Computer Science 331

Trees, Spanning Trees, and Subgraphs

Mike Jacobson

Department of Computer Science
University of Calgary

Lecture #31

Outline

Trees, Spanning Trees and Subgraphs

Goals for the Lecture:

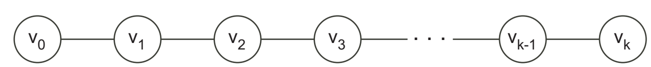
- We will introduce a particular type of graph — a *(free) tree* — that will be used in definitions of graph problems, and graph algorithms, throughout the rest of this course
- Additional important definitions and graph properties will also be introduced

Paths and Simple Paths

Definition: A *path* in an undirected graph $G = (V, E)$ is a sequence of zero or more edges in G

$$(v_0, v_1), (v_1, v_2), (v_2, v_3), \ldots, (v_{k-1}, v_k)$$

where the second vertex (shown) in each edge is the first vertex (shown) in the next edge.



The path shown above is a path *from* v_0 (the first vertex in the first edge) *to* v_k (the second vertex in the final edge).

This is a *simple path* if $v_0, v_1, \ldots, v_k$ are *distinct*.

Paths and Simple Paths

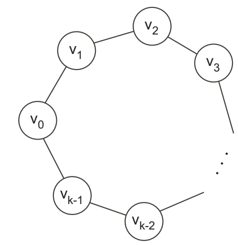
Definition: The *length* of a path is the length of the *sequence* of edges in it.

Thus the path shown in the previous slide has length k.

Definition: An undirected graph $G = (V, E)$ is a *connected* graph if there is a path from u to v, for *every* pair of vertices $u, v \in V$.

Cycles and Simple Cycles

Definition: A *cycle* (in an undirected graph $G = (V, E)$) is a path with length greater than zero from some vertex **to itself**:



A cycle $(\mathrm{v}_0, \mathrm{v}_1), (\mathrm{v}_1, \mathrm{v}_2), \ldots, (\mathrm{v}_{k-2}, \mathrm{v}_{k-1}), (\mathrm{v}_{k-1}, \mathrm{v}_0)$ is a *simple cycle* if $\mathrm{v}_0, \mathrm{v}_1, \ldots, \mathrm{v}_{k-1}$ are distinct.

A graph $G = (V, E)$ is *acyclic* if it does not have any cycles.

Problem: There is No Completely Standard Terminology!

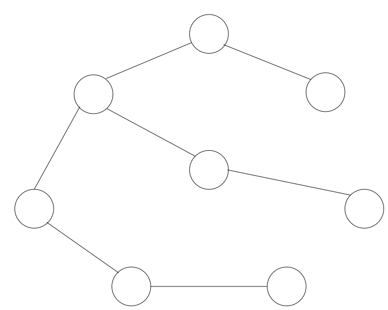
Problem with Terminology

- Different references tend to use these terms differently!
- For example, in some textbooks, a simple cycle is considered to be a kind of *simple path*, and the definition of "cycle" given is the same as the definition of *simple cycle* given above
- Other references only call something a "path" if it is a *simple path*, as defined above; they only call something a "cycle" if it is a *simple cycle*; and they use the term *walk* to refer to the more general kind of "path" that is defined in these notes

Consequence: You should check the definitions of these terms in any other references that you use!

Trees

Definition: A *free tree* is a connected acyclic graph.



Frequently we just call a free tree a "tree."

- If we identify one vertex as the "root," then the result is the kind of "rooted tree" we have seen before.

Properties 1

Consider graph $G = (V, E)$:

1. If G is connected then $|E| \geq |V| - 1$
2. If G is acyclic then $|E| \leq |V| - 1$
3. If G is connected and acyclic then $|E| = |V| - 1$

See the lecture supplement for proofs.

Properties 2

Consider graph $G = (V, E)$. We will use the following properties to characterize trees:

1. If G is a tree then it has $|V| - 1$ edges
2. An acyclic graph with $|V| - 1$ edges is a tree
3. A connected graph with $|V| - 1$ edges is a tree

See the lecture supplement for proofs.

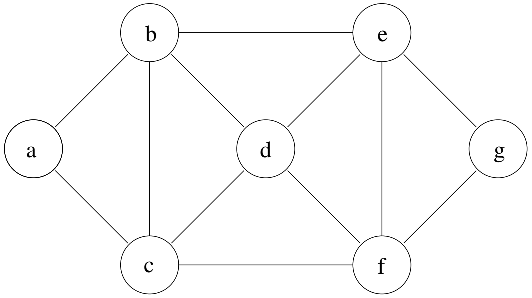
Spanning Trees

If $G = (V, E)$ is a connected undirected graph, then a *spanning tree* of G is a subgraph $\widehat{G} = (\widehat{V}, \widehat{E})$ of G such that

- $\widehat{V} = V$ (so that $\widehat{G}$ includes all the vertices in G)
- $\widehat{E} \subseteq E$
- $\widehat{G}$ is a tree.

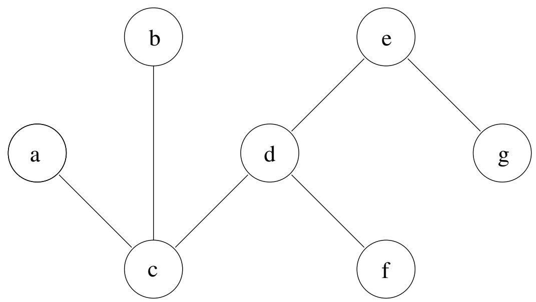
Example

Suppose $G = (V, E)$ is as follows.



