32. Treewidth:

Goal: • Find new algorithms for graph problems.
- Capture structure of a graph in tree-like decomposition
- Measure "how well" the decomposition is choosen and "how far" away the graph is from a tree.
  - → Technically, introduce treewidth.
- Do dynamic programming on the tree-like decomposition and use treewidth as new parameter.
  - → Problems on graphs of bounded treewidth can be solved efficiently.

Before we start with trees, we decompose graphs into paths.

32.1 Path decomposition:

Definition: Let G be a graph.
A path decomposition of G is a sequence $P=(X_1,\dots,X_r)$ ~~of~~ bags, where $X_i \subseteq V(G)$ and the following holds:

(P1) $\bigcup_{i=1}^{r} X_i = V(G)$,

(P2) For each edge $uv \in E(G)$ there is an $l \in [1..r]$: $u,v \in X_l$,

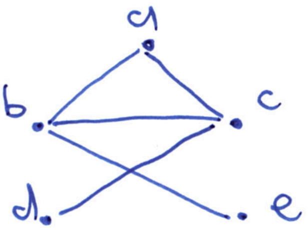
(P3) For each $v \in V(G)$, if $v \in X_i \cap X_k$ with $i \le k$, then $v \in X_j$ for any ~~$\le$~~ $i \le j \le k$.

Intuition: $X_1 - X_2 - \dots - \overbrace{X_i - \dots - X_k} - \dots - X_r$

if $v \in X_i$ and $v \in X_k$
→ v is in all intermediary bags.

In other words: The set $\{j \mid v \in X_j\}$ forms an interval in $[1..r]$.

Example:
Let G be



Then P: $abc - bce - bcd$ is a path decomposition of G.

Definition:
- The width of a path decomposition $P=(X_1,\dots,X_r)$ is defined to be $\max\limits_{1\le i\le r} |X_i| - 1$.
- The pathwidth, denoted by $pw(G)$, is the minimal possible width of a path decomposition of G.

Example:

- In the above example, width$(P) = 2$. In fact, $pw(G) = 2$.
- Paths have pathwidth 1: Let G be a path:

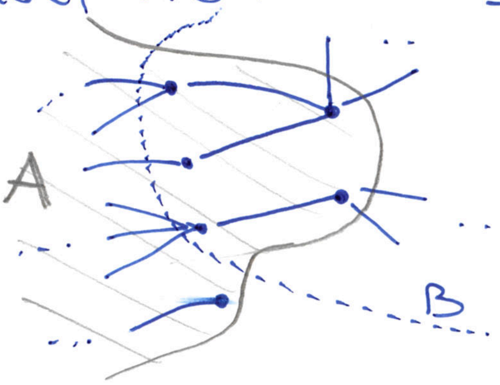
Then ab — bc — cd is a path decomposition of width 1.

A path decomposition defines a sequence of separators in the graph. These will be handy for dynamic programming as they "control" the edges between two parts of a graph.

Definition: Let G be a graph.

- Let $A, B \subseteq V(G)$. We say (A, B) is a <u>separation</u> of G if $A \cup B = V(G)$ and there is no edge between $A \setminus B$ and $B \setminus A$.
- In this case, $A \cap B$ is called <u>separator</u> of (A, B).

Illustration:



Each path between A and B must contain a vertex of $A \cap B$

- For $A \subseteq V(G)$ we define the <u>border</u> $\delta(A)$ of A to be the set of vertices of A that have a neighbor in $V(G) \setminus A$

Fact:

a) The tuple $(A, V(G) \setminus A \cup \delta(A))$ is a separation with separator $\delta(A)$.

b) (A, B) is a separation iff $A \cup B = V(G)$ and $\delta(A) \subseteq A \cap B$.

<u>Lemma:</u>

Let $P=(X_1,\dots,X_r)$ be a path decomposition of G.

For each $j \in [1..r-1]$ we have $\delta(\bigcup_{i=1}^{j} X_i) \subseteq X_j \cap X_{j+1}$.

In other words: $(\bigcup_{i=1}^{j} X_i, \bigcup_{i=j+1}^{r} X_i)$ is a separation of G with separator $X_j \cap X_{j+1}$.

<u>Proof:</u> Let j be fixed and $(A,B) = (\bigcup_{i=1}^{j} X_i, \bigcup_{i=j+1}^{r} X_i)$.

We show that $\delta(A) \subseteq X_j \cap X_{j+1}$:

Assume there is a $u \in \delta(A)$: $u \notin X_j \cap X_{j+1}$.

This means there is an edge $uv \in E(G)$: $u \in A, v \notin A, u \notin X_j \cap X_{j+1}$.

Let i be the largest index with $u \in X_i$.

Let k be the smallest index with $v \in X_k$.

Since $u \in A$ and $u \notin X_j \cap X_{j+1}$, we get: $i \leq j$.

Since $v \notin A$, we get $k \geq j+1$.

Hence, $i < k$.

By (P2) there is a bag X_ℓ with $u, v \in X_\ell$.

Then $\ell \leq i < k \leq \ell$ which is a contradiction.

