Julius-Maximilians-
UNIVERSITÄT
WÜRZBURG

Lehrstuhl für
INFORMATIK I
Effiziente Algorithmen und
wissensbasierte Systeme

Institut für Informatik

Advanced Algorithms

Winter term 2019/20

Lecture 11. Alternative Parameterization: Tree Decomposition
Source: **PA §7.2, 7.3.1**

(slides by Thomas van Dijk & Alexander Wolff)

Steven Chaplick *Lehrstuhl für Informatik I*

Independent Set

Independent Set

Given: graph G, weight function $\omega : V \to \mathbb{N}$

Question: What is the maximum weight of a set $S \subseteq V$ where no pair in S forms an edge in G?

Independent Set

Independent Set

Given: graph G, weight function $\omega : V \to \mathbb{N}$

Question: What is the maximum weight of a set $S \subseteq V$ where no pair in S forms an edge in G?

Thm: Independent Set is NP-complete.

Independent Set

Independent Set

Given: graph G, weight function $\omega : V \to \mathbb{N}$

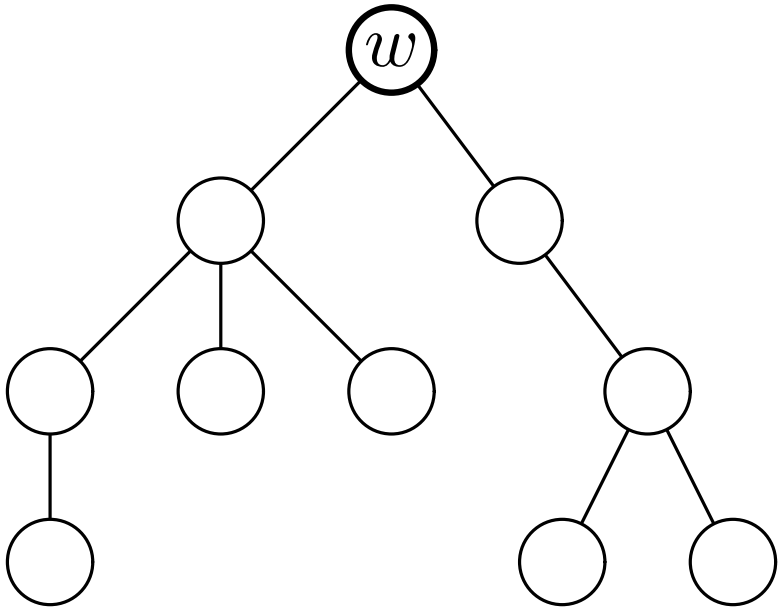
Question: What is the maximum weight of a set $S \subseteq V$ where no pair in S forms an edge in G?

Thm: Independent Set is NP-complete.

Thm: Independent Set can be solved in linear time on trees.

Independent Sets in Trees

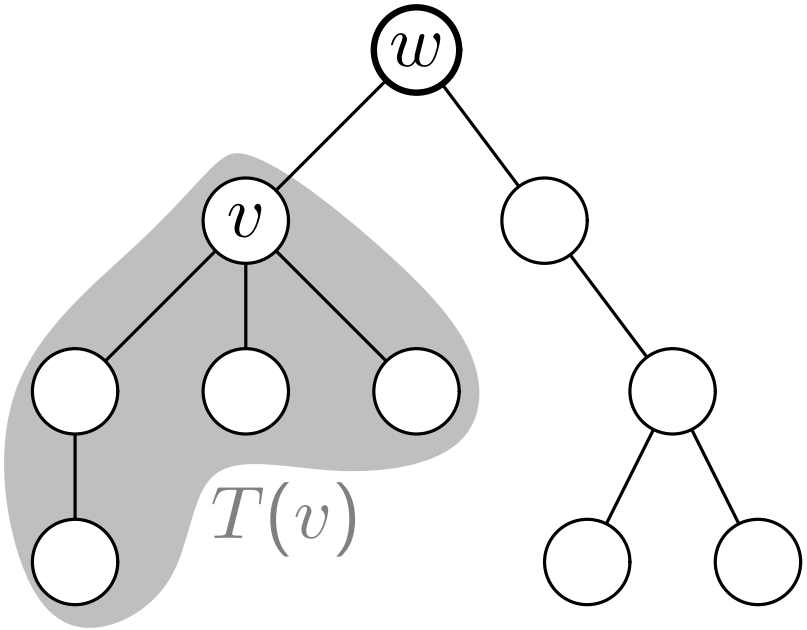
Choose an arbitrary root w.



Independent Sets in Trees

Choose an arbitrary root w.

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

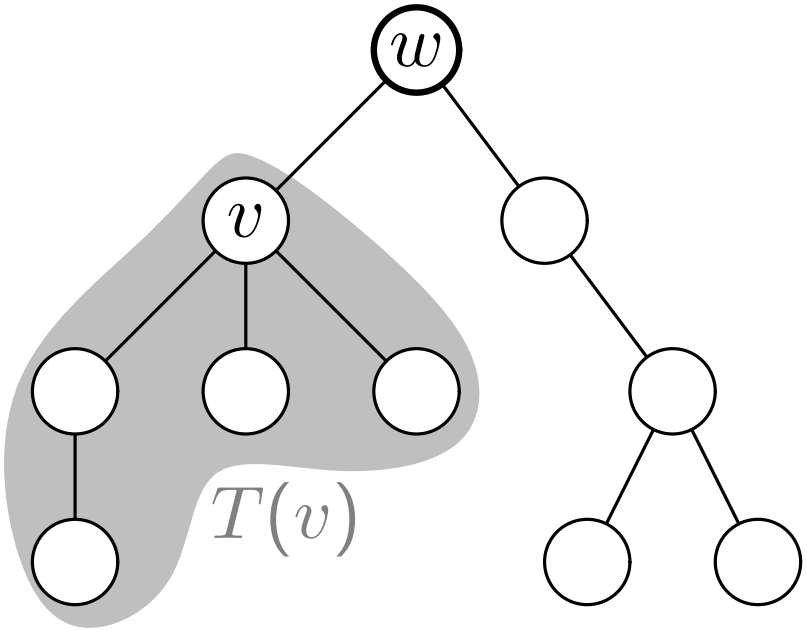


Independent Sets in Trees

Choose an arbitrary root w.

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

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



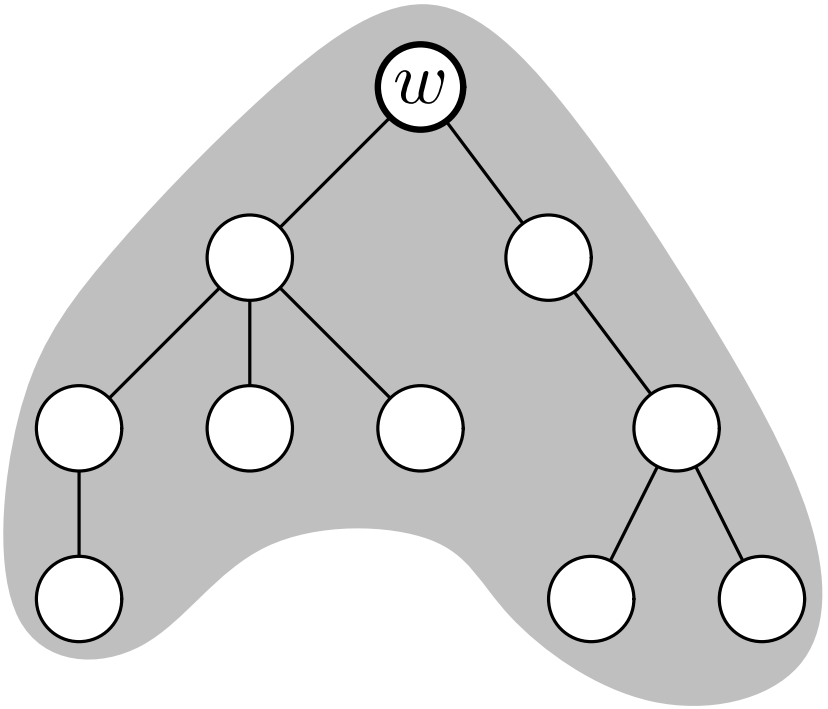
Independent Sets in Trees

$A(w) =$ solution

Choose an arbitrary root w.

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

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



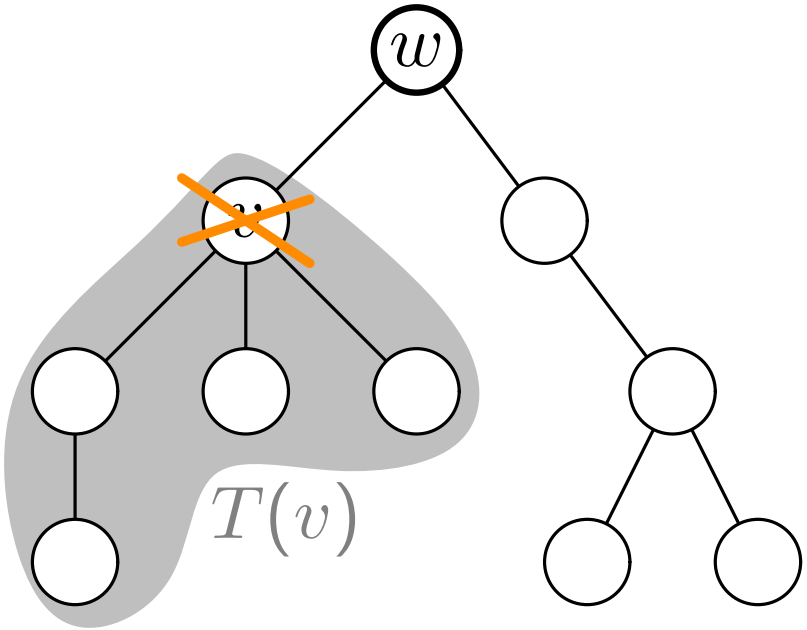
Independent Sets in Trees

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

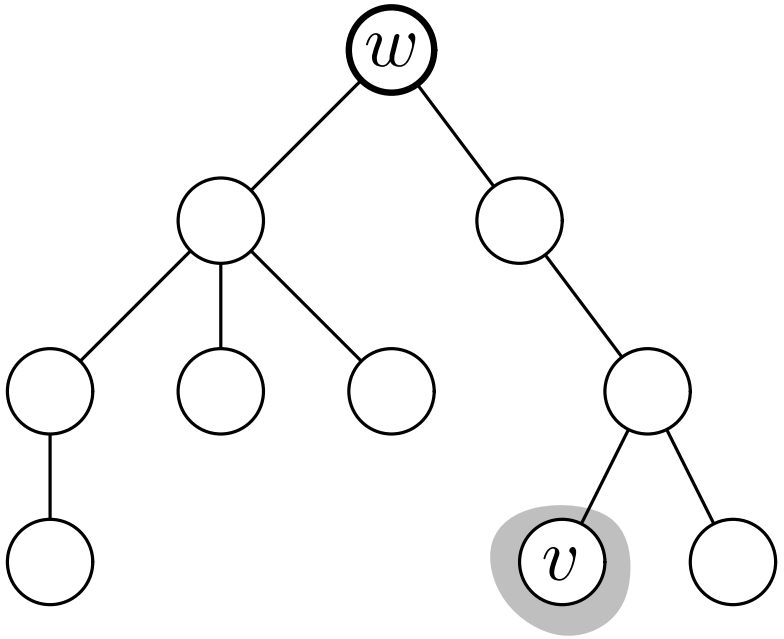
Choose an arbitrary root w.

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* $v \notin S$

When v is a leaf: $A(v) =$



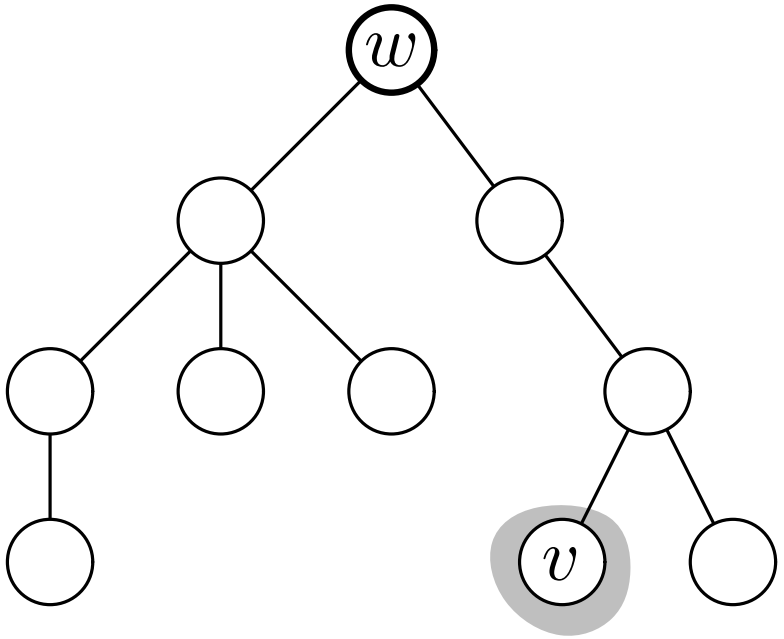
Independent Sets in Trees

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* $v \notin S$



When v is a leaf: $A(v) = \ \omega(v)$ and $B(v) =$

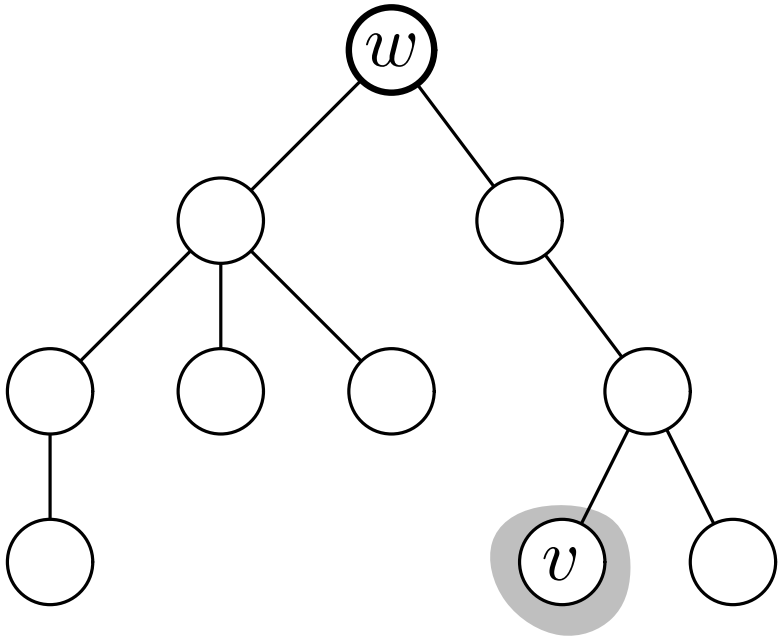
Independent Sets in Trees

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* $v \notin S$



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

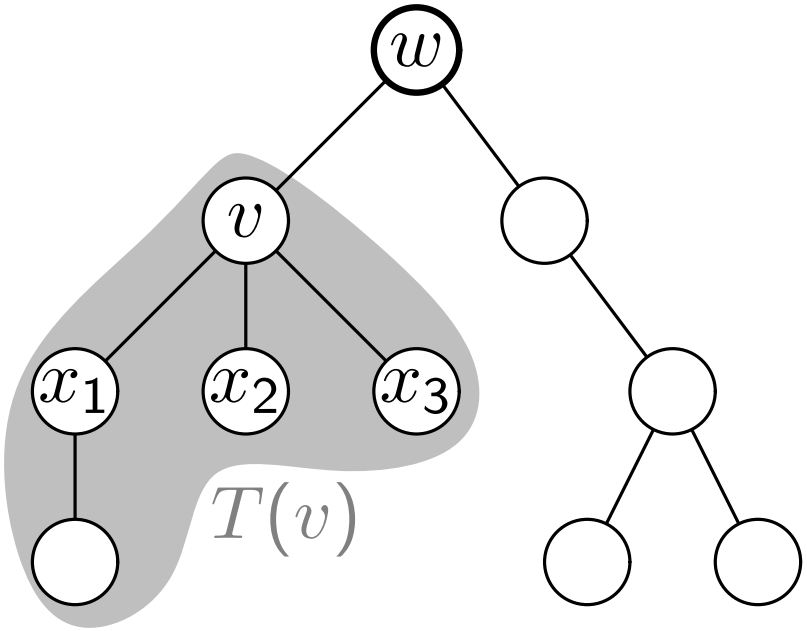
Independent Sets in Trees

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* $v \notin S$



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

When v has children $x_1, \ldots, x_r$:

$B(v) =$

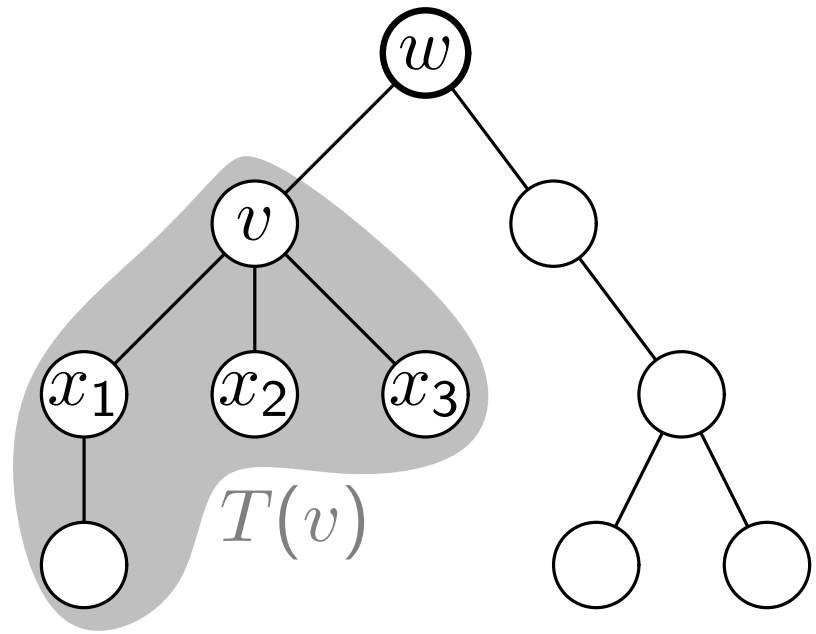
Independent Sets in Trees

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* $v \notin S$



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

When v has children $x_1, \ldots, x_r$:

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

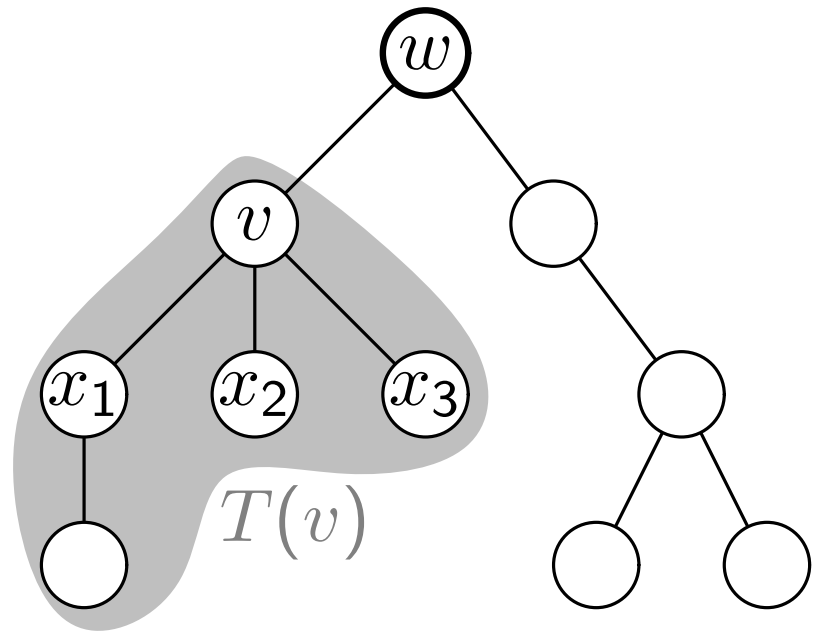
Independent Sets in Trees

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* $v \notin S$



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

When v has children $x_1, \ldots, x_r$:

