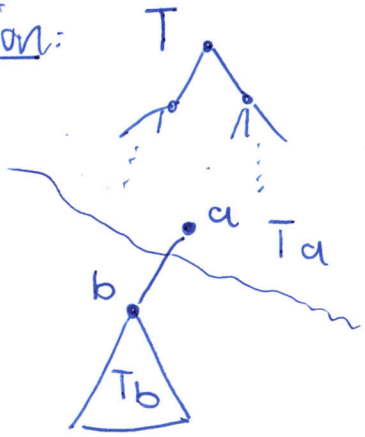
Illustration:



The Lemma separates the vertices in the bags of T_a from those vertices in the bags of T_b.

The separator is $X_a \cap X_b$ which is bounded by the width of the tree decomposition.

Like for paths, we define nice decompositions:

Definition:
Let $(T, \{X_t\}_{t \in V(T)})$ be a tree decomposition of graph G.
Let r be the root of T.
We call $(T, \{X_t\}_{t \in V(T)})$ a nice tree decomposition if

- $X_r = \emptyset$ and $X_l = \emptyset$ for each leaf l of T,
- each non-leaf node of T is of one of the following types:
  - Introduce node: A node t with exactly one child t' such that $X_t = X_{t'} \cup \{v\}$ for $v \notin X_{t'}$.
    We say v gets introduced at t.
  - Forget node: A node t with exactly one child t' such that $X_t = X_{t'} \setminus \{w\}$ for $w \in X_{t'}$.
    We say w is forgotten at t.
  - Join node: A node t with two children t_1, t_2 such that $X_t = X_{t_1} = X_{t_2}$.

Lemma: Given a tree decomposition of width $\leq k$, one can construct a nice tree decomposition of width $\leq k$ with $\mathcal{O}(k \cdot |V(G)|)$ nodes in time $\mathcal{O}(k^2 \cdot \max\{|V(T)|, |V(G)|\})$.

note: • In general, a tree decomposition is not known in advance.
• We assume that a decomposition is part of the input of the problem that we solve.
• There are fast algorithms that compute tree decompositions of approximately optimal width.

32.3 Dynamic Programming on Graphs of Bounded Treewidth:

We consider the following problem:

WEIGHTED INDEPENDENT SET (WIS)

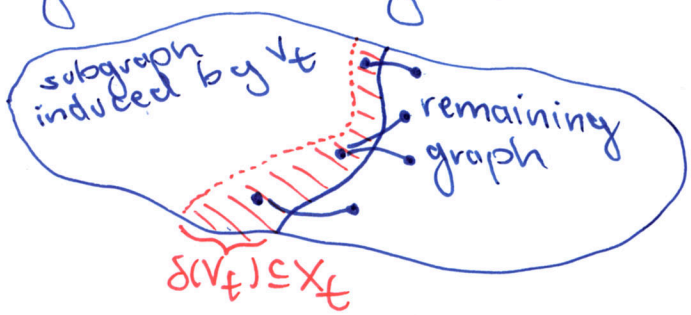
Given: A graph G with a weight function $w: V(G) \to \mathbb{R}_{\geq 0}$, and a tree decomposition of width k.

Parameter: k.

Quest: Find an independent set in G of max. weight.

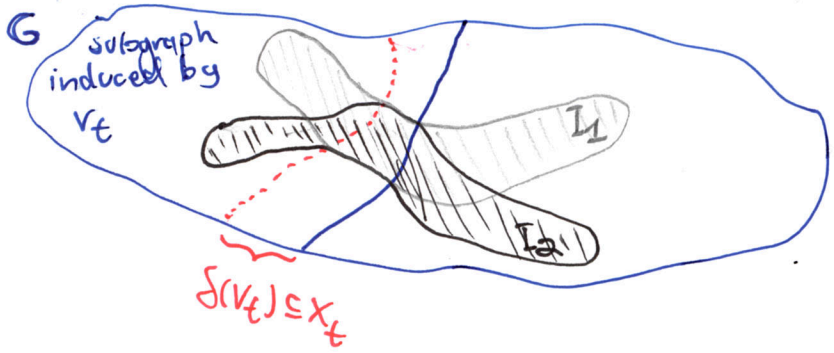
• Let $(T, \{X_t\}_{t \in V(T)})$ be the given tree decomposition. By the above lemma we can assume that it is a nice tree decomposition.
Let r be the root of T.
For $t \in V(T)$, set T_t = subtree rooted in t and $V_t = \bigcup_{v \in V(T_t)} X_v \subseteq V(G)$.

Observation:
By the separation Lemma we get that for each node $t \in V(T)$: $\delta(V_t) \subseteq X_t$.
This means that the subgraph induced by V_t can only communicate with the rest of the graph through the bag X_t.



Since $|X_t| \leq k+1$, we can, in some sense, controle the communication between V_t and the other vertices.

Let I_1, I_2 be two independent sets of G such that $I_1 \cap X_t = I_2 \cap X_t$ and $w(I_1 \cap V_t) > w(I_2 \cap V_t)$.



Then we claim that I_2 is not a maximal solution.
Indeed, we can construct an independent set $\hat{I}_2$ with $w(\hat{I}_2) > w(I_2)$.

Set $\hat{I}_2 := I_2 \setminus (I_2 \cap V_t) \cup (I_1 \cap V_t)$.

