CS787: Advanced Algorithms	**Scribe:** Silvia Di Gregorio
Lecture 4: Tree Width (continued)	**Date:** February 5, 2019

Let us start by recalling the following definitions.

Definition 4.0.1 *A* tree decomposition *of a graph $G = (V, E)$ is a tree T with vertices $X_1, X_2, \dots, X_k$ such that*

1. *$X_i \subseteq V$ for every $i = 1, \dots, k$;*
2. *$\bigcup_{i=1}^{k} X_i = V$;*
3. *for every edge $(u, v) \in E$, there exists i such that $\{u, v\} \subseteq X_i$;*
4. *for every $u \in V$, the set of all X_i's that contain u is connected.*

Observe that a tree decomposition of a graph could look completely different from the original graph. The purpose of the tree decomposition is to capture the presence of separators.

Definition 4.0.2 *The* tree width *of a graph G is defined as follows:*

$$\mathit{TreeWidth}(G) = \min_{\substack{\textit{all tree decomp} \\ T \textit{ of } G}} \max_{X_i \in T} (|X_i| - 1).$$

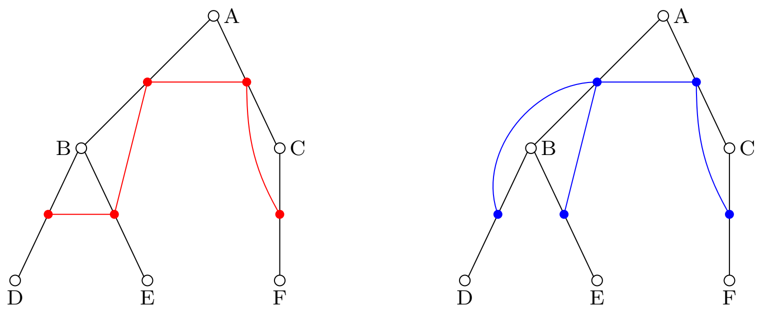
4.1 Examples of Tree Decomposition

In this section we provide different examples of Tree Decompositions.

4.1.1 Trees

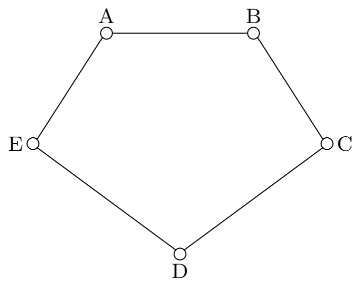
Let $G = (V, E)$ be a tree. We construct its tree decomposition T by introducing one X_i for every edge of G, which means that every X_i is of the form $X_i = \{u, v\}$, for some $(u, v) \in E$. If two edges of G share an endpoint, then their corresponding nodes in T must be connected.

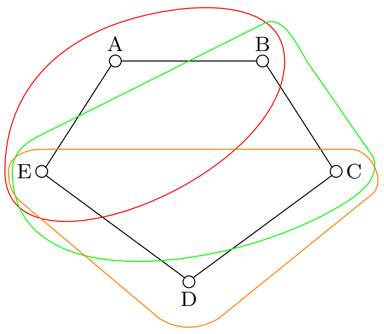
Note that the tree decomposition of G is not unique. In the picture below we represent a tree G with two different tree decompositions. The graph G is in black, whereas the two tree decompositions are in red and blue respectively.

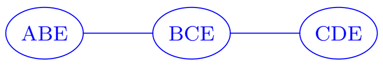


4.1.2 Cycles

Here we focus on the case when G is a cycle. An example is represented in the picture below on the left. The approach used for trees does not work here, in fact it results in another cycle and not a tree. Therefore we need to think about something different. For example, we can select one node of the graph, let it be E, and choose the vertices of the tree decomposition T so that E belongs to all of them. In the second picture, each colored line represents one vertex of the tree decomposition. The resulting tree decomposition is shown on the right.







