MATH 239 — LECTURE 7

Bartosz Antczak *Instructor: Luke Postle* *January 18, 2017*

Review of last lecture

Let X be a subset of $V(G)$. The **cut** induced by X, denoted by $\delta(X)$, is the set of edges with one end in X and the other not in X.
We also studied a **theorem** relating connectivity with $\delta(X)$:

Graph G is disconnected if and only if there exists a proper, non-empty subset of X of $V(G)$ such that $\delta(X)$ is empty

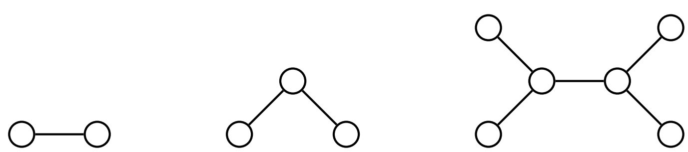
We also mentioned three concepts of algorithm analysis:

- P: polynomial time to answer/decide
- NP: polynomial time to convince there exists a solution
- Co-NP: polynomial time to convince there does not exists a solution

We also learned about **trees** and **forests**:

- Tree: a connected graph containing no cycle
- Forest: a graph (can be either connected or disconnected) containing no cycle

Example 7.0.1. *Some examples of trees*



The question we'll focus on today is *how can I convince you that a tree doesn't contain a cycle?*

7.1 More on Trees and Forests

7.1.1 Leaves

A **leaf** is a vertex of degree one.

Theorem 7.1.1:

If T is a tree with at least two vertices, then T has a leaf

Proof of Theorem 7.1.1: to prove this theorem, we'll instead prove the following (it will end up proving that T will have at least *two* leaves):

If G has a minimum degree of at least 2, then G contains a cycle

Proof: Let $P = v_0v_1 \cdots v_k$ be a longest path in G. Since G has minimum degree of at least 2, then v_0 has at least two neighbours (one of which is v_1). Thus, there exists a neighbour u of v_0 distinct from v_1.
If $u \in V(P)$, say $u = v_i$, then $C = v_0v_1, \cdots, v_iv_0$ is a cycle of G, as desired.
Now suppose $u \notin V(P)$, but then $P' = uv_0v_1 \cdots v_k$ is a path that is longer than P, a contradiction.
Now, let's prove

If T is a tree with at least two vertices, then T has at least two leaves

Proof: Let $P = v_0v_1 \cdots v_k$ be a longest path. Note that $k \geq 1$ because T is connected with at lesat two vertices. Now v_0 does not have a neighbour $u \neq v_i$, for if $u \notin V(P)$, then $p' - uv_0v_1 \cdots v_k$ is a longer path, a contradiction.
If $u \in V(P)$, say $u = v_i$, $C = v_0v_1, \cdots, v_1v_0$ is a cycle, contradicting that it's a tree.
So v_0 has degree one (i.e., is a leaf). But by symmetry, v_k is also a leaf, and since $k \geq 1$, $v_0 \neq v_k$, so T has at least two leaves, as desired.

Theorem 7.1.2:

If T is a tree and v is a leaf of T, then $T - v$ is a tree

Proof of Theorem 7.1.2: $T - v$ is a forest, because $T - v$ is a subgraph of T, and every subgraph of a forest is a forest (T is a tree, but by definition, it's a forest too).
Now we want to claim $T - v$ is connected and hence a tree. To see this, let $x, y \in V(T) - v$. Since T is connected, there exists a path P from x to y in T. Note that v is <u>not</u> in P because v has degree one and is not an end of P. But then P s a path from x to y in $T - v$, so $T - v$ is connected, as desired.

Corollary

T is a tree if and only if there exists a sequence of vertices $v_1, v_2, \cdots, v_{n-1}$ such that if $T_i = T - \{v_1, v_2, \cdots, v_{n-1}\}$, then T_i is a tree

This shows that deciding if G is a tree is in NP.

Algorithm for Deciding if G is a Tree

From the proven theorems shown above, we can create an algorithm for determining if G is a tree:

- If $|V(G)| = 1$, then yes!
- If $|V(G)| \geq 1$, determine if G has a leaf:
  - If yes, delete the leaf v and recurse on $G - v$
  - If no, then no!

This means that deciding if G is a tree is in polynomial time.

Theorem 7.1.3:

$$\textit{If } T \textit{ is a tree, then } |E(G)| = |V(G)| - 1$$

Proof of Theorem 7.1.3: Proceed by induction on $|V(T)|$:
Base case: If $|V(T)| = 1$, then $|E(G)| = 0$, as desired.
Inductive case: If $|V(G)| \geq 2$, then T has a leaf v, by our earlier theorem. by previous theorem, $T - v$ is a tree. By induction, $|E(T-v)| = |V(T-v)|-1$. However, $|V(T)| = |V(T-v)|+1$, and $|E(T)| = |E(T-v)|+1$, since v has degree one. Thus, $|E(G)| = |V(G)| - 1$, as desired.