$A(v) =$

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

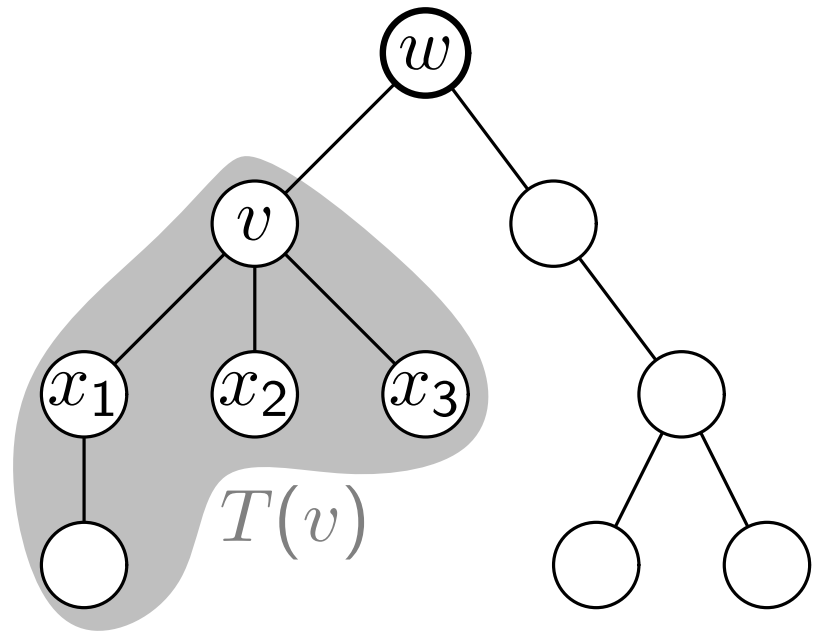
Independent Sets in Trees

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* $v \notin S$



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

When v has children $x_1, \ldots, x_r$:

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

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

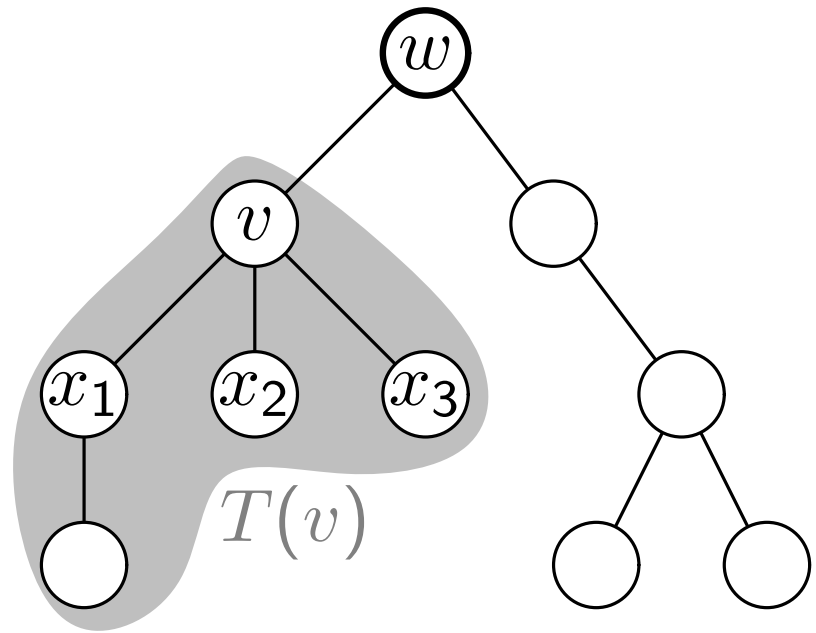
Independent Sets in Trees

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* $v \notin S$



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

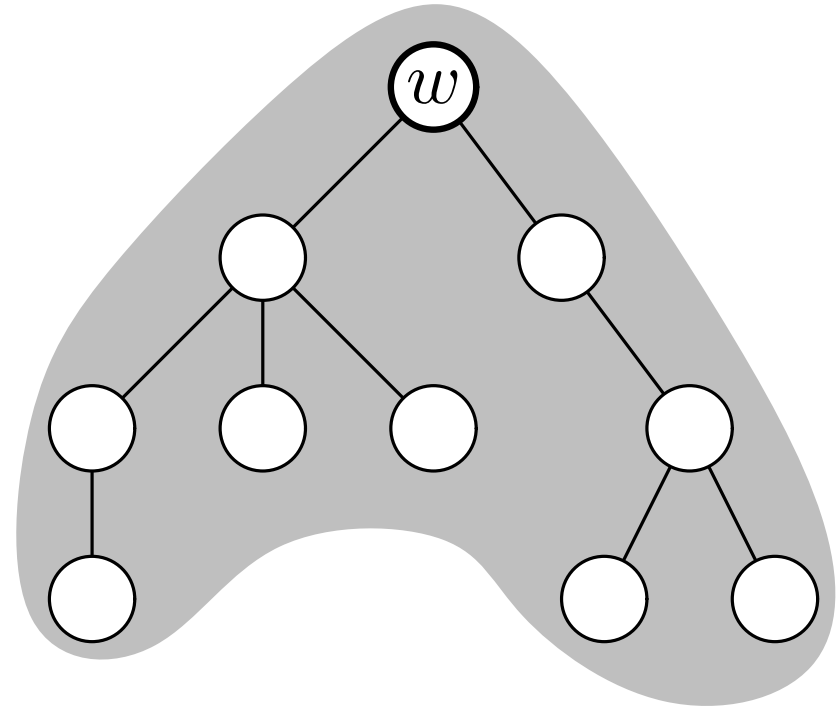
When v has children $x_1, \ldots, x_r$:

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

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

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* $v \notin S$

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

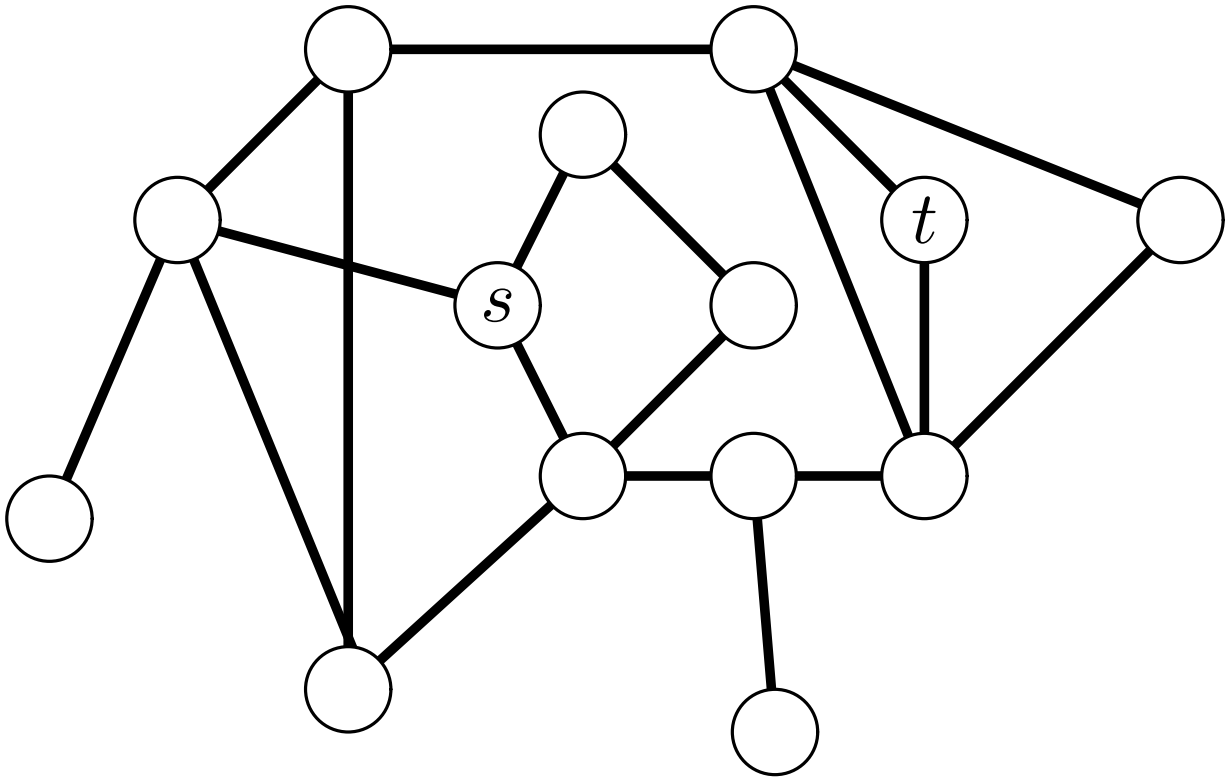
When v has children $x_1, \ldots, x_r$:

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

$B(v) = \sum_{i=1}^{r} A(x_i)$ **Algo:** Compute $A(\cdot)$ and $B(\cdot)$ bottom-up

s, t-series parallel graphs

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



s, t-series parallel graphs

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

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

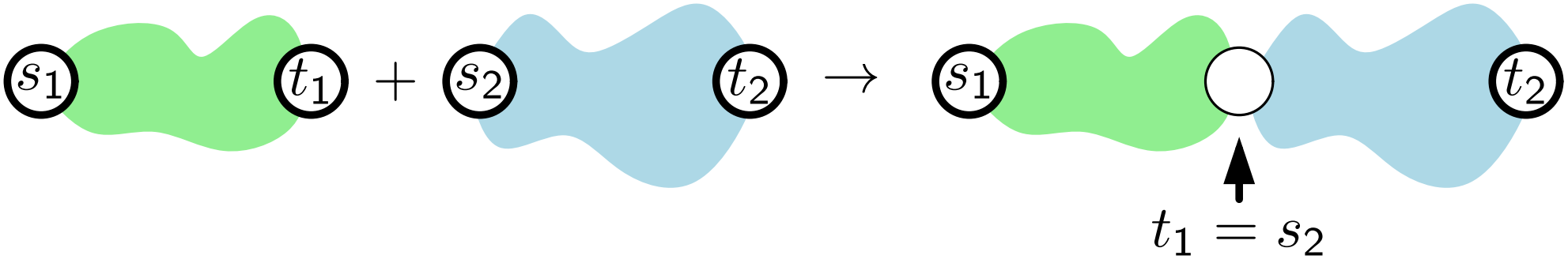
- G is a single edge (s, t)

s, t-series parallel graphs

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

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

- G is a single edge (s, t)
- G is a *series composition* of two series parallel graphs

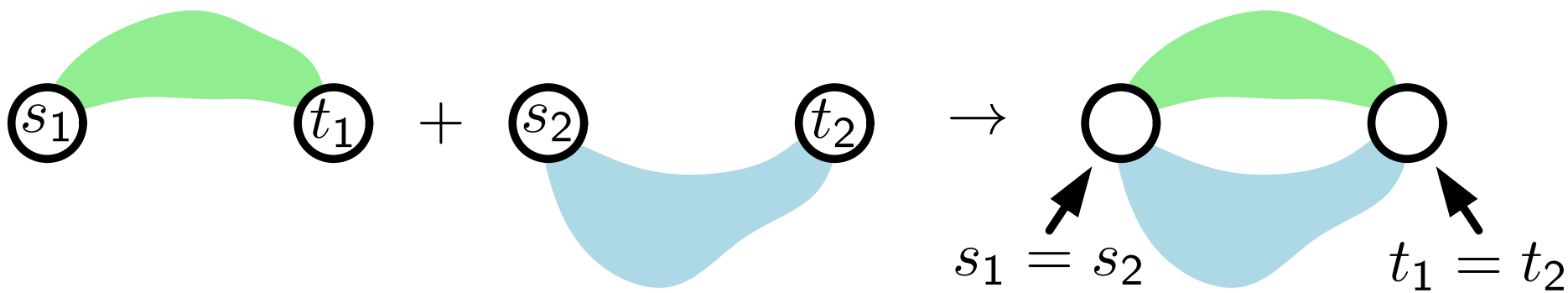


s, t-series parallel graphs

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

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

- 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* when it contains two special vertices s and t

