Tree Width

Q) Why study Tree Width?

Ans) Many imp. graph algo's are NP-hard. Trees are better in that respect as they are "well-structured". We can therefore think of decomposing a graph into trees and nodes of that tree are sets of graph vertices that satisfy a particular property. The "tree-ness" can be measured by the minimum size of the nodes of its tree-decomposition, which we call its TREE-WIDTH. We can therefore restrict classes of problem-instances in their tree-width, which proves to be beneficial to the complexity of the given problem, while not restricting so much so that the resulting algorithm loses its usefulness in the real world.

What is the paper about? Linear time algorithm for computing treewidth, which is based on reductions to smaller graphs using contractions along maximum matchings and deletions of suitable vertices.

Tree-Decompositions ⇒

Definition → $G = (V, E)$ is a graph. A tree-decomposition of G is a pair $T = (T, X)$, where $T = (I, F)$ is a tree and $X = (X_i)_{i \in I} \subseteq P(V)$ is a family of subsets X_i of V (also called bags), such that the following property holds →

i) $\bigcup_{i \in I} X_i = V$ ii) $\forall \{v, w\} \in E$, $\exists i \in I$ with $\{v, w\} \subseteq X_i$.

iii) $\forall v \in V$, $X^{-1}(v) := \{i \in I \mid v \in X_i\}$ is connected in T.

($P(V)$ is power set of the set of vertices V)

For any two nodes i & $j \in I$, last property is equivalent to $X_i \cap X_j \subseteq X_k$ $\forall k$ on the path from i to j in T.

The vertices of G correspond to subtrees in T which intersect

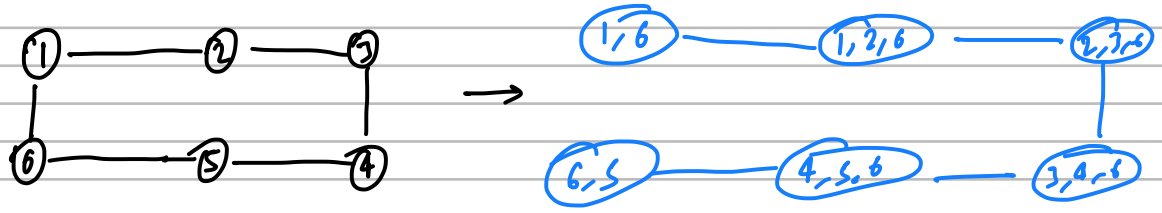
if (but not only if) the corresponding vertices are adjacent in G_T. The structure of G_T is therefore "preserved" in tree decomposition.
→ Size of bags X_i are indication of the structural complexity of G_T.

Width → For any tree-decomposition $\mathcal{T} = (T, X = (X_i)_{i \in I})$ of a graph G_T, the **width of $\mathcal{T}$** is $\boxed{\max\{|X_i| \mid i \in I\} - 1.}$

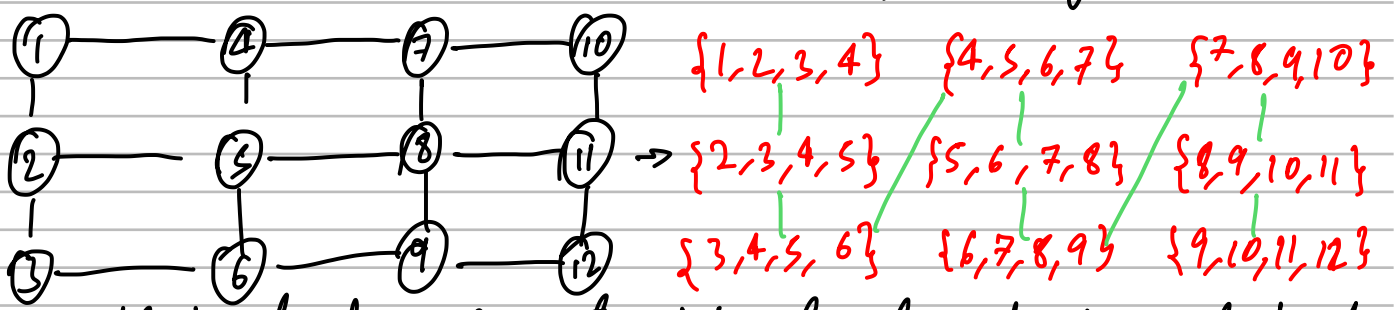
treewidth of a graph G_T is defined as
$tw(G_T) := \min(\{\text{width}(T) \mid T \text{ is tree-decomposition of } G_T\})$

→ **Some Examples** Treewidth of a tree is 1, as you can always find a tree decomposition of width 1 which is isomorphic to the tree T.
- Graphs with treewidth 1 are forests, i.e. acyclic graphs.

Treewidth of a cycle is 2. From last example, treewidth cannot be less than 2. We can always construct a $\mathcal{T}$ for which width = 2.