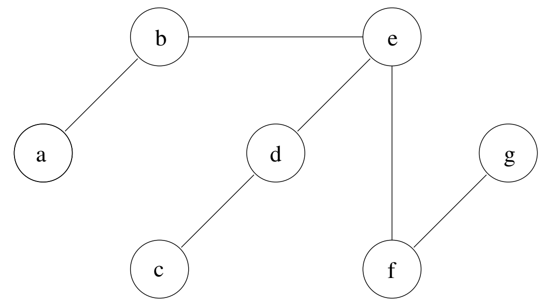
Example Tree 1

Is the following graph $G_1 = (V_1, E_1)$ a spanning tree of G? Yes!



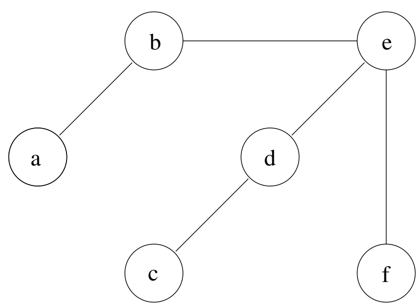
Example Tree 2

Is the following graph $G_2 = (V_2, E_2)$ also a spanning tree of G? Yes!



Example Tree 3

Is the following graph $G_3 = (V_3, E_3)$ is also a spanning tree of G? No!
Doesn't span G (vertex g missing)



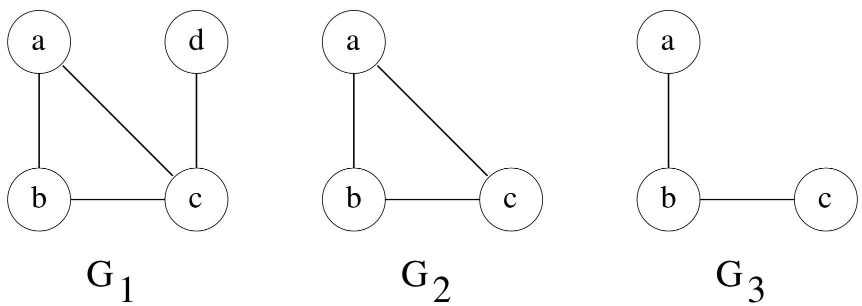
Subgraphs and Induced Subgraphs

Suppose $G = (V, E)$ is a graph.

- $\widehat{G} = (\widehat{V}, \widehat{E})$ is a *subgraph* of G if $\widehat{G}$ is a graph such that $\widehat{V} \subseteq V$ and $\widehat{E} \subseteq E$
- $\widetilde{G} = (\widetilde{V}, \widetilde{E})$ is an *induced subgraph* of G if
  - $\widetilde{G}$ is a subgraph of G and, furthermore
  - $\widetilde{E} = \left\{(u, v) \in E \mid u, v \in \widetilde{V}\right\}$, that is, $\widetilde{G}$ includes *all* the edges from G that it possibly could

Example

G_2 is an *induced subgraph* of G_1.
G_3 is a *subgraph* of G_1, but G_3 is **not** an *induced subgraph* of G_1.



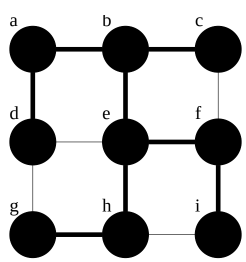
Predecessor Subgraphs

Let $G = (V, E)$ and let $s \in V$. Construct a subset V_p of V, a subset E_p of E, and a function $\pi : V \to V \cup \{\text{NIL}\}$ as follows.

- Initially, $V_p = \{s\}$, $E_p = \emptyset$, and $\pi(v) = \text{NIL}$ for *every* vertex $v \in V$.
- The following step is performed, between 0 and $|V| - 1$ times:
  - Pick some vertex u from the set V_p.
  - Pick some vertex $v \in V$ such that $v \notin V_p$ and $(u, v) \in E$. (The process must end if this is not possible to do.)
  - Set $\pi(v)$ to be u, add the vertex v to the set V_p, and add the edge $(u, v) = (\pi(v), v)$ to E_p

Note that $V_p \subseteq V$, $E_p \subseteq E$, and each edge in E_p connects pairs of vertices that each belongs to V_p each time the above (interior) step is performed — so that $G_p = (V_p, E_p)$ is always a *subgraph* of G.

Example



	a	b	c	d	e	f	g	h	i
π	NIL	a	b	a	b	e	h	e	f

Predecessor Subgraph Property

The graph $G_p = (V_p, E_p)$ that has been constructed is called a *predecessor subgraph*.

Claim:

Let $G_p = (V_p, E_p)$ be a predecessor subgraph of an undirected graph G.

a) *G_p is a subgraph of G and G_p is a tree.*

b) *If $V_p = V$ then G_p is a spanning tree of G.*

Proof.

Part (a) is true because $|E_p| = |V_p| - 1$, by the construction of V_p and of E_p, and G_p is always connected, so G_p is a tree, as well as a subgraph of G.

Part (b) now follows by the fact that E_p is a subset of E, so that G_p is a subgraph of G, and by the fact that $V_p = V$. □