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

- 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

recursive definition:
series parallel graphs have a natural tree-structure

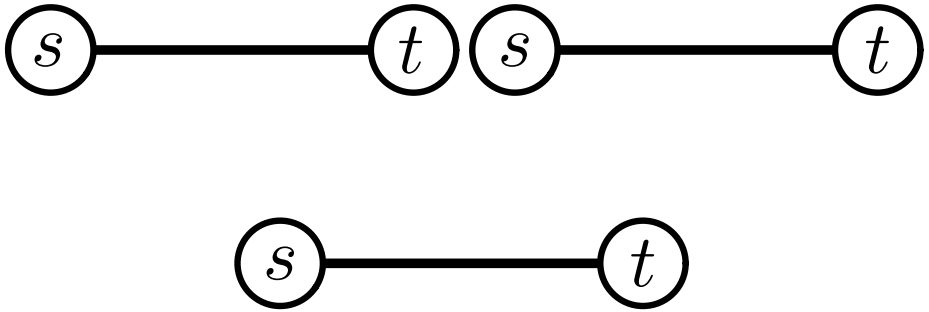
SP-tree

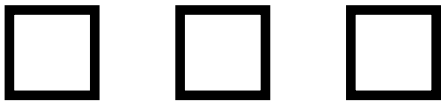
$\square$

SP-tree

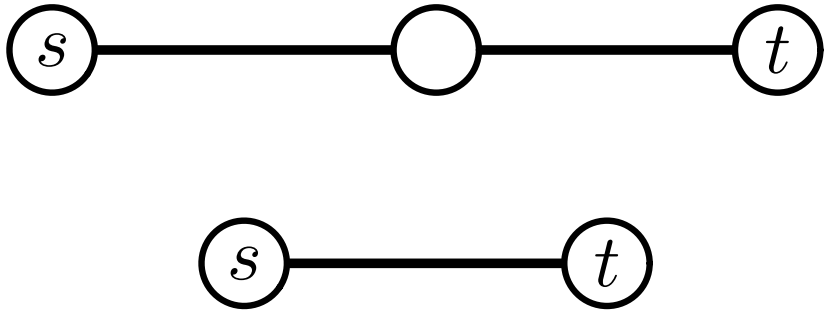
$\square \quad \square$

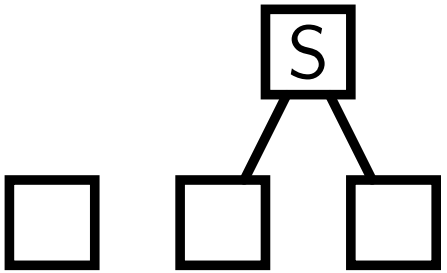
SP-tree



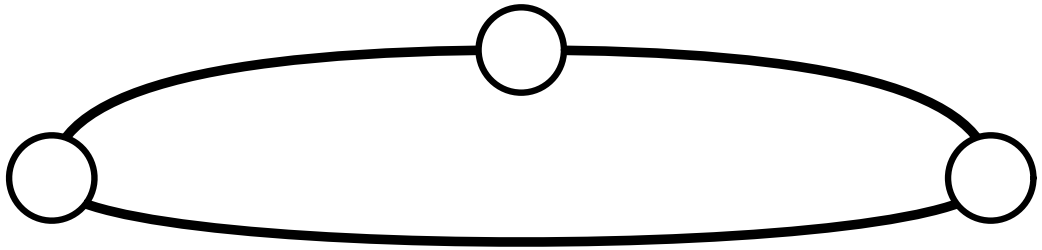


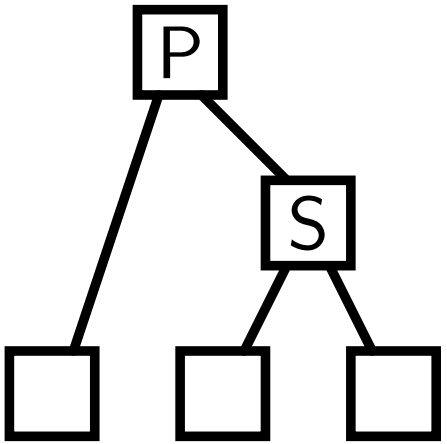
SP-tree



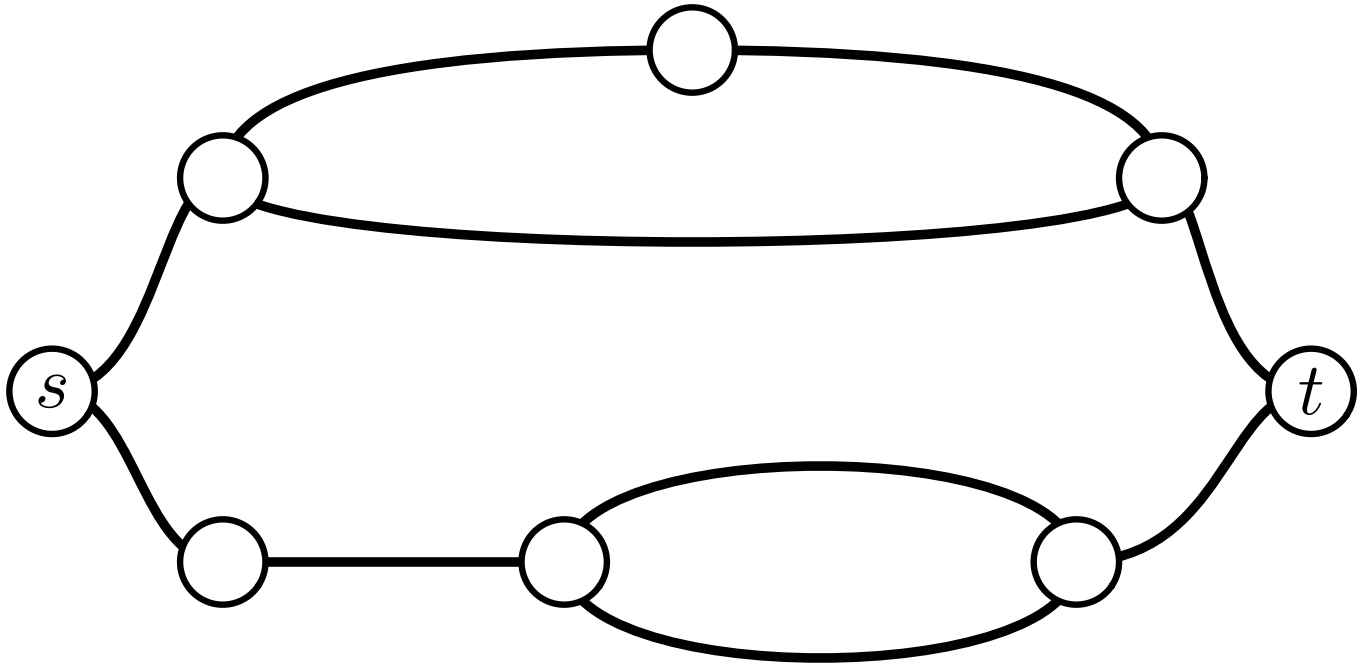


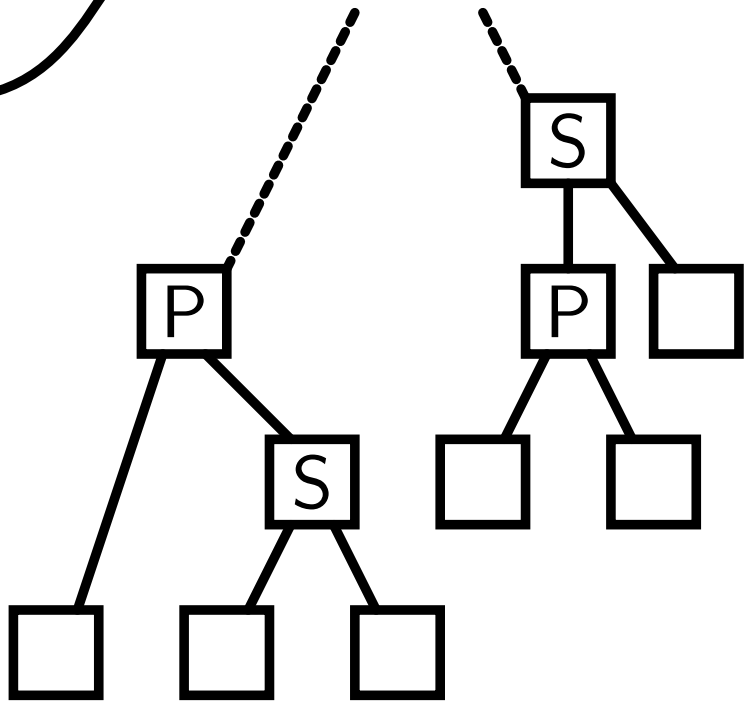
SP-tree



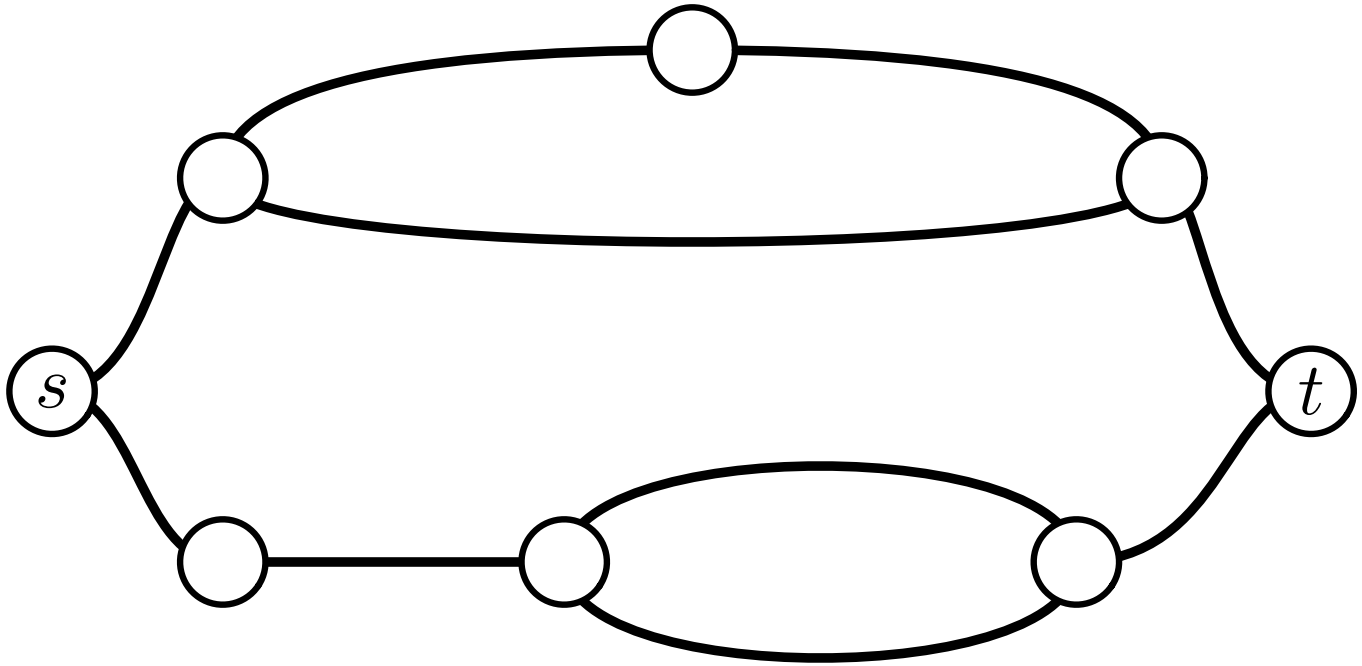


SP-tree

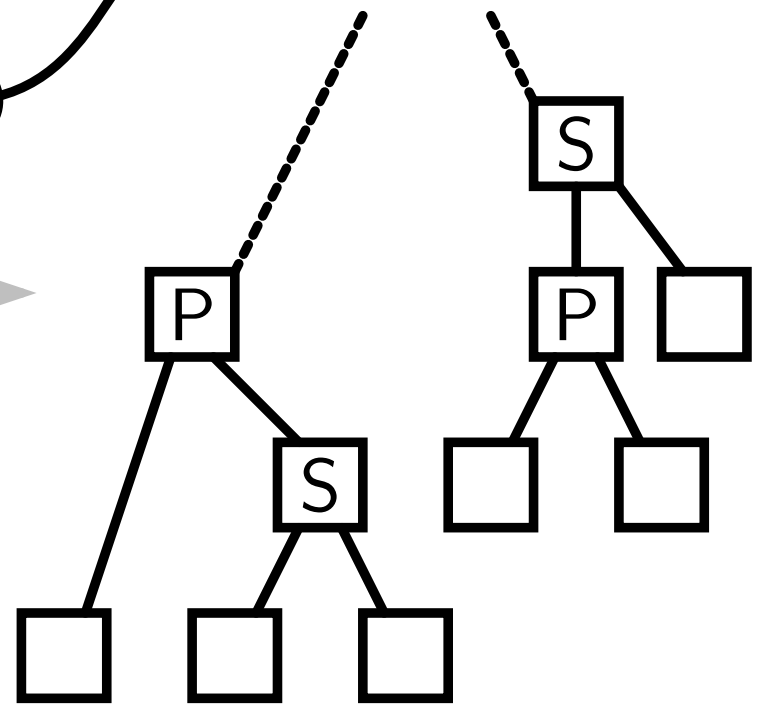




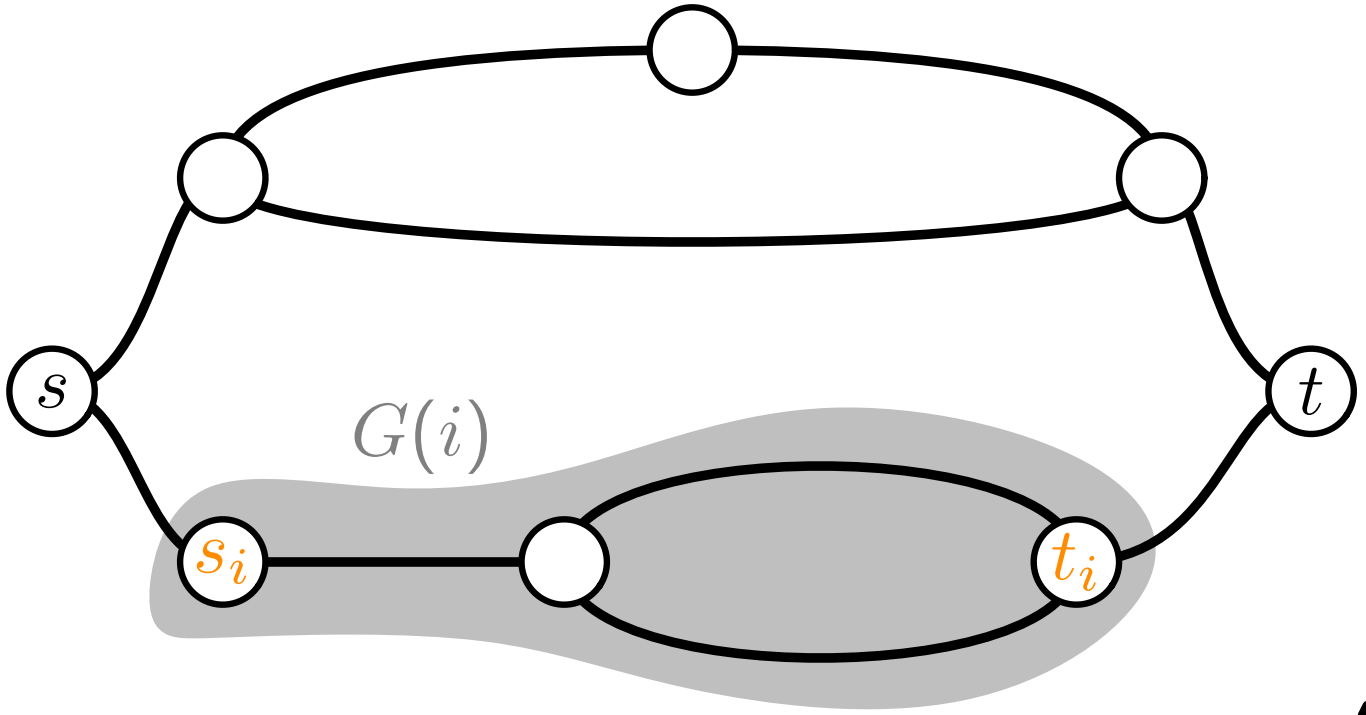
SP-tree



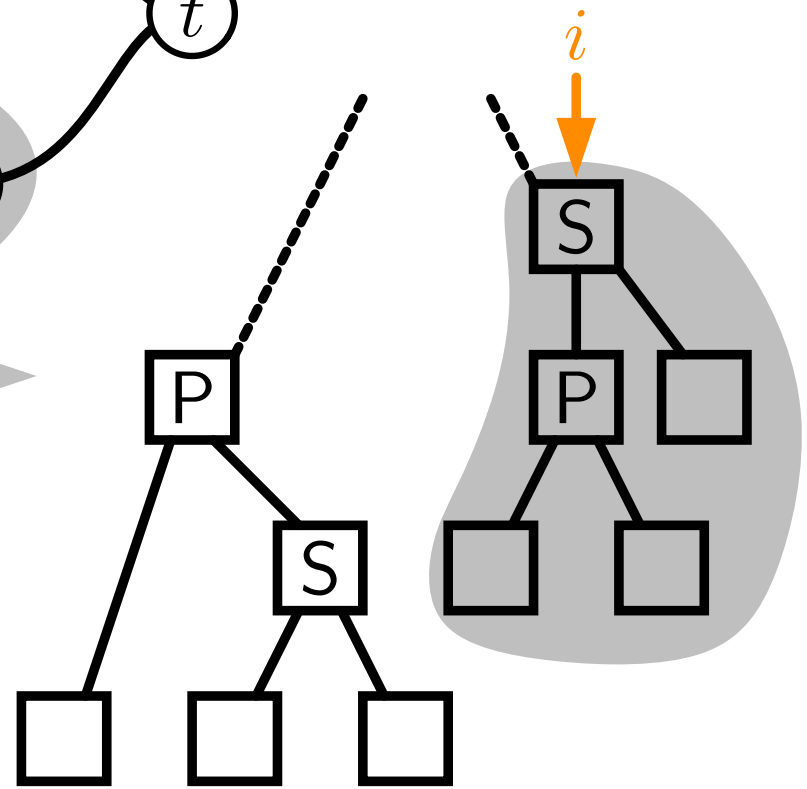
Let i be a node in an SP-tree.
$G(i) :=$ graph represented by the subtree rooted at i



SP-tree

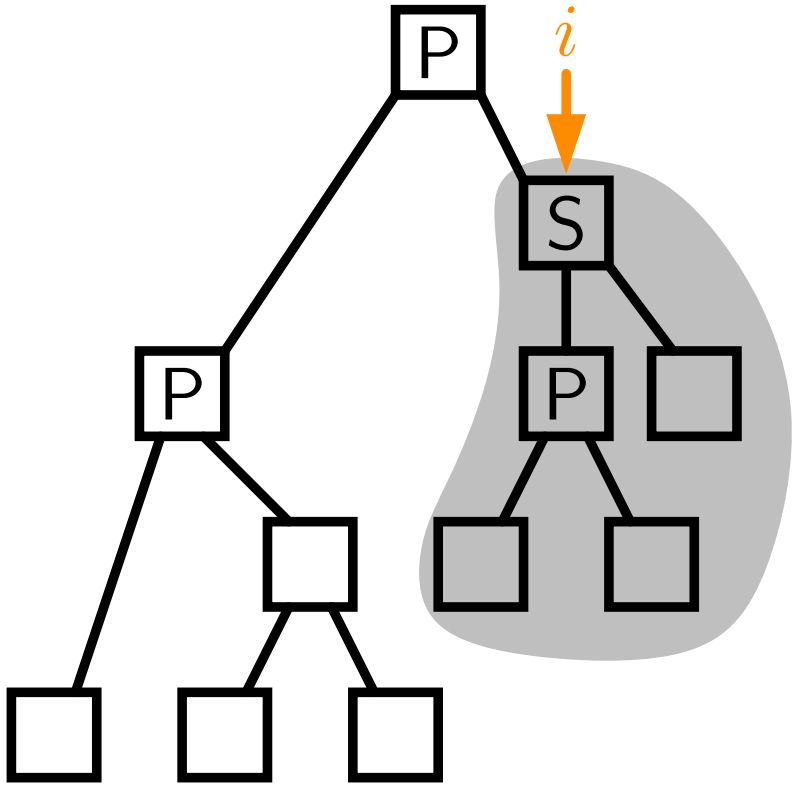


Let i be a node in an SP-tree.
$G(i) :=$ graph represented by the subtree rooted at i



Independent Set on SP-trees

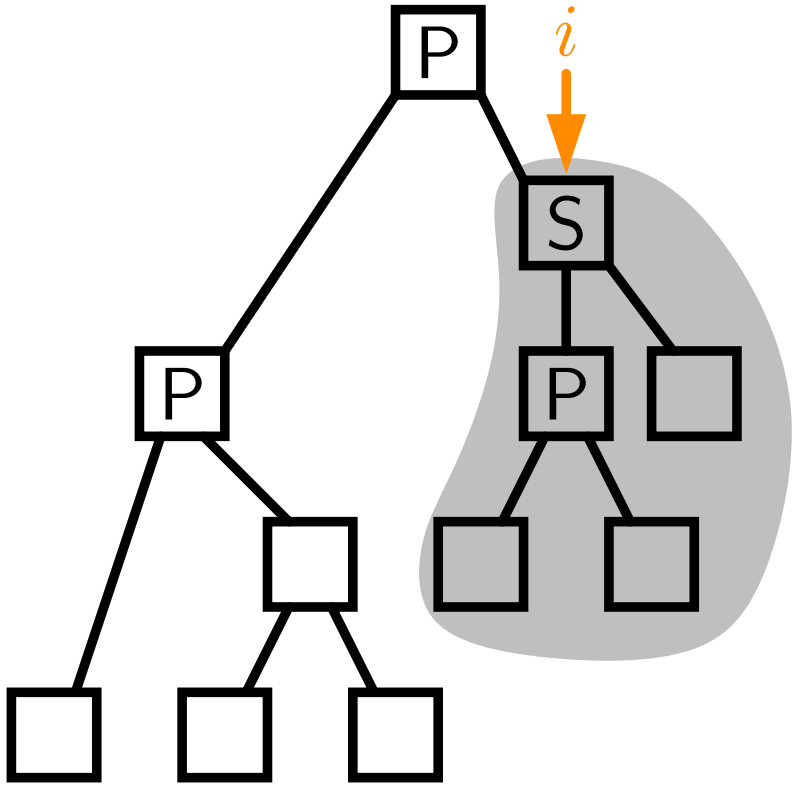
Dynamic program on SP-tree indexed by $G(i)$



Independent Set on SP-trees

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

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

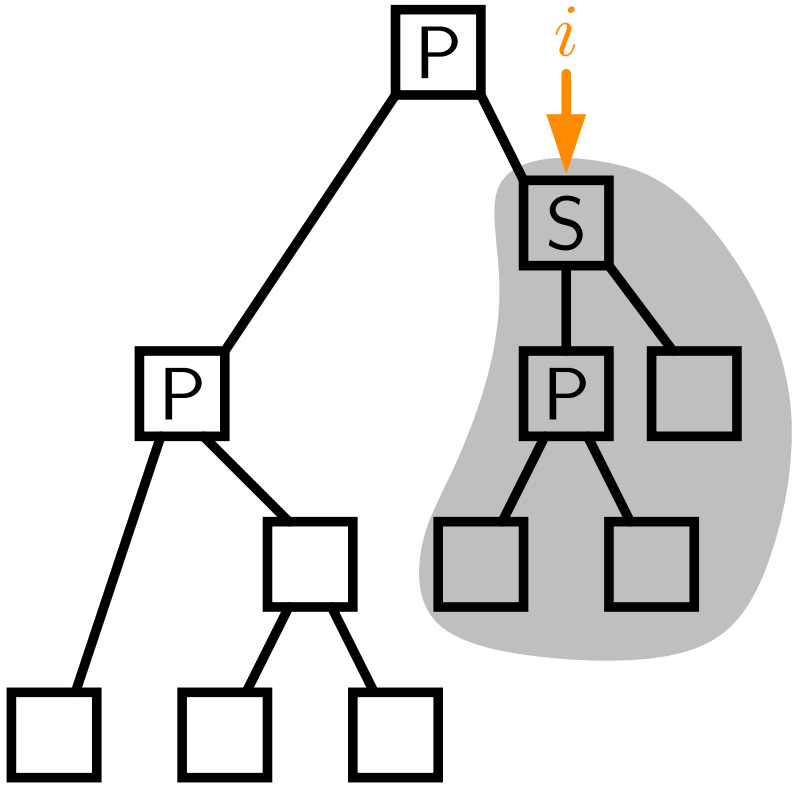


Independent Set on SP-trees

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

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

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



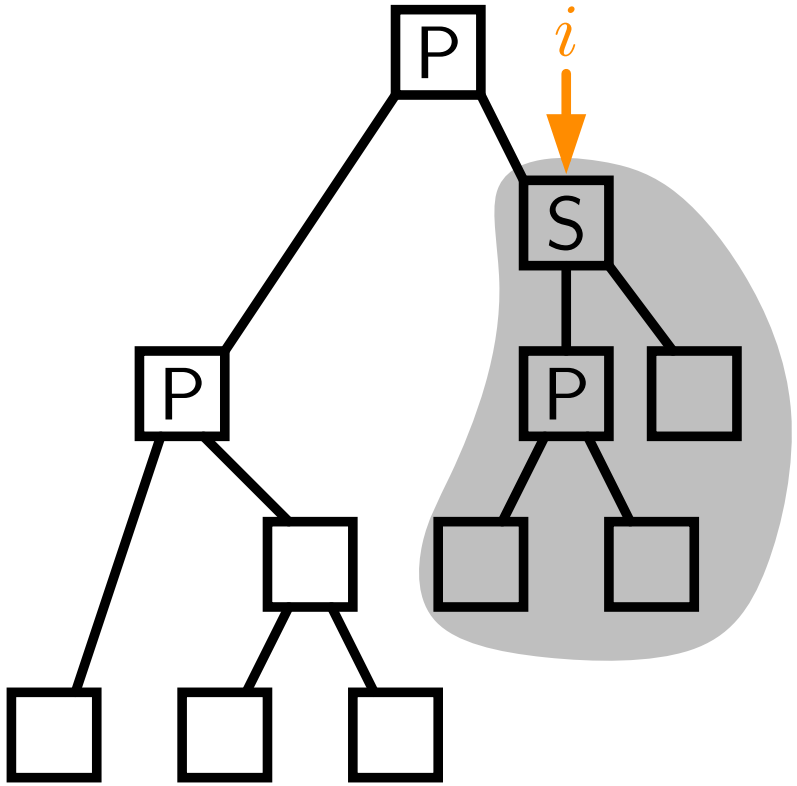
Independent Set on SP-trees

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

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

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

$AB(i)$ and $BB(i)$ def. similarly



Independent Set on SP-trees

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

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

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

$AB(i)$ and $BB(i)$ def. similarly

When i is a leaf...

$BB(i) =$

Independent Set on SP-trees

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

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

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

$AB(i)$ and $BB(i)$ def. similarly

When 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 independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

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

$AB(i)$ and $BB(i)$ def. similarly

When 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 independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

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

$AB(i)$ and $BB(i)$ def. similarly

When 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 independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

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

$AB(i)$ and $BB(i)$ def. similarly

When i is a leaf...

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

Independent Set on SP-trees

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

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

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

$AB(i)$ and $BB(i)$ def. similarly

When i is a leaf...

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

Independent Set on SP-trees

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

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

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

$AB(i)$ and $BB(i)$ def. similarly

When i is a leaf...

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

Independent Set on SP-trees

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

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

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

$AB(i)$ and $BB(i)$ def. similarly

When 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

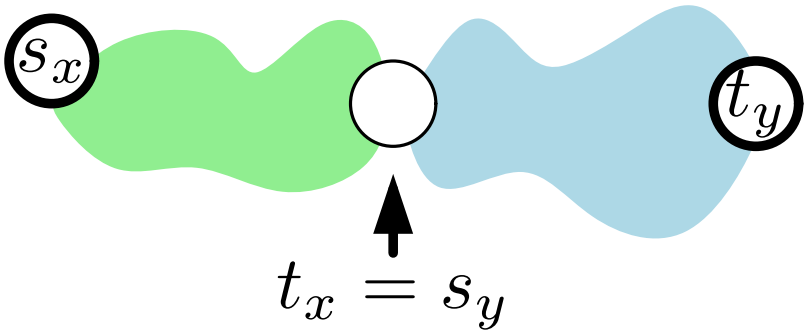
Dynamic program on SP-tree indexed by $G(i)$

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

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

$AB(i)$ and $BB(i)$ def. similarly

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



$AA(i) =$

Independent Set on SP-trees

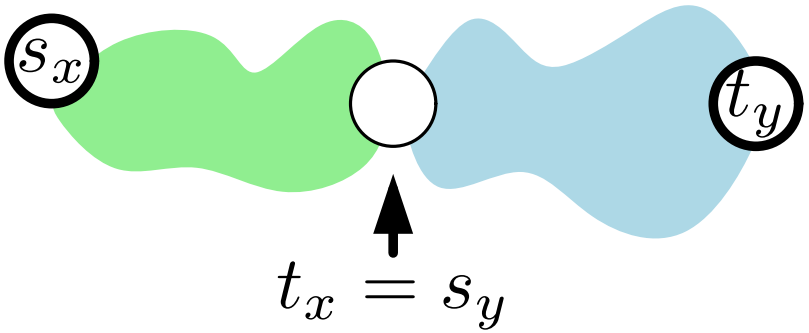
Dynamic program on SP-tree indexed by $G(i)$

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

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

$AB(i)$ and $BB(i)$ def. similarly

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



$AA(i) = \max\{$

Independent Set on SP-trees

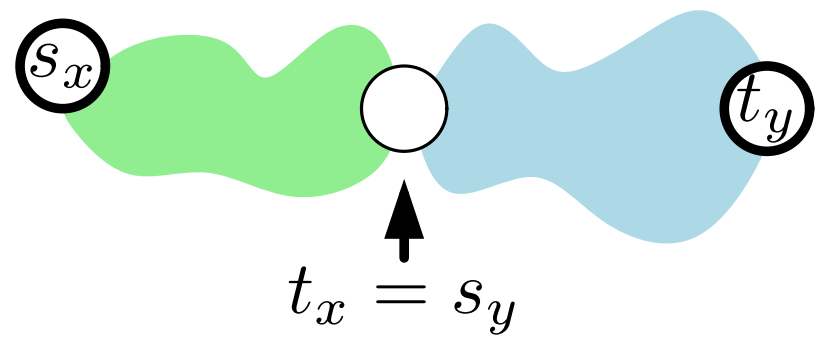
Dynamic program on SP-tree indexed by $G(i)$

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

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

$AB(i)$ and $BB(i)$ def. similarly

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



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

Independent Set on SP-trees

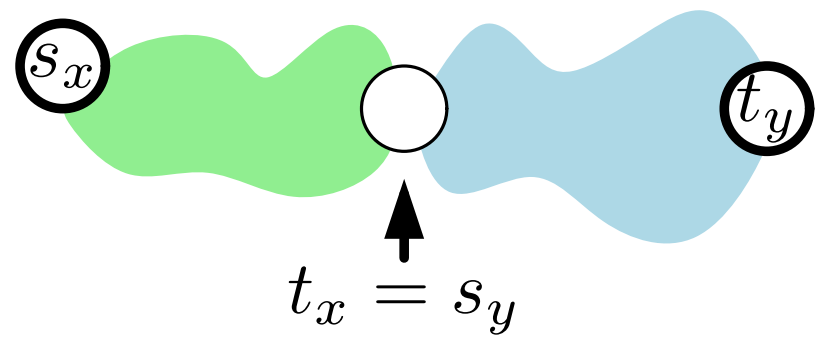
Dynamic program on SP-tree indexed by $G(i)$

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

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

$AB(i)$ and $BB(i)$ def. similarly

When 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 independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

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

$AB(i)$ and $BB(i)$ def. similarly

other cases omitted... (easy exercise)

Independent Set on SP-trees

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

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

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

$AB(i)$ and $BB(i)$ def. 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 independent set S in $G(i)$ where $s_i \in S$ and $t_i \in S$

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

$AB(i)$ and $BB(i)$ def. similarly

other cases omitted... (easy exercise)

$O(1)$ time per SP-node

Thm: INDEPENDENT SET on series parallel graphs with a given SP-tree can be solved in $O(n)$ time.

