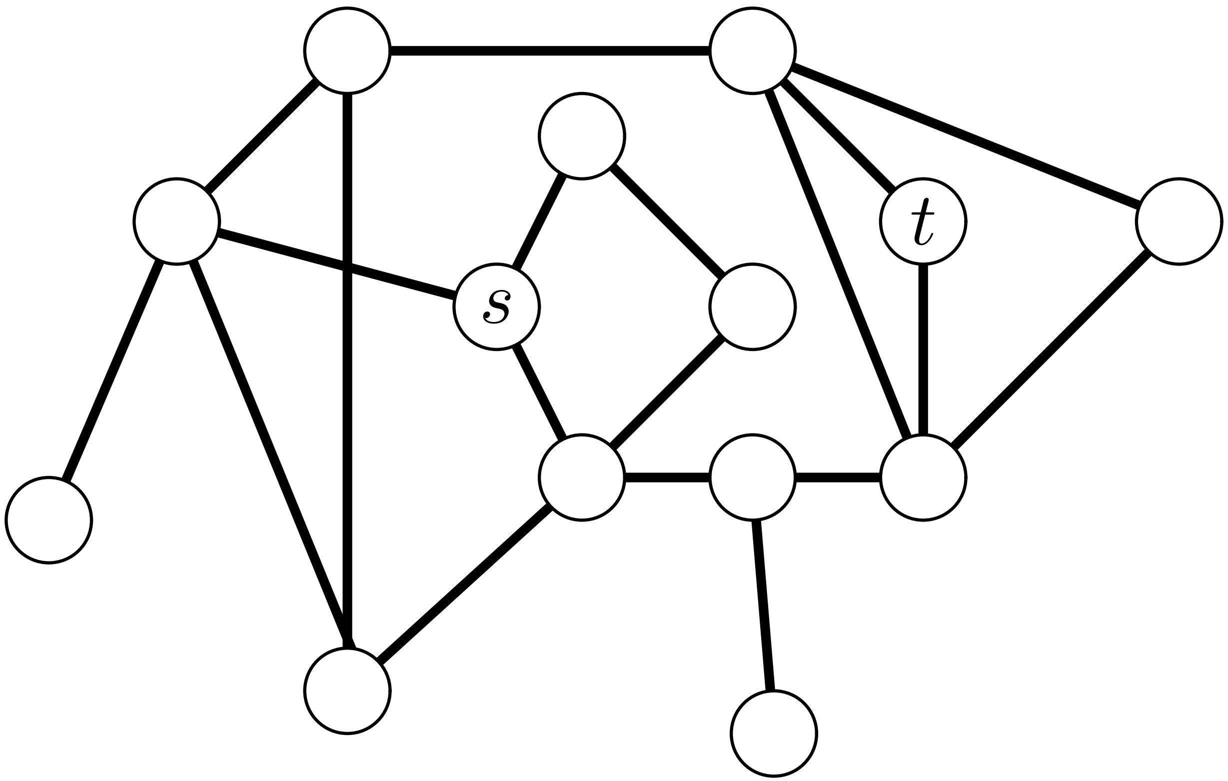
– If v has children $x_1, \ldots, x_r$:

$B(v) = \sum_{i=1}^{r} A(x_i)$; $A(v) = \max\{B(v), \omega(v) + \sum_{i=1}^{r} B(x_i)\}$

Algorithm: Compute $A(\cdot)$ and $B(\cdot)$ bottom-up!

(s,t)-Series-Parallel Graphs

Def. A graph $G = (V, E)$ is *2-terminal* if it contains two special vertices s and t.



(s, t)-Series-Parallel Graphs

Def. A graph $G = (V, E)$ is *2-terminal* if it contains two special vertices s and t.

Def. A 2-terminal graph G is *series-parallel* if:

- G is a single edge (s, t) 

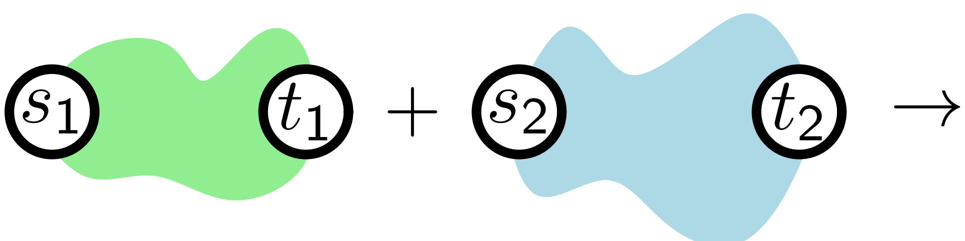
(s, t)-Series-Parallel Graphs

Def. A graph $G = (V, E)$ is *2-terminal* if it contains two special vertices s and t.

Def. A 2-terminal graph G is *series-parallel* if:

- G is a single edge (s, t)

- G is a *series composition* of two series-parallel graphs



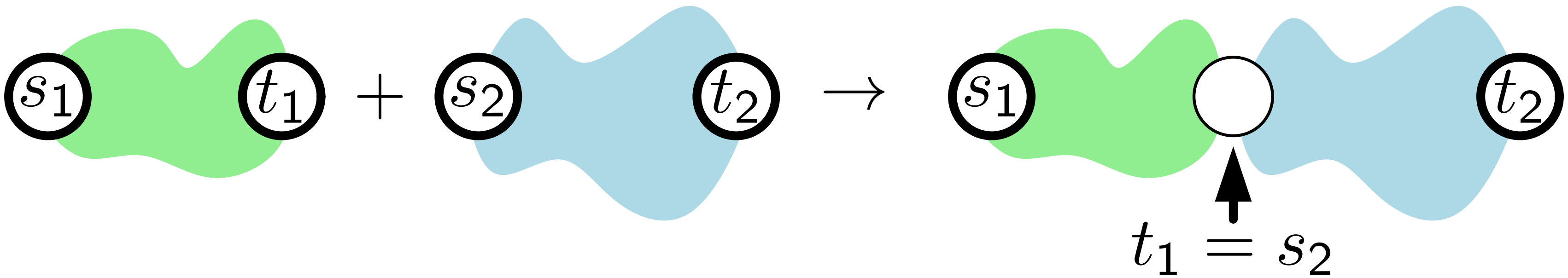
(s, t)-Series-Parallel Graphs

Def. A graph $G = (V, E)$ is *2-terminal* if it contains two special vertices s and t.

Def. A 2-terminal graph G is *series-parallel* if:

- G is a single edge (s, t) 

- G is a *series composition* of two series-parallel graphs



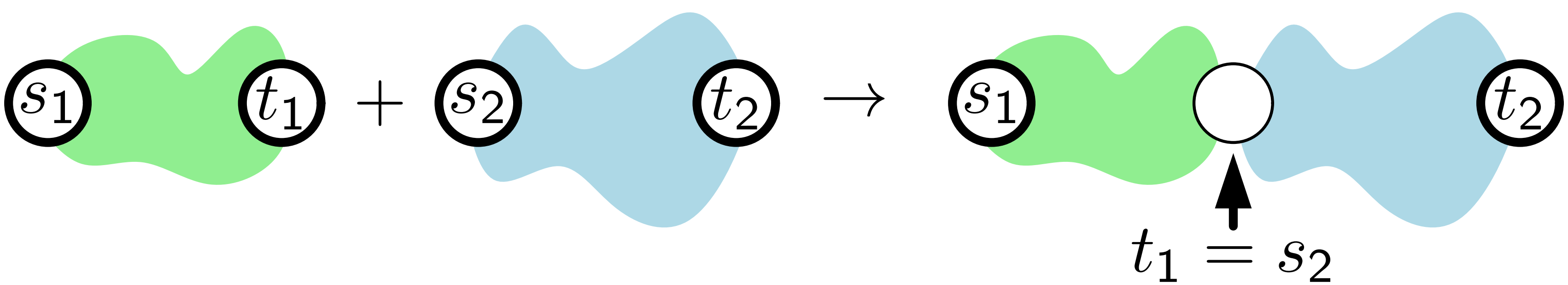
(s, t)-Series-Parallel Graphs

Def. A graph $G = (V, E)$ is *2-terminal* if it contains two special vertices s and t.

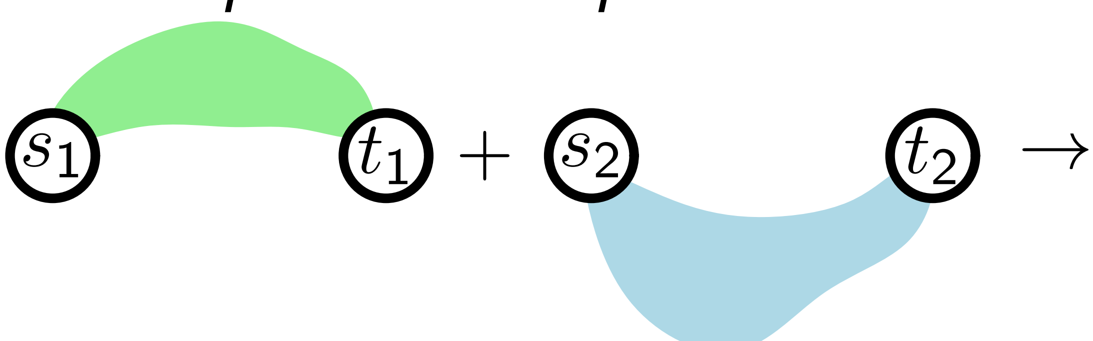
Def. A 2-terminal graph G is *series-parallel* if:

- G is a single edge (s, t)

- G is a *series composition* of two series-parallel graphs



- G is a *parallel composition* of two series-parallel graphs



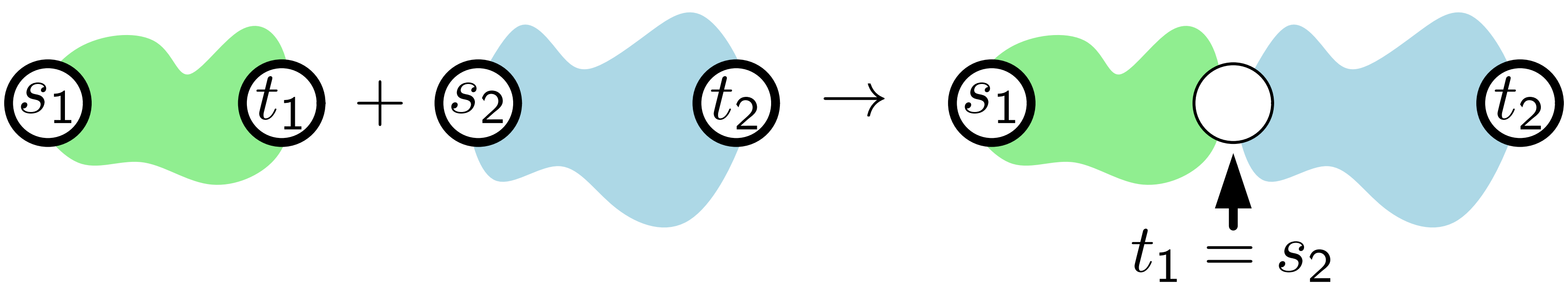
(s, t)-Series-Parallel Graphs

Def. A graph $G = (V, E)$ is *2-terminal* if it contains two special vertices s and t.

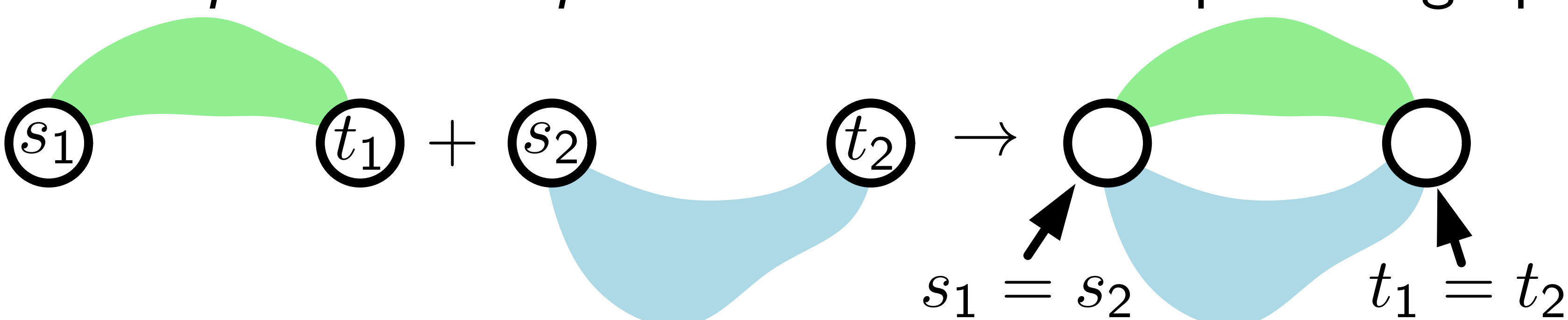
Def. A 2-terminal graph G is *series-parallel* if:

- G is a single edge (s, t) 

- G is a *series composition* of two series-parallel graphs



- G is a *parallel composition* of two series-parallel graphs



(s,t)-Series-Parallel Graphs

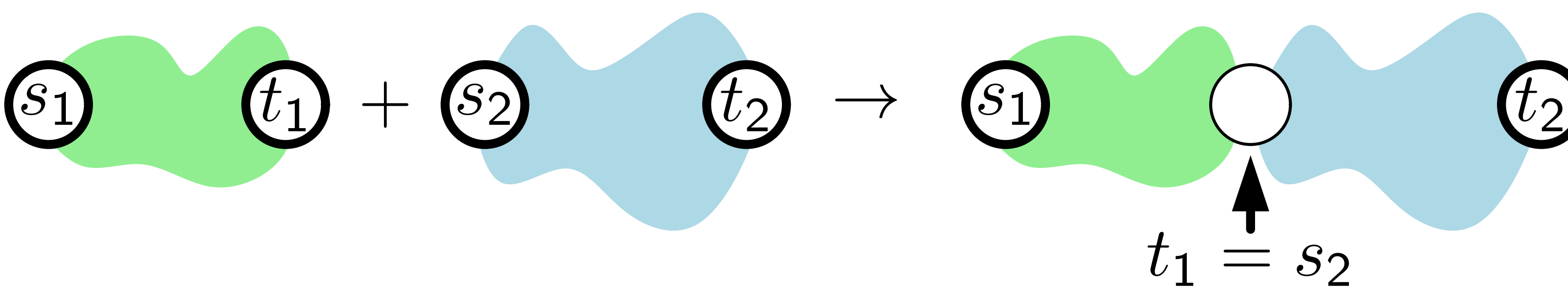
Def. A graph $G = (V, E)$ is *2-terminal* if it contains two special vertices s and t.

recursive definition

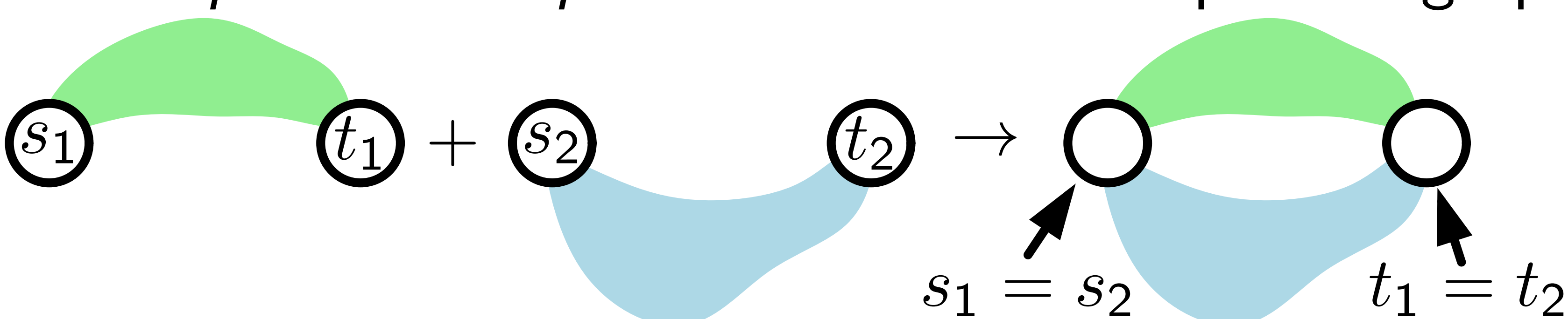
Def. A 2-terminal graph G is *series-parallel* if:

- G is a single edge (s,t)

- G is a *series composition* of two series-parallel graphs



- G is a *parallel composition* of two series-parallel graphs



(s, t)-Series-Parallel Graphs

Def. A graph $G = (V, E)$ is *2-terminal* if it contains two special vertices s and t.

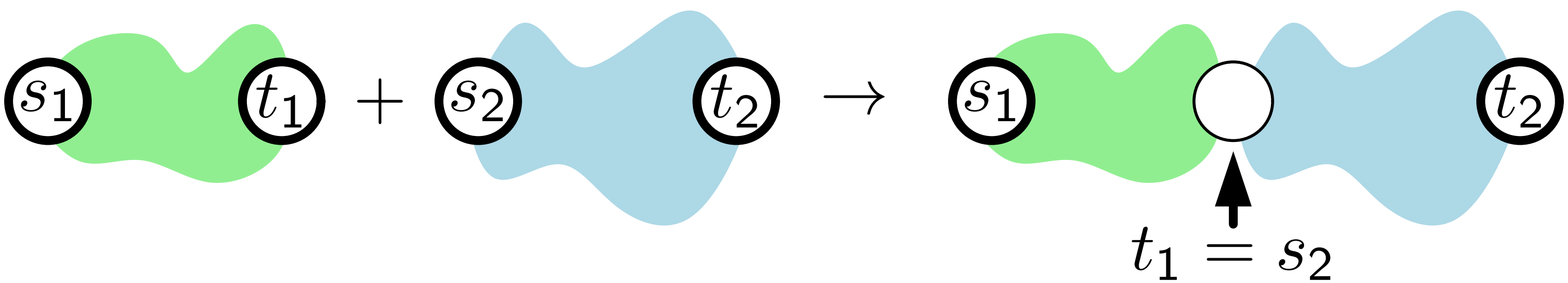
recursive definition

Series-parallel graphs have a natural tree structure!

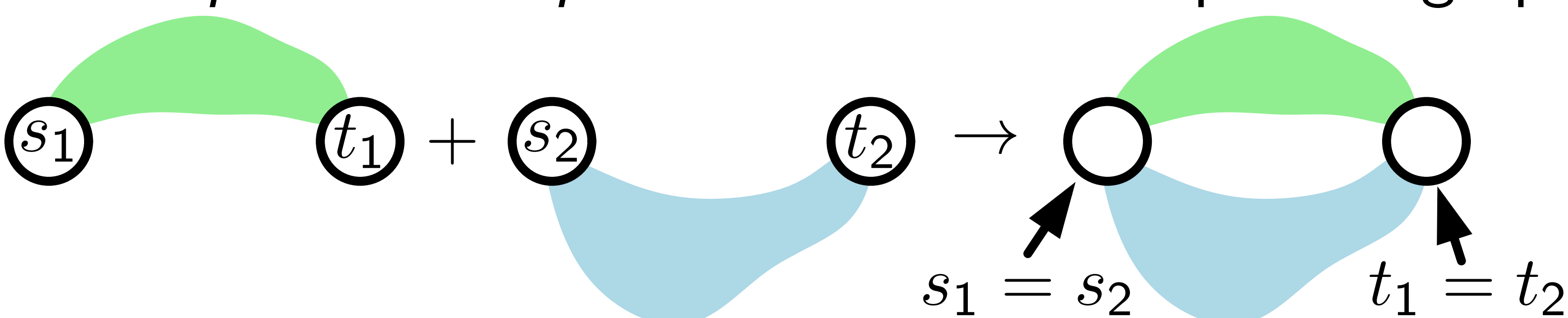
Def. A 2-terminal graph G is *series-parallel* if:

- G is a single edge (s, t)

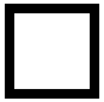
- G is a *series composition* of two series-parallel graphs



