Trees

Hongfei Fu

John Hopcroft Center for Computer Science
Shanghai Jiao Tong University
planar graphs

graph coloring
Today’s Topic

- trees
- spanning trees
Trees (树)

main textbook, Page 745 – Page 755
source: from internet
A tree is a connected undirected (simple) graph without simple circuits.

A forest is a undirected graph without simple circuits (i.e., a finite union of trees).
Trees

Definitions

- A **tree** is a connected undirected (simple) graph without simple circuits.
- A **forest** is a undirected graph without simple circuits (i.e., a finite union of trees).

Examples

- [Images of trees and forests]
Theorem

An undirected graph $G$ is a tree iff any two vertices in $G$ is connected by a unique simple path.
Theorem

An undirected graph $G$ is a tree iff any two vertices in $G$ is connected by a unique simple path.

Proof

- "if" part: Uniqueness of simple paths leads to non-existence of simple circuits.
- "only if" part: Distinct paths between two vertices lead to existence of a simple circuit.
A simple graph $G = (V, E)$ is a tree iff $G$ is connected and $|E| = |V| - 1$.

A connected simple graph $G = (V, E)$ satisfies that $|E| \geq |V| - 1$. 

Proof
By induction on the number of vertices.

Inductive Step: Pick any vertex $v \in V$ and list its neighbours as $u_1, \ldots, u_k$. Let $C_j = (V_j, E_j)$ be the connected component in $G - v$ that contains $u_j$. We have that $|E_j| \geq |V_j| - 1$. Then $G$ is a tree iff all $C_j$'s are distinct and each $C_j$ is a tree iff all $C_j$'s are distinct and $|E_j| = |V_j| - 1$ for all $j$ iff $|\bigcup_j E_j| = |\bigcup_j V_j| - k$ iff $|E| = |V| - 1$. 

Hongfei Fu (SJTU JHC)
A simple graph $G = (V, E)$ is a tree iff $G$ is connected and $|E| = |V| - 1$.

A connected simple graph $G = (V, E)$ satisfies that $|E| \geq |V| - 1$.

Proof

By induction on the number of vertices.

- **Base Step:** $|V| = 1$. Straightforward.

- **Inductive Step:** Pick any vertex $v \in V$ and list its neighbours as $u_1, \ldots, u_k$. Let $C_j = (V_j, E_j)$ be the connected component in $G - v$ that contains $u_j$. We have that $|E_j| \geq |V_j| - 1$. Then $G$ is a tree
  - iff all $C_j$'s are distinct and each $C_j$ is a tree
  - iff all $C_j$'s are distinct and $|E_j| = |V_j| - 1$ for all $j$
  - iff $|\bigcup_j E_j| = |\bigcup_j V_j| - k$
  - iff $|E| = |V| - 1$
A rooted tree is a tree with a designated vertex (as the root).
Rooted Trees

A root tree is a tree with a designated vertex (as the root).

Examples
Levels

- \( G = (V, E) \): a rooted tree with the root \( r \)

Then we define that
Trees

Levels

- \( G = (V, E) \): a rooted tree with the root \( r \)

Then we define that

- the level of a vertex \( v \) is the length of the unique path from \( r \) to \( v \);
- the height of the tree is the maximal level of its vertices;
Trees

Levels

- $G = (V, E)$: a rooted tree with the root $r$

Then we define that

- the **level** of a vertex $v$ is the length of the unique path from $r$ to $v$;
- the **height** of the tree is the maximal level of its vertices;
- the partial order $\preceq$ on $V$ consists of all ordered pairs $(u, v)$ of vertices such that the unique simple path from $r$ to $v$ contains $u$. 
Terminologies

- \( G = (V, E) \): a rooted tree with the root \( r \)

We have:

- The parent of a vertex \( v \neq r \) is the unique vertex \( u \) such that \( \{u, v\} \in E \) and \( u \preceq v \). Then \( v \) is a child of \( u \).
- Vertices \( u, v \) are siblings if they have the same parent.
- The ancestors of a vertex \( v \neq r \) is the vertices on the path from \( r \) to \( v \), excluding \( v \) but including \( r \).
- The descendants of a vertex \( v \) are all vertices that have \( v \) as one of their ancestors.
- A vertex is a leaf if it has no children, otherwise it is an internal vertex.
- The subtree at a vertex \( v \) is the induced subgraph of the vertex set consisting of the vertex \( v \) (as its root) and all the descendants of \( v \).
Trees

**Terminologies**

- $G = (V, E)$: a rooted tree with the root $r$

We have:

- The parent of a vertex $v \neq r$ is the unique vertex $u$ such that $\{u, v\} \in E$ and $u \preceq v$. Then $v$ is a child of $u$. 

Vertices $u, v$ are siblings if they have the same parent.

The ancestors of a vertex $v \neq r$ is the vertices on the path from $r$ to $v$, excluding $v$ but including $r$.

The descendants of a vertex $v$ are all vertices that have $v$ as one of their ancestors.

A vertex is a leaf if it has no children, otherwise it is an internal vertex.

