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
Independent Set

**INDEPENDENT SET**

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

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

**Thm:** Independent Set is NP-complete.

**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 $\omega : V \rightarrow \mathbb{N}$

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 $w$. 

![Tree Diagram]

Choose an arbitrary root $w$. 

![Tree Diagram]
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 \( T(v) \)

\[ A(w) = \text{solution} \]
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 \( v \not\in S \)
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 $v \not\in S$

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 \( v \not\in S \)

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 \( x_1, \ldots, x_r \):

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

\[
B(v) = \sum_{i=1}^{r} A(x_i)
\]
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 \( x_1, \ldots, x_r \):
\[
A(v) = \max \{ \sum_{i=1}^{r} A(x_i), \omega(v) + \sum_{i=1}^{r} B(x_i) \}
\]
\[
B(v) = \sum_{i=1}^{r} A(x_i)
\]
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 \( v \not\in S \)

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 $t$.
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

![Diagram of series parallel graphs](image)
$s, t$-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

\[
s_1 = s_2 \quad t_1 = t_2
\]
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
SP-tree

\[ s \quad t \quad s \quad t \]
SP-tree
SP-tree

\[ s \quad \text{---} \quad t \]

\[ s \quad \text{---} \quad t \]
SP-tree
SP-tree
Let $i$ be a node in an SP-tree.

$G(i) :=$ graph represented by the subtree rooted at $i$
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 $G(i)$

$AA(i) := \text{maximum weight independent set } S \text{ in } G(i) \text{ where } s_i \in S \text{ and } t_i \in S$
Independent Set on SP-trees

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

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

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

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

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

$AB(i) \text{ and } BB(i) \text{ def. similarly}$

When $i$ is a leaf...

$$BB(i) = 0$$
Independent Set on SP-trees

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

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

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

$AB(i) \text{ and } BB(i) \text{ def. similarly}$

When $i$ is a leaf...

$BA(i) =$

$BB(i) = 0$
Independent Set on SP-trees

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

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

$BA(i) := \text{maximum weight independent set } S \text{ in } G(i) \text{ where } s_i \not\in S \text{ and } t_i \in S$

$AB(i)$ 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 $s_i \not\in S$ and $t_i \in S$

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

When $i$ 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 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...

$$AB(i) = \omega(s_i)$$
$$BA(i) = \omega(t_i)$$
$$BB(i) = 0$$
Independent Set on SP-trees

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

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

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

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

When $i$ 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 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 $s_i \not\in S$ and $t_i \in S$

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

When $i$ is a leaf...

$AA(i) = -\infty$

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

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

$BB(i) = 0$
Independent Set on SP-trees

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

$AA(i) :=$ maximum weight 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 $y$, ...

$$AA(i) =$$

$t_x = s_y$
Independent Set on SP-trees

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

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

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

$AB(i)$ 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 $s_i \not\in S$ 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 $G(i)$

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

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

$AB(i)$ 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 $G(i)$

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

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

$AB(i) \text{ and } BB(i) \text{ def. similarly}$

other cases omitted... (easy exercise)
Independent Set on SP-trees

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

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

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

$AB(i)$ 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 $G(i)$

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

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

$AB(i)$ 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

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.
2. These bags are connected.

Graph $G = (V, E)$:

Tree Decomposition:
Example: Tree Decomposition

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

Graph $G = (V, E)$:

Tree Decomposition:
Example: Tree Decomposition

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

Graph $G = (V, E)$:

Tree Decomposition:
Example: Tree Decomposition

Graph $G = (V, E)$:

Tree Decomposition:

Each edge is contained in at least one bag.
Example: Tree Decomposition

Graph $G = (V, E)$:

- $V = \{a, b, c, d, e, f, g, h\}$
- $E = \{a, b, c, d, e, f, g, h\}$

Tree Decomposition:

- Each edge is contained in at least one bag.
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 $G = (V, E)$:

Tree Decomposition:
Example: Tree Decomposition

Graph \( G = (V, E) \):

\[
\{b, d\} \not\in E
\]