4.1.3 Series-Parallel Graphs

Definition 4.1.1 *A* series-parallel *graph is a graph that can be constructed by iteratively applying one of the following operations:*

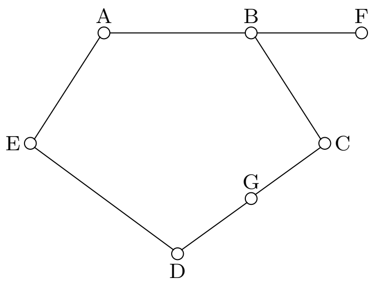
1. *Add a new node and connect it to any existing node;*
2. *Add an edge in parallel to another edge;*
3. *Split an edge into two edges.*

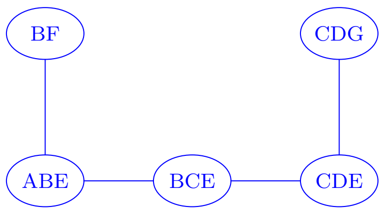
Cycles and sequence of cycles are examples of series-parallel graphs. It can be shown that

Proposition 4.1.2 *A graph G has $TreeWidth(G) = 2$ if and only if G is series-parallel and G is not a tree.*

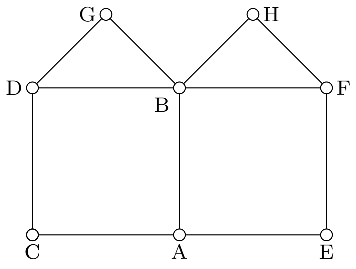
This is because every time one performs an operation of Definition 4.1.1 either the tree decomposition does not change (operation 2), or one node is added to T with size 2 or 3 (operations 1,3).

Let us show this with an example. Consider the cycle of the previous subsection and suppose we add a new node F and we connect it to B. Assume also that we split the edge (C, D) in two. In the following picture, the new graph is shown on the left and its tree decomposition on the right.





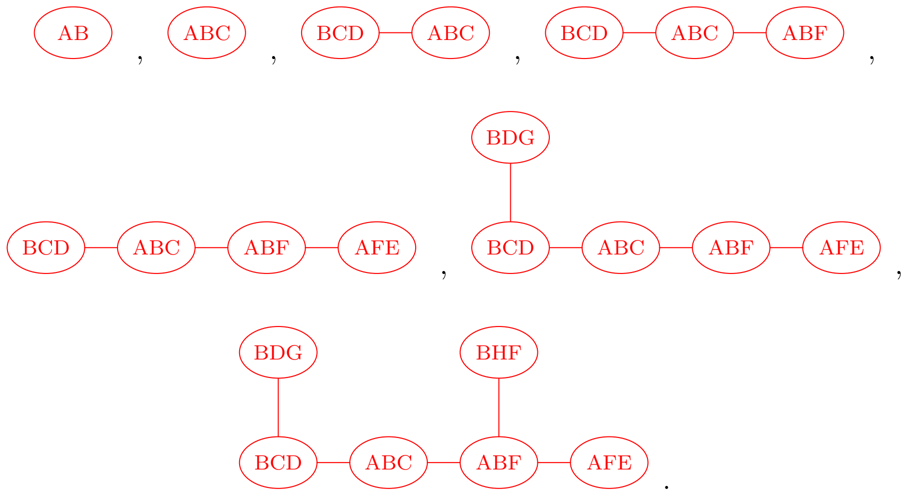
Another example of series-parallel graph is the following.



This graph can be constructed iteratively as follows:

1. Start with a single node A.
2. Add a new node B and connect it to A.
3. Add an edge $(A, B)'$ in parallel to the edge (A, B).
4. Split the edge $(A, B)'$ by adding a new node C.
5. Split the edge (B, C) by adding a new node D.
6. Add an edge $(A, B)''$ in parallel to the edge (A, B).
7. Split the edge $(A, B)''$ by adding a new node F.
8. Split the edge (A, F) by adding a new node E.
9. Add an edge $(B, D)'$ in parallel to the edge (B, D).
10. Split the edge $(B, D)'$ by adding a new node G.
11. Add an edge $(B, F)'$ in parallel to the edge (B, F).
12. Split the edge $(B, F)'$ by adding a new node H.

Following the same order of the operations, we can define also its tree decomposition. This construction is represented here:



4.1.4 Square Grid

We close this section with an example regarding the case when G is a square grid with n nodes. We will see that its tree width is at least $\sqrt{n} - 1$. This shows that planar graphs do not have a small tree width, since the grid is a planar graph.

We need a proposition that was already stated and proved in the last lecture. For ease of exposition, we recall it here.

Proposition 4.1.3 *If G is such that $TreeWidth(G) = p$, then there exists $S \subseteq V$ with $|S| \leq p + 1$ such that every connected component in $G \setminus S$ is of size less than or equal to $\frac{n}{2}$, i.e. there exists a balanced separator S.*

Moreover, it can be shown that

Proposition 4.1.4 *The $\sqrt{n} \times \sqrt{n}$ grid has no balanced separator of size smaller than $\sqrt{n}$.*

Therefore, by Proposition 4.1.3, the tree width of the $\sqrt{n} \times \sqrt{n}$ grid must be at least $\sqrt{n} - 1$. This implies that the tree width of planar graphs is in general not small.