Generalization?

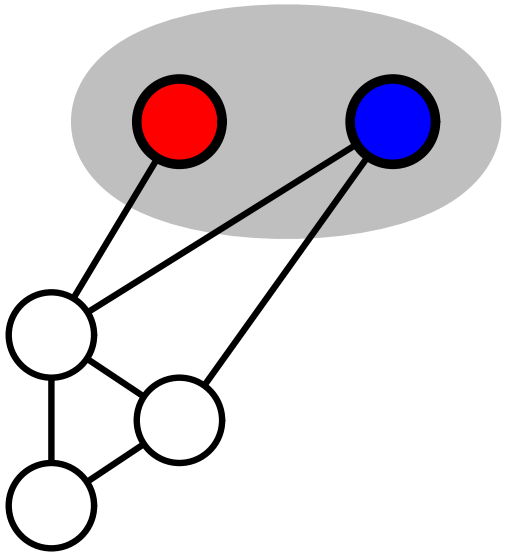
Many ways to generalize the concept of having a "tree structure"

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

Generalization?

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

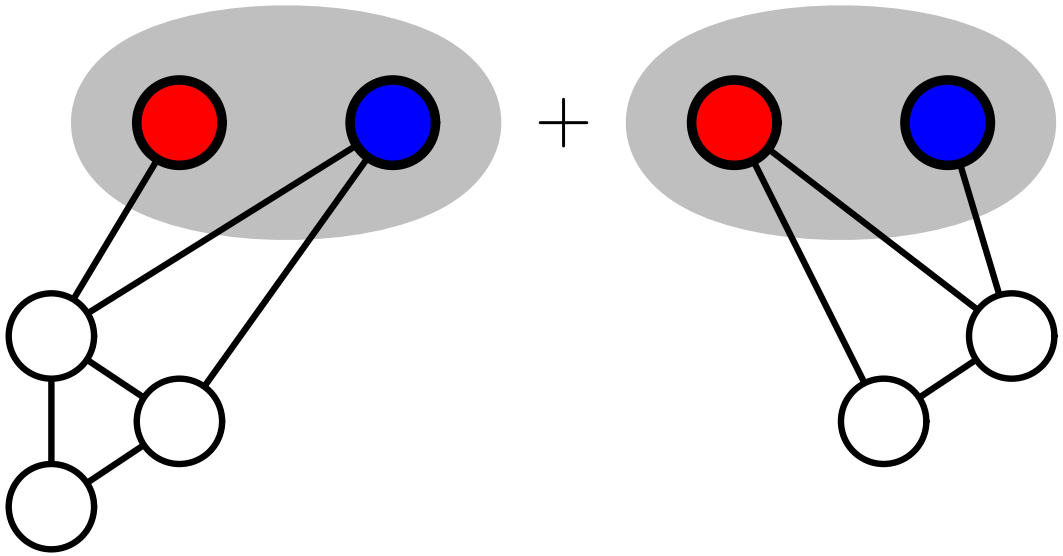
Ex.: k-terminal graph $G = (V, E, T)$, $|T| = k$



Generalization?

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

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

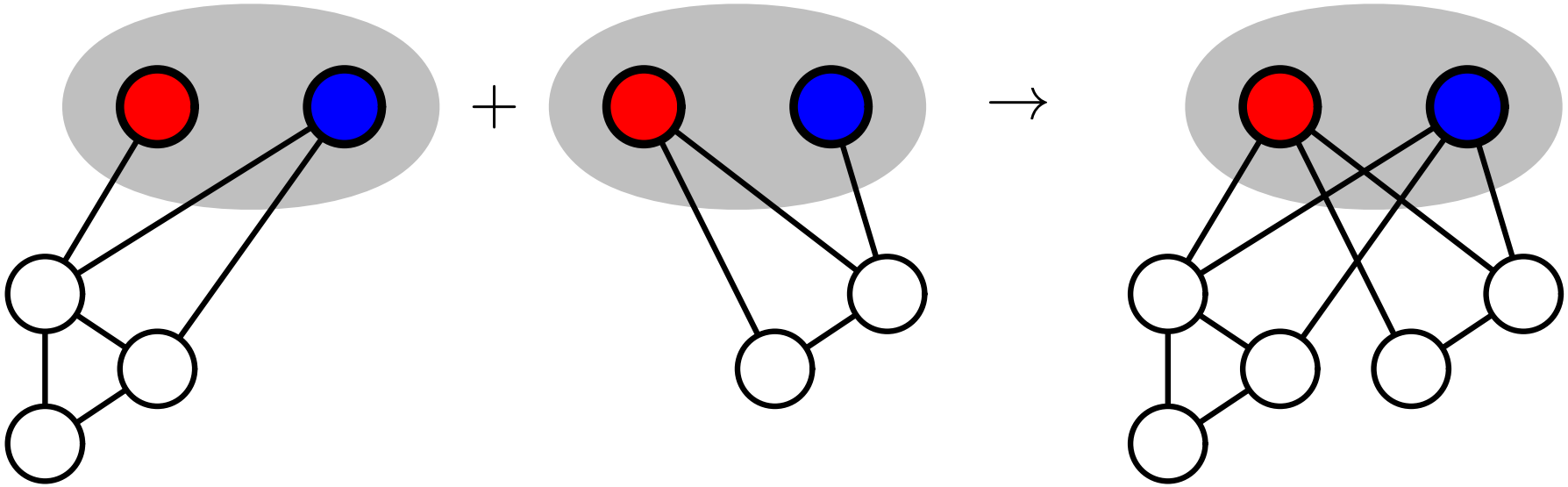


Generalization?

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

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

Example Operation: "gluing"

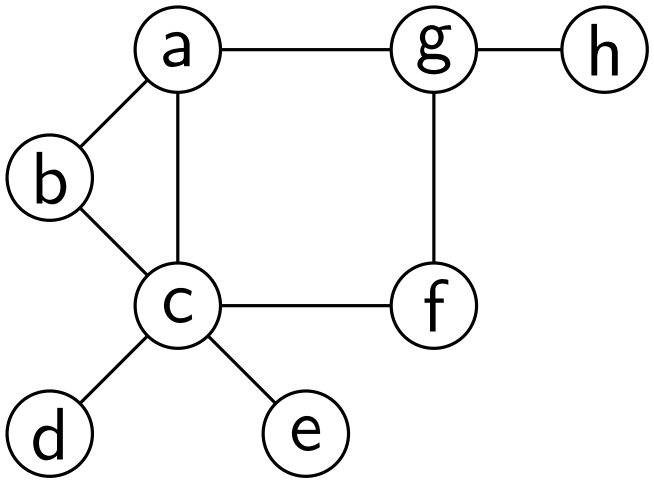


Example: Tree Decomposition

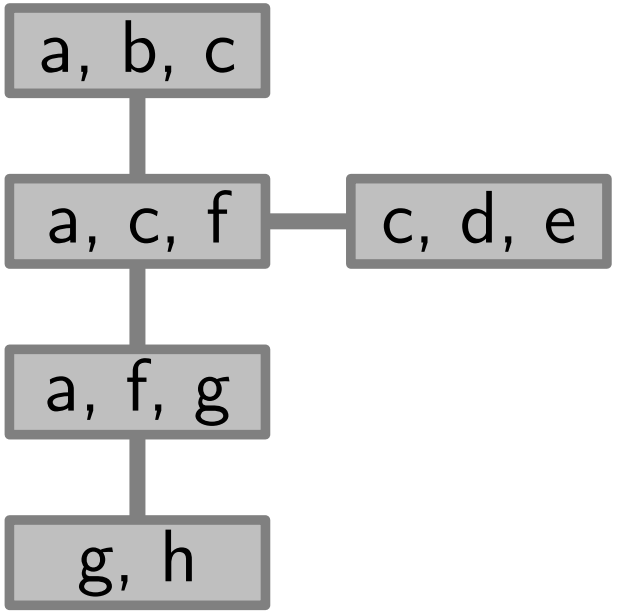
"bags"

Tree Decomposition is a tree where the nodes map to subsets of V so that...

Graph $G = (V, E)$:



Tree Decomposition:

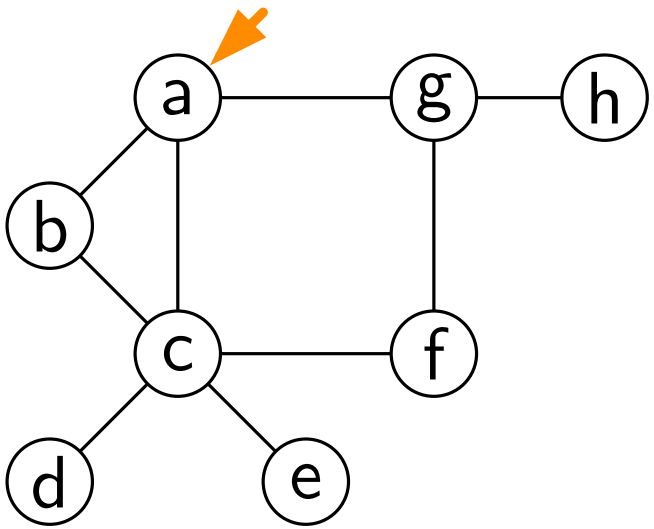


Example: Tree Decomposition

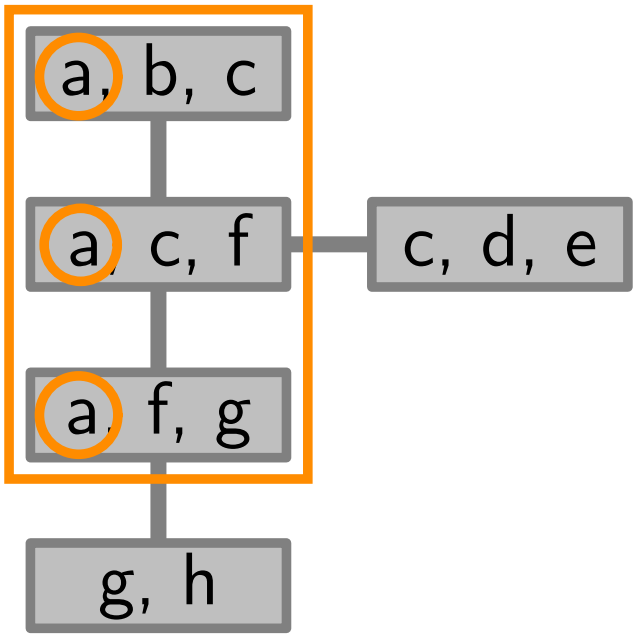
1 each vertex belongs to at least one bag

these bags are connected

Graph $G = (V, E)$:



Tree Decomposition:

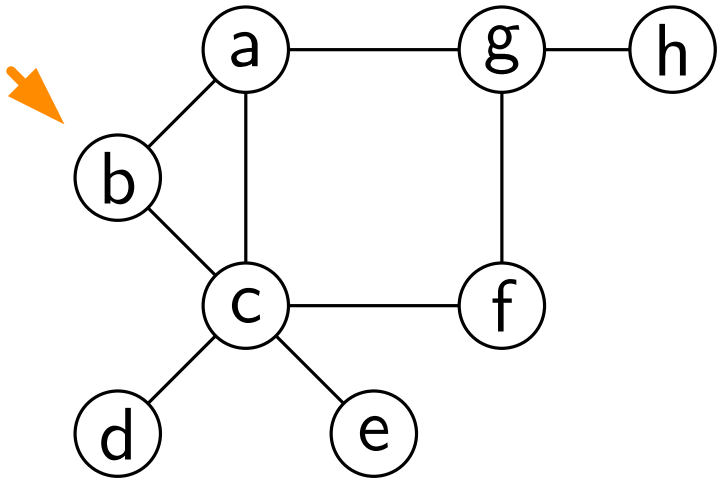


Example: Tree Decomposition

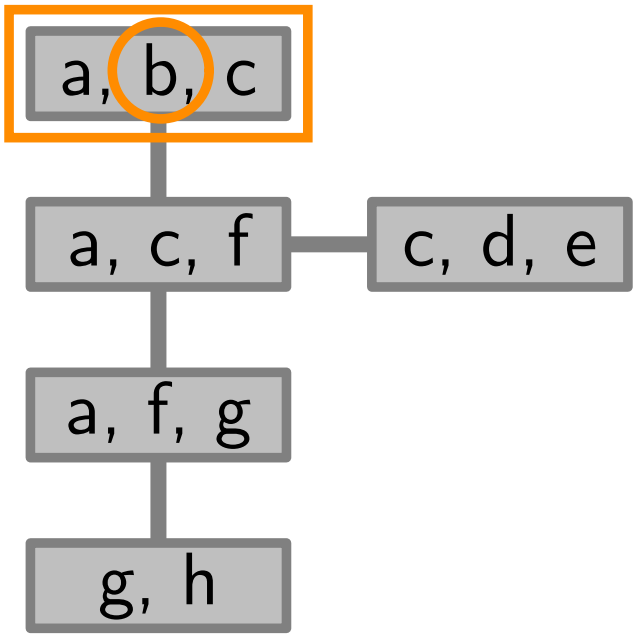
1 each vertex belongs to at least one bag

these bags are connected

Graph $G = (V, E)$:



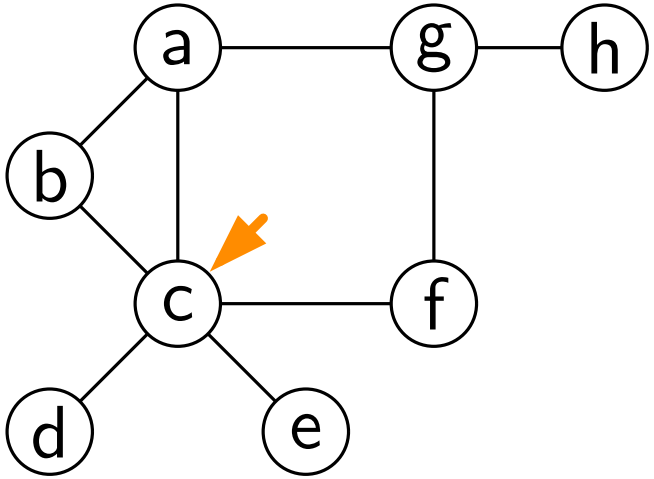
Tree Decomposition:



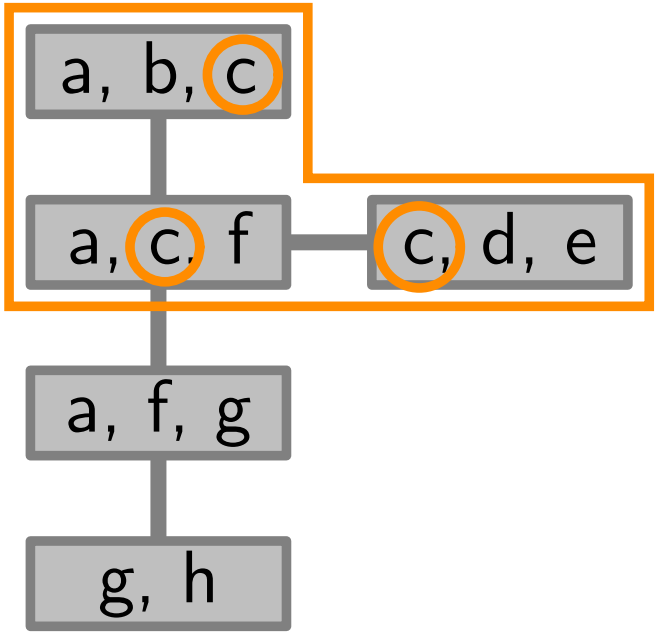
Example: Tree Decomposition

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

Graph $G = (V, E)$:



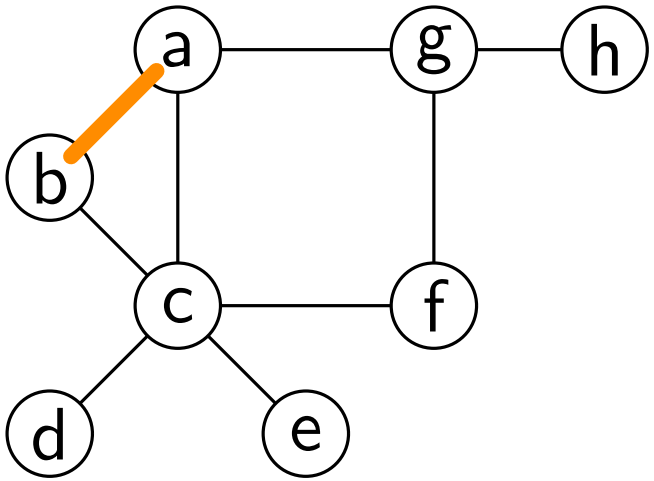
Tree Decomposition:



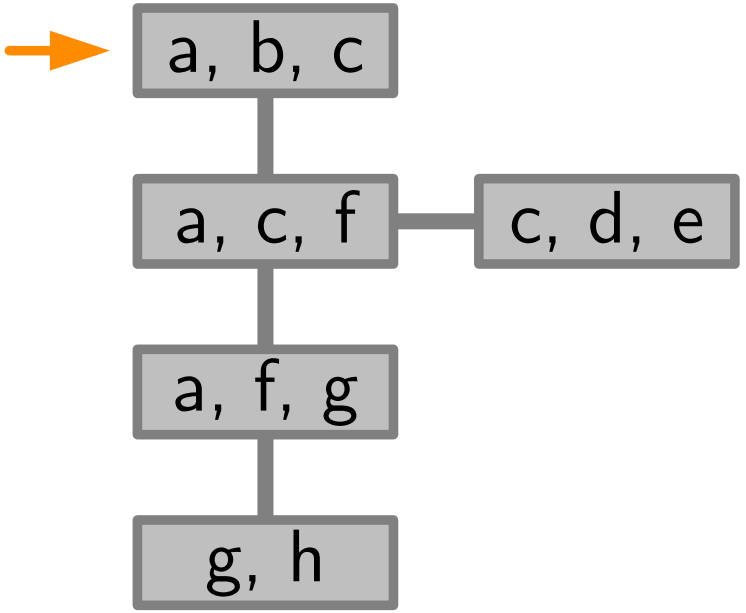
Example: Tree Decomposition

2 each edge is contained in at least one bag

Graph $G = (V, E)$:



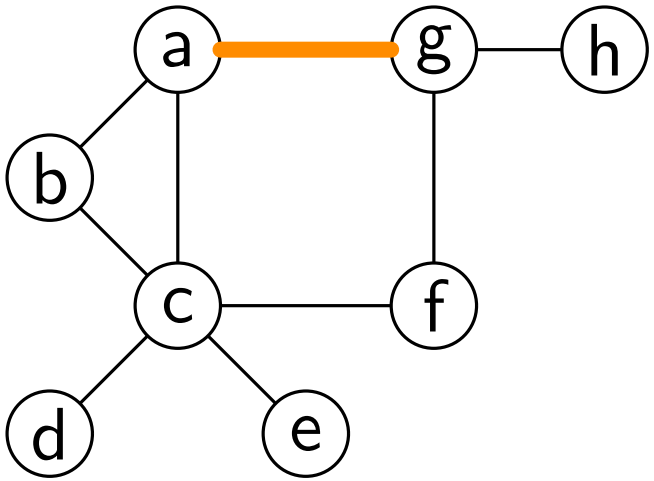
Tree Decomposition:



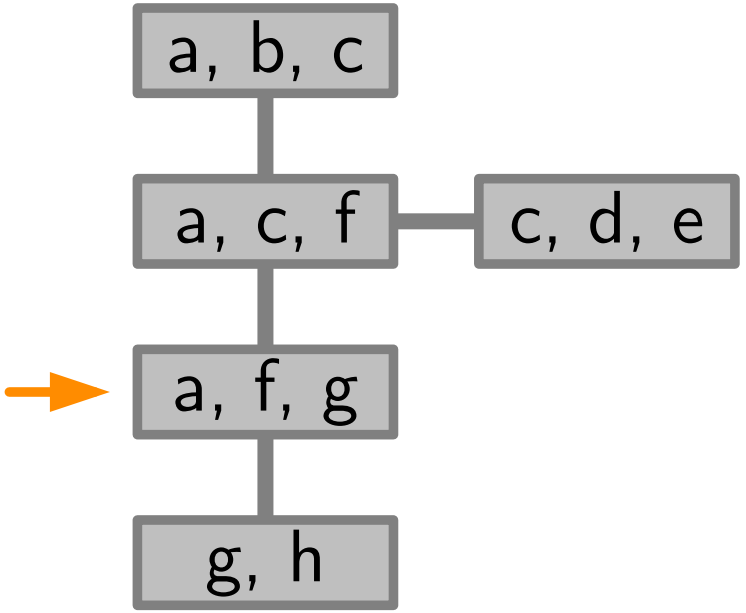
Example: Tree Decomposition

2 each edge is contained in at least one bag

Graph $G = (V, E)$:

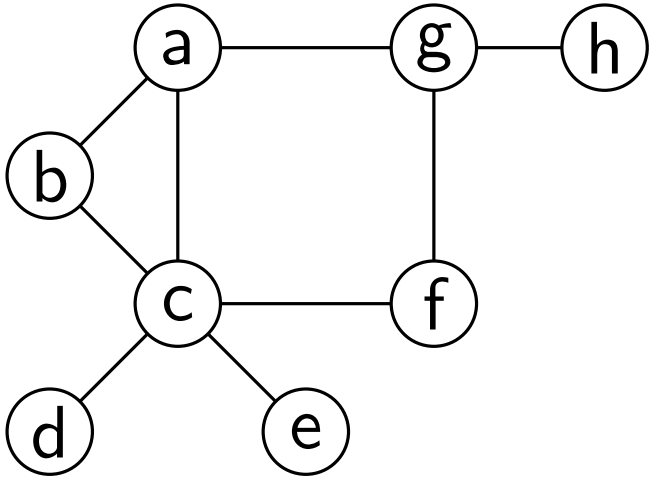


Tree Decomposition:



Example: Tree Decomposition

Graph $G = (V, E)$:



Tree Decomposition:

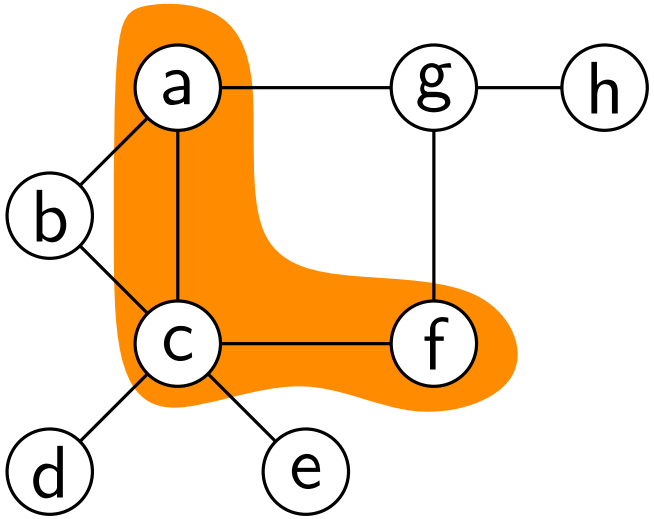
a, b, c,

d, e, f,

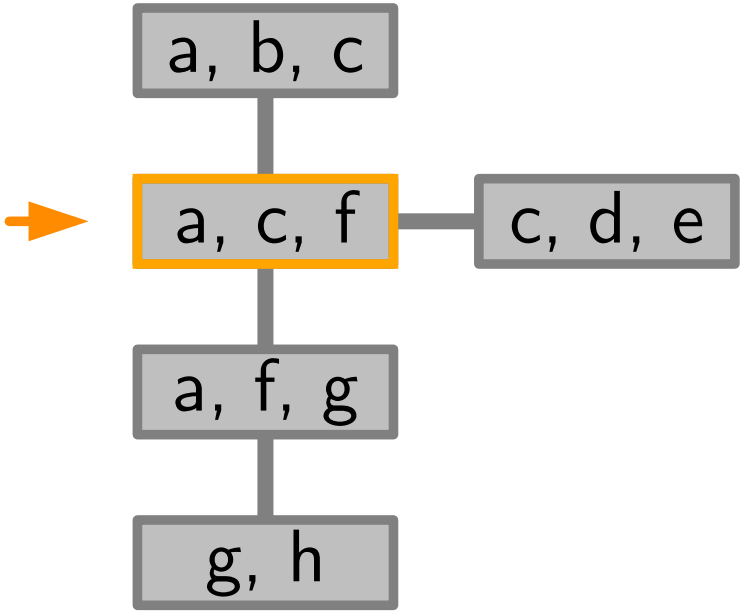
g, h

Example: Tree Decomposition

Graph $G = (V, E)$:

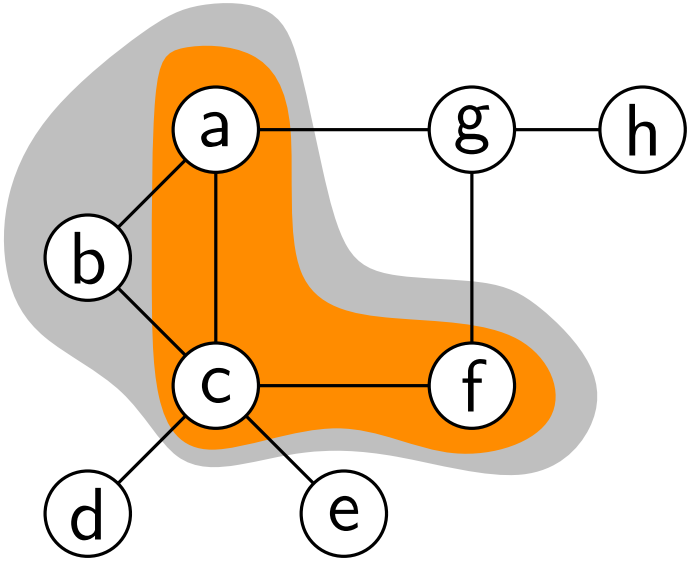


Tree Decomposition:

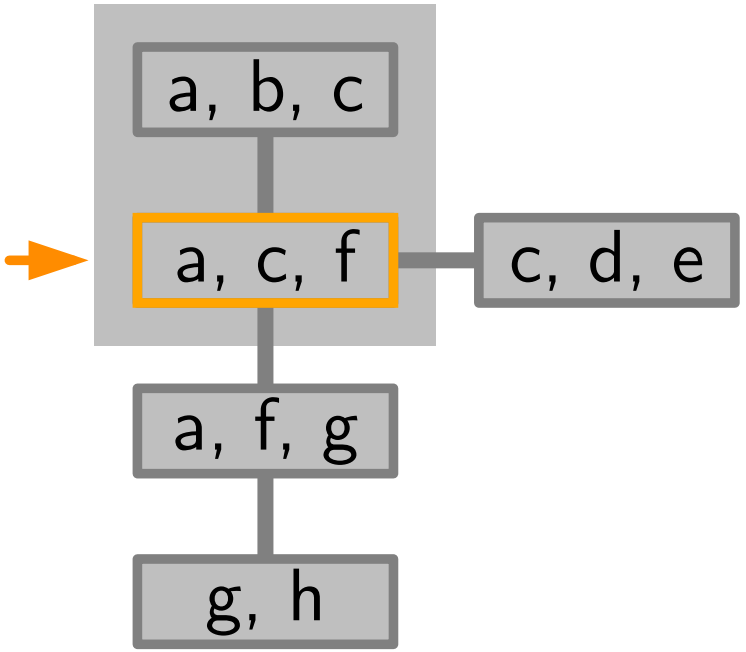


Example: Tree Decomposition

Graph $G = (V, E)$:

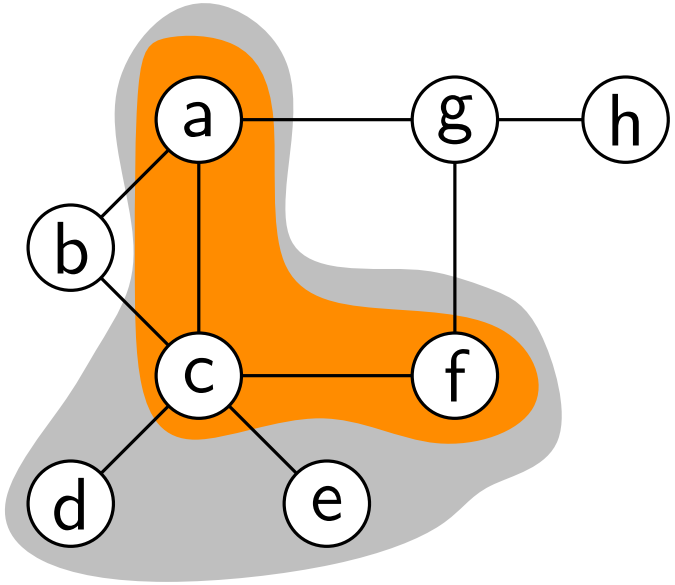


Tree Decomposition:

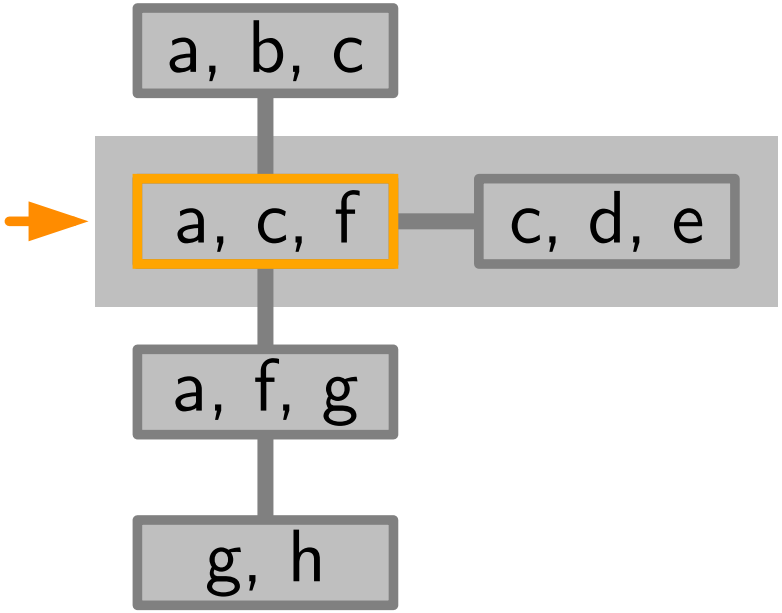


Example: Tree Decomposition

Graph $G = (V, E)$:

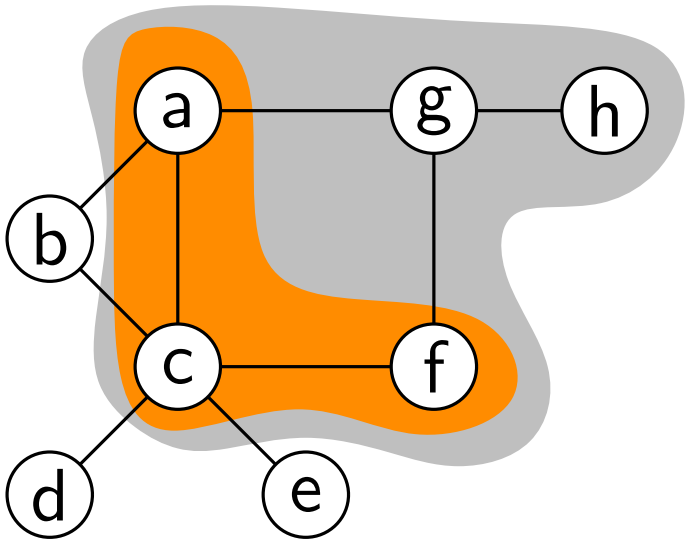


Tree Decomposition:

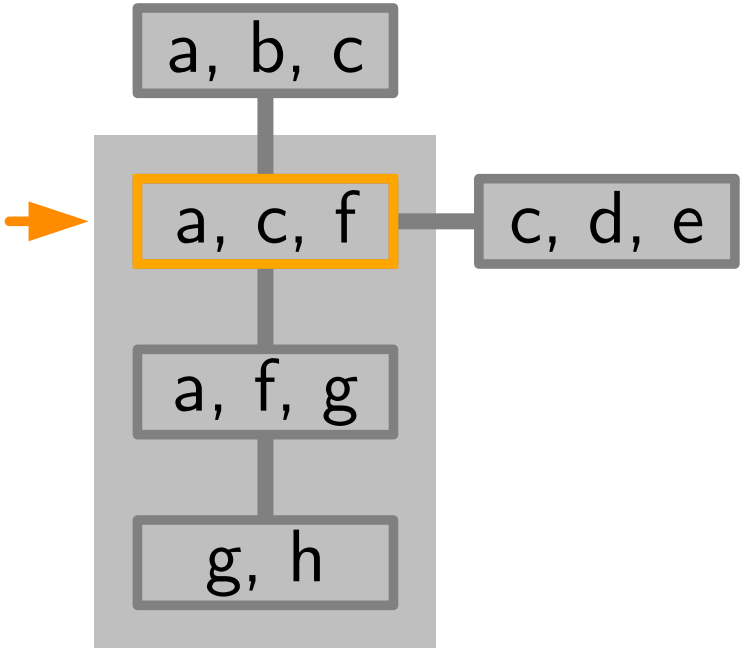


Example: Tree Decomposition

Graph $G = (V, E)$:

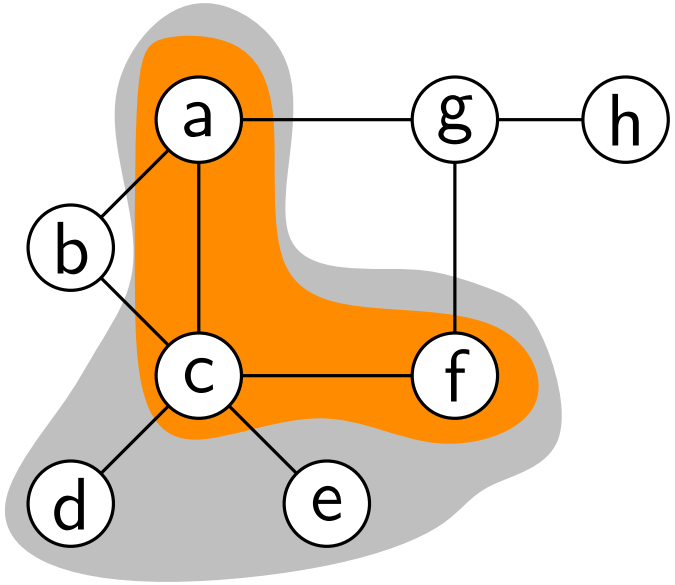


Tree Decomposition:

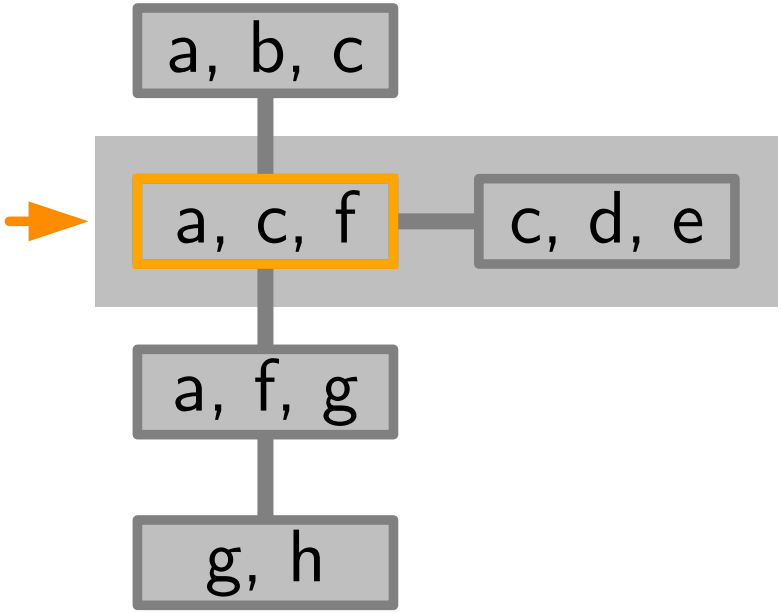


Example: Tree Decomposition

Graph $G = (V, E)$:

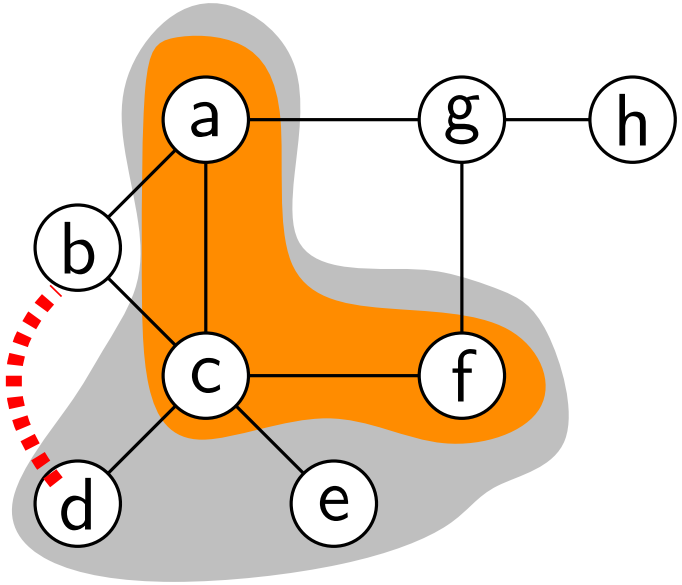


Tree Decomposition:



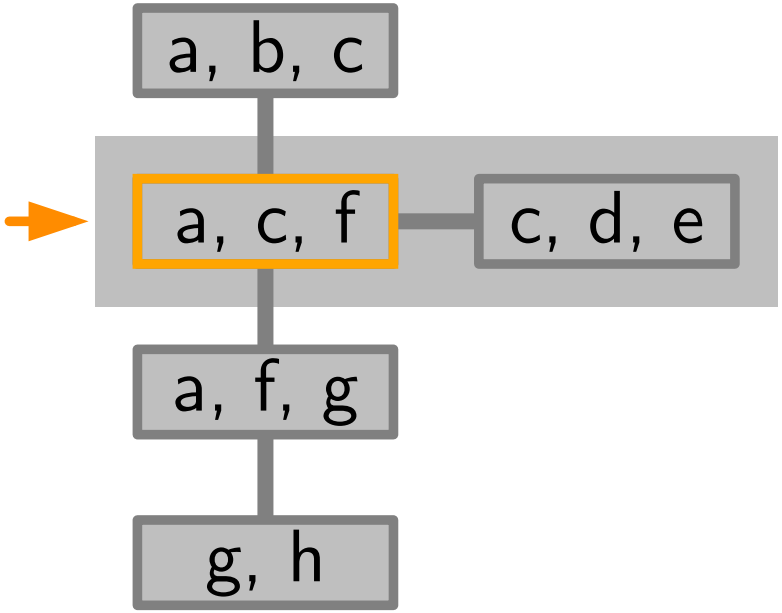
Example: Tree Decomposition

Graph $G = (V, E)$:



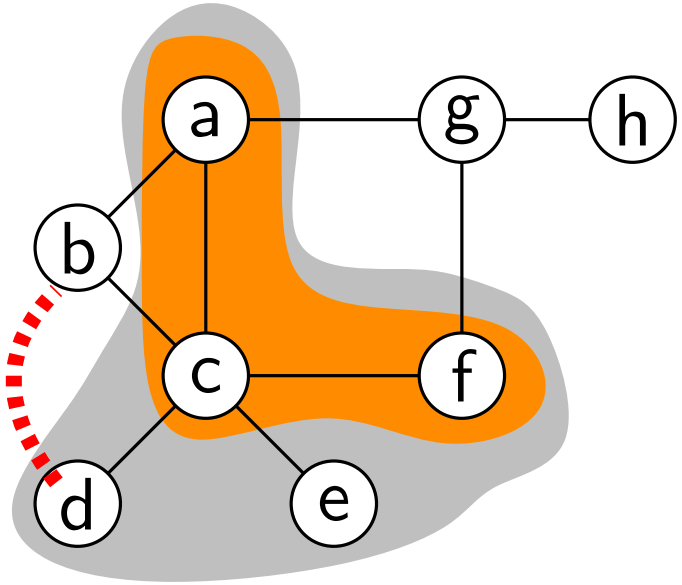
$\{b, d\} \notin E$

Tree Decomposition:



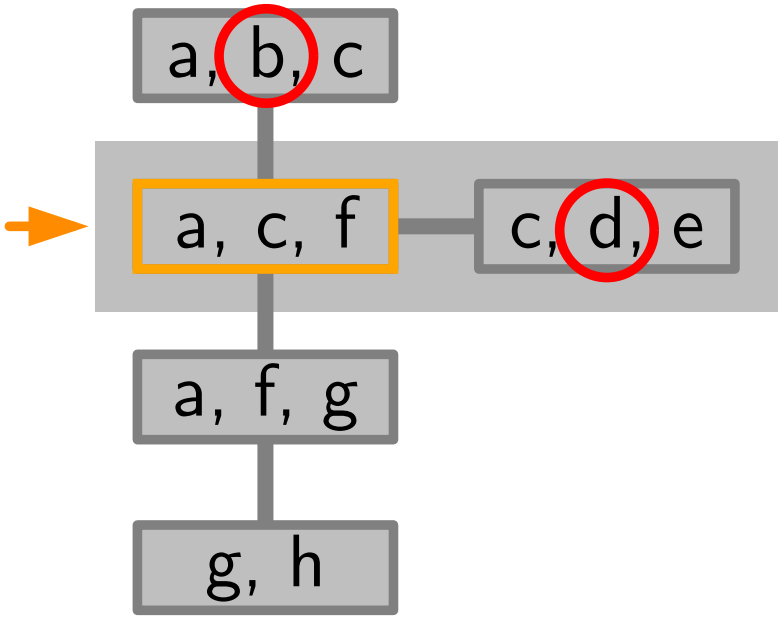
Example: Tree Decomposition

Graph $G = (V, E)$:



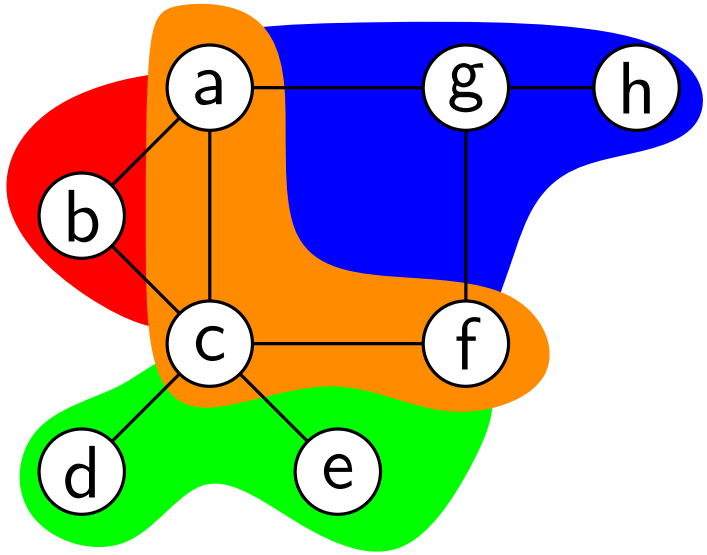
$\{b, d\} \notin E$

Tree Decomposition:

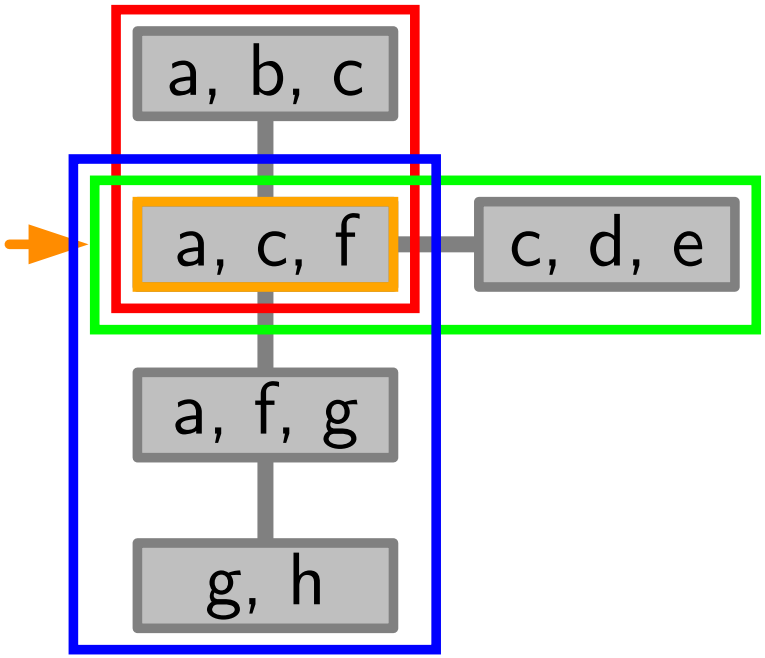


Example: Tree Decomposition

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 (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 (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 (one for each node in P)
- $\bigcup_{p \in P} X_p = V$
- $\forall \{u, v\} \in E \; \exists p \in P$ where $u, v \in X_p$

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 (one for each node in P)
- $\bigcup_{p \in P} X_p = V$
- $\forall \{u, v\} \in E \; \exists p \in P$ where $u, v \in X_p$
- $\forall v \in V : \{p \in P \mid v \in X_p\}$ 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 $\mathrm{tw}(G)$ is the minimum width of a tree decomposition of G

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

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

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

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? $E = \varnothing$

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? $E = \varnothing$

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

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

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

Exercise: Trees have treewidth 1

Exercise: Series parallel graphs have treewidth 2

Thm: There is a tree decomposition of width $\mathrm{tw}(G)$ where $|P|$ is polynomial in n, i.e., the tree has polynomial size 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 likely not FPT, $W[1]$-**comp.**

Parameterized Problems

See **PA §13.3**

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 — likely not FPT, $W[1]$-**comp.**

Parameterized Problems

See **PA §13.3**

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 — likely not FPT, $W[1]$-**comp.**

k-Dominating Set

Parameterized Problems

See **PA §13.3**

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 — likely not FPT, $W[1]$-**comp.**

k-Dominating Set — likely not FPT, $W[2]$-**comp.**

Parameterized Problems

See **PA §13.3**

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	likely not FPT, $W[1]$-**comp.**
k-Dominating Set	likely not FPT, $W[2]$-**comp.**
k-Coloring	

Parameterized Problems

See **PA §13.3**

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	likely not FPT,	$W[1]$**-comp.**
k-Dominating Set	likely not FPT,	$W[2]$**-comp.**
k-Coloring		**NP-comp.** $k \geq 3$

Parameterized Problems

See **PA §13.3**

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	likely not FPT,	$W[1]$-**comp.**
k-Dominating Set	likely not FPT,	$W[2]$-**comp.**
k-Coloring		**NP-comp.** $k \geq 3$

"natural parameterization"

Parameterized Problems

See **PA §13.3**

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	likely not FPT, $W[1]$-**comp.**
k-Dominating Set	likely not FPT, $W[2]$-**comp.**
k-Coloring	**NP-comp.** $k \geq 3$

Independent Set (treewidth)

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

Parameterized Problems

See **PA §13.3**

Given: Instance of size n and parameter k

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

Ex.:

k-Vertex Cover	**FPT**
k-Independent Set	likely not FPT, $W[1]$-**comp.**
k-Dominating Set	likely not FPT, $W[2]$-**comp.**
k-Coloring	**NP-comp.** $k \geq 3$

Independent Set (treewidth)

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

Parameterized Problems

See **PA §13.3**

Given: Instance of size n and parameter k

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

Ex.:

k-Vertex Cover	**FPT**
k-Independent Set	likely not FPT, $W[1]$-**comp.**
k-Dominating Set	likely not FPT, $W[2]$-**comp.**
k-Coloring	**NP-comp.** $k \geq 3$

Independent Set (treewidth) **FPT**

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

Parameterized Problems

See **PA §13.3**

Given: Instance of size n and parameter k

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

Ex.:

k-Vertex Cover		**FPT**
k-Independent Set	likely not FPT,	$W[1]$**-comp.**
k-Dominating Set	likely not FPT,	$W[2]$**-comp.**
k-Coloring		**NP-comp.** $k \geq 3$
Independent Set (treewidth)		**FPT**
List Coloring (treewidth)		

Parameterized Problems

See **PA §13.3**

Given: Instance of size n and parameter k

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

Ex.:

k-Vertex Cover		**FPT**
k-Independent Set	likely not FPT,	$W[1]$**-comp.**
k-Dominating Set	likely not FPT,	$W[2]$**-comp.**
k-Coloring		**NP-comp.** $k \geq 3$
Independent Set (treewidth)		**FPT**
List Coloring (treewidth)		$W[1]$**-comp.**

Parameterized Problems

See **PA §13.3**

Given: Instance of size n and parameter k

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

Ex.:

k-Vertex Cover		**FPT**
k-Independent Set	likely not FPT,	$W[1]$**-comp.**
k-Dominating Set	likely not FPT,	$W[2]$**-comp.**
k-Coloring		**NP-comp.** $k \geq 3$

Independent Set (treewidth)	**FPT**
List Coloring (treewidth)	$W[1]$**-comp.**
Channel Assignment (treewidth)	

Parameterized Problems

See **PA §13.3**

Given: Instance of size n and parameter k

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

Ex.:

k-Vertex Cover		**FPT**
k-Independent Set	likely not FPT,	$W[1]$-**comp.**
k-Dominating Set	likely not FPT,	$W[2]$-**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: $\mathrm{tw}(G) \leq k$?

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

Thm: TREEWIDTH is NP-complete

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

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

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

See PA §7.6.

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 See PA §7.6.

actually fixed-parameter linear: $O(f(k)n)$
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

See PA §7.6.

How can we make "fixed-treewidth-tractable" algorithms?

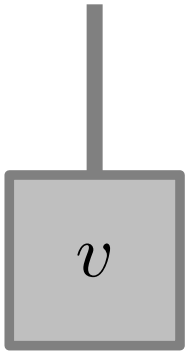
item #1: nice tree decompositoins

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

item #1: nice tree decompositoins

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

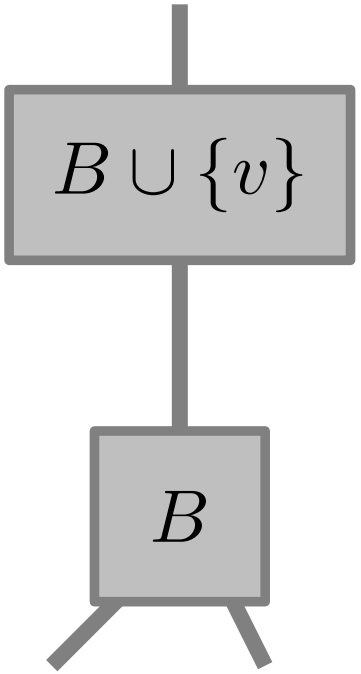
- **Leaf:** the bag is a leaf and contains only one vertex



item #1: nice tree decompositoins

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

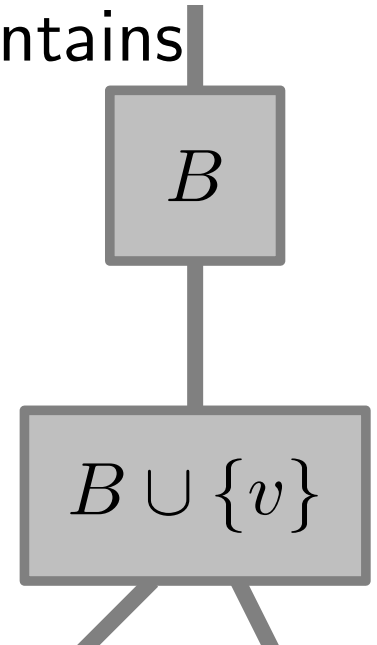
- **Leaf:** the bag is a leaf and contains only one vertex
- **Introduce:**
  The bag has exactly one child and contains the child's vertices and exactly one new vertex.



item #1: nice tree decompositoins

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

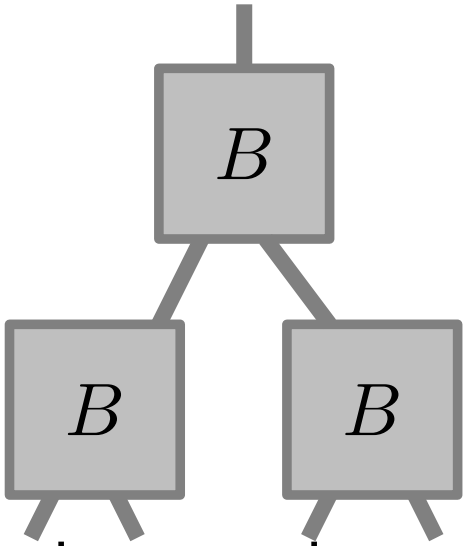
- **Leaf:** the bag is a leaf and contains only one vertex
- **Introduce:**
  The bag has exactly one child and contains the child's vertices and exactly one new vertex.
- **Forget:**
  The bag has exactly one child and contains one vertex fewer than the child.



item #1: nice tree decompositoins

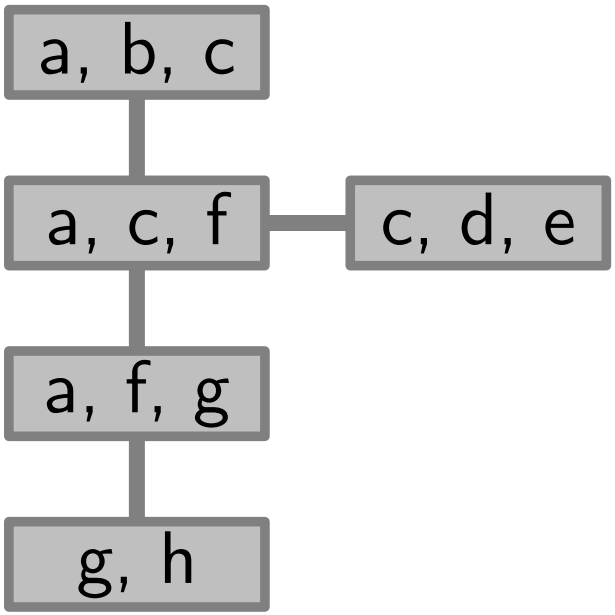
In a *nice* tree decomp., one bag is marked as the root and there are only 4 types of bags:

- **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

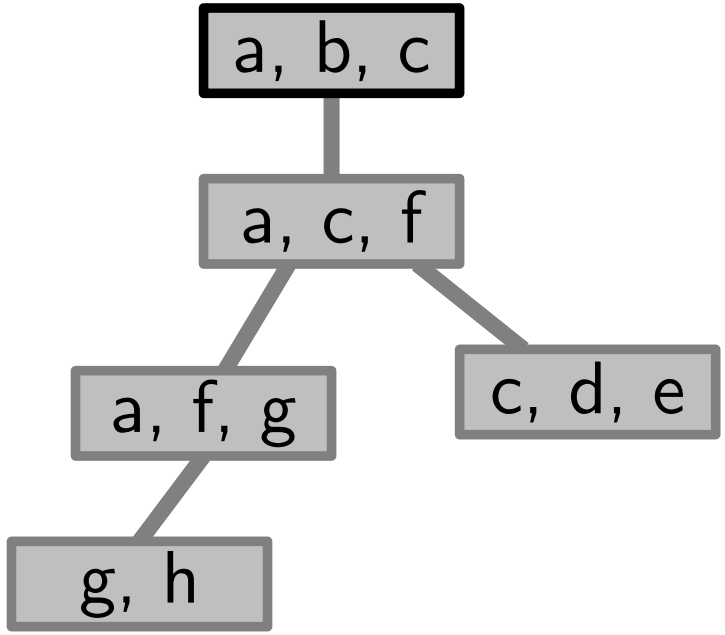


item #1: nice tree decompositoins

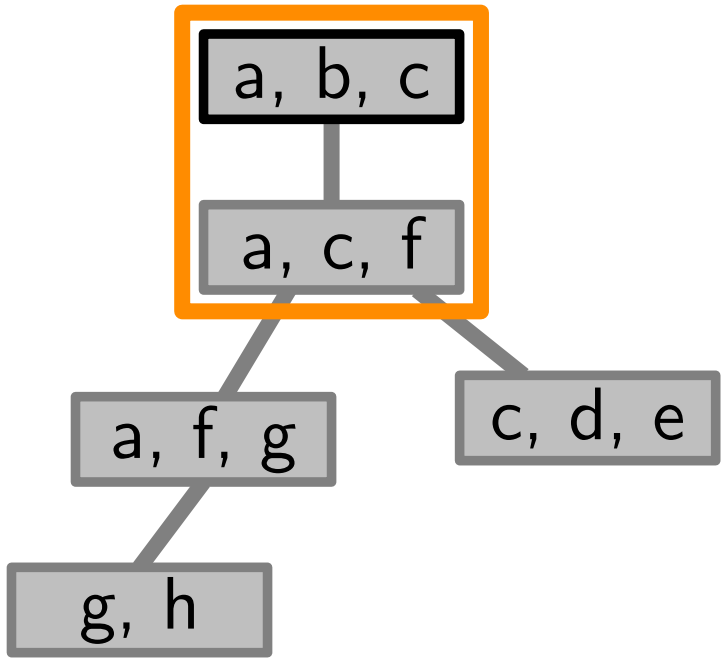
Thm: For each tree decomposition, there is a nice tree decomposition of the same width and polynomially many more bags. This can also be constructed in polynomial time.