Tree Decomposition:
Example: Tree Decomposition

Graph $G = (V, E)$:

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

- a tuple $D = (X, T)$
Tree Decomposition (formal)

**Def.** A *tree decomposition* of a graph $G = (V, E)$ is:
- a tuple $D = (X, T)$
- $T = (P, F)$ is a tree
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 $P$)
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 Decomposition (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 Decomposition (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 \( \text{tw}(G) \) 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 \( tw(G) \) is the minimum width of a tree decomposition of \( G \)

**Obs.** \( tw(G) < n \)
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 \( tw(G) \) is the minimum width of a tree decomposition of \( G \)

**Obs.** \( tw(G) < n \)

**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 $\text{tw}(G)$ 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
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? $E = \emptyset$

**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? $E = \emptyset$

**Exercise:** Trees have treewidth 1

**Exercise:** Series parallel graphs have treewidth 2

**Thm:** There is a tree decomposition of width $\text{tw}(G)$ 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 $O(f(k) \cdot \text{poly}(n))$ 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 $O(f(k) \cdot \text{poly}(n))$ time.

**Ex.:** $k$-Vertex Cover FPT
Parameterized Problems

Given: Instance of size $n$ and parameter $k$

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

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

$k$-Independent Set
Parameterized Problems

Given: Instance of size $n$ and parameter $k$

**Def.** Problem is FPT when solvable in $O(f(k) \cdot \text{poly}(n))$ 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 \( \text{FPT} \)
- \( k \)-Independent Set \( \text{likely not FPT, } W[1]-\text{comp.} \)

See PA §13.3
Parameterized Problems

Given: Instance of size $n$ and parameter $k$

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

**Ex.:**
- **$k$-Vertex Cover** (FPT)
- **$k$-Independent Set** (likely not FPT, W[1]-comp.)
- **$k$-Dominating Set**

See PA §13.3
Parameterized Problems

Given: Instance of size $n$ and parameter $k$

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

**Ex.:**
- \textit{k-Vertex Cover} \quad \text{FPT}
- \textit{k-Independent Set} \quad \text{likely not FPT, W[1]-comp.}
- \textit{k-Dominating Set} \quad \text{likely not FPT, W[2]-comp.}

See PA §13.3
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
- $k$-Independent Set
- $k$-Dominating Set
- $k$-Coloring

- FPT

See PA §13.3
Parameterized Problems

Given: Instance of size $n$ and parameter $k$

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

**Ex.:**

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

See PA §13.3
Parameterized Problems

Given: Instance of size $n$ and parameter $k$

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

**Ex.:**
- **$k$-Vertex Cover**
- **$k$-Independent Set**
- **$k$-Dominating Set**
- **$k$-Coloring**

“natural parameterization”

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

See **PA §13.3**
Parameterized Problems

Given: Instance of size \( n \) and parameter \( k \)

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

**Ex.:**

- \( k \)-**Vertex Cover**  
  - **FPT**
- \( k \)-**Independent Set**  
  - likely not FPT, \( W[1] \)-comp.
- \( k \)-**Dominating Set**  
- \( k \)-**Coloring**  
  - \( \text{NP-comp.} \) \( k \geq 3 \)

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

**Ex.:**
- **$k$-Vertex Cover**  
  FPT
- **$k$-Independent Set**  
- **$k$-Dominating Set**  
- **$k$-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 $O(f(k) \cdot \text{poly}(n))$ time.