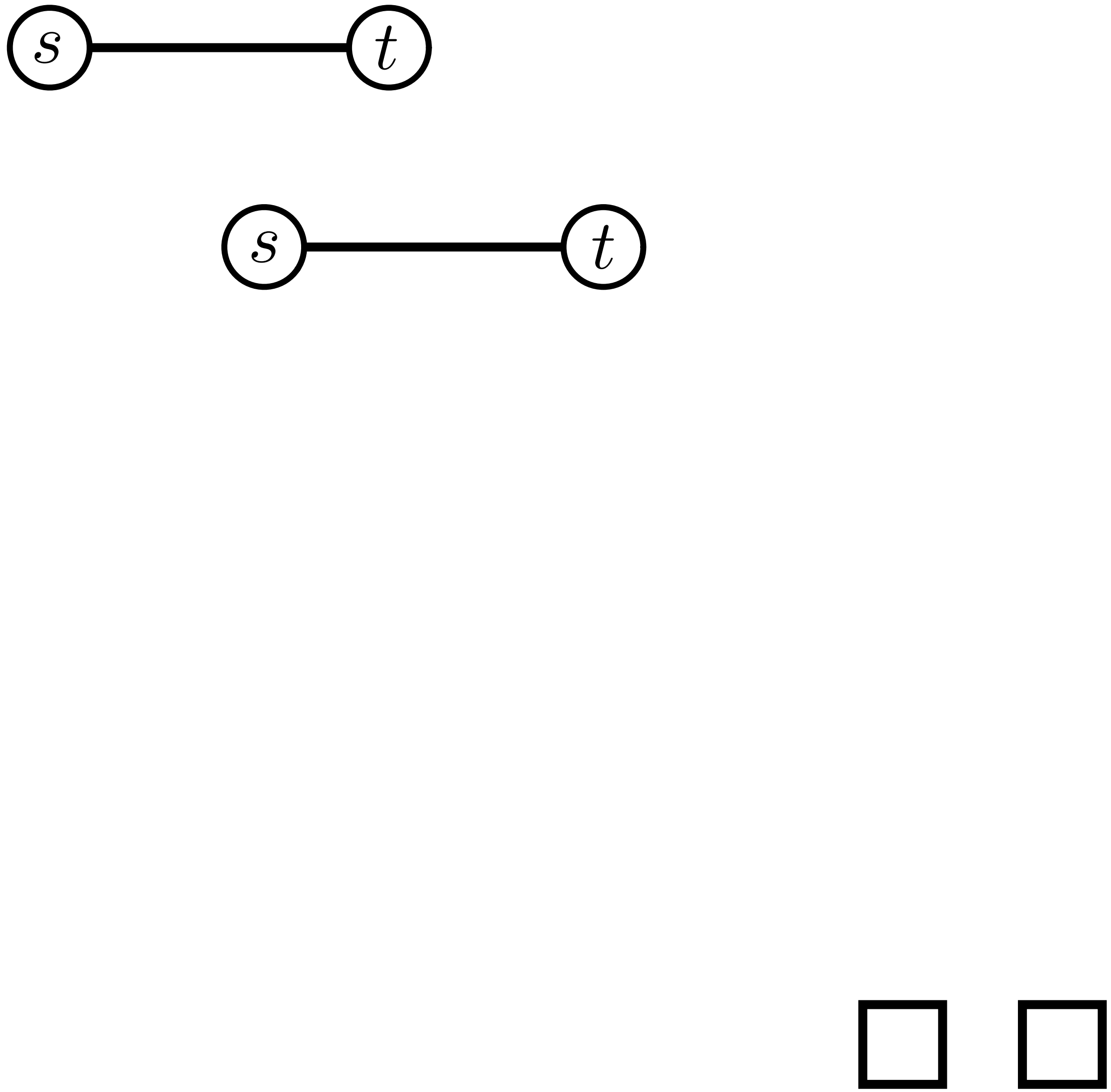
- G is a *parallel composition* of two series-parallel graphs



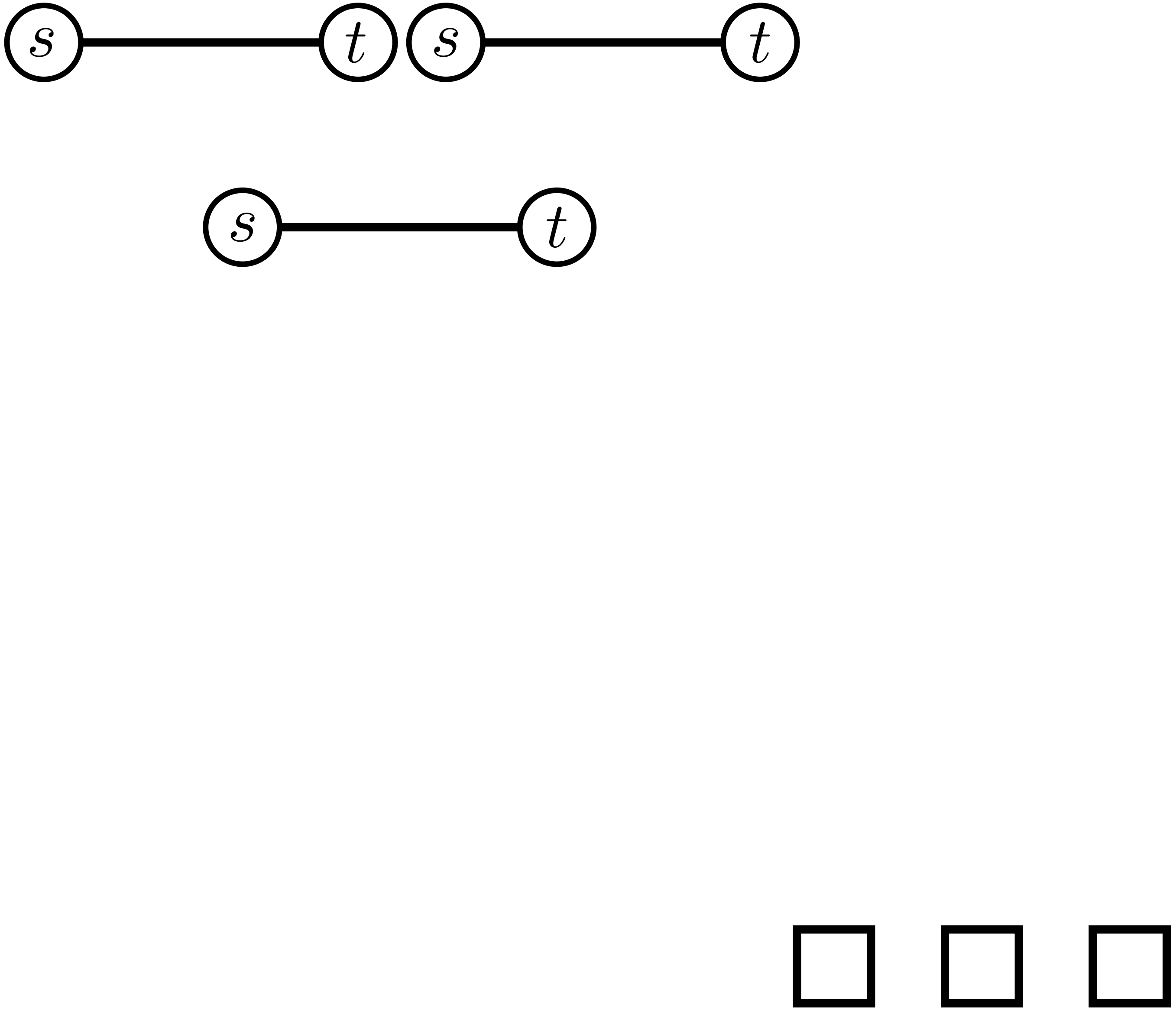
SP-Tree



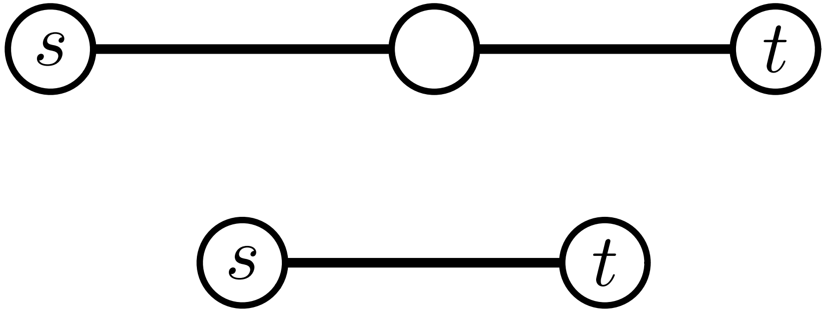
SP-Tree

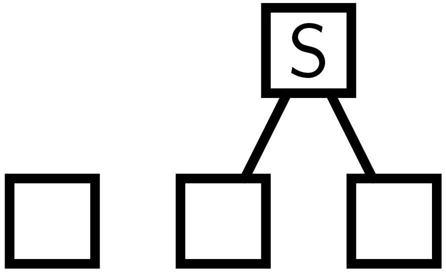


SP-Tree

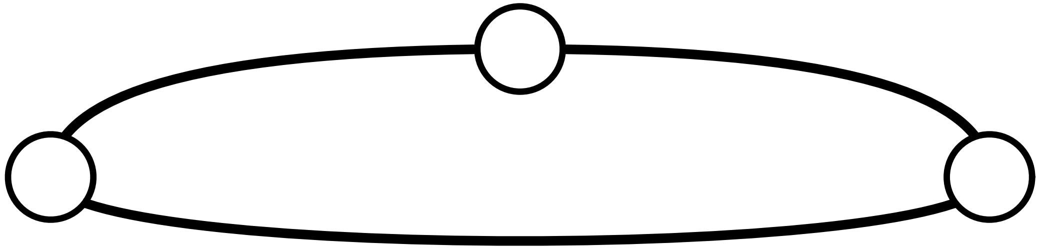


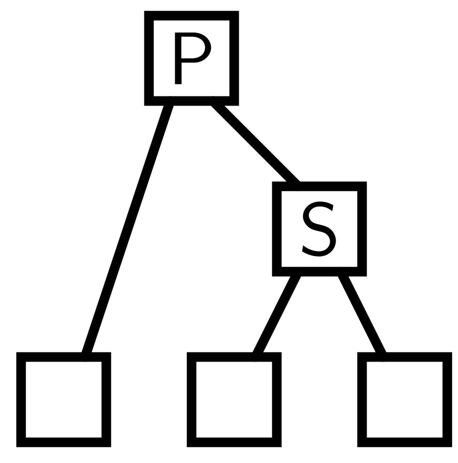
SP-Tree



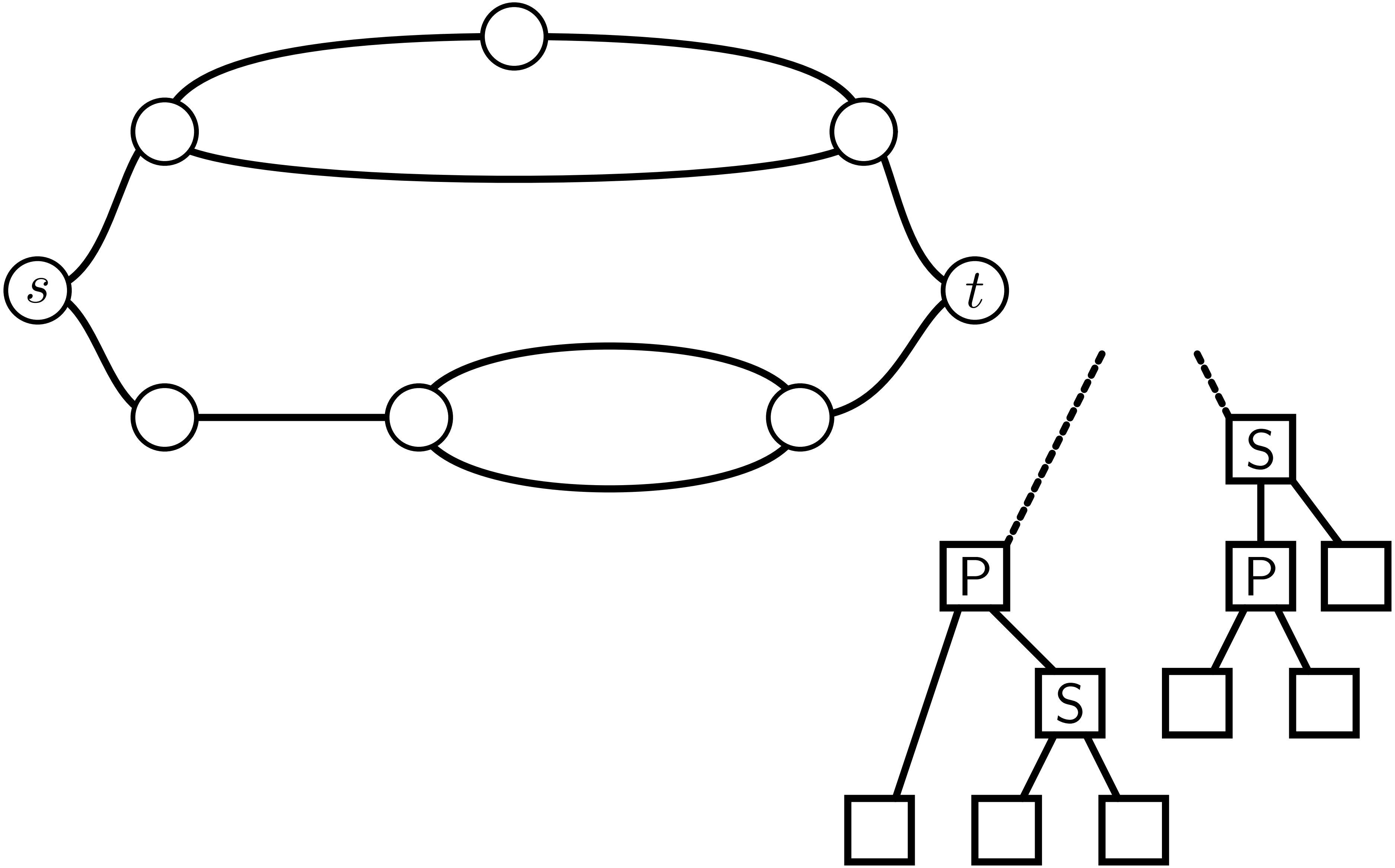


SP-Tree

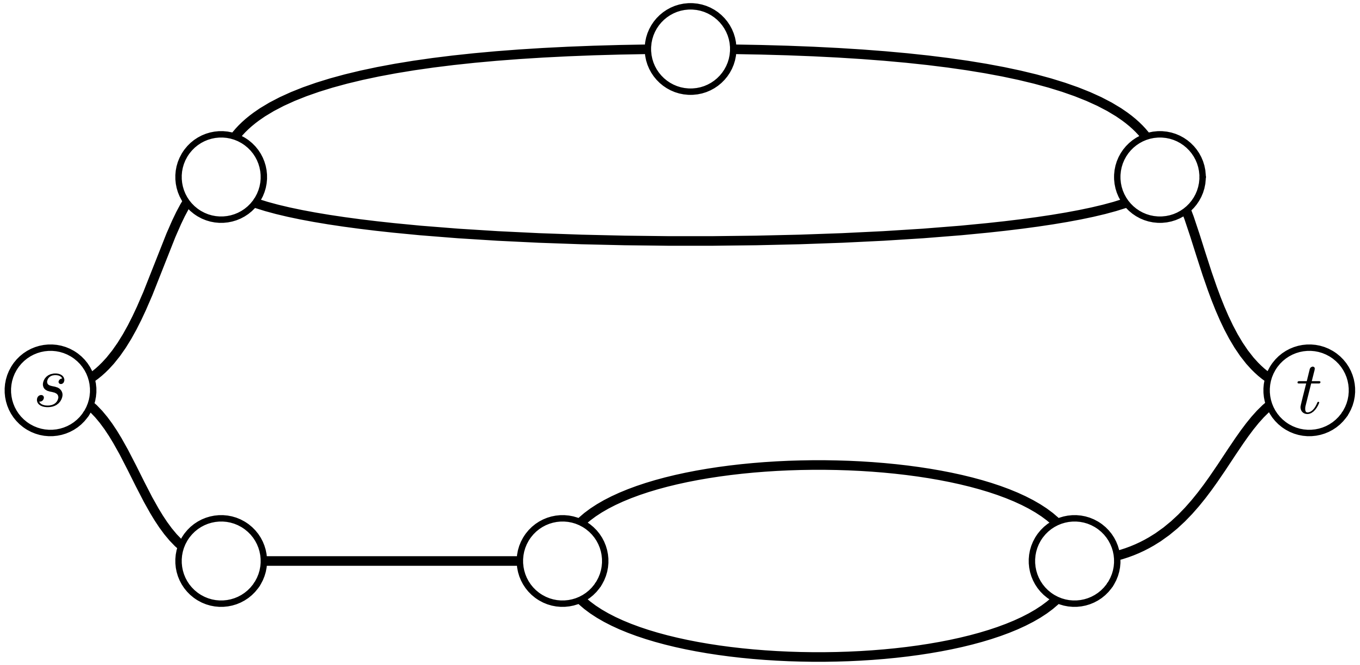


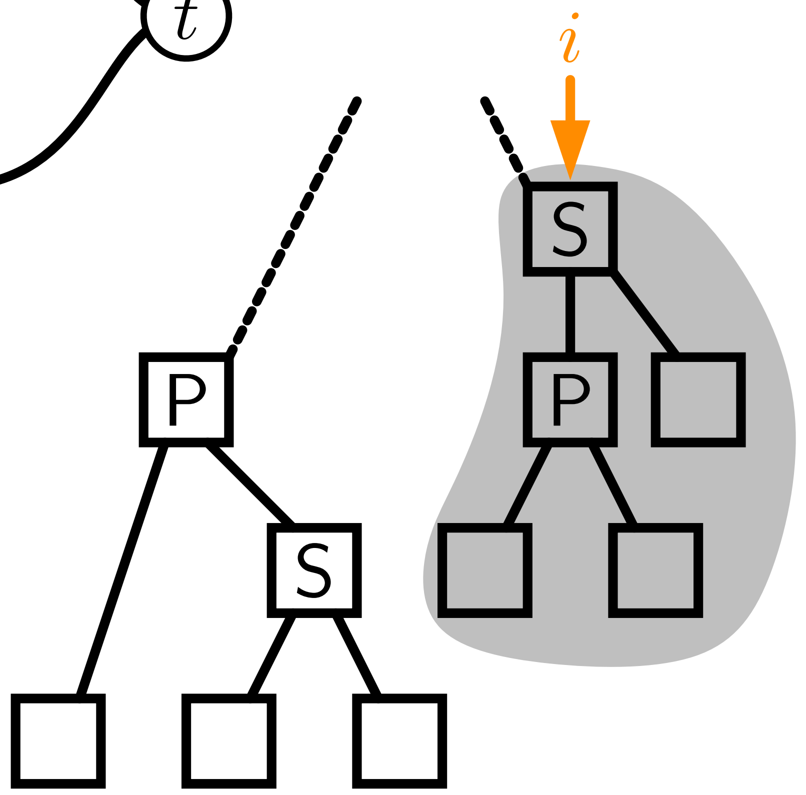


SP-Tree



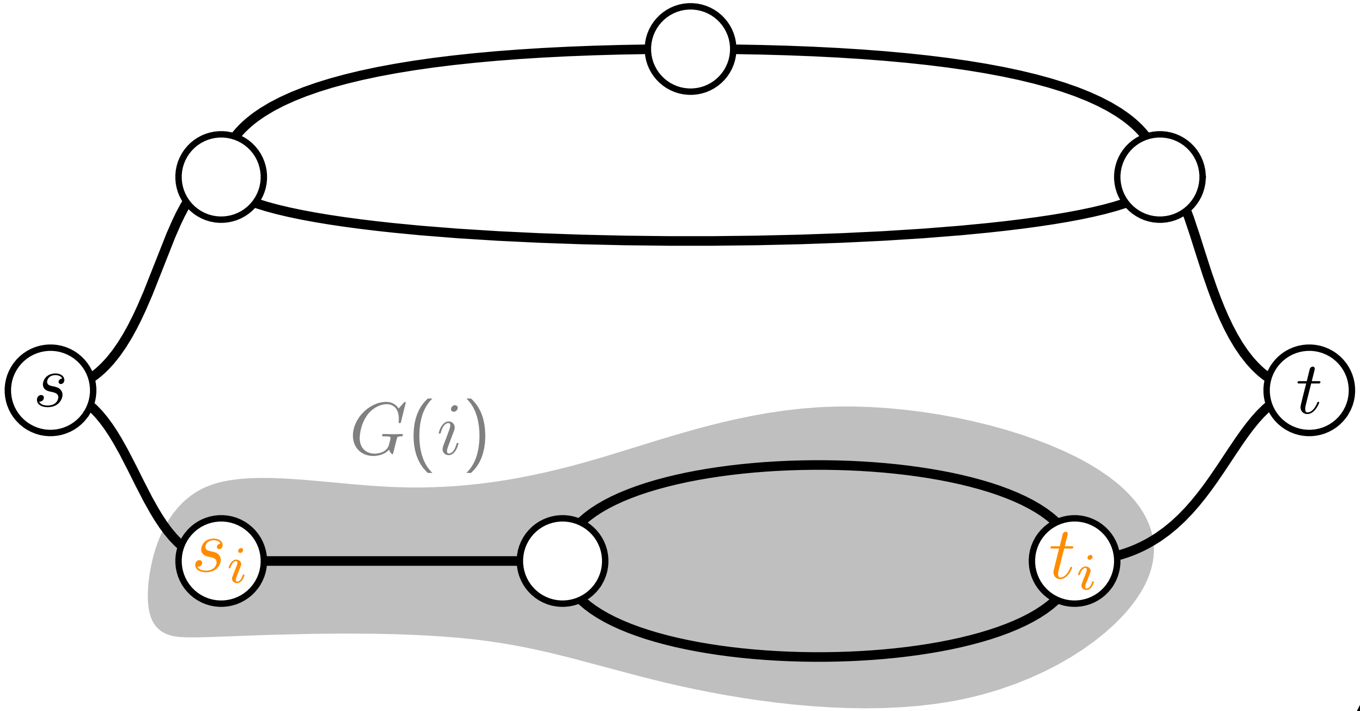
SP-Tree

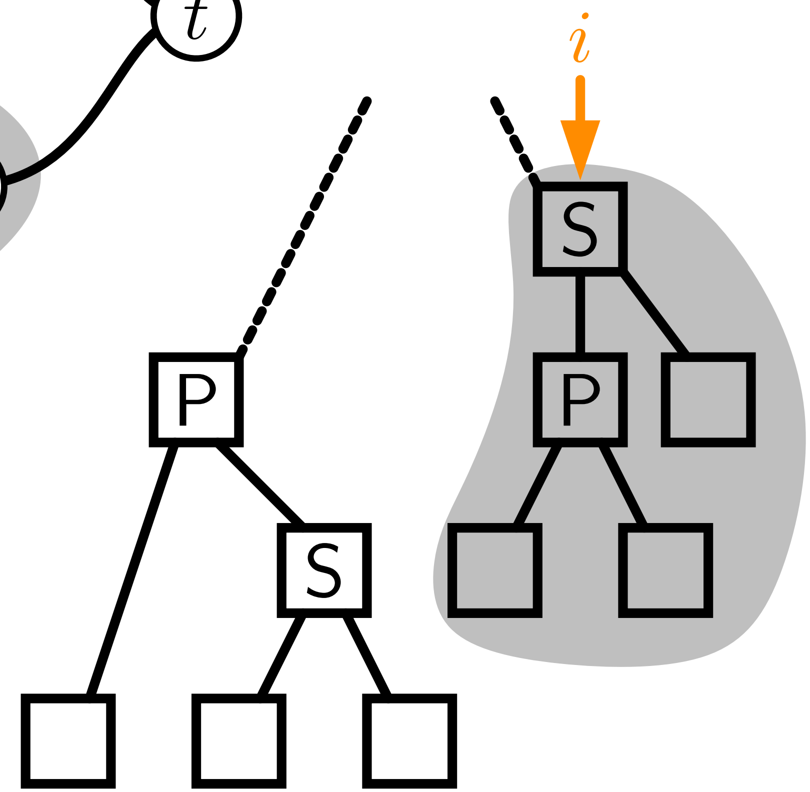




Let i be a node in an SP-tree.
$G(i) :=$ graph represented by the subtree rooted at i