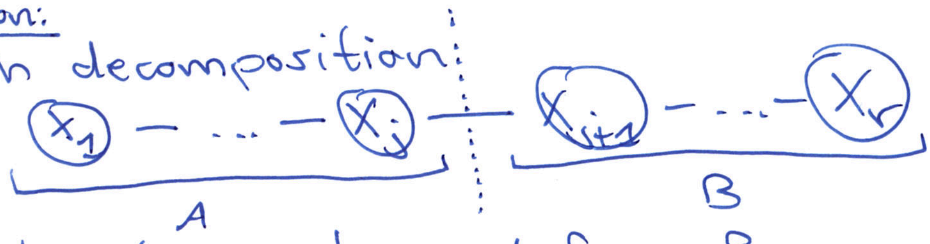
Hence, $\delta(A) \subseteq X_j \cap X_{j+1}$.

By (P3) we get that $X_j \cap X_{j+1} = A \cap B$.

Then the claim follows by Fact b). ∎

<u>Illustration:</u>

A path decomposition:



separates the vertices A from B.

It controls the size of the separator.

For dynamic programming it is particularly useful to restrict to path decompositions that have an easy bag structure:

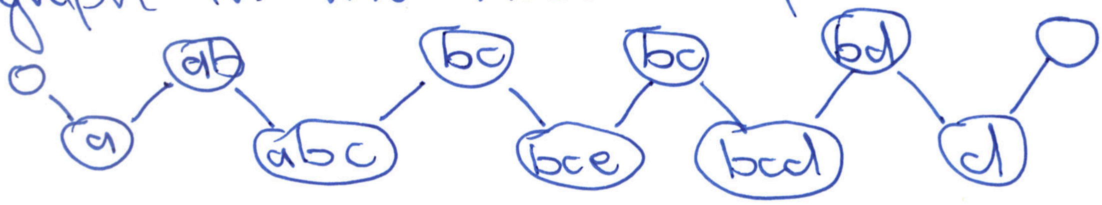
Definition:

A path decomposition $P = (X_1, \dots, X_r)$ of a graph G is *nice* if:

- $X_1 = X_r = \emptyset$,
- for each $i \in \{1, 2, \dots, r-1\}$ there is either a vertex $v \notin X_i$ such that $X_{i+1} = X_i \cup \{v\}$, or a vertex $w \in X_i$ such that $X_{i+1} = X_i \setminus \{w\}$.

Example:

We construct a nice path decomposition for the graph in the first example:



Notation:

- Bags of the form $X_{i+1} = X_i \cup \{v\}$ are *introduce bags*.
- Bags of the form $X_{i+1} = X_i \setminus \{w\}$ are *forget bags*.

Lemma:

Given a path decomposition $P = (X_1, \dots, X_r)$ of width $\leq p$, one can compute a nice path decomposition of width $\leq p$ in time $\mathcal{O}(p^2 \cdot \max(r, |V(G)|))$.

32.2 Tree decomposition:

Tree decompositions are a generalization of path decompositions.
In fact, the tree decompositions that are paths are exactly the path decompositions.

Definition:

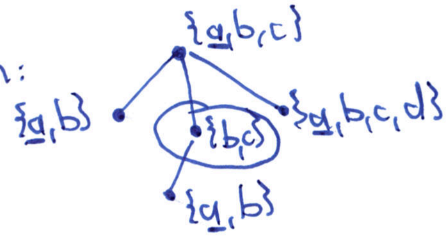
- A <u>tree decomposition</u> of a graph G is a pair $(T, \{X_t\}_{t \in V(T)})$ where T is a tree the nodes of which are assigned <u>bags</u> $X_t \subseteq V(G)$ such that the following holds:

  (T1) $\bigcup_{t \in V(T)} X_t = V(G)$,

  (T2) For each edge $uv \in E(G)$, there is a node t of T such that $u, v \in X_t$.

  (T3) For each $u \in V(G)$, the set $T_u = \{t \in V(T) \mid u \in X_t\}$ is a connected subtree of T.

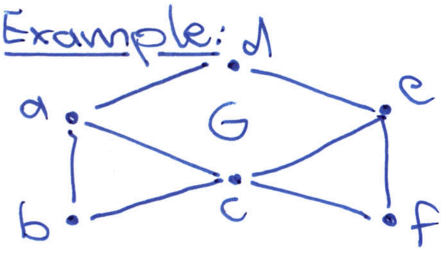
Illustration:



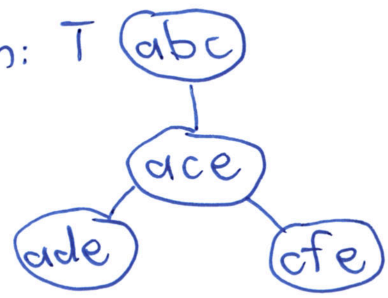
This tree would violate (T3) in vertex a, T_a is not connected.

- The <u>width</u> of a tree decomposition is $\max_{t \in V(T)} |X_t| - 1$.
- The <u>treewidth</u> of G, denoted by $tw(G)$, is the minimal possible width of a tree decomposition.

<u>Example:</u>



Then: T



is a tree decomp. of width 2. In fact, $tw(G) = 2$.

Also a tree decomposition induces separations of the graph:

<u>Lemma:</u> Let $(T, \{X_t\}_{t \in V(T)})$ be a tree decomposition of a graph G and let ab be an edge of T. The forest $T \setminus \{ab\}$ obtained from T by deleting ab consists of two connected components T_a (containing a) and T_b (containing b).

Let $A = \bigcup_{t \in V(T_a)} X_t$ and $B = \bigcup_{t \in V(T_b)} X_t$.

Then $\delta(A), \delta(B) \subseteq X_a \cap X_b$ or equivalently:

(A, B) is a separation of G with separator $X_a \cap X_b$.

Proof: As for path decompositions. ■