CS787: Advanced Algorithms	**Scribe:** Bowen Hu
Lecture 3: FPT; Tree Decomposition	**Date:** Jan 29, 2019

Outlines for the following lecture

- Definition of Tree Decomposition and TreeWidth
- Dynamic Programming over Tree Decomposition in time of $O(2^{\text{TreeWidth}} \cdot poly(n))$.
- Properties of Tree-Width
- FPT algorithm to find a tree decomposition (Next Lecture)

Recall: A Fixed Parameter Tractable problem FPT is a problem who has an algorithm that runs in $O(f(k) \cdot poly(n))$ algorithm, where k is parameter and n is the size of input.

Motivation: In the following lecture, we are trying to define a parameter that represent the "tree-likeness" of a graph. And it should be small when the graph is a tree, and large when the graph is "far" from tree.

3.1 Tree Decomposition

Definition 3.1.1 *Tree decomposition of graph $G = (V, E)$ is a tree T with vertices X_1, X_2, $\cdots$, X_k such that:*

1. *each X_i is a subset of vertices V of G. ($X_i \subset V,\ \forall i = 1, \ldots, k$)*
2. $\bigcup_i X_i = V$
3. *for every edges $(u, v) \in E$, there exists X_i containing both u, v.*
4. *for every vertices $v \in V$, the collection of X_i's containing v, forms a connected subtree.*

Example: Tree decomposition for a tree.

Consider graph G =(V,E) where $V = \{A, B, C, D, E\}$ and $E = \{\{A, C\}, \{A, B\}, \{C, D\}, \{C, E\}\}$ as Figure 3.1.1. One example of its tree decomposition can be considered as: $T = \{X_1, X_2, X_3, X_4\}$ where $X_1 = \{A, C\}$, $X_2 = \{C, D\}$, $X_3 = \{C, E\}$, $X_1 = \{A, B\}$

Proof: Property 1 and 2 are obvious from definition. $E = \{AC, CD, CE, AB\}$ corresponds to $\{X_1, X_2, X_3, X_4\}$ in T. Thus, we have property 3. We can enumerate all vertices in G to prove property 4. For $A \in V$, we have $X_1, X_4 \subset V(T)$ containing A, and they forms a subtree. Similarly we can prove for other vertices in V. ∎

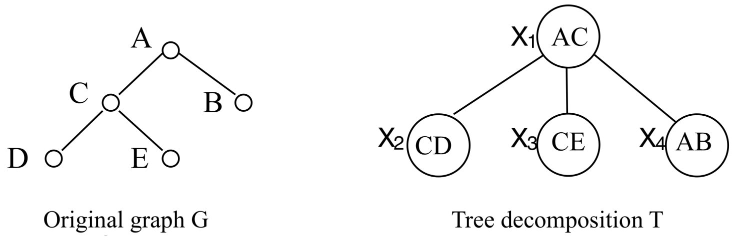


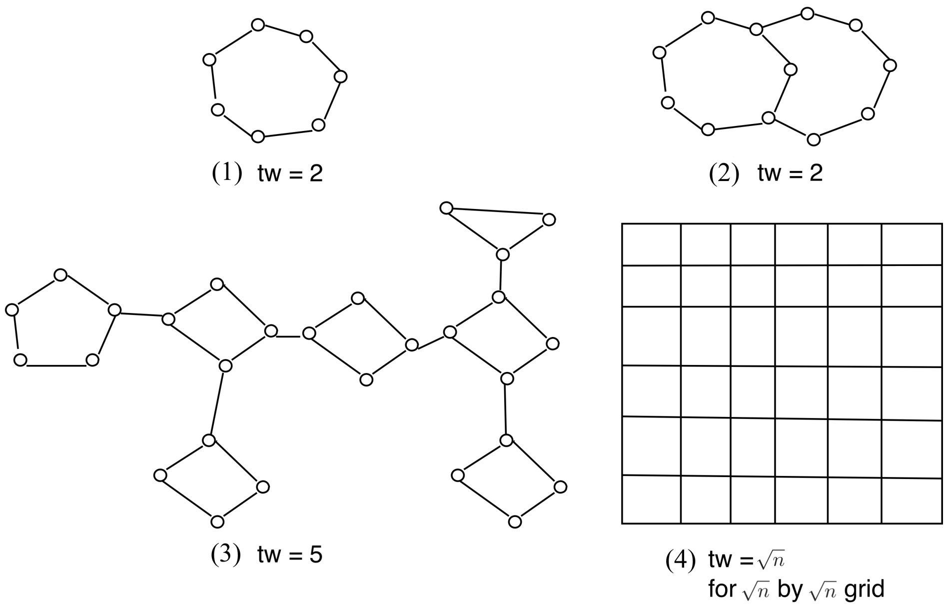
Figure 3.1.1: Example of tree decomposition

Definition 3.1.2

$$TreeWidth = \min_{all\ tree\ decomposition\ T\ of\ G} \left(\max_{vertices\ X_i\ of\ T} |X_i| - 1 \right)$$

Treewidth is a measure of the tree-likeness of a graph (Introduced by Robertson and Seymour). The width of a tree decomposition is the size of largest set X_i minus 1. $tw(T) = \max_i |X_i| - 1$. And the tree-width $tw(G)$ of a graph G is the minimum width among all possible tree decompositions of G. Note that a graph can have several tree decomposition with minimum tree-width.