SP-Tree

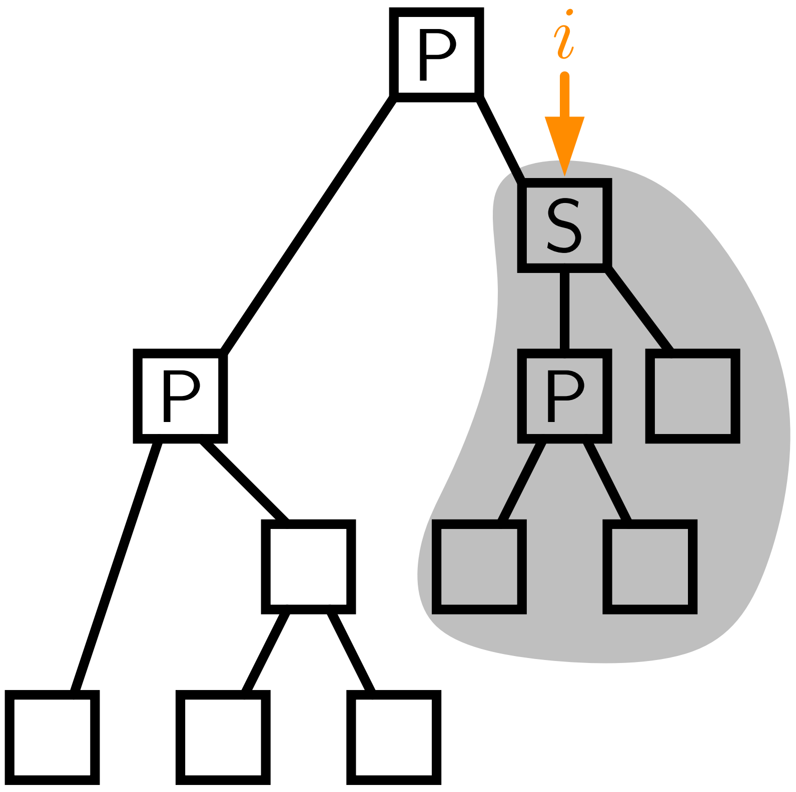




Let i be a node in an SP-tree.
$G(i) :=$ graph represented by the subtree rooted at i

Independent Set on SP-Trees

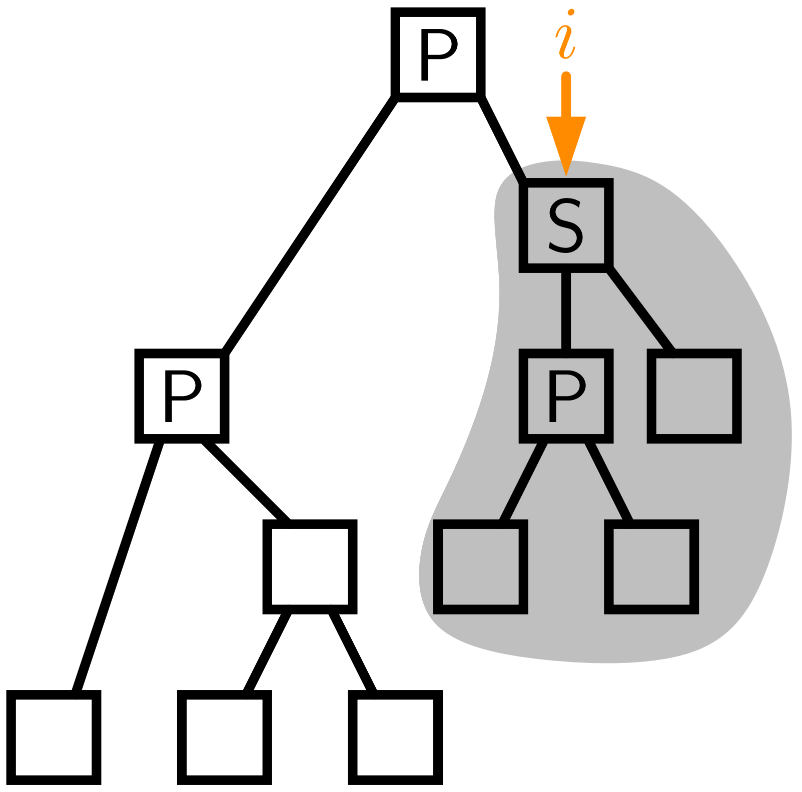
Dynamic program on SP-tree indexed by $G(i)$



Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

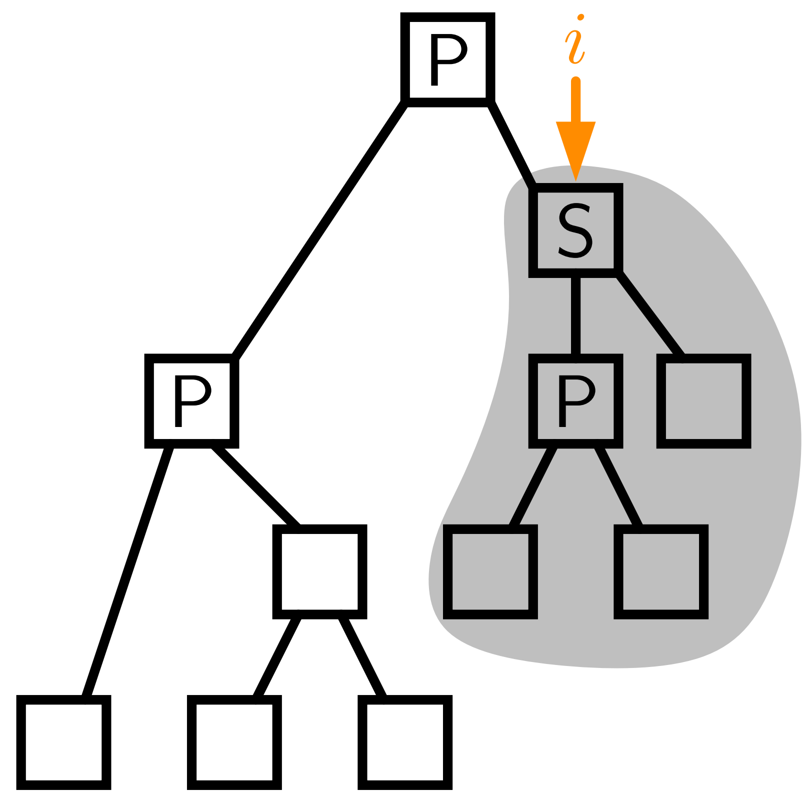


Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$



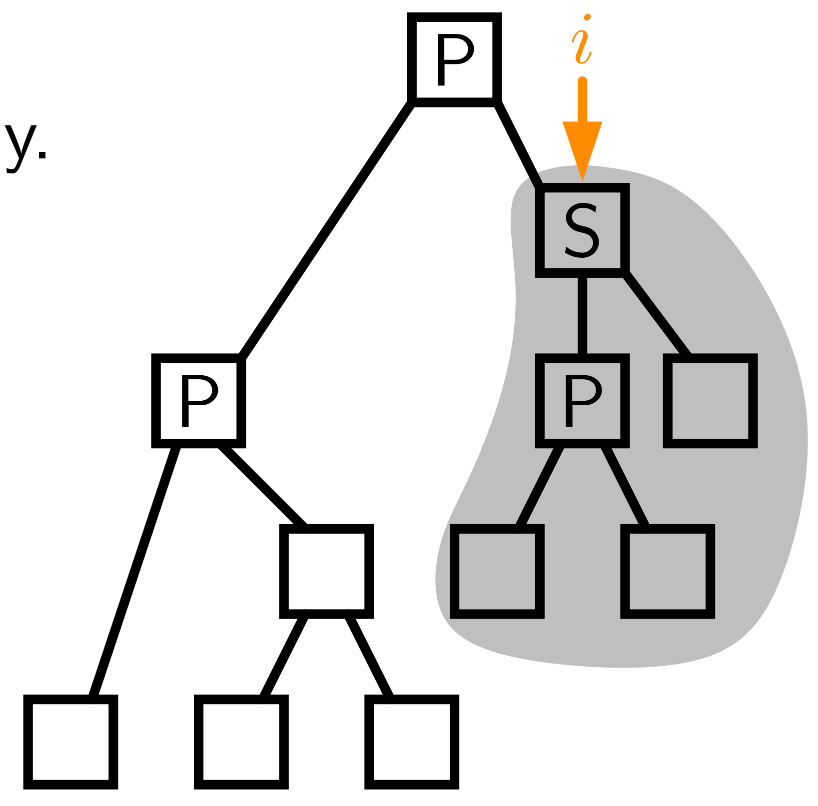
Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.



Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.

– If i is a leaf...

$BB(i) =$

Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.

– If i is a leaf...

$BB(i) =$ 0

Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.

– If i is a leaf...

$BA(i) =$

$BB(i) =$ 0

Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.

– If i is a leaf...

$$BA(i) = \omega(t_i)$$
$$BB(i) = 0$$

Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.

– If i is a leaf...

$$AB(i) =$$
$$BA(i) = \omega(t_i)$$
$$BB(i) = 0$$

Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.

– If i is a leaf...

$$AB(i) = \omega(s_i)$$
$$BA(i) = \omega(t_i)$$
$$BB(i) = 0$$

Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.

– If i is a leaf...

$AA(i) =$

$AB(i) = \omega(s_i)$

$BA(i) = \omega(t_i)$

$BB(i) = 0$

Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.

– If i is a leaf...

$$\begin{aligned} AA(i) &= -\infty \\ AB(i) &= \omega(s_i) \\ BA(i) &= \omega(t_i) \\ BB(i) &= 0 \end{aligned}$$

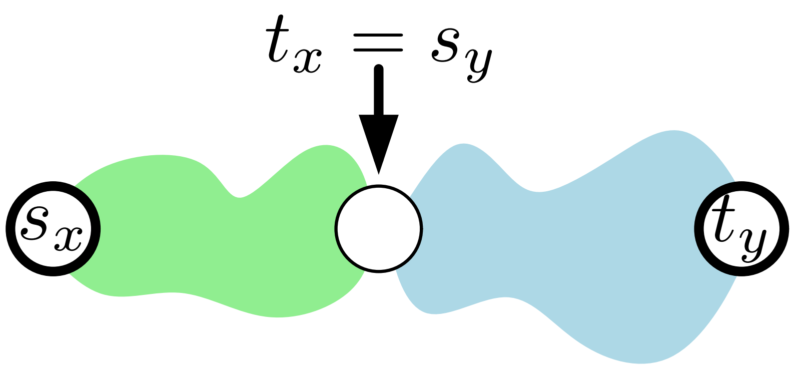
Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.



– If i is a series composition with children x and y, ...

$AA(i) =$

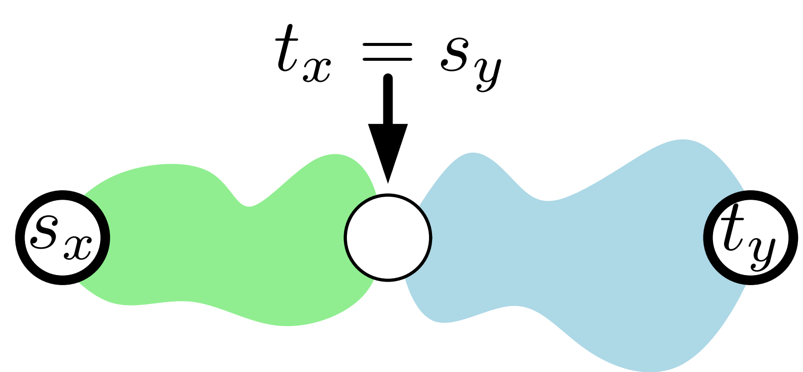
Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.



– If i is a series composition with children x and y, ...

$AA(i) = \max\{$

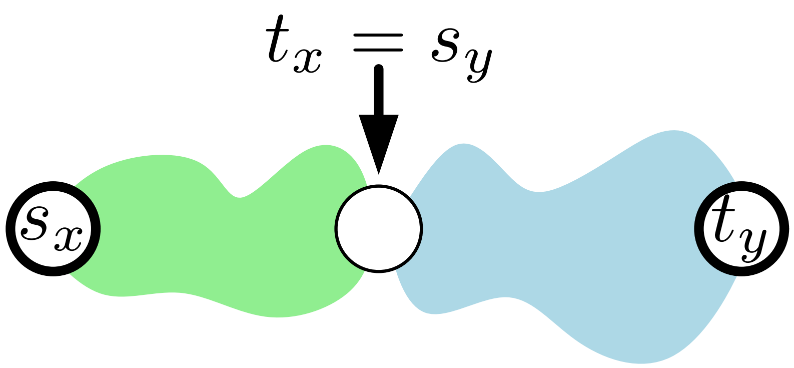
Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.



– If i is a series composition with children x and y, ...

$$AA(i) = \max\{\ AB(x) + BA(y),$$

Independent Set on SP-Trees

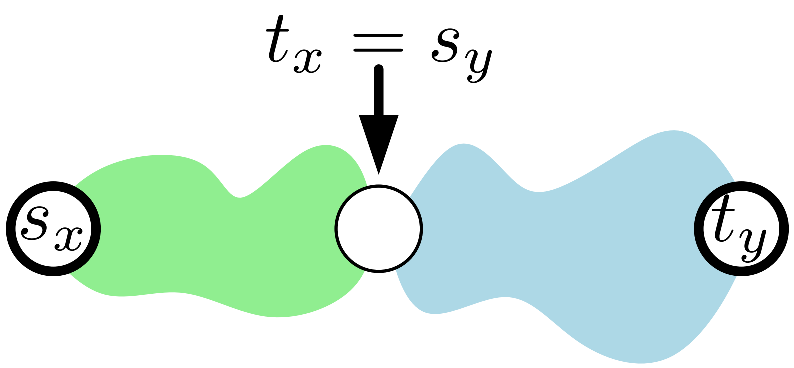
Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.

– If i is a series composition with children x and y, ...



$$AA(i) = \max\{\ AB(x) + BA(y),$$

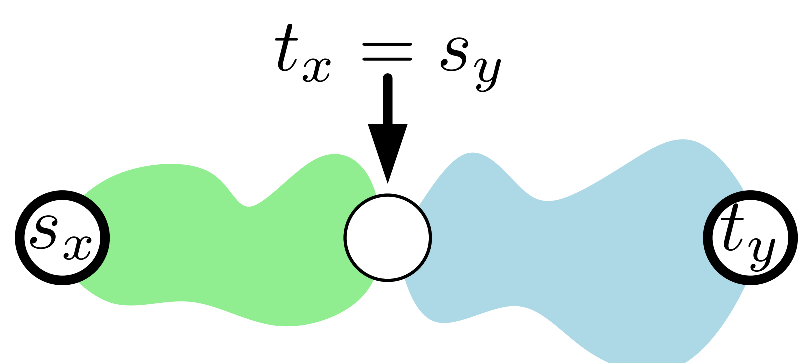
Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

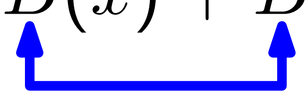
$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.



– If i is a series composition with children x and y, ...

$$AA(i) = \max\{\ AB(x) + BA(y),\ AA(x) + AA(y) - \omega(t_x)\ \}$$



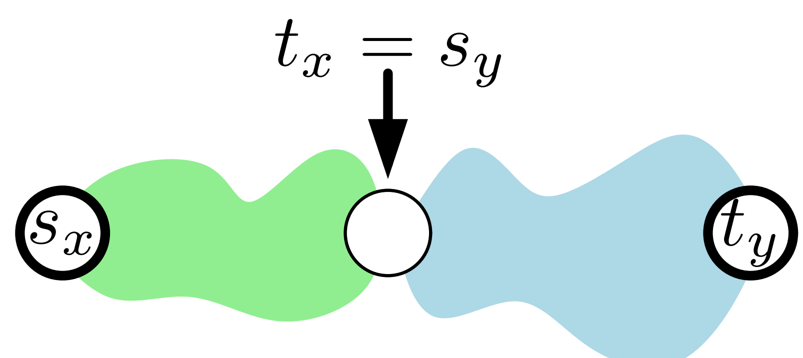
Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.



– If i is a series composition with children x and y, ...

$$AA(i) = \max\{\ AB(x) + BA(y),\ AA(x) + AA(y) - \omega(t_x)\ \}$$

Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$
where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$
where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.

– Other cases omitted... (easy exercise).

Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.

– Other cases omitted... (easy exercise).

– $O(1)$ time per SP-node.

Independent Set on SP-Trees

Dynamic program on SP-tree indexed by $G(i)$

$AA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

$BA(i) :=$ maximum weight of an independent set S in $G(i)$ where $s_i \notin S$ and $t_i \in S$

$AB(i)$ and $BB(i)$ are defined similarly.

– Other cases omitted... (easy exercise).

– $O(1)$ time per SP-node.

Thm. Given an n-vertex series-parallel graph G with its SP-tree, Independent Set on G can be solved in $O(n)$ time.

Generalization?

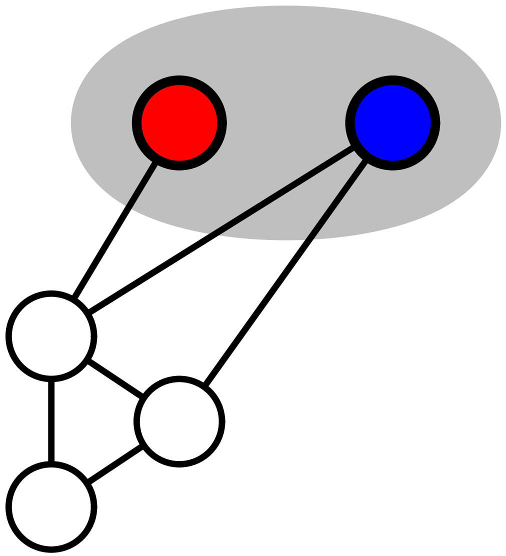
Many ways to generalize the concept of having a “tree structure”

Example: k-terminal graph $G = (V, E, T)$, $|T| = k$

Generalization?

Many ways to generalize the concept of having a "tree structure"

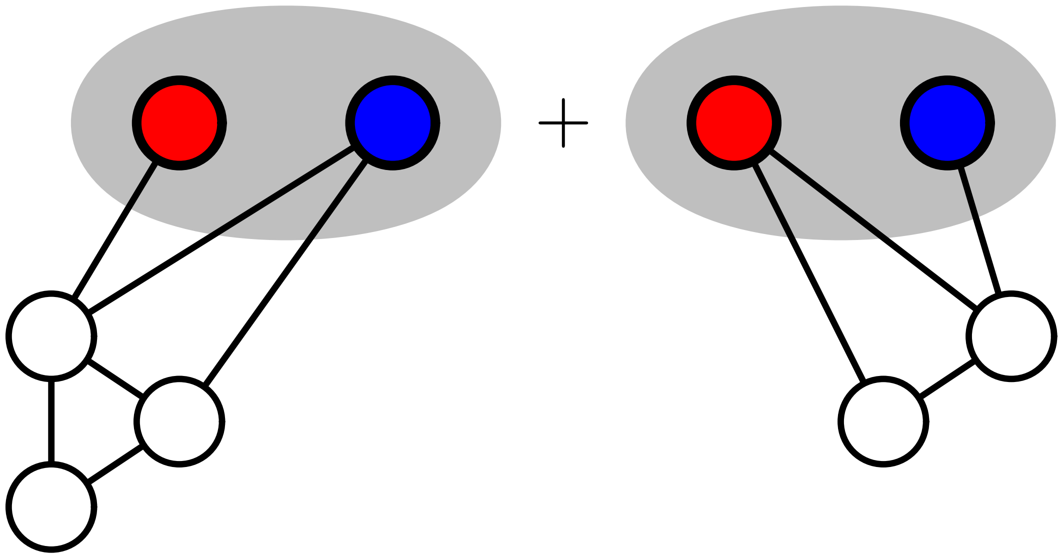
Example: k-terminal graph $G = (V, E, T)$, $|T| = k$



Generalization?