**Ex.:**
- **$k$-Vertex Cover**
  - FPT
- **$k$-Independent Set**
- **$k$-Dominating Set**
- **$k$-Coloring**

**Independent Set (treewidth)**

<table>
<thead>
<tr>
<th><strong>Given:</strong></th>
<th>Graph $G$, number $k$</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Parameter:</strong></td>
<td>$\text{tw}(G)$</td>
</tr>
<tr>
<td><strong>Question:</strong></td>
<td>Does $G$ have an independent set of size $\geq k$?</td>
</tr>
</tbody>
</table>
Parameterized Problems

Given: Instance of size $n$ and parameter $k$

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

*Ex.:*  
- $k$-Vertex Cover  
  - FPT  
- $k$-Independent Set  
- $k$-Dominating Set  
- $k$-Coloring  
  - NP-comp. $k \geq 3$

Independent Set (treewidth)  FPT  
List Coloring (treewidth)

See PA §13.3
Parameterized Problems

Given: Instance of size $n$ and parameter $k$

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

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

See PA §13.3
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**
- $k$-**Dominating Set**
- $k$-**Coloring**
  - NP-comp. $k \geq 3$

**Independent Set (treewidth)**
- FPT

**List Coloring (treewidth)**

**Channel Assignment (treewidth)**
Parameterized Problems

Given: Instance of size \( n \) and parameter \( k \)

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

Ex.:
- \( k \)-Vertex Cover \( \text{FPT} \)
- \( k \)-Independent Set likely not FPT, \( W[1]\)-comp.
- \( k \)-Dominating Set likely not FPT, \( W[2]\)-comp.
- \( k \)-Coloring \( \text{NP-comp.} \quad k \geq 3 \)

Independent Set (treewidth) \( \text{FPT} \)
List Coloring (treewidth) \( W[1]\)-comp.
Channel Assignment (treewidth) \( \text{NP-comp.} \quad k \geq 3 \)

See PA §13.3
Computing Treewidth

**TreeWidth**

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

See PA §7.6.
Computing Treewidth

**Treeewidth**

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

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

**Thm:** Treeewidth is NP-complete

**k-Treeewidth**

**Given:** graph $G = (V, E)$

**Parameter:** number $k$

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

**Thm:** $k$-Treeewidth is FPT

- actually fixed-parameter linear: $O(f(k)n)$
- algorithm is constructive (provides an optimal tree decomp.)

See PA §7.6.
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

See PA §7.6.

How can we make “fixed-treewidth-tractable” algorithms?
item #1: nice tree decompositions

In a *nice* tree decomp., one bag is marked as the root and there are only 4 types of bags:
item #1: nice tree decompositions

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

\[ v \]
**item #1: nice tree decompositions**

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 decompositions

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 decompositions

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.
**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 decompositions

```
a, b, c
  /    
\a, c, f/
   /    
\a, f, g|   c, d, e
   /    
\g, h  /
```
item #1: nice tree decompositions
item #1: nice tree decompositions

- Introduce b
- Forget f
item #1: nice tree decompositions

Introduce b

Forget f
item #1: nice tree decompositions

```
Introduce b
Forget f
Join
```

```
g, h

a, f, g

a, c, f

a, c

a, b, c
```
item #1: nice tree decompositions

Introduce b

Forget f

Join

a, b, c

a, c

a, c, f

a, c, f

a, f, g

c, d, e

g, h
item #1: nice tree decompositions

Introduce b

Forget f

Join

Introduce f

Introduce a

Forget d

Forget e

Introduce e

Introduce d
item #1: nice tree decompositions

- Introduce b
- Forget f
- Join
- Introduce f
- Introduce a
- Forget d
- Forget e
- Introduce e
- Introduce d
item #1: nice tree decompositions

Introduce b

Forget f

Join

Introduce a

Introduce f

Introduce e

Forget d

Introduce d

Forget e

Introduce e

Introduce c

Forget d

Introduce c

Introduce g

Forget e

Introduce d

Introduce a
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 $G(i) := \text{Graph induced by the vertices in the subtree at } i$
Indep.Set on Nice Tree Decomp.

Let $G(i) := \text{Graph induced by the vertices in the subtree at } i$
Indep.Set on Nice Tree Decomp.

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

For bag $i$ and $S \subseteq X_i$, let:
Indep.Set on Nice Tree Decomp.

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

For bag $i$ and $S \subseteq X_i$, let:
$$R(i, S) := \text{maximum weight of an indep. set } I \text{ in } G(i)$$
with $I \cap X_i = S$
Indep.Set on Nice Tree Decomp.