Then $w(\hat{I}_2) = w(I_2) - w(I_2 \cap V_t) + w(I_1 \cap V_t) > w(I_2)$.
($w(I_1 \cap V_t) - w(I_2 \cap V_t) > 0$)

and $\hat{I}_2$ is still independent since $I_1 \cap X_t = I_2 \cap X_t$ and there is no edge from $V_t \setminus X_t$ to the remaining graph.

Hence, all maximal solutions I with $I \cap X_t = S$, for a fixed set S, have the same weight on V_t.

Idea:
- Compute for each node t and set $S \subseteq X_t$ a maximal independent set $\hat{S}$ such that $S \subseteq \hat{S} \subseteq V_t$ and $\hat{S} \cap X_t = S$.
  → Then we want the result for $t = r$ and $S = \emptyset$.
- In any independent set $I \subseteq V(G)$ with $I \cap X_t = S$, we can then replace $I \cap V_t$ by $\hat{S}$. This will only increase the weight and preserves independence.

Dynamic Programming Algorithm:

For every node t and set $S \subseteq X_t$, define

$$c[t,S] = \max\left\{ w(\hat{S}) \;\middle|\; \begin{array}{l} S \subseteq \hat{S} \subseteq V_t, \\ \hat{S} \text{ independent} \end{array} \; \hat{S} \cap X_t = S \right\}.$$

Note:
- $c[t,S] = -\infty$ if such an $\hat{S}$ does not exist. This happens if and only if S is not independent.
- $c[r, \emptyset]$ is the value we are looking for.

We compute $c[\cdot,\cdot]$ bottom-up.

1) Leaf node: If t is a leaf, the only value we have to compute is $c[t,\emptyset] = 0$.

2) Introduce node: Let t be an introduce node with child t' such that $X_t = X_{t'} \cup \{v\}$, for a $v \notin X_{t'}$. Let $S \subseteq X_t$.
   - If S is not independent, set $c[t,S] = -\infty$.
   - If S is independent, we use the following recursion:

$$c[t,S] = \begin{cases} c[t',S], & \text{if } v \notin S \quad (1) \\ c[t',S\setminus\{v\}] + w(v), & \text{if } v \in S \quad (2) \end{cases}$$

Proof of the recursion:

(1): The sets $\hat{S}$ considered in the definitions of $c[t,S]$ and $c[t',S]$ are the same.

(2): „$\leq$" Let $\hat{S}$ be a set which attains the maximum in $c[t,S]$. Then $w(\hat{S}) = c[t,S]$, $S \subseteq \hat{S} \subseteq V_t$, $\hat{S}$ is independent, and $\hat{S} \cap X_t = S$.
Then $\hat{S}\setminus\{v\}$ is also independent, $S\setminus\{v\} \subseteq \hat{S}\setminus\{v\} \subseteq V_{t'}$, and $\hat{S}\setminus\{v\} \cap X_{t'} = S\setminus\{v\}$.

$$\Rightarrow c[t', S\setminus\{v\}] \geq w(\hat{S}\setminus\{v\}) = w(\hat{S}) - w(v) = c[t,S] - w(v).$$

$$\Rightarrow c[t,S] \leq c[t', S\setminus\{v\}] + w(v).$$

„≥" Let $\hat{U}$ be an independent set that attains the maximum in $c[t', S \setminus \{v\}]$.
Then $\hat{U} \cup \{v\}$ is a set considered in $c[t, S]$.
$\Rightarrow c[t,S] \geq w(\hat{U} \cup \{v\}) = w(\hat{U}) + w(v) = c[t', S \setminus \{v\}] + w(v)$. ∎

3) Forget node: Let t be a forget node with child t' such that $X_t = X_{t'} \setminus \{w\}$, for all $w \in X_{t'}$.
Let $S \subseteq X_t$.
- If S is not independent, set $c[t,S] = -\infty$.
- If S is independent, we use the recursion
$$c[t,S] = \max\{c[t',S], c[t', S \cup \{w\}]\}.$$

The proof is similar to the above recursion.

4) Join node: Let t be a join node with children t_1, t_2 such that $X_t = X_{t_1} = X_{t_2}$.
Let $S \subseteq X_t$.
- If S is not independent, set $c[t,S] = -\infty$.
- If S is independent,
$$c[t,S] = c[t_1,S] + c[t_2,S] - w(S).$$

Since we have a tree decomposition of width k, we have for each bag: $|X_{t'}| \leq k+1$.
Hence, at any node t, we have to compute $2^{|X_{t'}|} \leq 2^{k+1}$ values for $c[\cdot,\cdot]$.
Any of these computations take poly time and we have $\mathcal{O}(k \cdot |V(G)|)$ many nodes.

<u>Theorem:</u> WIS can be solved in time $2^k \cdot n^{\mathcal{O}(1)}$ if a tree decomposition of width k is given.

Since a graph has a vertex cover of size at most ℓ if and only if it has an independent set of size $\geq n-\ell$, we get:

Corollary:
VERTEX COVER can be solved in time $2^k \cdot n^{O(1)}$ if a tree decomposition of width k is given.

Proof:
For a given graph G, define the weight function $w: V(G) \to \mathbb{R}_{\geq 0}$, $v \mapsto 1$.
Then run the algorithm for WIS. If the result is $\geq n-\ell$, return yes. Otherwise, return no. ∎