Many ways to generalize the concept of having a "tree structure"

Example: k-terminal graph $G = (V, E, T)$, $|T| = k$

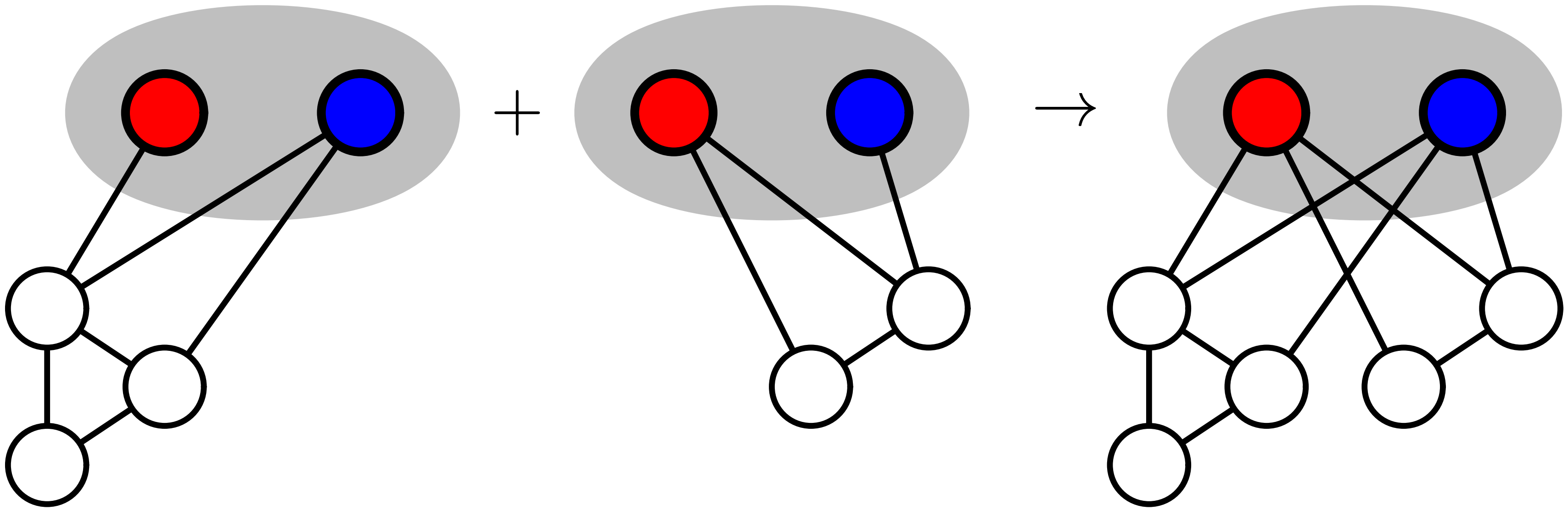


Generalization?

Many ways to generalize the concept of having a "tree structure"

Example: k-terminal graph $G = (V, E, T)$, $|T| = k$

Operation: "gluing"

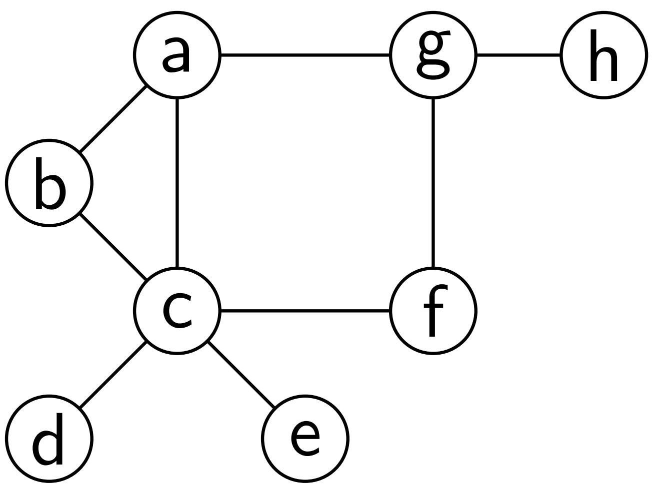


Example: Tree Decomposition

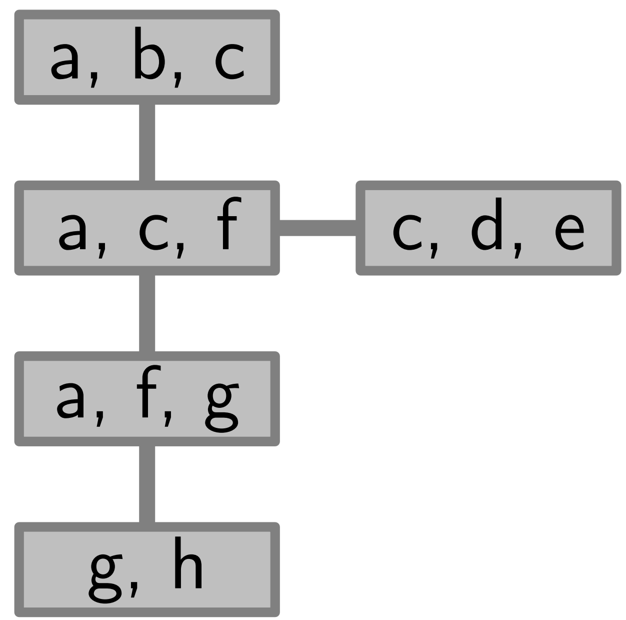
"bags"

A *tree decomposition* is a tree whose nodes map to subsets of V so that...

Graph $G = (V, E)$:



Tree Decomposition:

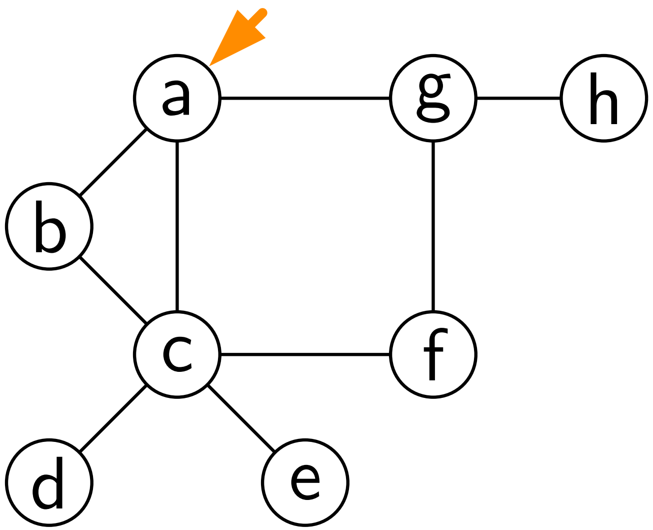


Example: Tree Decomposition

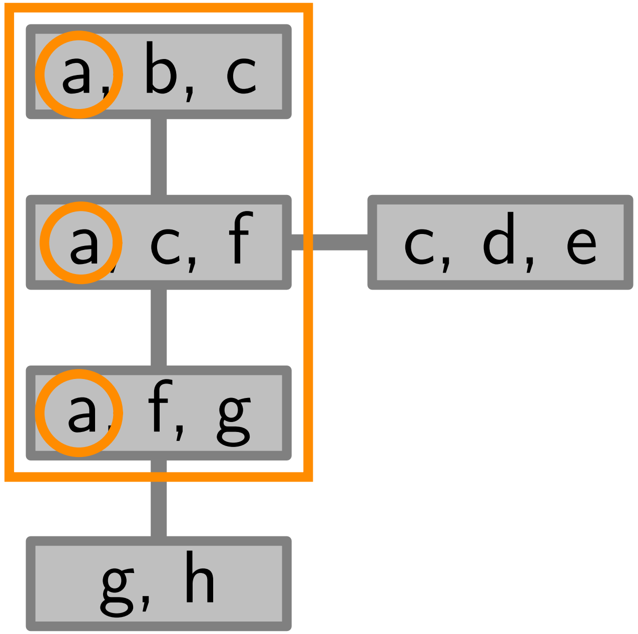
1 each vertex belongs to at least one bag

these bags are connected

Graph $G = (V, E)$:



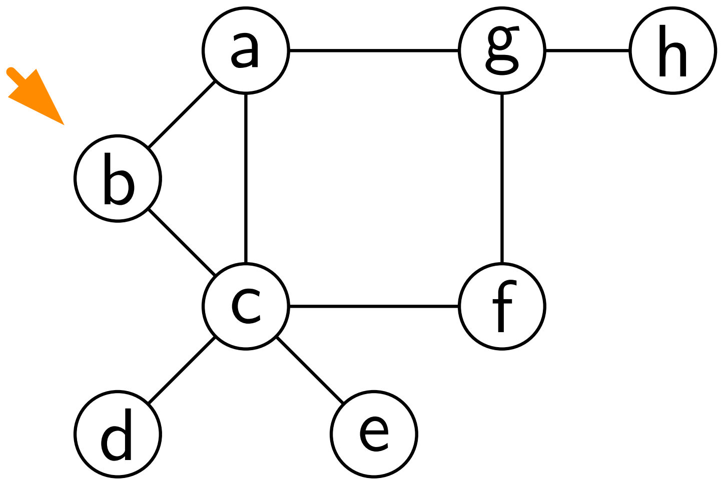
Tree Decomposition:



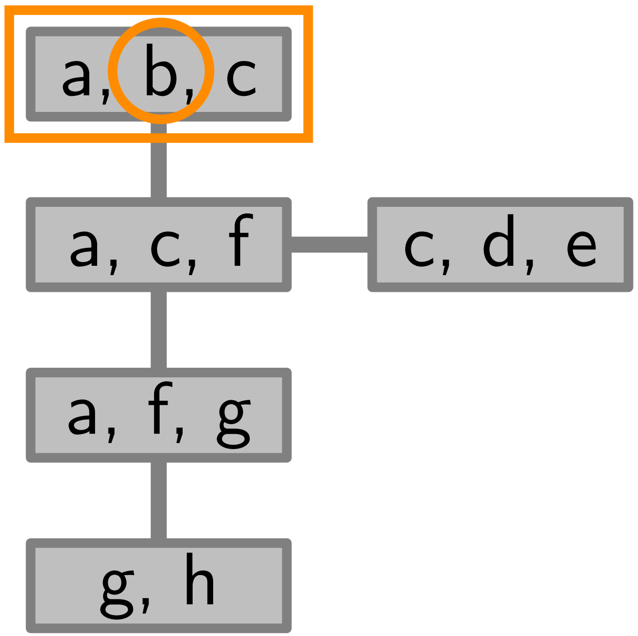
Example: Tree Decomposition

1 each vertex belongs to at least one bag
these bags are connected

Graph $G = (V, E)$:



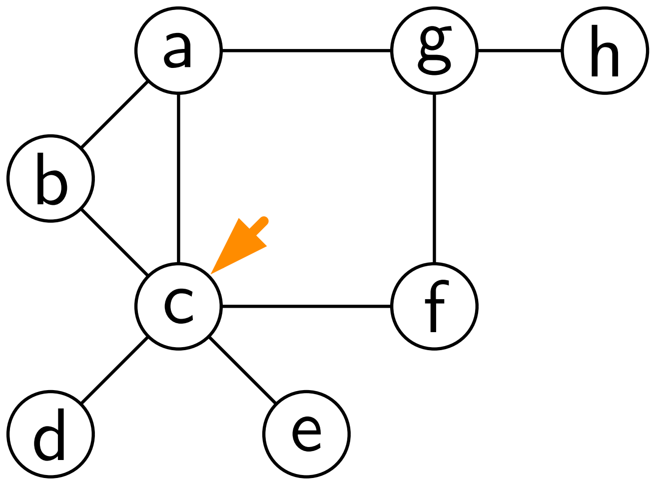
Tree Decomposition:



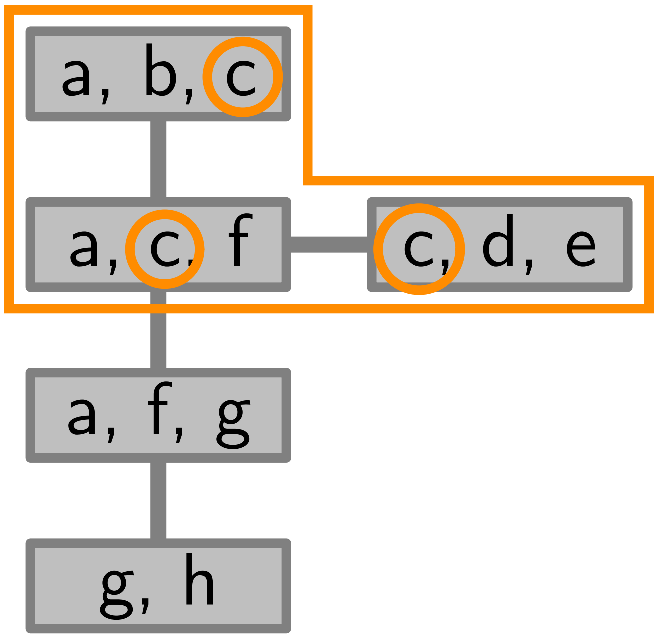
Example: Tree Decomposition

1 each vertex belongs to at least one bag
these bags are connected

Graph $G = (V, E)$:



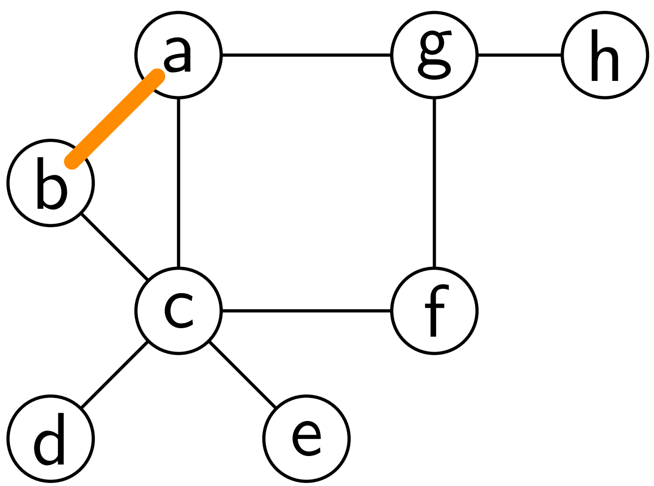
Tree Decomposition:



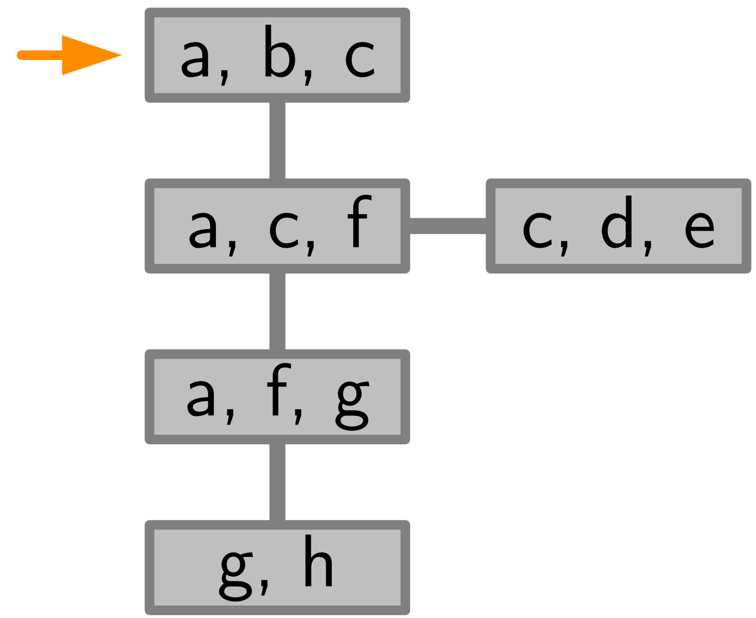
Example: Tree Decomposition

2 each edge is contained in at least one bag

Graph $G = (V, E)$:



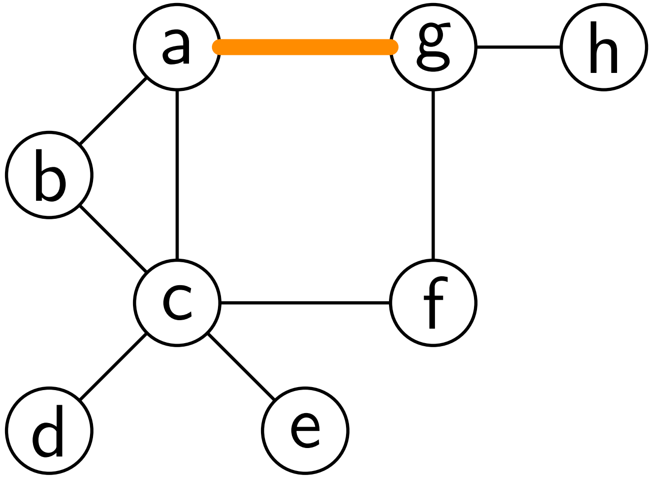
Tree Decomposition:



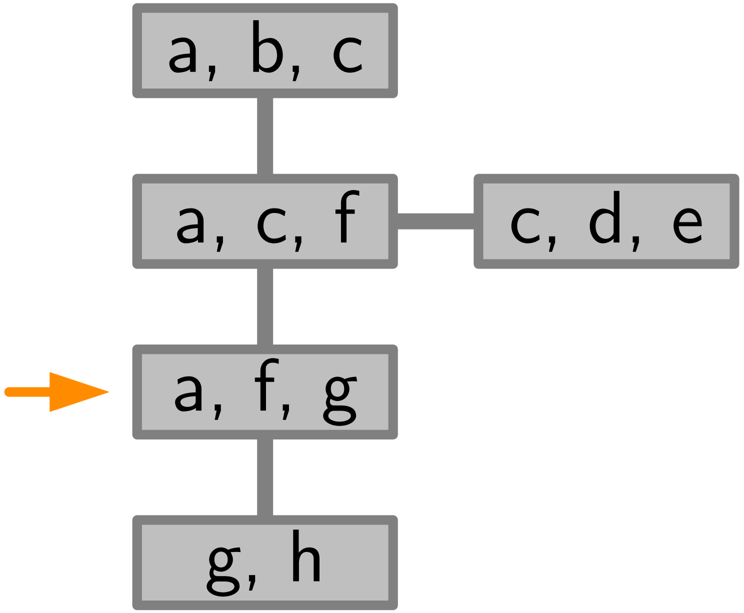
Example: Tree Decomposition

2 each edge is contained in at least one bag

Graph $G = (V, E)$:

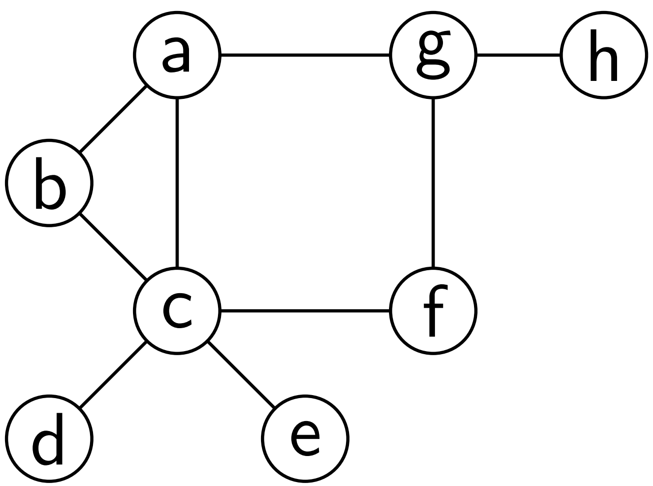


Tree Decomposition:



Example: Tree Decomposition

Graph $G = (V, E)$:

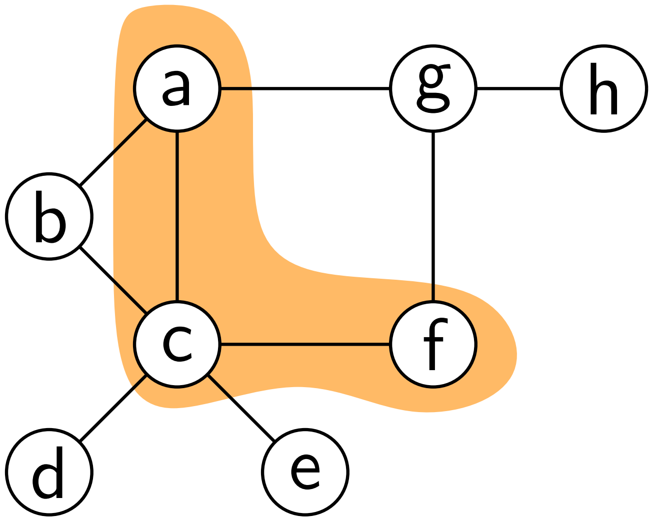


Tree Decomposition:

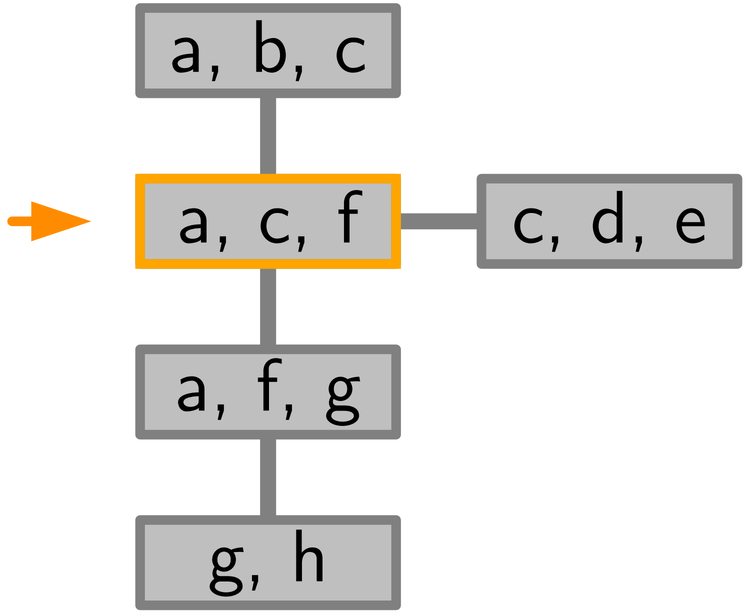
a, b, c,
d, e, f,
g, h

Example: Tree Decomposition

Graph $G = (V, E)$:

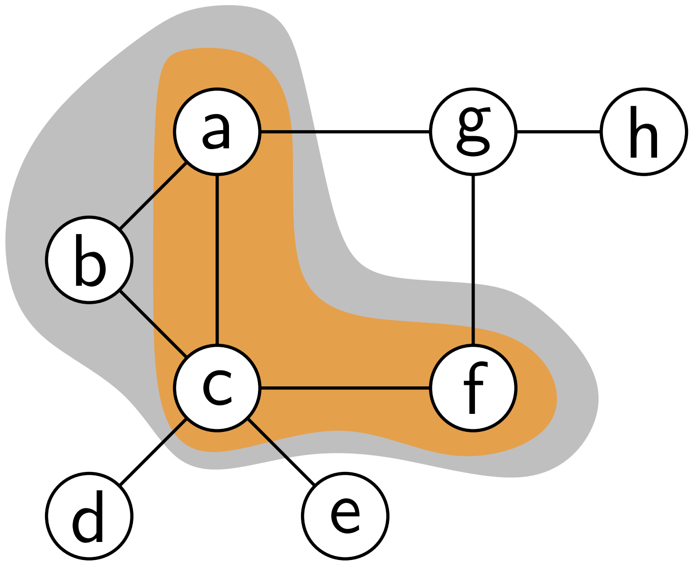


Tree Decomposition:

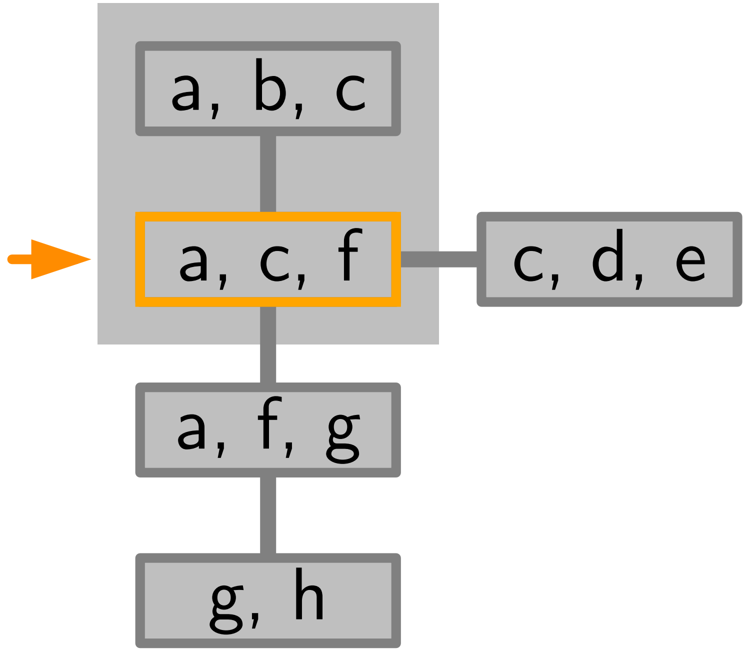


Example: Tree Decomposition

Graph $G = (V, E)$:

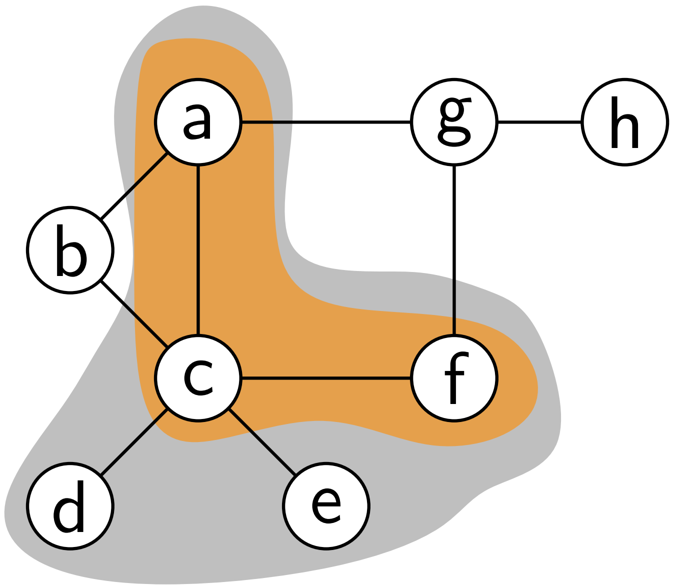


Tree Decomposition:

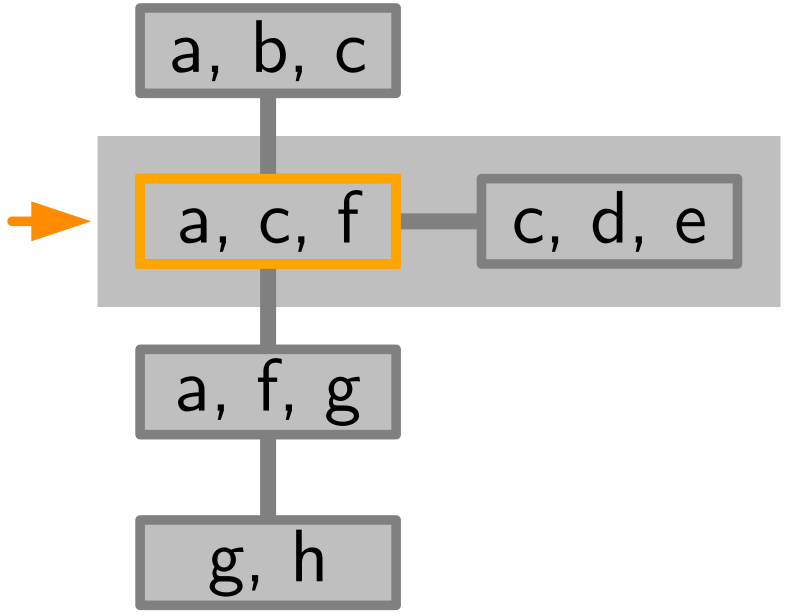


Example: Tree Decomposition

Graph $G = (V, E)$:

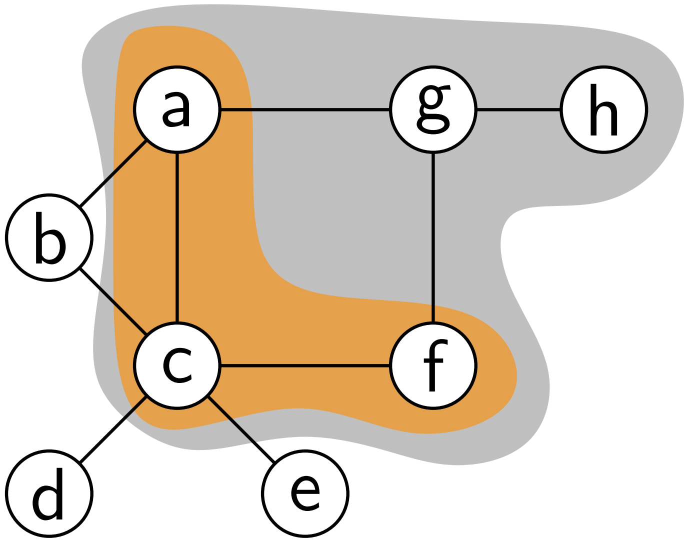


Tree Decomposition:

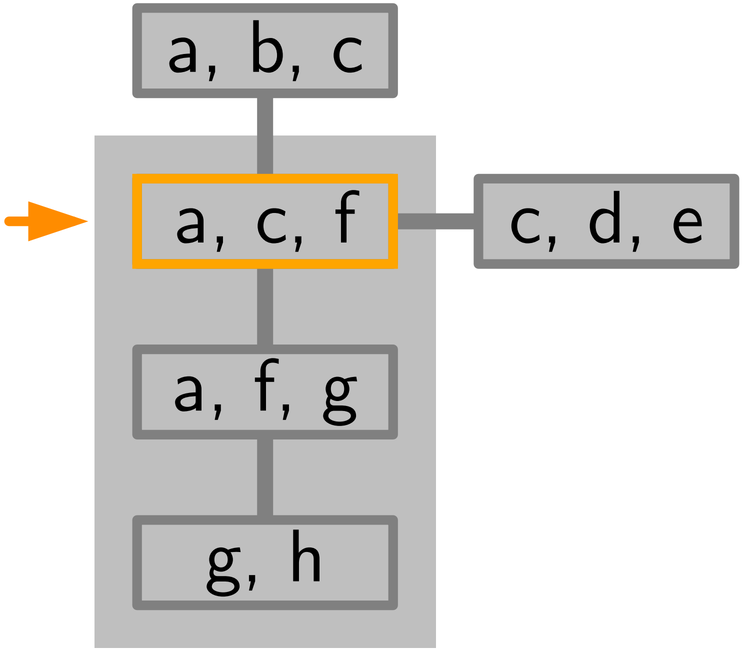


Example: Tree Decomposition

Graph $G = (V, E)$:

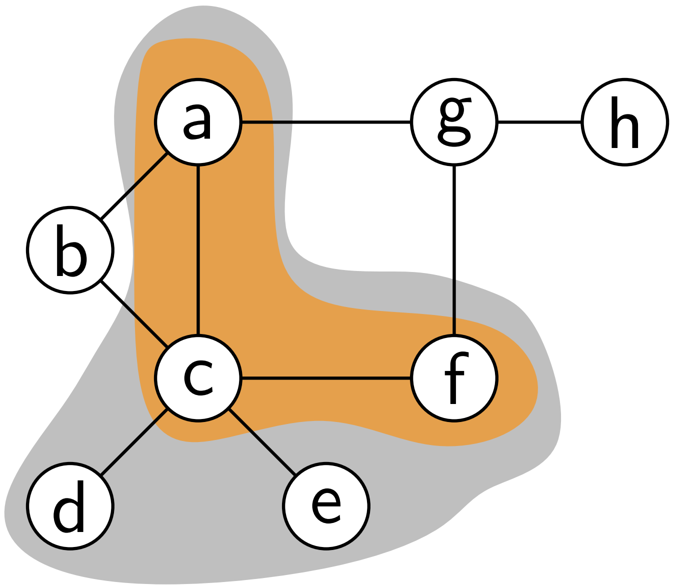


Tree Decomposition:

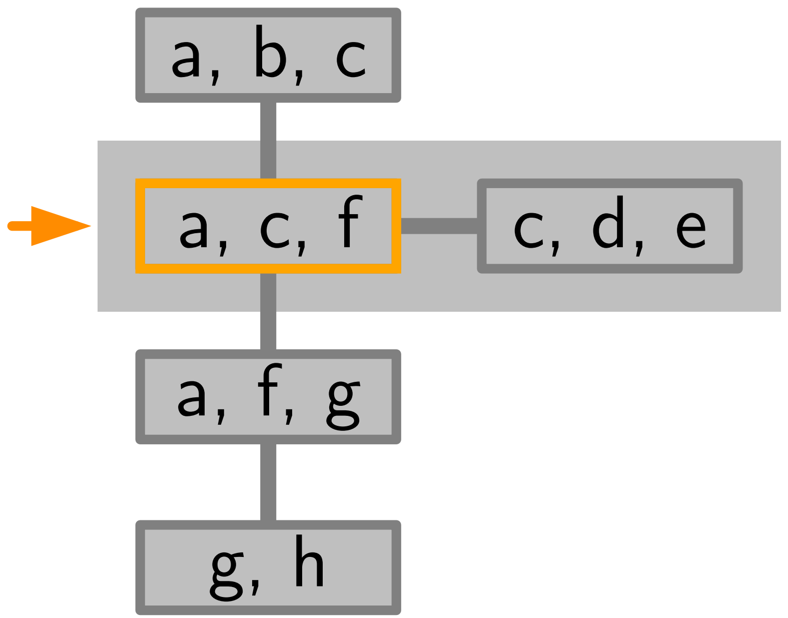


Example: Tree Decomposition

Graph $G = (V, E)$:

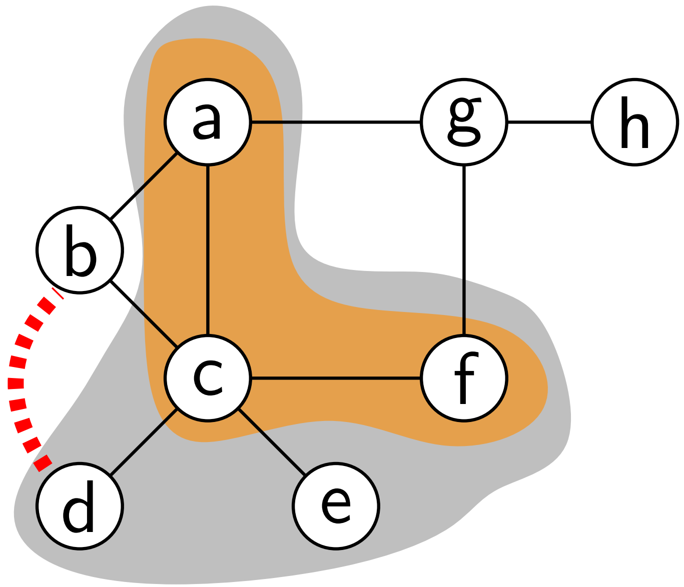


Tree Decomposition:



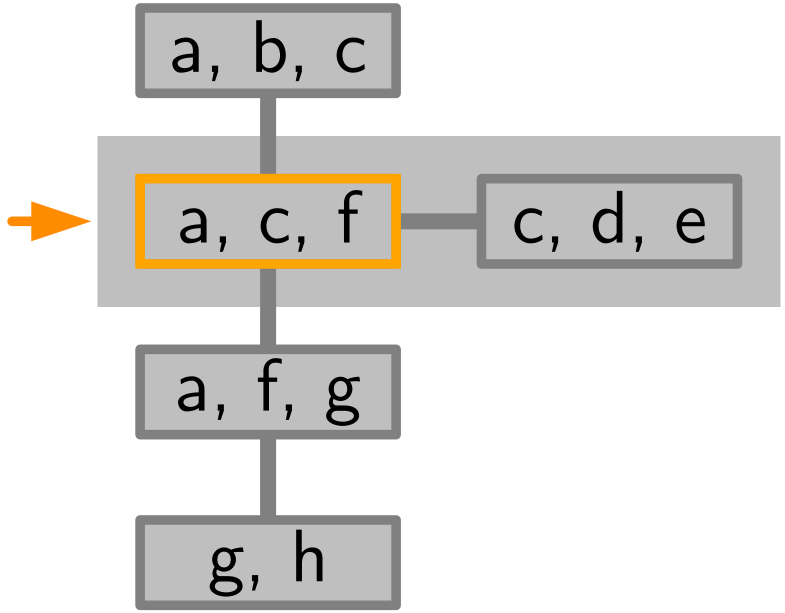
Example: Tree Decomposition

Graph $G = (V, E)$:



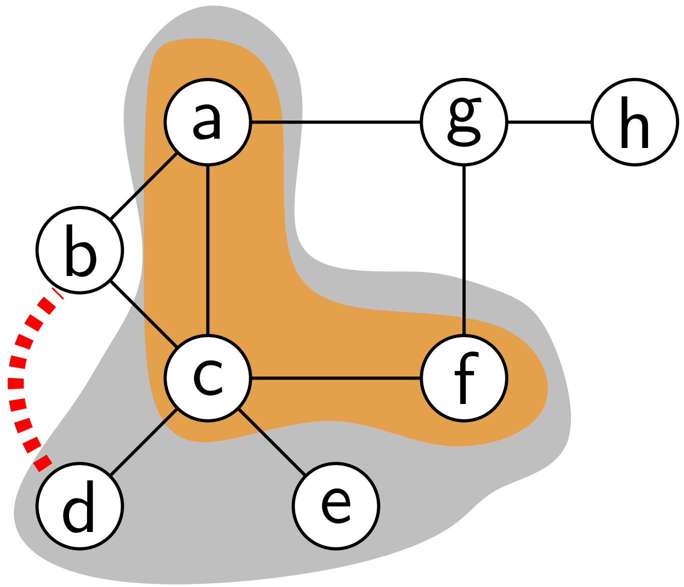
$\{b, d\} \notin E$

Tree Decomposition:



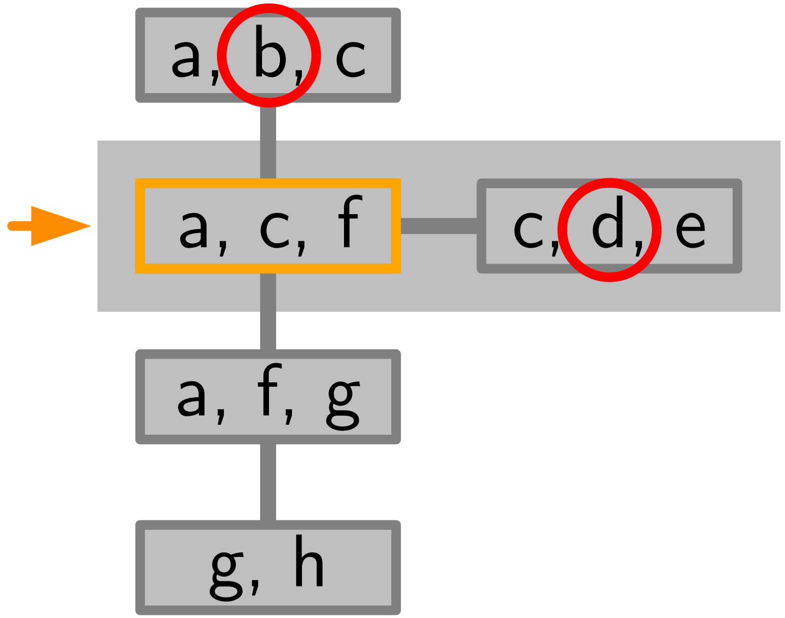
Example: Tree Decomposition

Graph $G = (V, E)$:



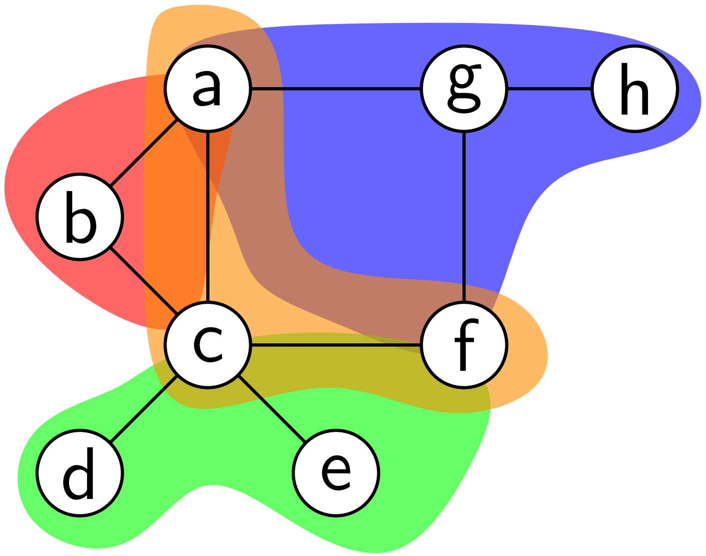
$\{b, d\} \notin E$

Tree Decomposition:

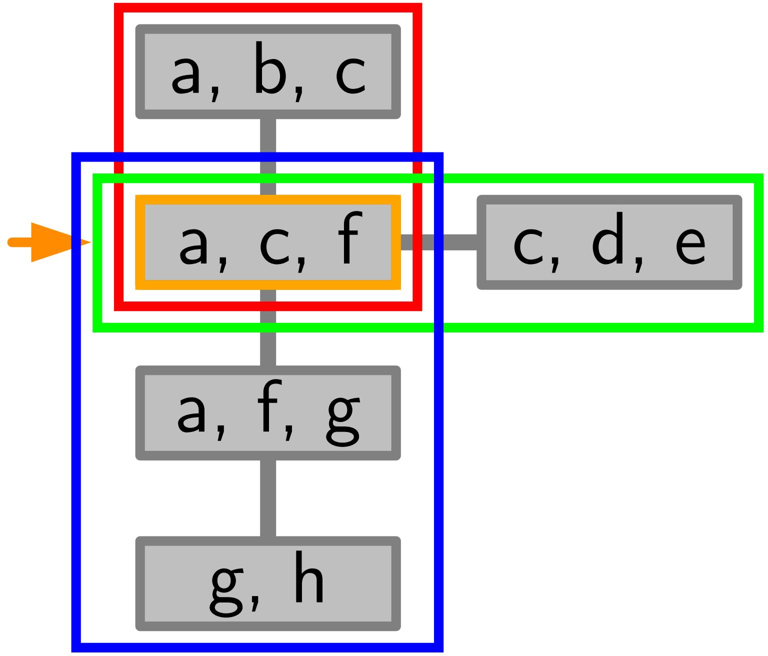


Example: Tree Decomposition

Graph $G = (V, E)$:



Tree Decomposition:



Tree Decompostion (formal)

Def. A *tree decomposition* of a graph $G = (V, E)$ is:

- a tuple $D = (X, T)$

Tree Decompostion (formal)

Def. A *tree decomposition* of a graph $G = (V, E)$ is:

- a tuple $D = (X, T)$
- $T = (P, F)$ is a tree

Tree Decompostion (formal)

Def. A *tree decomposition* of a graph $G = (V, E)$ is:

- a tuple $D = (X, T)$
- $T = (P, F)$ is a tree
- $X = \{X_p \mid p \in P\}$ is a set family of subsets of V ("bags"; one for each node in P)

Tree Decompostion (formal)

Def. A *tree decomposition* of a graph $G = (V, E)$ is:

- a tuple $D = (X, T)$
- $T = (P, F)$ is a tree
- $X = \{X_p \mid p \in P\}$ is a set family of subsets of V ("bags"; one for each node in P)
- $\bigcup_{p \in P} X_p = V$

Tree Decompostion (formal)

Def. A *tree decomposition* of a graph $G = (V, E)$ is:

- a tuple $D = (X, T)$
- $T = (P, F)$ is a tree
- $X = \{X_p \mid p \in P\}$ is a set family of subsets of V ("bags"; one for each node in P)
- $\bigcup_{p \in P} X_p = V$
- $\forall \{u, v\} \in E$ there is a $p \in P$ such that $u, v \in X_p$.