Example:



3.2 Dynamic Programming over Tree Decomposition

Example: VertexCover

Consider the VertexCover given graph G, tree decomposition T of G, tree-width k of T.

Fact 3.2.1 *Given any tree decomposition T of G and any vertex X_i of T, removing X_i from G results in connected components (c.c.) C_1, C_2, $\cdots$ such that each C_i intersects with at most one c.c. of $T \setminus \{X_i\}$.*

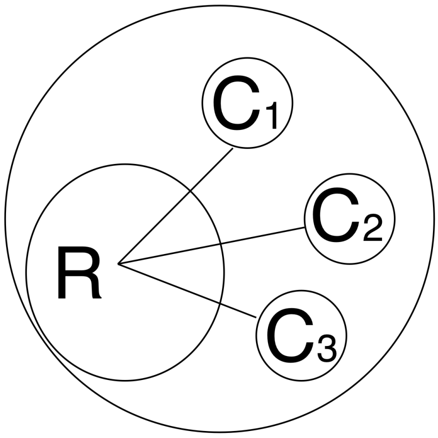


Figure 3.2.2: Graph G with four connected components: R, C_1, C_2 and C_3

Recall: Find VertexCover over a tree using dynamic programming as follows. Split it to two sub-problems as the VertexCover either contains the root or the VertexCover does not contain root. Solve recursively for two cases and return the smallest VertexCover.

Now consider VertexCover problem for graph G in Figure 3.2.2. G's VertexCover should contain a subset of R. For every subset $R' \subset R$, and for every c.c. C_i in $G \setminus R$, solve recursively for the smallest VertexCover of C_i with edges incident on R' covered but those on $R \setminus R'$ not covered. That gives us the optimal solution corresponding to this choice R'. Return minimum VertexCover over all $R' \subset R$. ■

3.3 Some Properties of Tree Width

Lemma 3.3.1 *If graph G has tree-width p, then G contains a vertex of degree at most p.*

Proof: Consider a **leaf** vertex X_i in the tree decomposition T of G and X_j is its only neighbor. Observe $X_i \not\subset X_j$, otherwise we can just remove X_i from T. So there exists vertex $v \in X_i \setminus X_j$. Thus, v is not in any other node of T. In that case, all neighbors of v in G are in X_i. Since $|X_i| \leq tw(G) + 1 = p + 1$, v have at most p neighbors.

Proposition 3.3.2 *Empty graphs have tree-width = 0.*

Proof: Let $G = (V, E)$ be an empty graph. Consider tree decomposition $T = V$ and we get tree width 0 for such tree decomposition.

Proposition 3.3.3 *A graph is a forest iff tree-width = 1.*

Proof: $\Leftarrow$ Consider an arbitrary tree G. We build a tree decomposition for it with tree-width 1 as follows. Pick a root r in G, and orient the edges away from r. Each oriented edge (u, v) in G makes one node labeled u, v in T . (The label u, v is equivalent to the label v,u, but we shall keep the orientation of the edge for ease of description.) Two nodes $\{u_1, v_1\}$ and $\{u_2, v_2\}$ are adjacent in T iff $v_1 = u_2$. It can be seen that this gives a tree decomposition of G. A forest is a disjoin union of trees and the tw of the disjoint components are the max of the tw of each component.
$\Rightarrow$ Conversely, assume that G has tree-width 1. Then it has a vertex of degree at most 1 (by Lemma 3.3.1). Removing this vertex from G, the tree-width cannot increase. Hence we can recursively remove vertices of degree at most 1, until we exhaust all vertices in G. This implies that G is a forest.

Proposition 3.3.4 *A graph of* n *vertices has tree-width* $n-1$ *iff it's* K_n

Proof: $\Rightarrow$ Suppose a graph of n vertices has tree-width $n - 1$. If the graph is not K_n, then some edge (u, v) is missing. Consider the tree decomposition V\{u} and V\{v}. Then $tw = n - 2$. Contradiction.
$\Leftarrow$ Conversely, if G is connected and has a tree decomposition with tree-width below $n - 1$, then it has a vertex of degree at most $n - 2$ (by Lemma 3.3.1), and cannot be K_n. Contradiction.

Proposition 3.3.5 *If a graph has tree-width k, then it has a balanced separator of size* $\leq k + 1$*, where a balanced separator* S *is a set of vertices such taht* $G \setminus S$ *has more than 2 c.c. and each c.c. has less than* $\frac{n}{2}$ *vertices.*

Proof: Consider a graph of tree-width k. Then it has a tree decomposition where $|X_i| = k + 1$ for some i. Let V_i be the set of nodes in subtree rooted at X_i, and $m(X_i) = |V_i|$. Pick X_i where $m(x_i) \geq \frac{n}{2}$ and $\forall$ children X_j of X_i, $m(X_j) < \frac{n}{2}$. Such X_i can be found as following: Since $m(root) = n > n/2$, start at root and move to a child with $m \geq n/2$ if there exists one. Return the node when none of its children have $m \geq n/2$.

Now we will prove X_i is a balanced separator. Pick an arbitrary root and orient edges away from it. If we remove X_i from G, the c.c.'s in G are either subsets of V_j for some child X_j of X_i or subsets of $V \setminus V_i$. Since $|V_j| \leq m(X_j) < n/2$ and $|V \setminus V_i| < n - m(X_i) \leq n/2$, we have X_i a balanced separator.