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

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

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

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:

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

When $i$ is a Leaf ...

Let $X_i = \{v\}$.

$$R(i, \{v\}) = \omega(v)$$
Indep.Set on Nice Tree Decomp.

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

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

$R(i, S) := \text{maximum weight of an indep. set } I \text{ in } G(i)$

with $I \cap X_i = S$

When $i$ is a Leaf ...

Let $X_i = \{v\}$.

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

$R(i, \emptyset) =$
Indep.Set on Nice Tree Decomp.

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

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

$R(i, S) := \text{maximum weight of an indep. set } I \text{ in } G(i)$

with $I \cap X_i = S$

When $i$ is a Leaf ... 

Let $X_i = \{v\}$.

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

$R(i, \emptyset) = 0$
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 $j_2$
$R(i, S) =$
Let $G(i) := \text{Graph induced by the vertices in the subtree at } i$

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

$R(i, S) := \text{maximum weight of an indep. set } I \text{ in } G(i)$

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 $G(i) := \text{Graph induced by the vertices in the subtree at } i$

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

$R(i, S) := \text{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 $G(i) := \text{Graph induced by the vertices in the subtree at } i$

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

$R(i, S) := \text{maximum weight of an indep. set } I \text{ in } G(i)$

with $I \cap X_i = S$

When $i$ is an \textbf{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 $G(i) := \text{Graph induced by the vertices in the subtree at } i$

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

$R(i, S) := \text{maximum weight of an indep. set } I \text{ in } G(i)$

with $I \cap X_i = S$

When $i$ is an Introduce ...

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

For each $S \subseteq X_j$: $R(i, S) = R(j, S)$
Indep.Set on Nice Tree Decomp.

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

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

$R(i, S) := \text{maximum weight of an indep. set } I \text{ in } G(i)$

with $I \cap X_i = S$

When $i$ is an Introduce ...

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

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

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:
\[ R(i, S) := \text{maximum weight of an indep. set } I \text{ in } G(i) \]
with $I \cap X_i = S$

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

For each $S \subseteq X_j$: $R(i, S) = R(j, S)$
and if $v$ has neighbors in $S$, $R(i, S \cup \{v\}) = -\infty$
Indep.Set on Nice Tree Decomp.

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

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

$R(i, S) := \text{maximum weight of an indep. set } I \text{ in } G(i)$

with $I \cap X_i = S$

When $i$ is an Introduce ...

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

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

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:
\[
R(i, S) := \text{maximum weight of an indep. set } I \text{ in } G(i) \text{ with } I \cap X_i = S
\]

When \( i \) is an Introduce ...

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

For each \( S \subseteq X_j \):
\[
R(i, S) = R(j, S)
\]

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 $G(i) := \text{Graph induced by the vertices in the subtree at } i$

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

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

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 $X_i = X_j \setminus \{v\}$

$R(i, S) = \max \{ \}$
Let \( G(i) \) := Graph induced by the vertices in the subtree at \( i \)

For bag \( i \) and \( S \subseteq X_i \), let:
\[ R(i, S) := \text{maximum weight of an indep. set } I \text{ in } G(i) \]

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:

\[ R(i, S) := \text{maximum weight of an indep. set } I \text{ in } G(i) \]
\[ \text{with } I \cap X_i = S \]

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 $G(i) := \text{Graph induced by the vertices in the subtree at } i$

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

$R(i, S) := \text{maximum weight of an indep. set } I \text{ in } G(i)$

with $I \cap X_i = S$

**Algo.:** Compute $R(i, S)$ for all $i$ and corresponding $S$
Indep. Set on Nice Tree Decomp.

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

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

$R(i, S) := \text{maximum weight of an indep. set } I \text{ in } G(i)$

with $I \cap X_i = S$

\textbf{Algo.:} Compute $R(i, S)$ for all $i$ and corresponding $S$

\textbf{Runtime:} ?
Indep.Set on Nice Tree Decomp.

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

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

$R(i, S) := \text{maximum weight of an indep. set } I \text{ in } G(i)$

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.