Tree Decompostion (formal)

Def. A *tree decomposition* of a graph $G = (V, E)$ is:

- a tuple $D = (X, T)$
- $T = (P, F)$ is a tree
- $X = \{X_p \mid p \in P\}$ is a set family of subsets of V ("bags"; one for each node in P)
- $\bigcup_{p \in P} X_p = V$
- $\forall \{u, v\} \in E$ there is a $p \in P$ such that $u, v \in X_p$.
- $\forall v \in V$ the set $\{p \in P \mid v \in X_p\}$ is connected in T.

Treewidth (formal)

- a tuple $D = (X, T)$
- $T = (P, F)$ is a tree

Def. Width (tree decomposition): $\max_{p \in P} |X_p| - 1$,
i.e., cardinality of the largest bag -1

Treewidth (formal)

- a tuple $D = (X, T)$
- $T = (P, F)$ is a tree

Def. Width (tree decomposition): $\max_{p \in P} |X_p| - 1$, ?!
i.e., cardinality of the largest bag -1

Treewidth (formal)

- a tuple $D = (X, T)$
- $T = (P, F)$ is a tree

Def. Width (tree decomposition): $\max_{p \in P} |X_p| - 1$, i.e., cardinality of the largest bag -1

Def. Treewidth $\mathrm{tw}(G)$ is the minimum width of a tree decomposition of G

Treewidth (formal)

- a tuple $D = (X, T)$
- $T = (P, F)$ is a tree

Def. Width (tree decomposition): $\max_{p \in P} |X_p| - 1$, i.e., cardinality of the largest bag -1

Def. Treewidth $\text{tw}(G)$ is the minimum width of a tree decomposition of G

Obs. $\text{tw}(G) < n$

Treewidth (formal)

- a tuple $D = (X, T)$
- $T = (P, F)$ is a tree

Def. Width (tree decomposition): $\max_{p \in P} |X_p| - 1$,
i.e., cardinality of the largest bag -1

Def. Treewidth $\text{tw}(G)$ is the minimum width of a tree decomposition of G

Obs. $\text{tw}(G) < n$

Question: Which graphs have treewidth 0?

Treewidth (formal)

- a tuple $D = (X, T)$
- $T = (P, F)$ is a tree

Def. Width (tree decomposition): $\max_{p \in P} |X_p| - 1$, i.e., cardinality of the largest bag -1

Def. Treewidth $\mathrm{tw}(G)$ is the minimum width of a tree decomposition of G

Obs. $\mathrm{tw}(G) < n$

Question: Which graphs have treewidth 0? $E = \varnothing$

Treewidth (formal)

- a tuple $D = (X, T)$
- $T = (P, F)$ is a tree

Def. Width (tree decomposition): $\max_{p \in P} |X_p| - 1$, ?!
i.e., cardinality of the largest bag -1

Def. Treewidth $\text{tw}(G)$ is the minimum width of a tree decomposition of G

Obs. $\text{tw}(G) < n$

Question: Which graphs have treewidth 0? $E = \varnothing$

Exercise: Trees have treewidth 1.

Treewidth (formal)

- a tuple $D = (X, T)$
- $T = (P, F)$ is a tree

Def. Width (tree decomposition): $\max_{p \in P} |X_p| - 1$, i.e., cardinality of the largest bag -1

Def. Treewidth $\mathrm{tw}(G)$ is the minimum width of a tree decomposition of G

Obs. $\mathrm{tw}(G) < n$

Question: Which graphs have treewidth 0?

Exercise: Trees have treewidth 1.

Exercise: Series-parallel graphs have treewidth 2

Treewidth (formal)

- a tuple $D = (X, T)$
- $T = (P, F)$ is a tree

Def. Width (tree decomposition): $\max_{p \in P} |X_p| - 1$, i.e., cardinality of the largest bag -1

Def. Treewidth $\text{tw}(G)$ is the minimum width of a tree decomposition of G

Obs. $\text{tw}(G) < n$

Question: Which graphs have treewidth 0?

Exercise: Trees have treewidth 1.

Exercise: Series-parallel graphs have treewidth 2

Thm. There is a tree decomposition of width $\text{tw}(G)$ where the tree size $|P|$ is polynomial in n.

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.

Ex.: k-Vertex Cover

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.

Ex.: k-Vertex Cover **FPT**

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.

Ex.: k-Vertex Cover **FPT**

k-Independent Set

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.

Ex.: k-Vertex Cover **FPT**

k-Independent Set $W[1]$-**comp.**

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.

Ex.: k-Vertex Cover **FPT**

k-Independent Set $W[1]$-**comp.**

k-Dominating Set

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.

Ex.:	k-Vertex Cover	**FPT**
	k-Independent Set	$W[1]$-**comp.**
	k-Dominating Set	$W[2]$-**comp.**

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.

Ex.: k-Vertex Cover — **FPT**

k-Independent Set — $W[1]$-**comp.**

k-Dominating Set — $W[2]$-**comp.**

k-Coloring

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.

Ex.: k-VERTEX COVER **FPT**

k-INDEPENDENT SET $W[1]$-**comp.**

k-DOMINATING SET $W[2]$-**comp.**

k-COLORING **NP-comp.** $k \geq 3$

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.

Ex.:	k-Vertex Cover	**FPT**
	k-Independent Set	$W[1]$-**comp.**
	k-Dominating Set	$W[2]$-**comp.**
	k-Coloring	**NP-comp.** $k \geq 3$

↑

"natural parameterization"

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.

Ex.:	k-Vertex Cover	**FPT**
	k-Independent Set	$W[1]$-**comp.**
	k-Dominating Set	$W[2]$-**comp.**
	k-Coloring	**NP-comp.** $k \geq 3$

Independent Set (treewidth)

Given: Graph G, number k
Parameter: tw(G)
Question: Does G have an independent set of size $\geq k$?

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.
$O(f(\text{tw}(G)) \cdot poly(n))$ time.

Ex.: k-Vertex Cover **FPT**

k-Independent Set $W[1]$-**comp.**

k-Dominating Set $W[2]$-**comp.**

k-Coloring **NP-comp.** $k \geq 3$

Independent Set (treewidth)

Given: Graph G, number k
Parameter: $\text{tw}(G)$
Question: Does G have an independent set of size $\geq k$?

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.
$O(f(\text{tw}(G)) \cdot poly(n))$ time.

Ex.:	k-Vertex Cover	**FPT**
	k-Independent Set	$W[1]$-**comp.**
	k-Dominating Set	$W[2]$-**comp.**
	k-Coloring	**NP-comp.** $k \geq 3$
	Independent Set (treewidth)	**FPT**

Given: Graph G, number k
Parameter: $\text{tw}(G)$
Question: Does G have an independent set of size $\geq k$?

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.

$O(f(\text{tw}(G)) \cdot poly(n))$ time.

Ex.:

k-Vertex Cover	**FPT**
k-Independent Set	$W[1]$-**comp.**
k-Dominating Set	$W[2]$-**comp.**
k-Coloring	**NP-comp.** $k \geq 3$
Independent Set (treewidth)	**FPT**
List Coloring (treewidth)	

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.
$O(f(\text{tw}(G)) \cdot poly(n))$ time.

Ex.:	k-VERTEX COVER	**FPT**
	k-INDEPENDENT SET	$W[1]$-**comp.**
	k-DOMINATING SET	$W[2]$-**comp.**
	k-COLORING	**NP-comp.** $k \geq 3$
	INDEPENDENT SET (TREEWIDTH)	**FPT**
	LIST COLORING (TREEWIDTH)	$W[1]$-**comp.**

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.
$O(f(\text{tw}(G)) \cdot poly(n))$ time.

Ex.:	k-Vertex Cover	**FPT**
	k-Independent Set	$W[1]$-**comp.**
	k-Dominating Set	$W[2]$-**comp.**
	k-Coloring	**NP-comp.** $k \geq 3$
	Independent Set (treewidth)	**FPT**
	List Coloring (treewidth)	$W[1]$-**comp.**
	Channel Assignment (treewidth)	

Parameterized Problems

Given: Instance of size n and parameter k

Def. Problem is FPT when solvable in $O(f(k) \cdot poly(n))$ time.

$O(f(\text{tw}(G)) \cdot poly(n))$ time.

Ex.:

k-Vertex Cover	**FPT**
k-Independent Set	$W[1]$-**comp.**
k-Dominating Set	$W[2]$-**comp.**
k-Coloring	**NP-comp.** $k \geq 3$
Independent Set (treewidth)	**FPT**
List Coloring (treewidth)	$W[1]$-**comp.**
Channel Assignment (treewidth)	**NP-comp.** $k \geq 3$

Computing Treewidth

TREEWIDTH
Given: Graph $G = (V, E)$, number k
Question: $\mathsf{tw}(G) \leq k$?

Computing Treewidth

TREEWIDTH

Given: Graph $G = (V, E)$, number k

Question: $\mathrm{tw}(G) \leq k$?

Thm. TREEWIDTH is NP-complete.

Computing Treewidth

TREEWIDTH
Given: Graph $G = (V, E)$, number k
Question: tw$(G) \leq k$?

Thm. TREEWIDTH is NP-complete.

k-TREEWIDTH
Given: graph $G = (V, E)$
Parameter: number k
Question: tw$(G) \leq k$?

Computing Treewidth

TREEWIDTH
Given: Graph $G = (V, E)$, number k
Question: $\text{tw}(G) \leq k$?

Thm. TREEWIDTH is NP-complete.

k-TREEWIDTH
Given: graph $G = (V, E)$
Parameter: number k
Question: $\text{tw}(G) \leq k$?

Thm. k-TREEWIDTH is FPT.

Computing Treewidth

TREEWIDTH
Given: Graph $G = (V, E)$, number k
Question: $\text{tw}(G) \leq k$?

Thm. TREEWIDTH is NP-complete.

k-TREEWIDTH
Given: graph $G = (V, E)$
Parameter: number k
Question: $\text{tw}(G) \leq k$?

Thm. k-TREEWIDTH is FPT.

– Actually *fixed-parameter linear*: runtime $O(f(k) \cdot n)$

Computing Treewidth

Treewidth
Given: Graph $G = (V, E)$, number k
Question: $\mathrm{tw}(G) \leq k$?

Thm. Treewidth is NP-complete.

k-Treewidth
Given: graph $G = (V, E)$
Parameter: number k
Question: $\mathrm{tw}(G) \leq k$?

Thm. k-Treewidth is FPT.

– Actually *fixed-parameter linear*: runtime $O(f(k) \cdot n)$
– Algorithm is constructive (provides an optimal tree decomp.)

Computing Treewidth

TREEWIDTH
Given: Graph $G = (V, E)$, number k
Question: $\mathrm{tw}(G) \leq k$?

Thm. TREEWIDTH is NP-complete.

k-TREEWIDTH
Given: graph $G = (V, E)$
Parameter: number k
Question: $\mathrm{tw}(G) \leq k$?

Thm. k-TREEWIDTH is FPT.

– Actually *fixed-parameter linear*: runtime $O(f(k) \cdot n)$
– Algorithm is constructive (provides an optimal tree decomp.)
– How can we make "fixed-treewidth-tractable" algorithms?

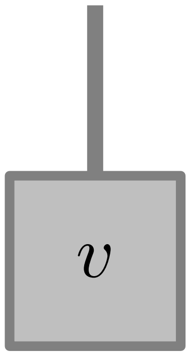
Tool #1: Nice Tree Decompositoins

In a *nice* tree decomp., one bag is marked as the root and there are only 4 types of bags:

Tool #1: Nice Tree Decompositoins

In a *nice* tree decomp., one bag is marked as the root and there are only 4 types of bags:

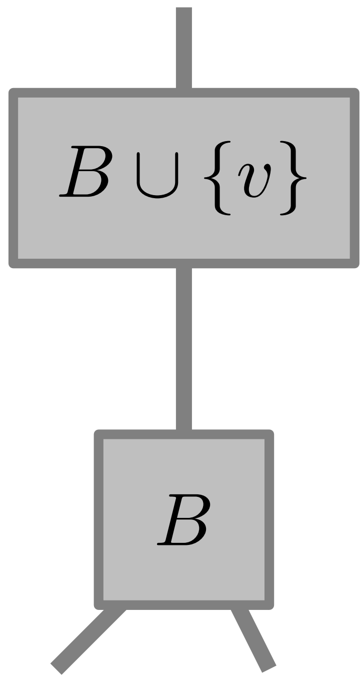
- **Leaf:** the bag is a leaf and contains only one vertex



Tool #1: Nice Tree Decompositoins

In a *nice* tree decomp., one bag is marked as the root and there are only 4 types of bags:

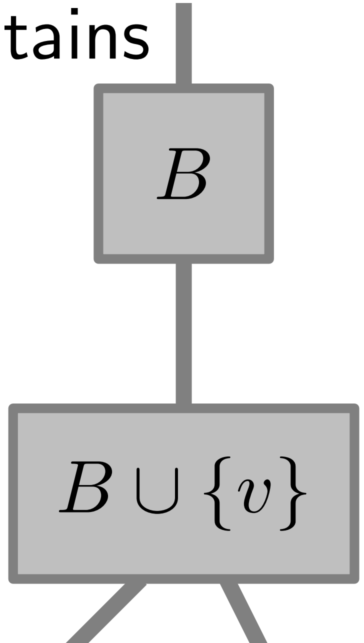
- **Leaf:** the bag is a leaf and contains only one vertex
- **Introduce:**
  The bag has exactly one child and contains the child's vertices and exactly one new vertex.



Tool #1: Nice Tree Decompositoins

In a *nice* tree decomp., one bag is marked as the root and there are only 4 types of bags:

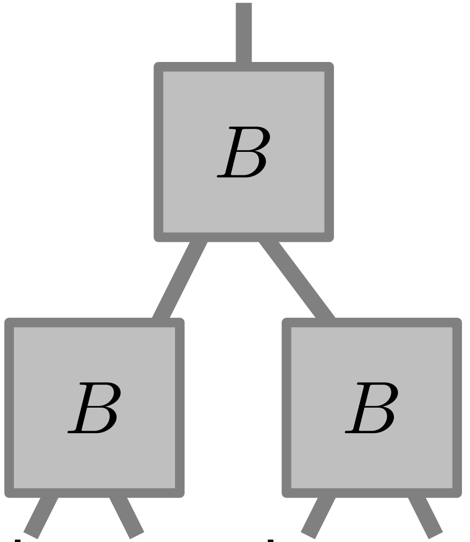
- **Leaf:** the bag is a leaf and contains only one vertex
- **Introduce:**
  The bag has exactly one child and contains the child's vertices and exactly one new vertex.
- **Forget:**
  The bag has exactly one child and contains one vertex fewer than the child.



Tool #1: Nice Tree Decompositoins

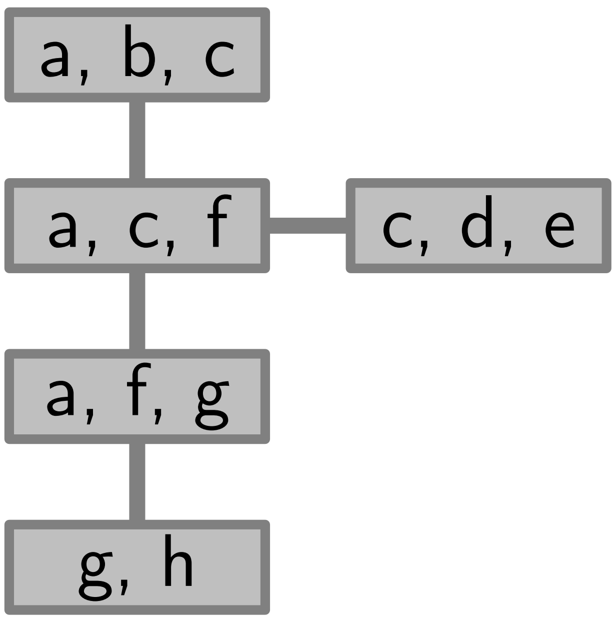
In a *nice* tree decomp., one bag is marked as the root and there are only 4 types of bags:

- **Leaf:** the bag is a leaf and contains only one vertex
- **Introduce:**
  The bag has exactly one child and contains the child's vertices and exactly one new vertex.
- **Forget:**
  The bag has exactly one child and contains one vertex fewer than the child.
- **Join:**
  The bag has exactly two children and these three nodes have exactly the same vertices

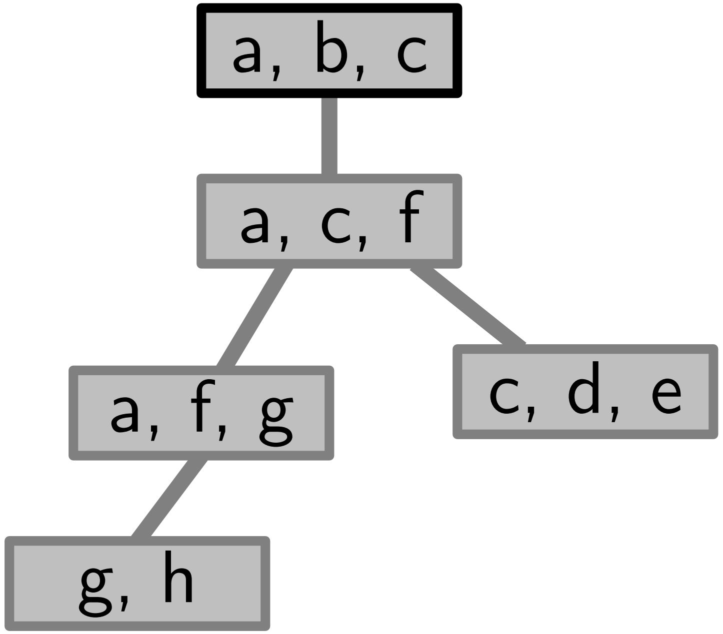


Tool #1: Nice Tree Decompositoins

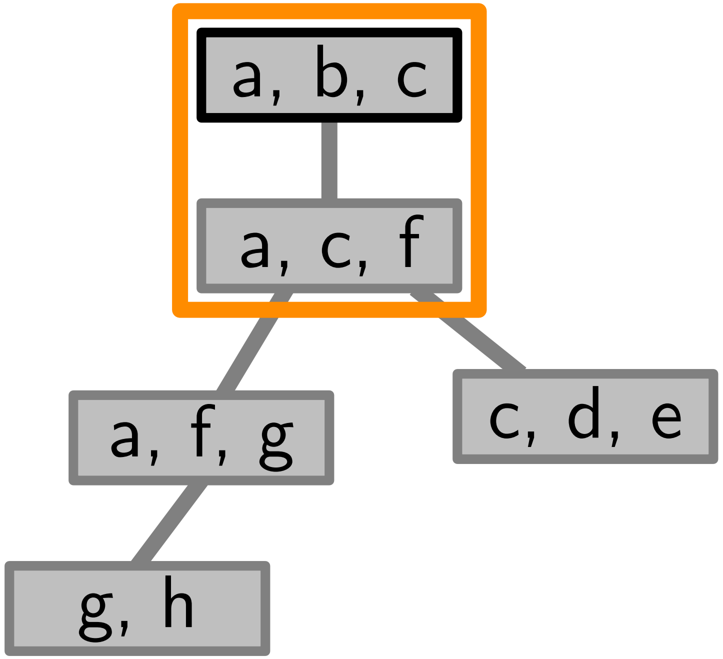
Thm. For each tree decomposition, there is a nice tree decomposition of the same width and polynomially many more bags. The nice decomposition can be constructed in polynomial time.