We do not prove Proposition 4.1.4. However, in order to give an idea of the proof, we show a weaker result. In fact, we now prove that the $\sqrt{n} \times \sqrt{n}$ grid has no balanced separator of size less than or equal to $\frac{\sqrt{n}}{2}$.

Proof: Let us assume that there exists a balanced separator S of size less than or equal to $\frac{\sqrt{n}}{2}$. Then S would not intersect at least $\frac{\sqrt{n}}{2}$ columns, since there are $\sqrt{n}$ columns in the grid. Similarly, there exists one row that has empty intersection with S. Therefore, there is a connected component in $G \setminus S$ of size at least $\frac{n}{2}$. Note that this connected component is formed by the $\frac{\sqrt{n}}{2}$ columns and the row whose intersection with S is equal to the emptyset. This contradicts the definition of balanced separator. ∎

4.2 Finding a Tree Decomposition

At this point we would like to devise an algorithm to find a tree decomposition of a generic graph G. Sadly, finding the optimal tree decomposition is NP-hard. However, the following holds.

Proposition 4.2.1 *If G has tree width equal to p, then we can find a tree decomposition of width $\leq 8p$ in time $O(2^{O(p)} poly(n))$.*

We construct the tree decomposition inductively. Assume that we already have a partial tree decomposition on a subset $V' \subseteq V$, of tree width less than or equal to $8p$, such that $G \setminus V'$ has multiple connected components. Let C be one of these connected components of $G \setminus V'$. We want to understand how to attach C to our partial tree decomposition to just one node of the tree.

Throughout this iterative process we wish to preserve two invariants:

1. Let C be any connected component of $G \setminus V'$, then there exists a vertex X in the partial tree decomposition that contains all neighbors of C.
2. Define N to be the set of neighbors of C in X, then $|N| \leq 6p$.

Before explaining how to construct the tree decomposition, let us first observe that if G has tree width equal to p, given $V' \subseteq V$, there exists $S \subseteq V$ with $|S| \leq p+1$, such that every connected component in $G \setminus S$ has less than or equal to $\frac{|V'|}{2}$ nodes of V'. The proof is similar to the one of Proposition 4.1.3. We are now ready to present the iterative procedure to find a tree decomposition of Proposition 4.2.1.

To start, pick $6p$ vertices of G and let them be the first vertex of the tree decomposition. It clearly satisfies the invariants. Let us move on to the inductive step.

We denote by V' the set of nodes for which we already have a tree decomposition. Then let C be a connected component of $G \setminus V'$. By the first invariant property, it follows that there exists one vertex X of the partial tree decomposition that contains the neighbors of C. Let N be this set of neighbors. By the invariant property number 2, $|N| \leq 6p$. We now use the observation in the previous paragraph, and obtain that there exists a set of nodes S of size less than or equal to $p+1$ that every connected component in $G \setminus S$ has less than of equal to $\frac{|N|}{2}$ nodes of N. Now select an arbitrary node v of C and define a new set $X_{new} = S \cup N \cup \{v\}$. We add X_{new} to the partial tree decomposition by connecting it to X.

We need to check that this operation has not affected either the tree width, nor the two invariant properties. Regarding the first issue, observe that $|X_{new}| \leq p+1+6p+1 = 7p+2 \leq 8p$, where the first inequality comes from the definition of X_{new} and the second follows from the implicit assumption that $p \geq 2$. In fact otherwise G would be a forest and we would not need to use this procedure.

The first invariant property still holds, since the connected components deriving from C in $(G \setminus V') \setminus S$ have all their neighbors only in X_{new}. Consider now the second invariant property. Every new connected component in $(G \setminus V') \setminus S$ deriving from C has a number of neighbors which is $\leq p+1+3p+1 \leq 6p$. In fact $p+1$ is the size of S, the term $3p$ follows from the fact that S is a balanced separator of N, while the $+1$ is to take into account the node v of C that was added to X_{new}.

The only problem remaining is how to find the balanced separator S. It can be done by enumeration, trying all the possibilities of different balanced partitions of N. This step is the reason of the term $2^{O(p)}$ in the running time of the algorithm.