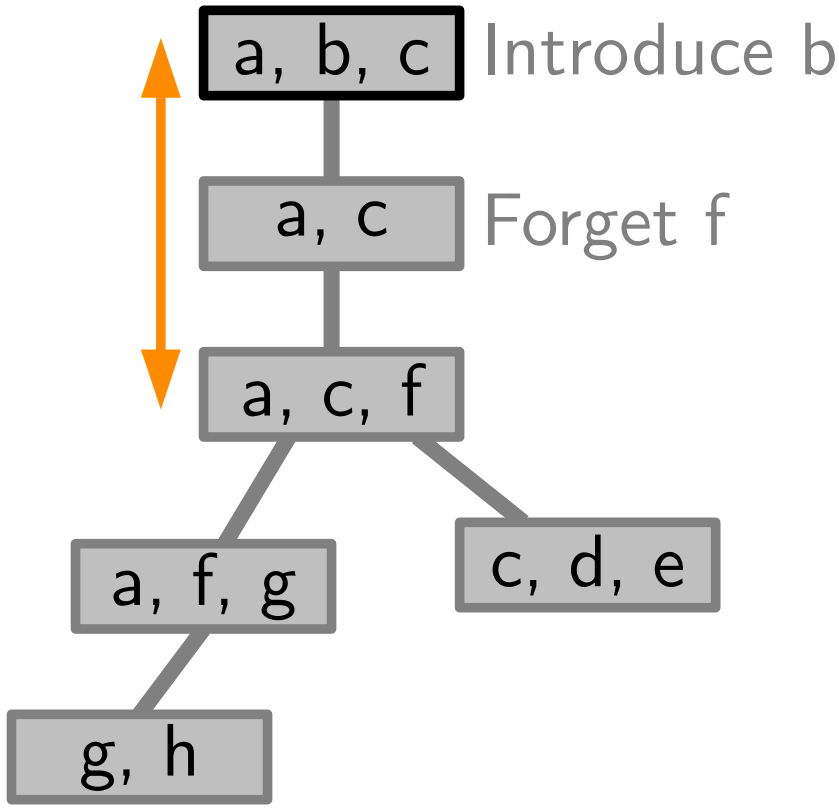
item #1: nice tree decompositoins



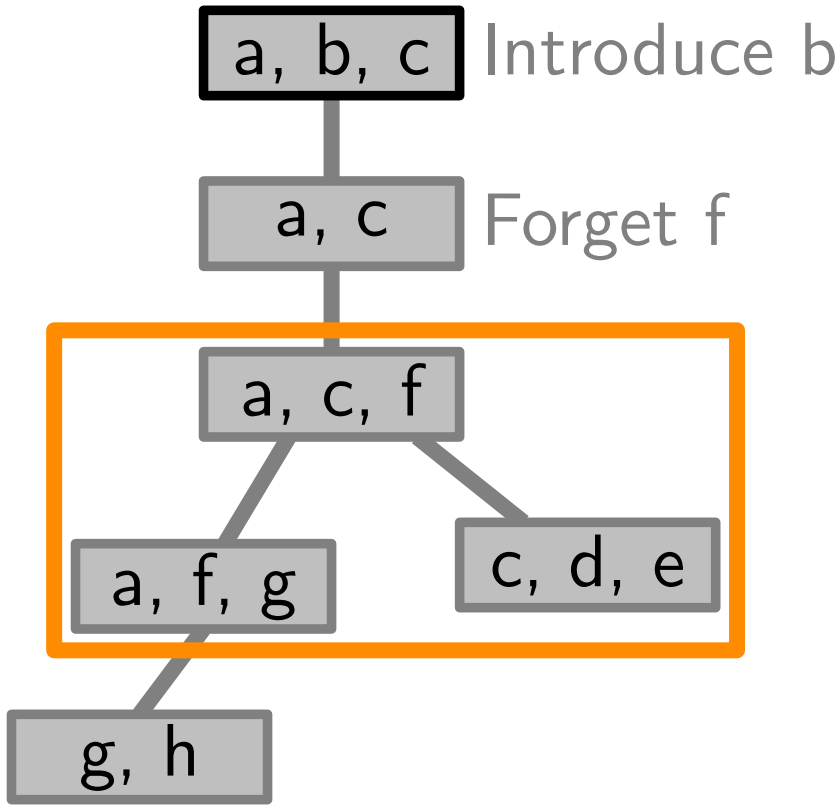
item #1: nice tree decompositoins



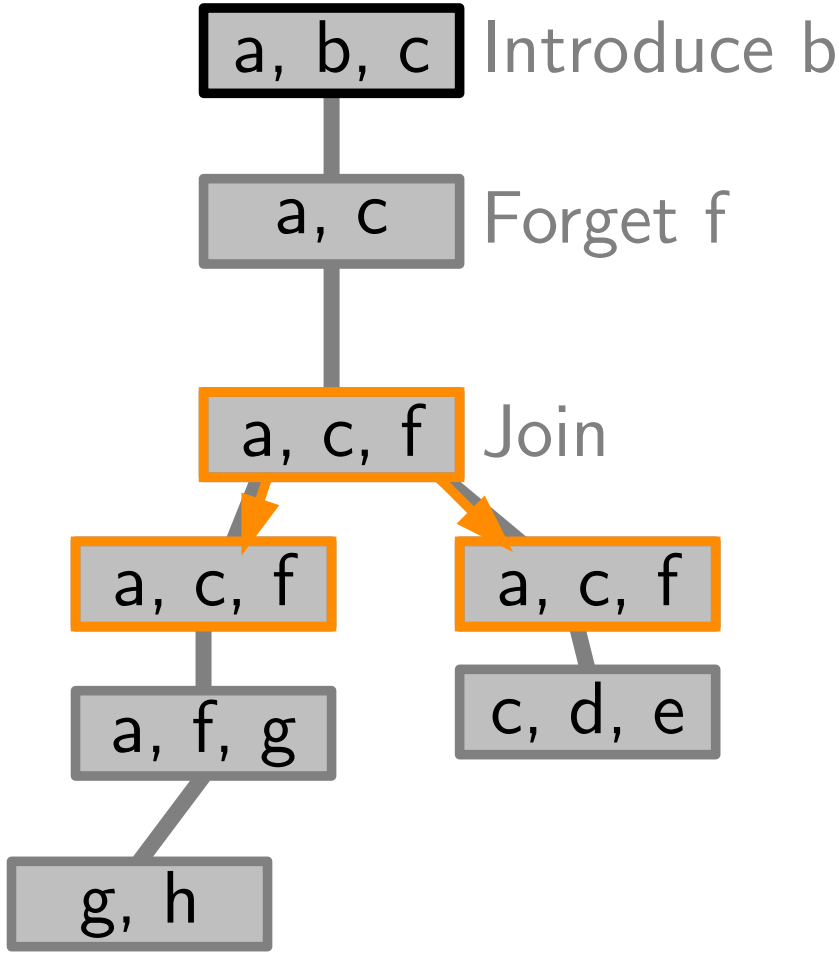
item #1: nice tree decompositoins



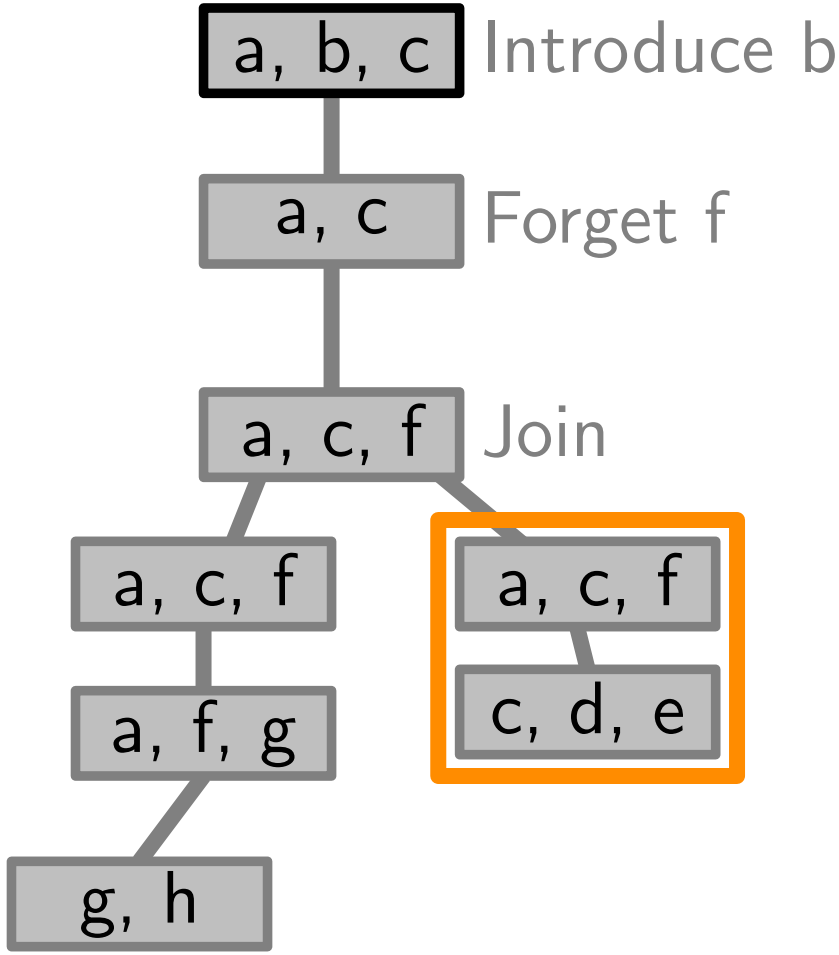
item #1: nice tree decompositoins



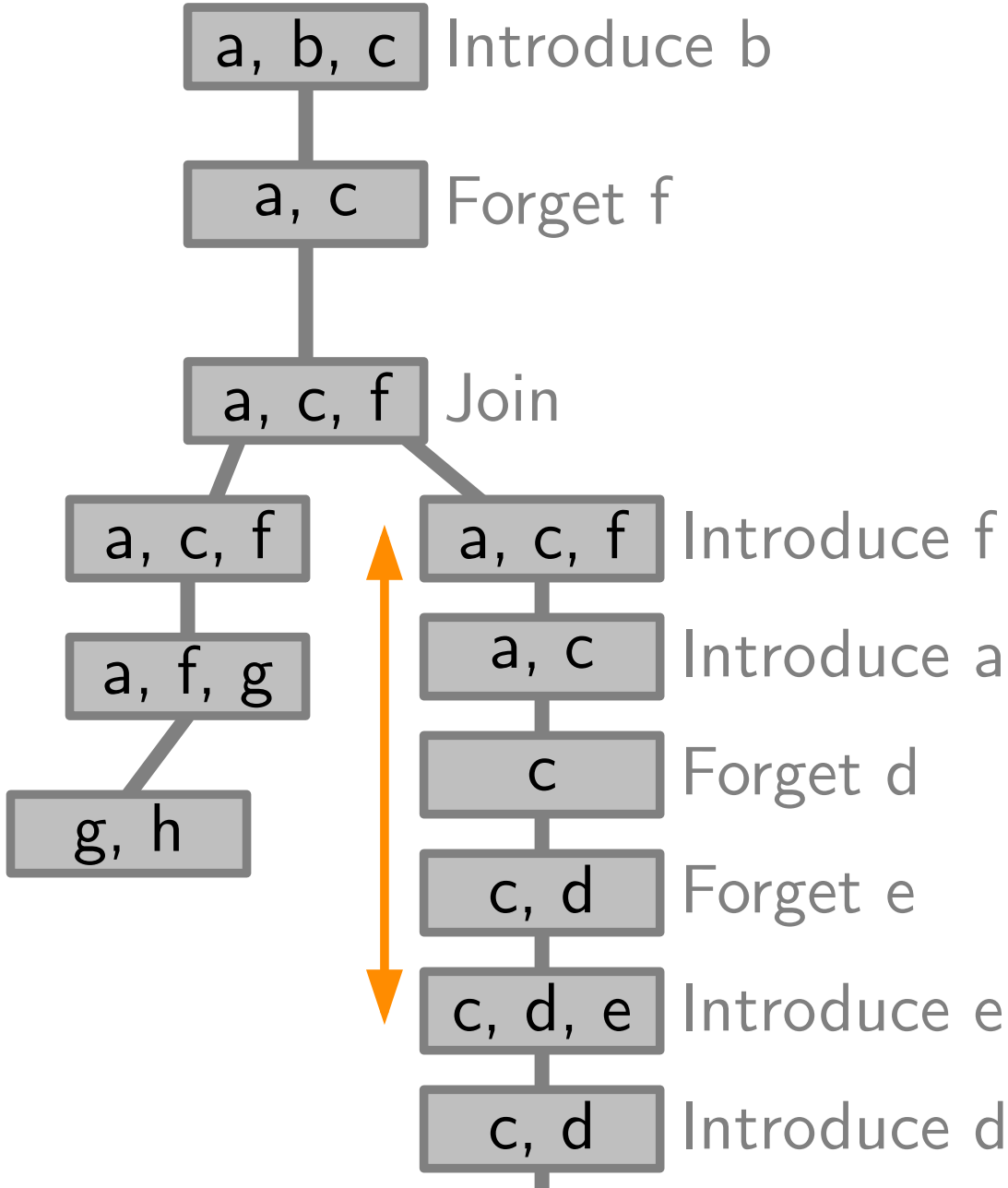
item #1: nice tree decompositoins



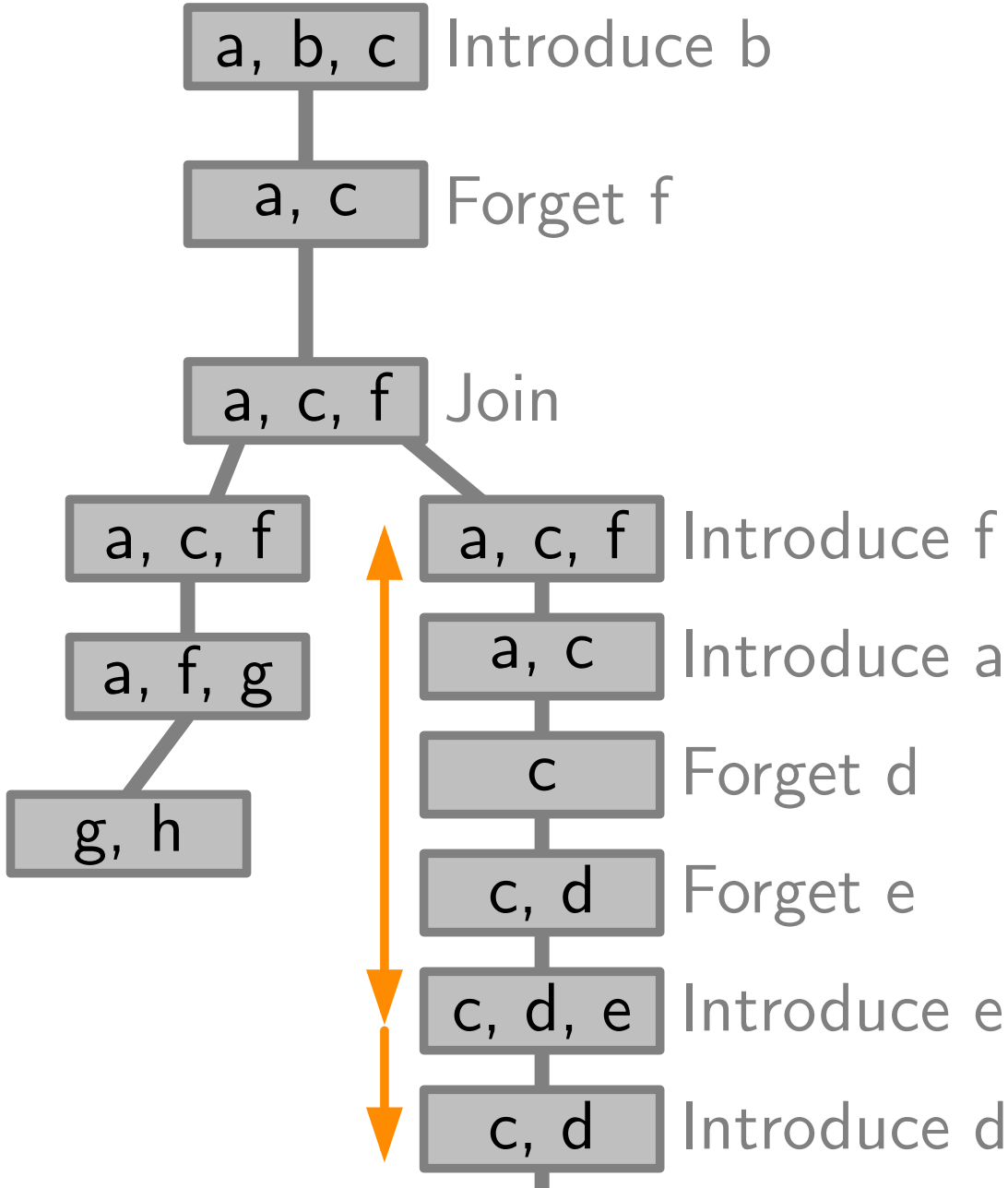
item #1: nice tree decompositoins



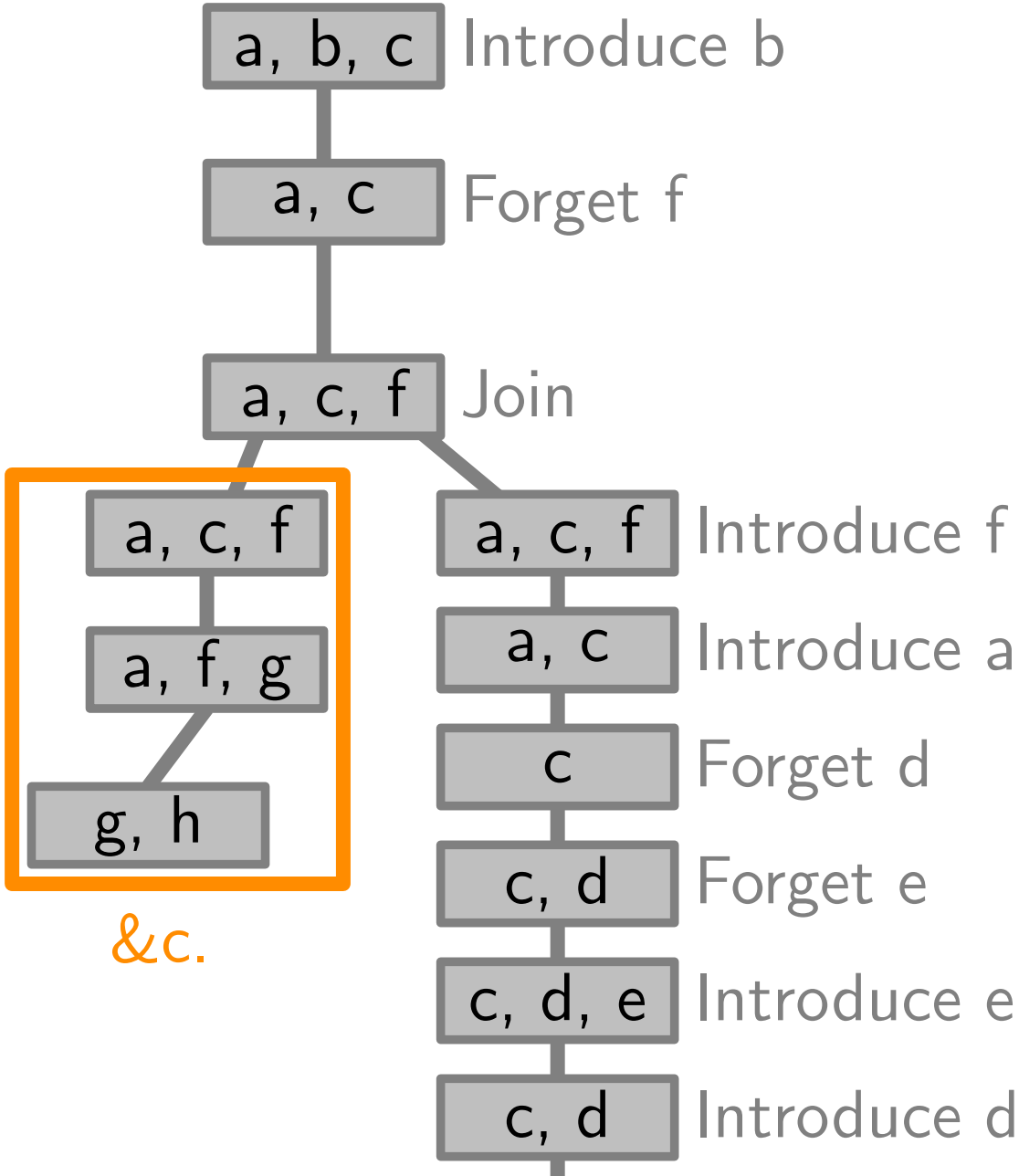
item #1: nice tree decompositoins



item #1: nice tree decompositoins



item #1: nice tree decompositoins



item #2: DP on nice Tree Decomp.

Thm: k-TREEWIDTH is FPT

item #2: DP on nice Tree Decomp.

Thm: k-TREEWIDTH is FPT

Thm: Tree decompositions $\rightarrow$ *nice* in polynomial time.

item #2: DP on nice Tree Decomp.

Thm: k-TREEWIDTH is FPT

Thm: Tree decompositions $\rightarrow$ *nice* in polynomial time.

Cor: For FPT-Algorithms it suffices to use nice tree decomp.

item #2: DP on nice Tree Decomp.

Thm: k-TREEWIDTH is FPT

Thm: Tree decompositions $\rightarrow$ *nice* in polynomial time.

Cor: For FPT-Algorithms it suffices to use nice tree decomp.