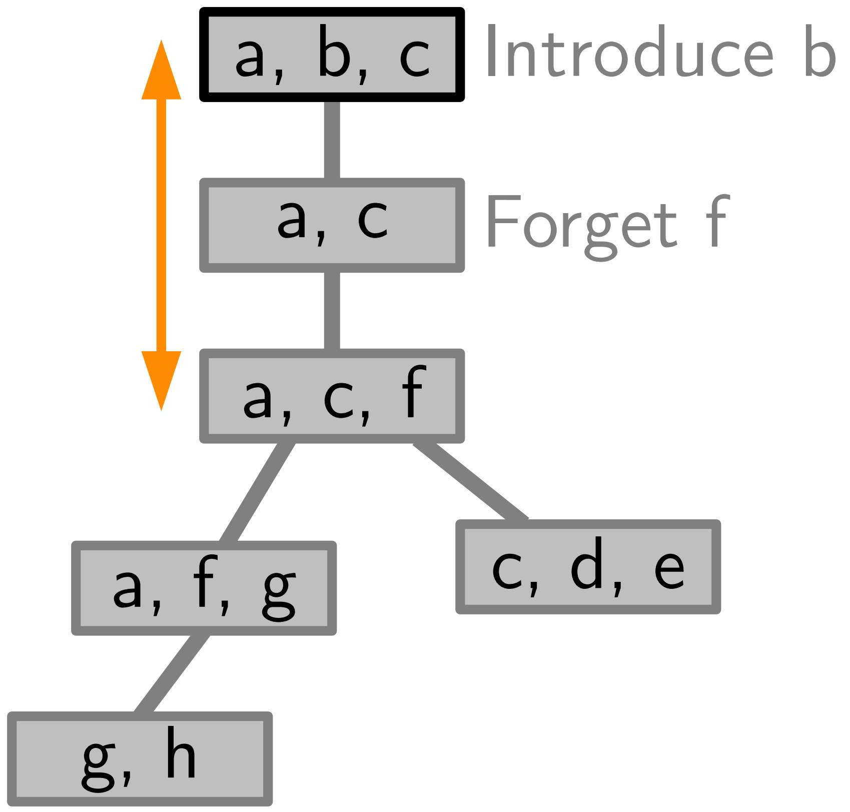
Tool #1: Nice Tree Decompositoins



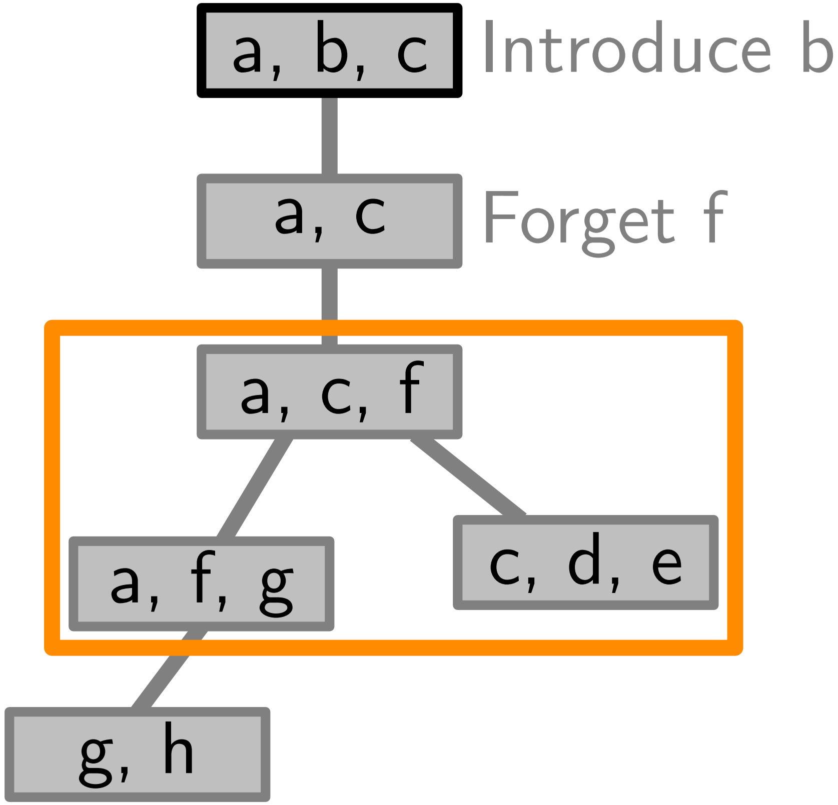
Tool #1: Nice Tree Decompositoins



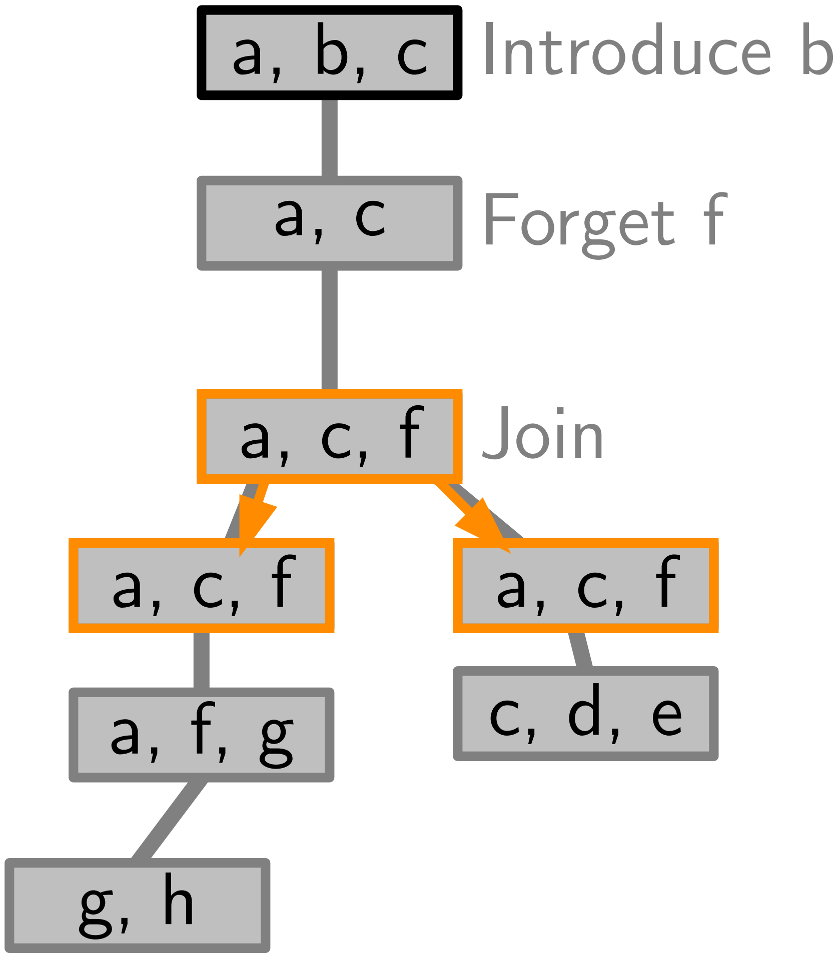
Tool #1: Nice Tree Decompositoins



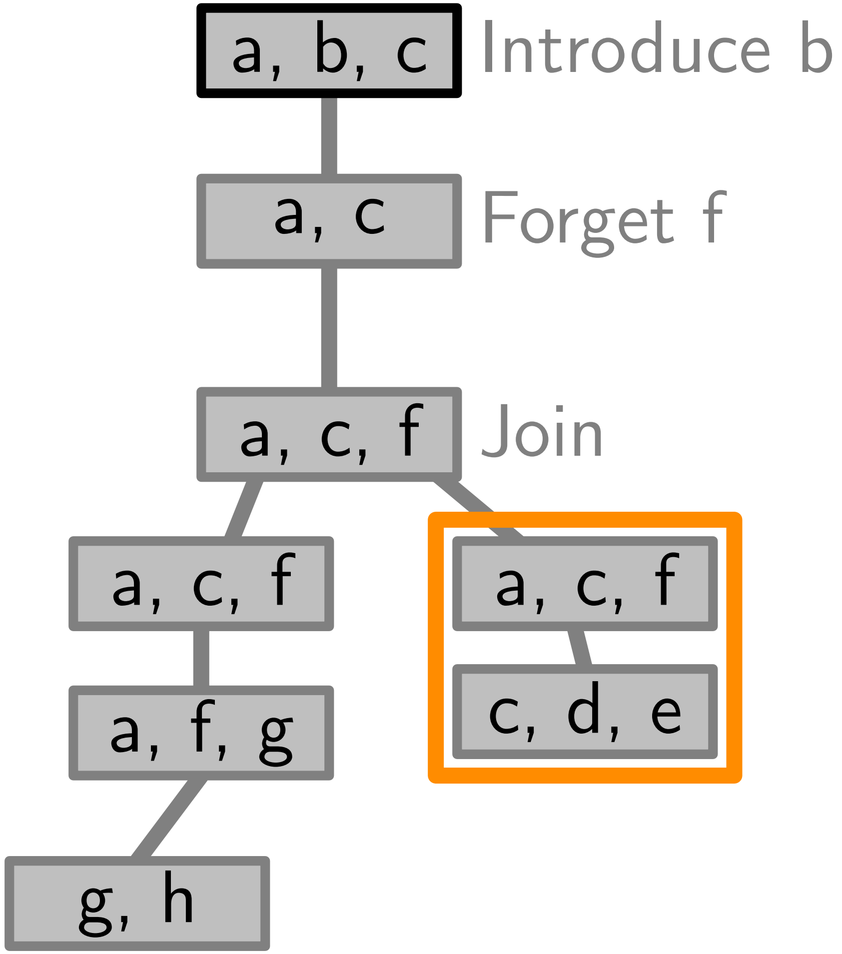
Tool #1: Nice Tree Decompositoins



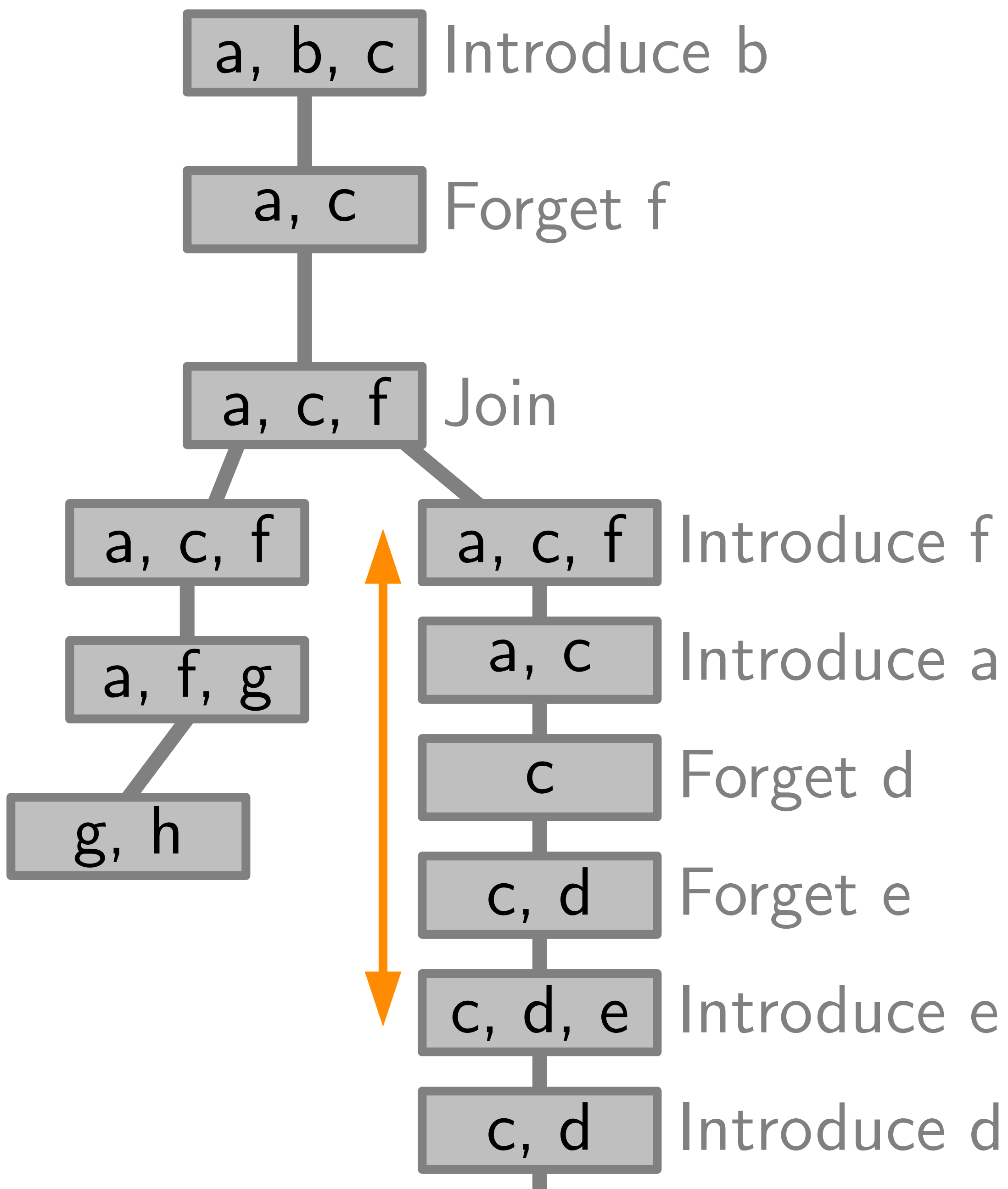
Tool #1: Nice Tree Decompositoins



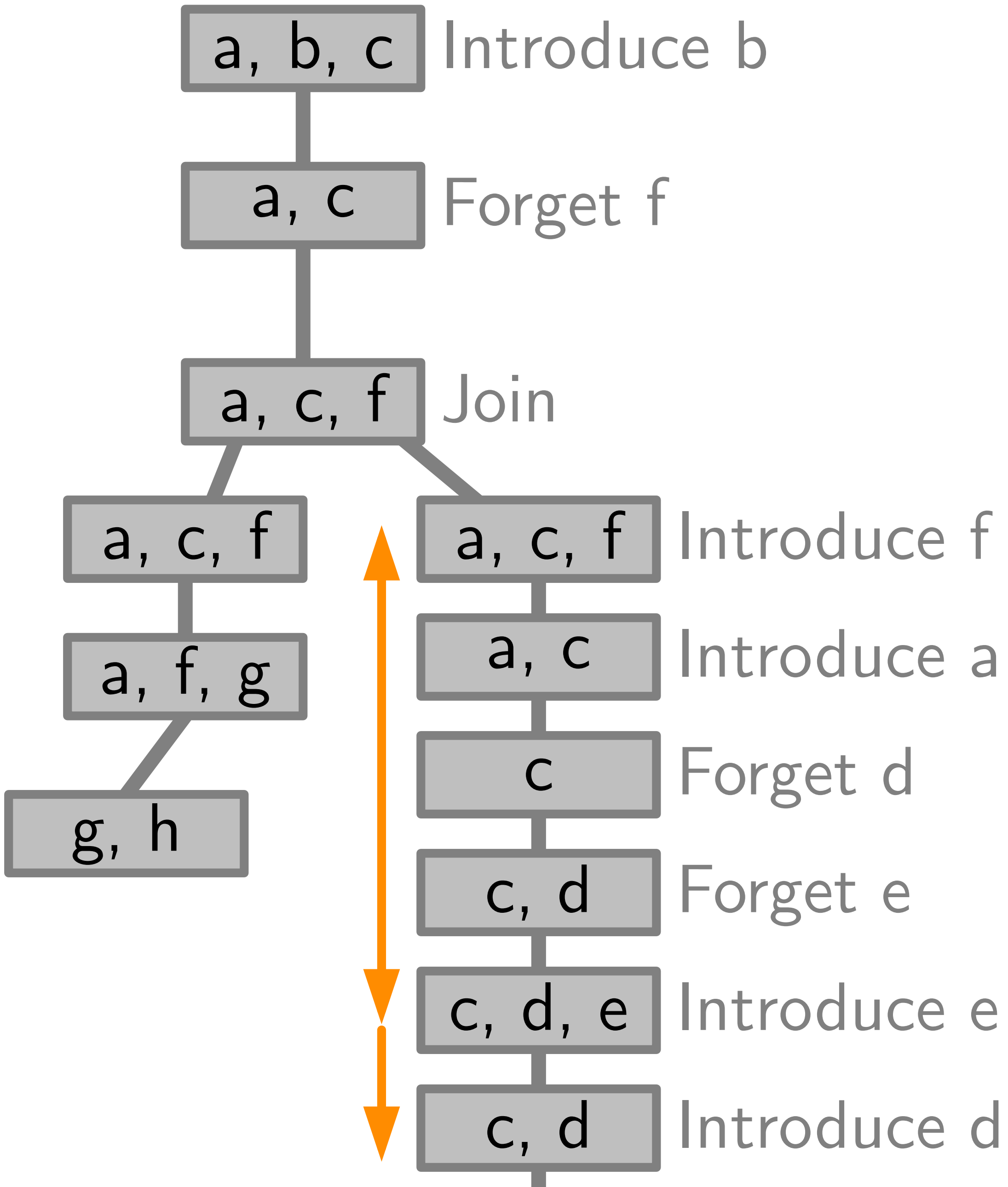
Tool #1: Nice Tree Decompositoins



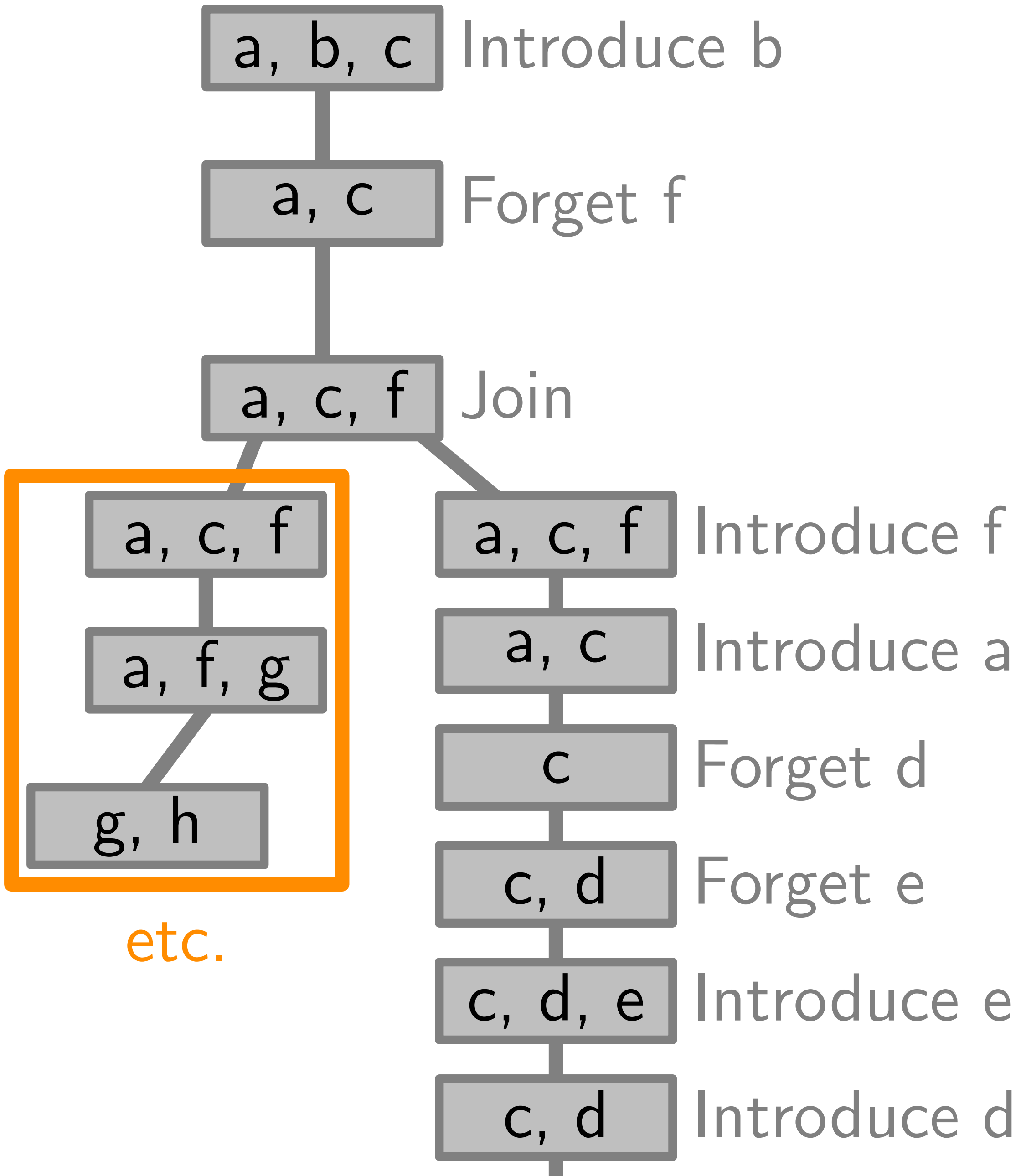
Tool #1: Nice Tree Decompositoins



Tool #1: Nice Tree Decompositoins



Tool #1: Nice Tree Decompositoins



Tool #2: DP (on Nice Tree Decompositions)

Thm. k-Treewidth is FPT

Tool #2: DP (on Nice Tree Decompositions)

Thm. k-Treewidth is FPT

Thm. Tree decompositions can be made *nice* in poly-time.