Treewidth of a $m \times n$ grid is $\min(m,n)$. The (m,n) grid $G_{m,n}$ with m rows and n columns has the following $\mathcal{T} = (T, X)$ →



This example hints at a connection b/w tree decomposition and seperators in G_T.

Terminology ⇒ For $\mathcal{T} = (T = (I, F), X)$ of a graph G_T and a subset $J \subseteq I$, $X(J) := \bigcup_{j \in J} X_j$

Theorem: $G_T = (V, E)$ be a graph with a tree-decomposition $\mathcal{T} = (T, X)$ and $\{i, j\}$ is edge in T. Deleting $\{i, j\}$ from T splits it into T_i & T_j which are two disjoint subtrees. The set $S = X_i \cap X_j$ separates $X(T_i)$ from $X(T_j)$ in graph G_T. Any path from $X(T_i)$ to $X(T_j)$ leads through S. (Note: $X(T_i) \cap X(T_j) = X_i \cap X_j = S$)

Corollary 1: $\mathcal{T} = (T, X)$ of graph $G_T = (V, E)$ and $C \subseteq V$ is a clique in G_T. $\exists\, i \in I$ with $C \subseteq X_i$. In particular if G_T has a clique of size 'k', then $tw(G_T) \geq k-1$

Proof ⇒ Use induction & contradiction. Induct on size of the clique.

Corollary 2: Let $\mathcal{T} = (T, X)$ be tree-decomposition of $G_T = (V, E)$. If $W_1, W_2 \subseteq V$ induce a complete bipartite subgraph in G_T, i.e. $W_1 \times W_2 \subseteq E$, then there exists an $i \in I$ with either $W_1 \subseteq X_i$ or $W_2 \subseteq X_i$.

Let this be true $\forall\, |C| \leq n-1$. Let $v \in C$ be a vertex in G_T. $C' = C / \{v\}$ ⇒ $|C'| = n-1$ ⇒ $\exists\, i \in I$ s.t. $C' \subseteq X_i$. Let $P = \{i \mid C' \subseteq X_i \;\; i \in I\}$. P will form a connected component in T. Let $\{i, j\}$ be edge connecting $T/\{P\}$ & P. ⇒ $T_i = P$ & $T_j = T/\{P\}$
⇒ $S = X_i \cap X_j$ will seperate $X(T_i)$ & $X(T_j)$ in G_T.
$v \in X(T_j)$ and $v \notin X(T_i)$. For any w in C, $\{v, w\} \in E$ and $\{v, w\}$ joins $X(T_i)$ & $X(T_j)$ ⇒ Either $v \in S$, or $w \in S$.
If $v \in S$ ⇒ $v \in X(T_i)$ ⟶⟵
If $w \in S$ & $v \notin S$ $\forall\, w \in C'$ ⇒ $C' \in S = X_i \cap X_j \subseteq X(X^{-1}(v))$
⇒ $T(C') \cap X(X^{-1}(v)) \neq \phi$.

Grome-Characterization of Treewidth →

Cops-and-Robber game →

- k cops, every cop either on a vertex or in a helicopter.
- Cop can either enter a helicopter, and cops in helicopter can either

choose to go to any arbitrary vertex of the graph, leave the helicopter and occupy that vertex.

- Robber can move arbitrarily fast along edges of the graph (he can elude cops before they land in particular).
- Robber cannot pass through nodes occupied by a cop.
- Goal is for cops to corner robber s.t. a cop can be placed on his vertex without him being able to elude.
- Cops are always aware of robber's position.

A state of this game is (C_i, R_i) where $|C_i| \leq k$ are vertices with cop on them. R_i is a connected component in $G_T / \{C_i\}$ (representing location of robber, whose exact posn in the component is not relevant).

$(C_0, R_0) = (\emptyset, V)$. Correct transitions are s.t. $C_i \subseteq C_{i+1}$ or $C_{i+1} \subseteq C_i$ and accordingly $R_{i+1} \subseteq R_i$ or $R_i \subseteq R_{i+1}$.

Cops win if $R_i \subseteq C_{i+1}$ or $R_{i+1} = \emptyset$. We say graph can be searched by k cops if they have winning strategy for the game.

Theorem: For a graph G_T, following two are equivalent → $(k \in \mathbb{N})$.

- $tw(G_T) \leq k-1$
- G_T can be searched by k cops.

Computing Treewidth →

Finding the treewidth of a graph is NP-complete, it becomes tractable if we fix 'k'. The algorithm is exponential in 'k'.

Theorem: $\forall k \in \mathbb{N}$, there exists a linear time algorithm that tests whether a graph $G_T = (V, E)$ has a treewidth at most 'k', and if so, outputs a tree decomposition of G_T with width at most 'k'.

The main idea of the algo is to recursively reduce the problem to smaller graphs.

Look at these two smaller algo's → $(G_T = (V, E)$ & $k \in \mathbb{N})$

Graphs with bounded tw are generally 'sparse' i.e. no of edges is linearly bounded by no of vertices.

Lemma: If $tw(G_T) \leq k$, then $|E| \leq k|V| - \frac{k(k+1)}{2}$