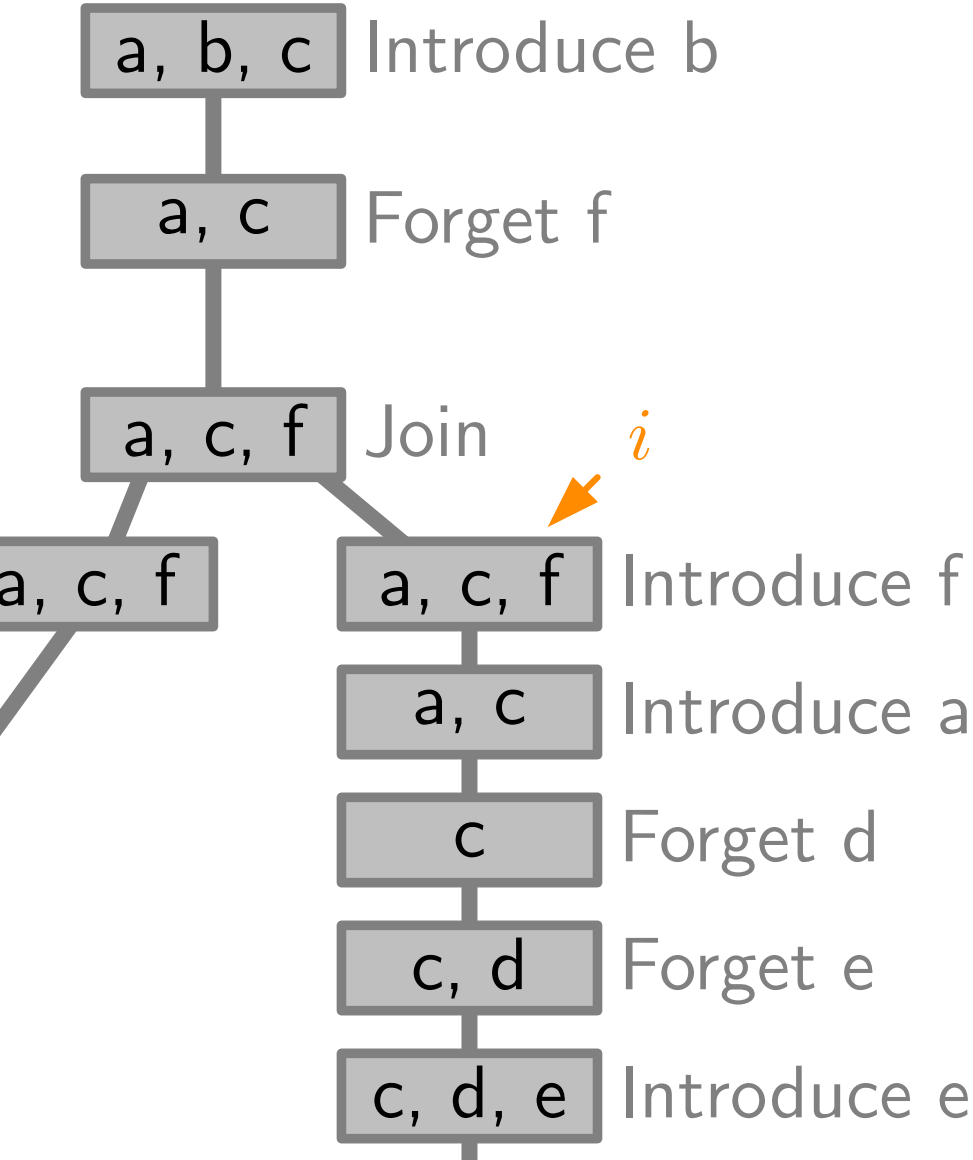
Strategy: Build a recurrence for each type of bag, and use dynamic programming.

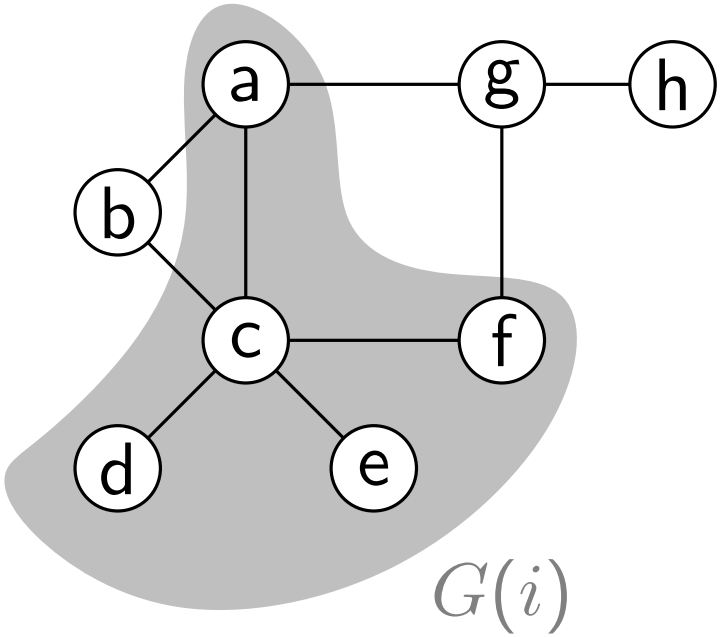
Indep.Set on Nice Tree Decomp.

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

Indep.Set on Nice Tree Decomp.

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





Indep.Set on 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:

Indep.Set on 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 indep. set I in $G(i)$
with $I \cap X_i = S$

Indep.Set on 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 indep. set I in $G(i)$
with $I \cap X_i = S$

When i is a **Leaf** ...

Indep.Set on 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 indep. set I in $G(i)$
with $I \cap X_i = S$

When i is a **Leaf** ...

Let $X_i = \{v\}$.

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

Indep.Set on 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 indep. set I in $G(i)$
with $I \cap X_i = S$

When i is a **Leaf** ...

Let $X_i = \{v\}$.

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

Indep.Set on 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 indep. set I in $G(i)$
with $I \cap X_i = S$

When i is a **Leaf** ...

Let $X_i = \{v\}$.

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

$R(i, \varnothing) =$

Indep.Set on 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 indep. set I in $G(i)$
with $I \cap X_i = S$

When i is a **Leaf** ...

Let $X_i = \{v\}$.

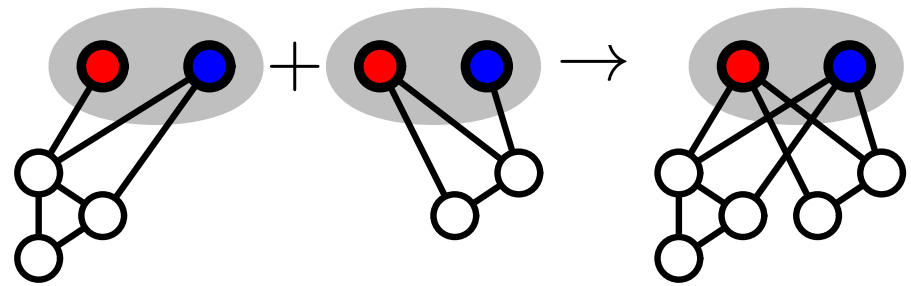
$R(i, \{v\}) = \omega(v)$

$R(i, \varnothing) = 0$

Indep.Set on Nice Tree Decomp.

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

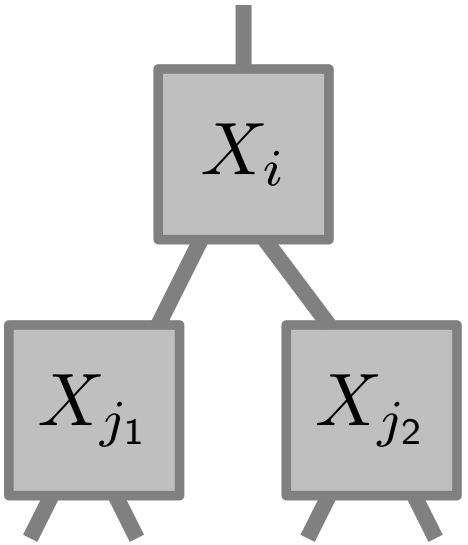
For bag i and $S \subseteq X_i$, let:
$R(i, S) :=$ maximum weight of an indep. set I in $G(i)$
with $I \cap X_i = S$



When i is a **Join** ...

with children j_1 and j_2

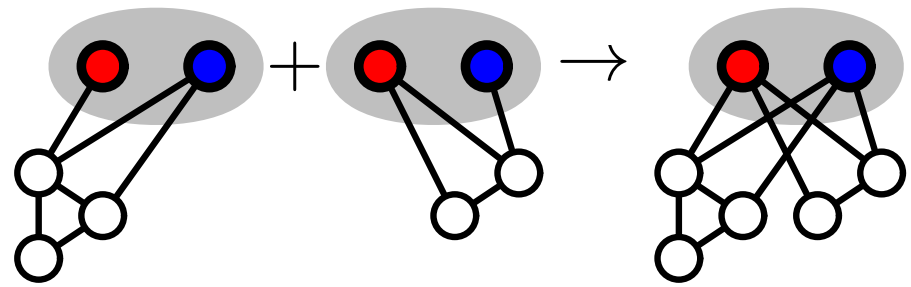
$R(i, S) =$



Indep.Set on Nice Tree Decomp.

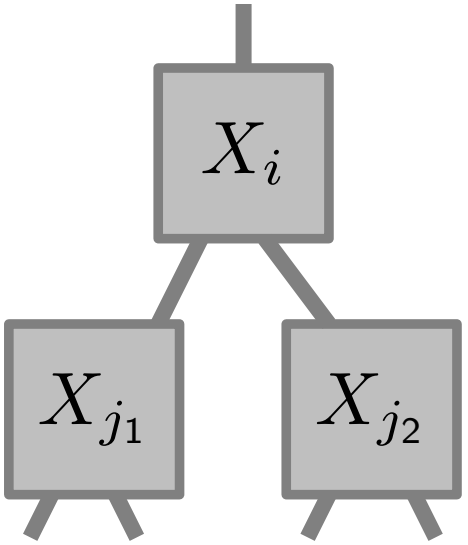
Let $G(i) :=$ Graph induced by the vertices in the subtree at i

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



When i is a **Join** ...
with children j_1 and j_2

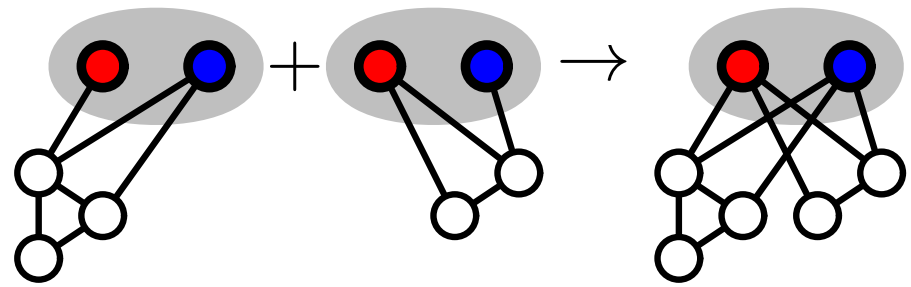
$$R(i, S) = R(j_1, S) + R(j_2, S)$$



Indep.Set on Nice Tree Decomp.

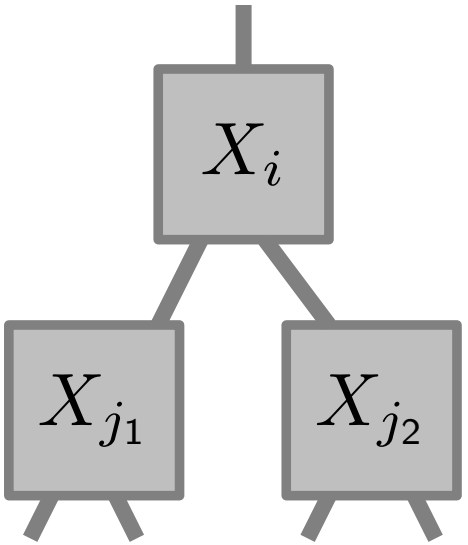
Let $G(i) :=$ Graph induced by the vertices in the subtree at i

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



When i is a **Join** ...
with children j_1 and j_2

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



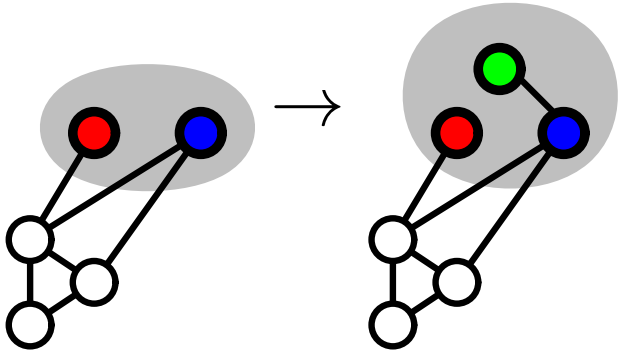
Indep.Set on Nice Tree Decomp.

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 indep. set I in $G(i)$
with $I \cap X_i = S$

When i is an **Introduce** ...
with child j and $X_i = X_j \cup \{v\}$

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



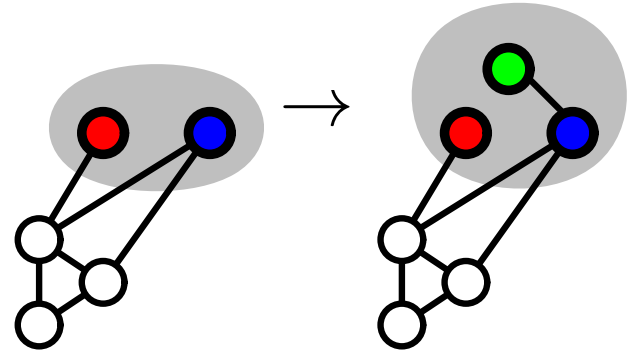
Indep.Set on Nice Tree Decomp.

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 indep. set I in $G(i)$
with $I \cap X_i = S$

When i is an **Introduce** ...
with child j and $X_i = X_j \cup \{v\}$

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

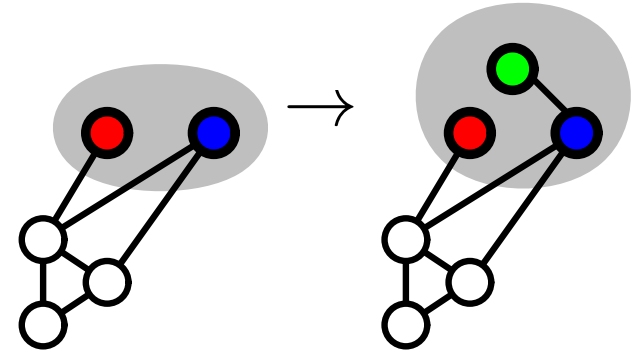


Indep.Set on Nice Tree Decomp.

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 indep. set I in $G(i)$
with $I \cap X_i = S$

When i is an **Introduce** ...
with child j and $X_i = X_j \cup \{v\}$



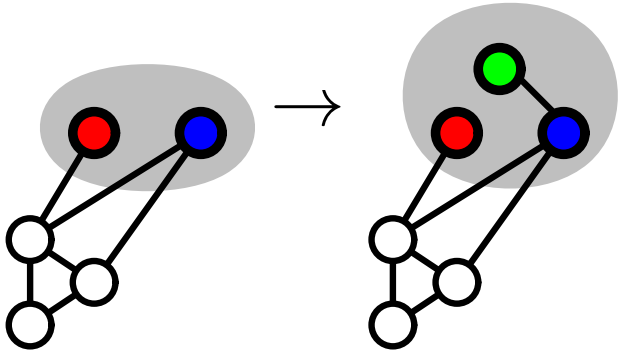
For each $S \subseteq X_j$: $R(i, S) = \quad R(j, S)$
and if v has neighbors in S, $R(i, S \cup \{v\}) =$

Indep.Set on 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 indep. set I in $G(i)$
with $I \cap X_i = S$

When i is an **Introduce** ...
with child j and $X_i = X_j \cup \{v\}$

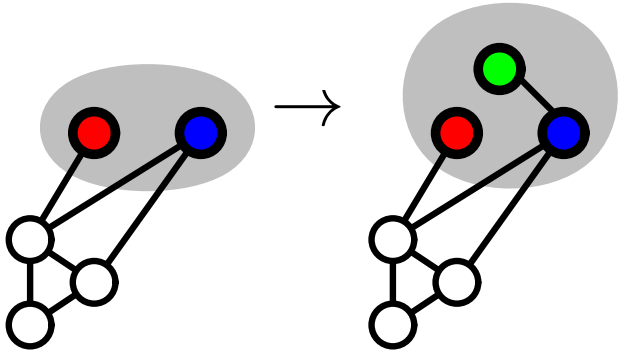


For each $S \subseteq X_j$: $R(i, S) = \quad R(j, S)$
and if v has neighbors in S, $R(i, S \cup \{v\}) = -\infty$

Indep.Set on 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 indep. set I in $G(i)$
with $I \cap X_i = S$



When i is an **Introduce** ...
with child j and $X_i = X_j \cup \{v\}$

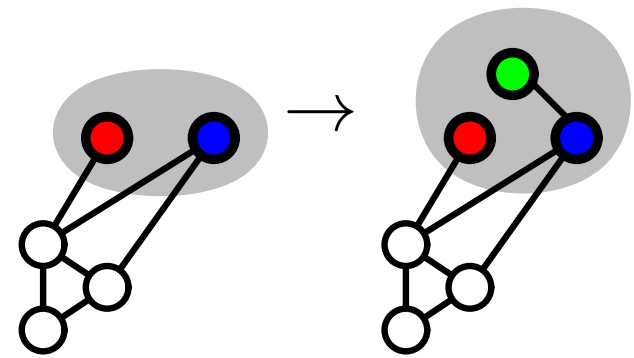
For each $S \subseteq X_j$: $R(i, S) = \quad R(j, S)$
and if v has neighbors in S, $R(i, S \cup \{v\}) = -\infty$
if v has no neighbors in S, $R(i, S \cup \{v\}) =$

Indep.Set on 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 indep. set I in $G(i)$
with $I \cap X_i = S$

When i is an **Introduce** ...
with child j and $X_i = X_j \cup \{v\}$



For each $S \subseteq X_j$: $R(i, S) = \quad R(j, S)$
and if v has neighbors in S, $R(i, S \cup \{v\}) = -\infty$
if v has no neighbors in S, $R(i, S \cup \{v\}) = R(j, S) + \omega(v)$

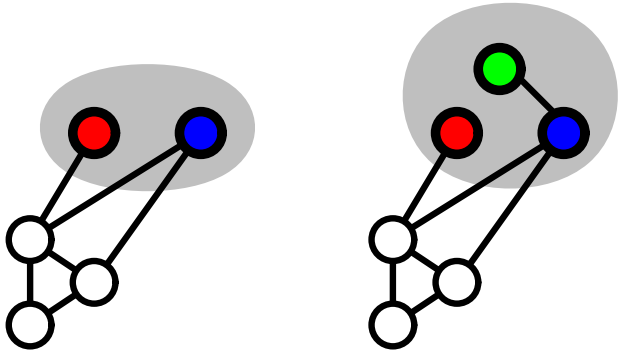
Indep.Set on Nice Tree Decomp.

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 indep. set I in $G(i)$
with $I \cap X_i = S$

When i is a **Forget** ...
with child j and $X_i = X_j \setminus \{v\}$

$R(i, S) =$



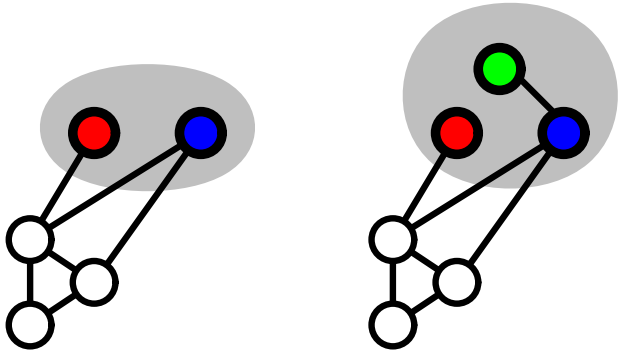
Indep.Set on Nice Tree Decomp.

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 indep. set I in $G(i)$
with $I \cap X_i = S$

When i is a **Forget** ...
with child j and $X_i = X_j \setminus \{v\}$

$R(i, S) =$ max{



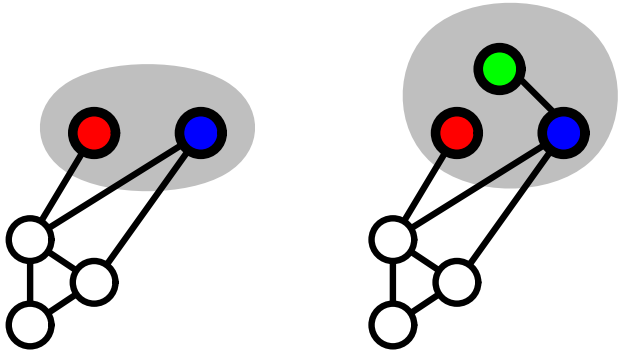
Indep.Set on Nice Tree Decomp.

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 indep. set I in $G(i)$
with $I \cap X_i = S$

When i is a **Forget** ...
with child j and $X_i = X_j \setminus \{v\}$

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



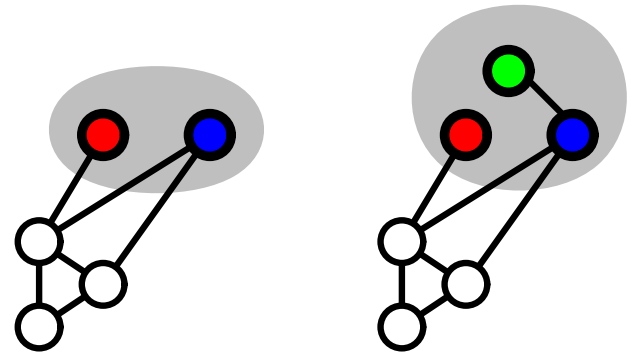
Indep.Set on Nice Tree Decomp.

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 indep. set I in $G(i)$
with $I \cap X_i = S$

When i is a **Forget** ...
with child j and $X_i = X_j \setminus \{v\}$

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



Indep.Set on 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 indep. set I in $G(i)$
with $I \cap X_i = S$

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

Indep.Set on 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 indep. set I in $G(i)$
with $I \cap X_i = S$

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

Runtime: ?

Indep.Set on 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 indep. set I in $G(i)$
with $I \cap X_i = S$

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

Runtime: ?

Thm: The independent set problem is FPT parameterized by treewidth.