Tool #2: DP (on Nice Tree Decompositions)

Thm. k-TREEWIDTH is FPT

Thm. Tree decompositions can be made *nice* in poly-time.

Corollary. For FPT-Algorithms it suffices to consider nice tree decompositions.

Tool #2: DP (on Nice Tree Decompositions)

Thm. k-Treewidth is FPT

Thm. Tree decompositions can be made *nice* in poly-time.

Corollary. For FPT-Algorithms it suffices to consider nice tree decompositions.

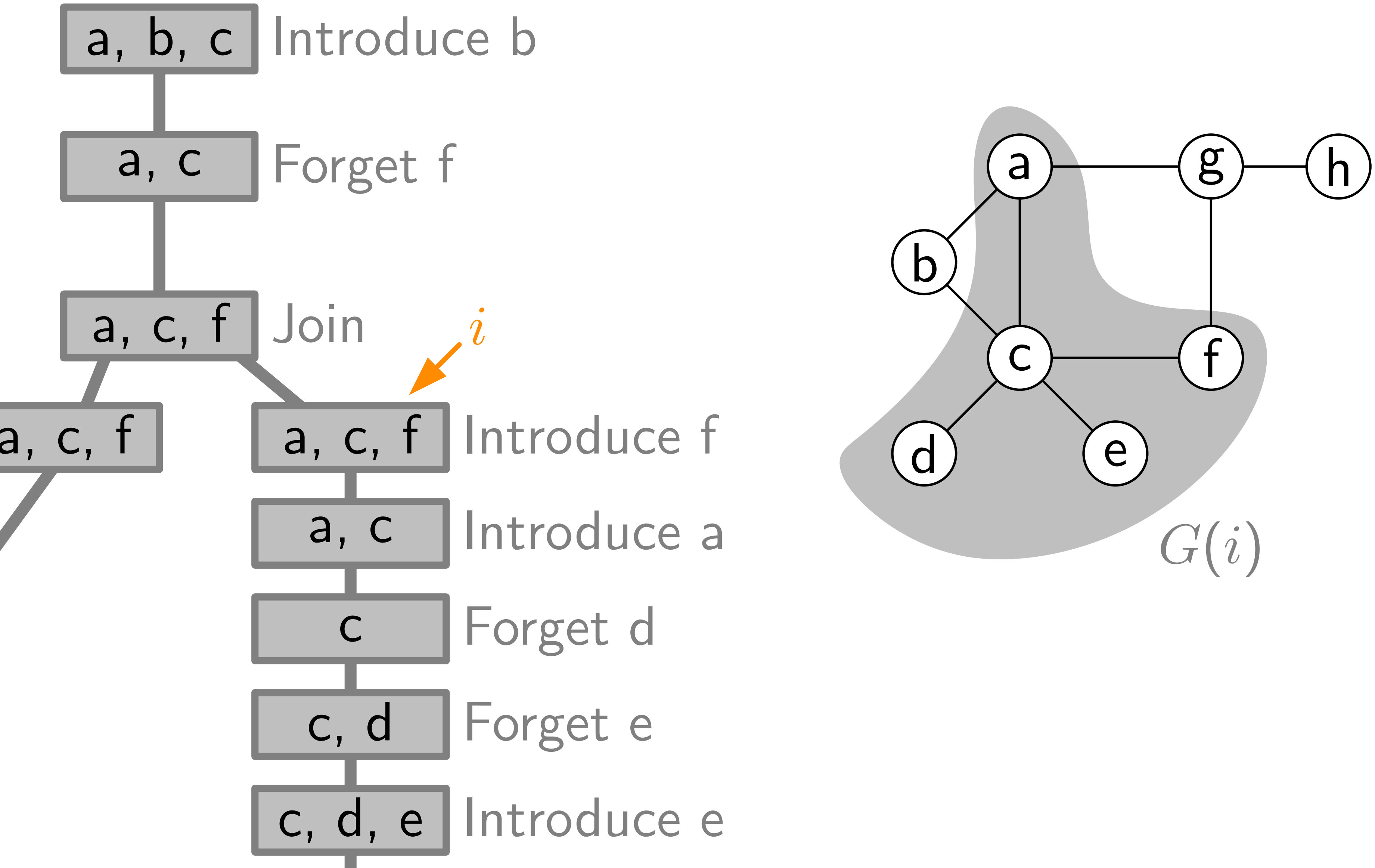
Strategy: Build a recurrence for each type of bag, and use dynamic programming.

Independent Set Using Nice Tree Decomp.

Let $G(i) :=$ graph induced by the vertices in the subtree at i.

Independent Set Using Nice Tree Decomp.

Let $G(i)$:= graph induced by the vertices in the subtree at i.



Independent Set Using Nice Tree Decomp.

Let $G(i) :=$ graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

Independent Set Using Nice Tree Decomp.

Let $G(i)$:= graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i, S)$:= maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.

Independent Set Using Nice Tree Decomp.

Let $G(i) :=$ graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i, S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.

If i is a **Leaf** ...

Independent Set Using Nice Tree Decomp.

Let $G(i) :=$ graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i, S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.

If i is a **Leaf** ...

Sei $X_i = \{v\}$.

$R(i, \{v\}) =$

Independent Set Using Nice Tree Decomp.

Let $G(i)$:= graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i, S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.

If i is a **Leaf** ...

Sei $X_i = \{v\}$.

$R(i, \{v\}) = \omega(v)$

Independent Set Using Nice Tree Decomp.

Let $G(i) :=$ graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i,S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.

If i is a **Leaf** ...

Sei $X_i = \{v\}$.

$R(i, \{v\}) = \omega(v)$

$R(i, \varnothing) =$

Independent Set Using Nice Tree Decomp.

Let $G(i)$:= graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i, S)$:= maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.

If i is a **Leaf** ...

Sei $X_i = \{v\}$.

$R(i, \{v\}) = \omega(v)$

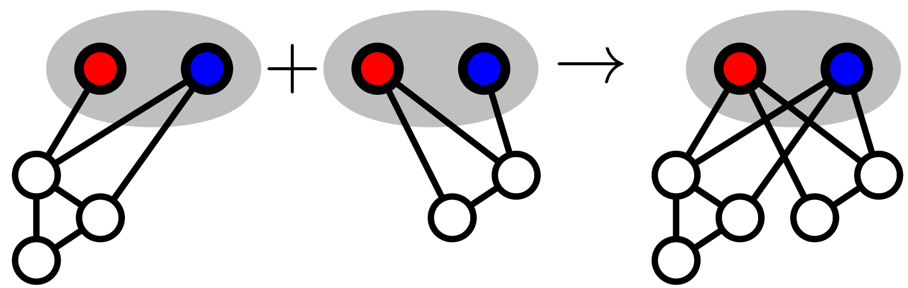
$R(i, \varnothing) = 0$

Independent Set Using Nice Tree Decomp.

Let $G(i) :=$ graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

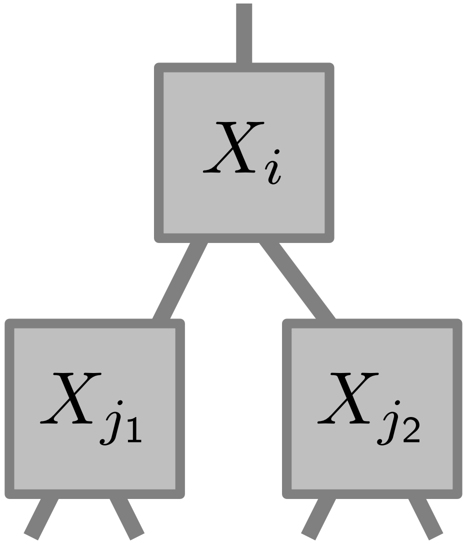
$R(i, S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.



If i is a **Join** ...

with children j_1 and j_2

$R(i, S) =$

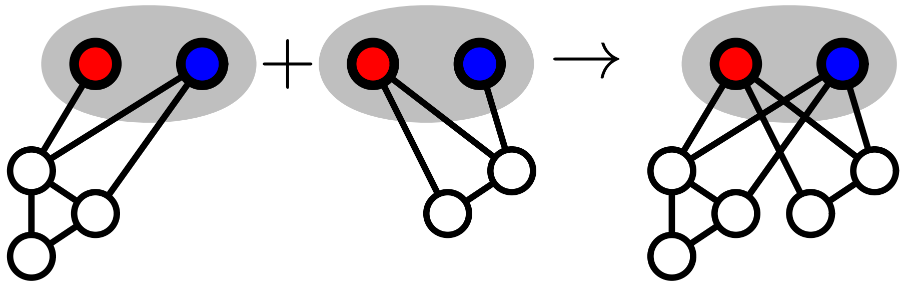


Independent Set Using Nice Tree Decomp.

Let $G(i)$:= graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

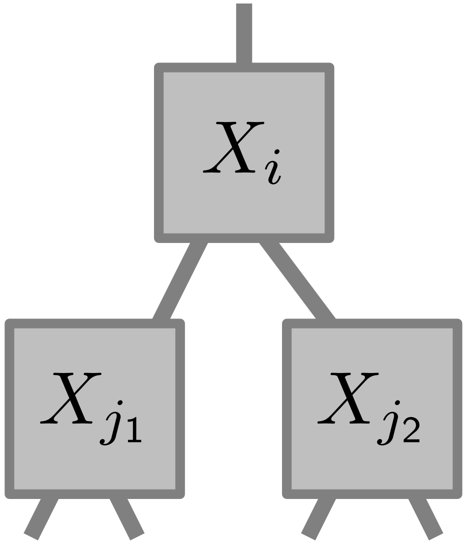
$R(i, S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.



If i is a **Join** ...

with children j_1 and j_2

$$R(i, S) = R(j_1, S) + R(j_2, S)$$

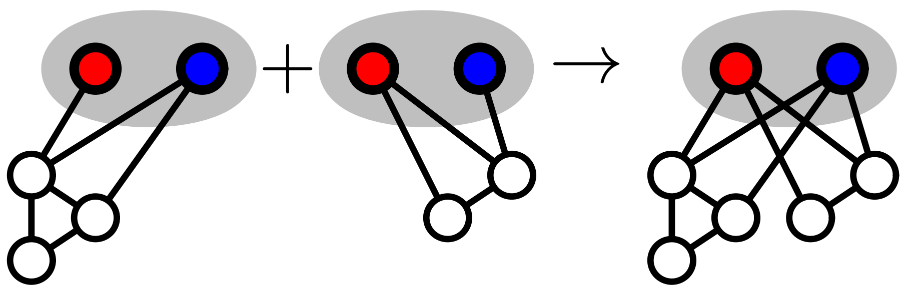


Independent Set Using Nice Tree Decomp.

Let $G(i)$:= graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

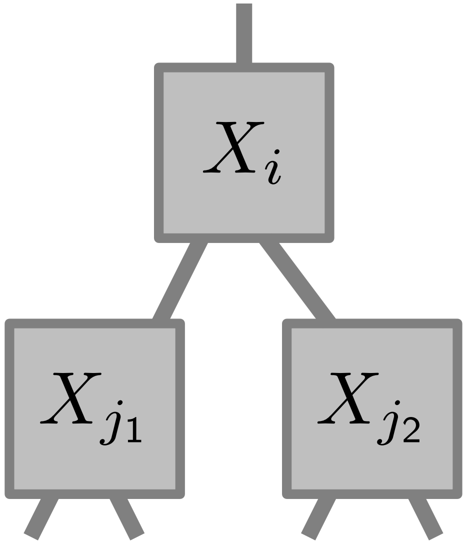
$R(i, S)$:= maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.



If i is a **Join** ...

with children j_1 and j_2

$$R(i, S) = R(j_1, S) + R(j_2, S) - \sum_{v \in S} \omega(v)$$



Independent Set Using Nice Tree Decomp.

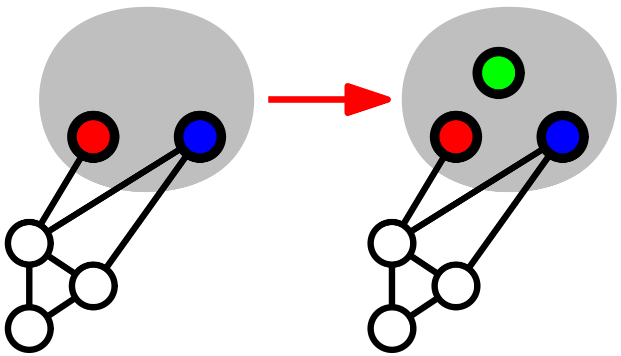
Let $G(i) :=$ graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i, S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.

If i is an **Introduce** ...

with child j and $X_i = X_j \cup \{v\}$

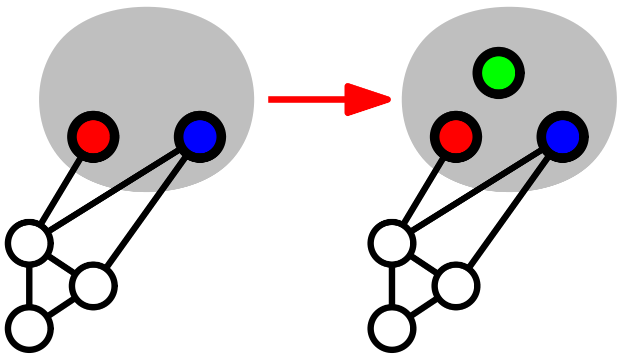


Independent Set Using Nice Tree Decomp.

Let $G(i)$:= graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i, S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.



If i is an **Introduce** ...

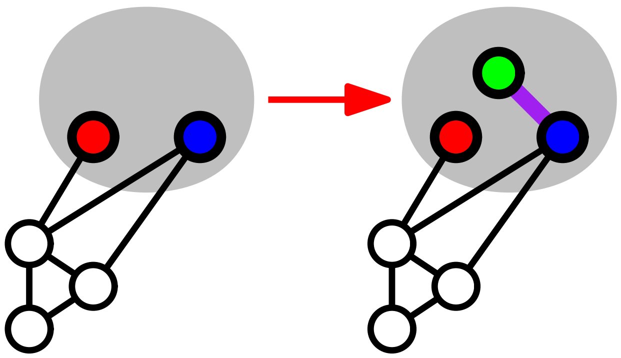
with child j and $X_i = X_j \cup \{v\}$

For each $S \subseteq X_j$: $R(i, S) = R(j, S)$.

Independent Set Using Nice Tree Decomp.

Let $G(i)$:= graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:
$R(i, S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.



If i is an **Introduce** ...

with child j and $X_i = X_j \cup \{v\}$

For each $S \subseteq X_j$: $R(i, S) = R(j, S)$.

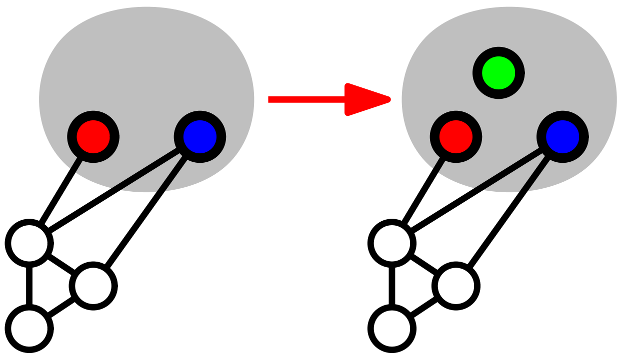
Otherwise if v has neighbors in S, $R(i, S) = -\infty$

Independent Set Using Nice Tree Decomp.

Let $G(i)$:= graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i, S)$:= maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.



If i is an **Introduce** ...

with child j and $X_i = X_j \cup \{v\}$

For each $S \subseteq X_j$: $R(i, S) = R(j, S)$.

Otherwise if v has neighbors in S, $R(i, S) = -\infty$

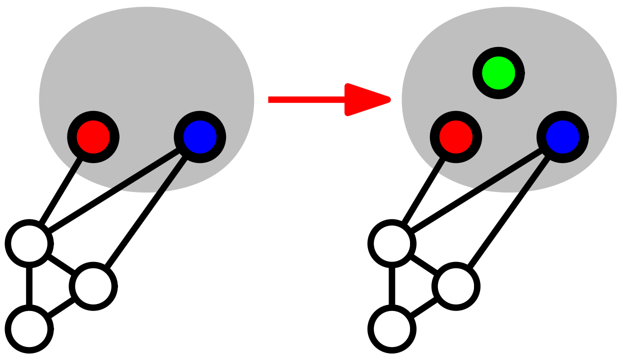
if v has no neighbors in S,

Independent Set Using Nice Tree Decomp.

Let $G(i)$:= graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i, S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.



If i is an **Introduce** ...

with child j and $X_i = X_j \cup \{v\}$

For each $S \subseteq X_j$: $R(i, S) = R(j, S)$.

Otherwise if v has neighbors in S, $R(i, S) = -\infty$

if v has no neighbors in S,

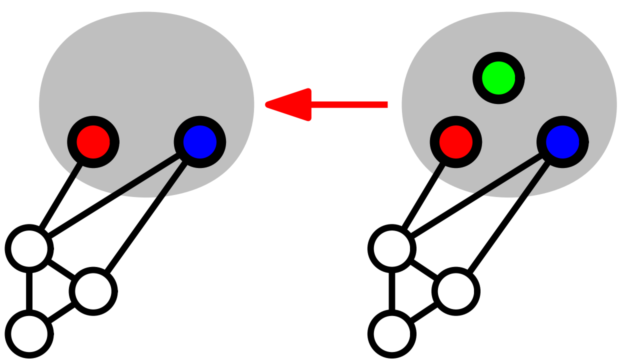
$$R(i, S) = R(j, S \setminus \{v\}) + \omega(v)$$

Independent Set Using Nice Tree Decomp.

Let $G(i)$:= graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i,S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.



If i is a **Forget** ...

with child j and $X_i = X_j \setminus \{v\}$

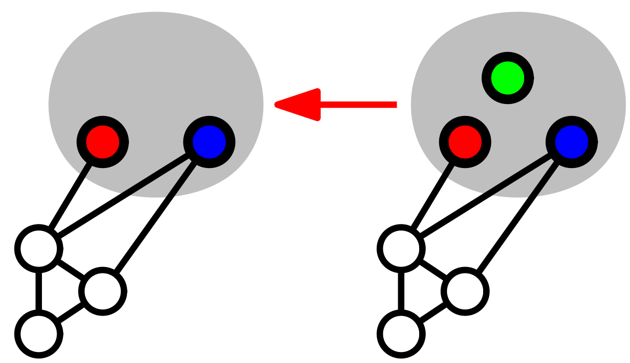
$R(i,S) =$

Independent Set Using Nice Tree Decomp.

Let $G(i)$:= graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i,S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.



If i is a **Forget** ...

with child j and $X_i = X_j \setminus \{v\}$

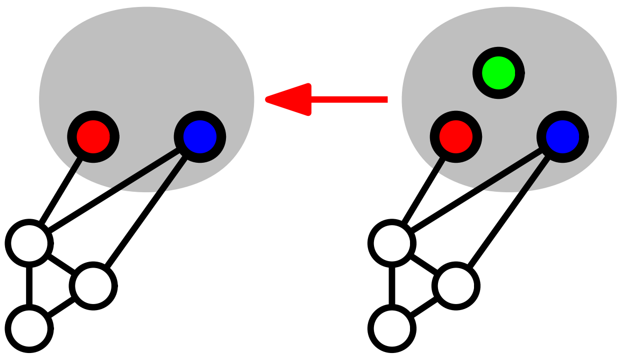
$R(i,S) =$ max{

Independent Set Using Nice Tree Decomp.

Let $G(i)$:= graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i, S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.



If i is a **Forget** ...

with child j and $X_i = X_j \setminus \{v\}$

$R(i, S) =$ max$\{$ $R(j, S)$

Independent Set Using Nice Tree Decomp.

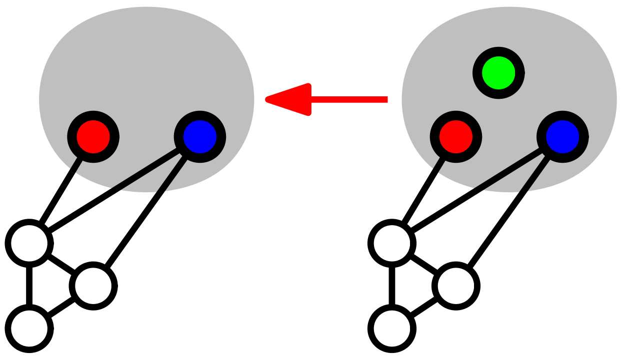
Let $G(i) :=$ graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i, S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.

If i is a **Forget** ...

with child j and $X_i = X_j \setminus \{v\}$



$$R(i, S) = \max\{\ R(j, S), R(j, S \cup \{v\})\ \}$$

Independent Set Using Nice Tree Decomp.

Let $G(i) :=$ graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i, S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.

Algorithm: Compute $R(i, S)$ for all i and corresponding S.

Independent Set Using Nice Tree Decomp.

Let $G(i) :=$ graph induced by the vertices in the subtree at i.

For bag i and $S \subseteq X_i$, let:

$R(i, S) :=$ maximum weight of an independent set I in $G(i)$ with $I \cap X_i = S$.

Algorithm: Compute $R(i, S)$ for all i and corresponding S.

Runtime: **?**