The subtree at a vertex $v$ is the induced subgraph of the vertex set consisting of the vertex $v$ (as its root) and all the descendants of $v$. 

Trees

Terminologies

- \( G = (V, E) \): a rooted tree with the root \( r \)

We have:

- The parent of a vertex \( v \neq r \) is the unique vertex \( u \) such that \( \{u, v\} \in E \) and \( u \preceq v \). Then \( v \) is a child of \( u \).
- Vertices \( u, v \) are siblings if they have the same parent.
- The ancestors of a vertex \( v \neq r \) is the vertices on the path from \( r \) to \( v \), excluding \( v \) but including \( r \).
Trees

Terminologies

- \( G = (V, E) \): a rooted tree with the root \( r \)

We have:

- The parent of a vertex \( v \neq r \) is the unique vertex \( u \) such that \( \{u, v\} \in E \) and \( u \preceq v \). Then \( v \) is a child of \( u \).

- Vertices \( u, v \) are siblings if they have the same parent.

- The ancestors of a vertex \( v \neq r \) is the vertices on the path from \( r \) to \( v \), excluding \( v \) but including \( r \).

- The descendants of a vertex \( v \) are all vertices that have \( v \) as one of their ancestors.
Terminologies

- \( G = (V, E) \): a rooted tree with the root \( r \)

We have:

- The **parent** of a vertex \( v \neq r \) is the unique vertex \( u \) such that \( \{u, v\} \in E \) and \( u \preceq v \). Then \( v \) is a **child** of \( u \).
- Vertices \( u, v \) are **siblings** if they have the same parent.
- The **ancestors** of a vertex \( v \neq r \) is the vertices on the path from \( r \) to \( v \), excluding \( v \) but including \( r \).
- The **descendants** of a vertex \( v \) are all vertices that have \( v \) as one of their ancestors.
- A vertex is a **leaf** if it has no children, otherwise it is an **internal vertex**.
Trees

Terminologies

- $G = (V, E)$: a rooted tree with the root $r$

We have:

- The parent of a vertex $v \neq r$ is the unique vertex $u$ such that \{u, v\} $\in E$ and $u \preceq v$. Then $v$ is a child of $u$.

- Vertices $u, v$ are siblings if they have the same parent.

- The ancestors of a vertex $v \neq r$ is the vertices on the path from $r$ to $v$, excluding $v$ but including $r$.

- The descendants of a vertex $v$ are all vertices that have $v$ as one of their ancestors.

- A vertex is a leaf if it has no children, otherwise it is an internal vertex.

- The subtree at a vertex $v$ is the induced subgraph of the vertex set consisting of the vertex $v$ (as its root) and all the descendants of $v$. 

**Examples**

$T_1$  

$T_2$  

$T_3$  

$T_4$
Trees

$m$-ary Trees

- $G$: a rooted tree

We say that
$m$-ary Trees

- $G$: a rooted tree

We say that

- $G$ is $m$-ary if every internal vertex has at most $m$ children;
Trees

$m$-ary Trees

- $G$: a rooted tree

We say that

- $G$ is $m$-ary if every internal vertex has at most $m$ children;
- $G$ is full $m$-ary if every internal vertex has exactly $m$ children;
$m$-ary Trees

- $G$: a rooted tree

We say that

- $G$ is $m$-ary if every internal vertex has at most $m$ children;
- $G$ is full $m$-ary if every internal vertex has exactly $m$ children;
- $G$ is binary if it is 2-ary.
Theorem

A full $m$-ary tree with $i$ internal vertices has $n = m \cdot i + 1$ vertices.

Proof
Each internal vertex has $m \cdot i$ children, except for the root vertex.
Theorem
A full $m$-ary tree with $i$ internal vertices has $n = m \cdot i + 1$ vertices.

Proof
Each internal vertices has $m \cdot i$ children, except for the root vertex.
Theorem

A full $m$-ary tree with $i$ internal vertices has $n = m \cdot i + 1$ vertices.

Proof

Each internal vertices has $m \cdot i$ children, except for the root vertex.

Theorem

An $m$-ary tree of height $h$ has at most $m^h$ leaves.
Theorem

A full \( m \)-ary tree with \( i \) internal vertices has \( n = m \cdot i + 1 \) vertices.

Proof

Each internal vertex has \( m \cdot i \) children, except for the root vertex.

Theorem

An \( m \)-ary tree of height \( h \) has at most \( m^h \) leaves.

- proof by induction on \( h \)
Balanced $m$-ary Trees

A rooted $m$-ary tree of height $h$ is balanced if all its leaves are at level $h$ or $h - 1$. 
Balanced \( m \)-ary Trees

A rooted \( m \)-ary tree of height \( h \) is balanced if all its leaves are at level \( h \) or \( h - 1 \).

Examples

\( T_1 \) \hspace{1cm} \( T_2 \) \hspace{1cm} \( T_3 \)
Spanning Trees  （生成树）

main textbook, Page 785 – 791
Spanning Trees

Motivation
How can we find a minimal subgraph of a connected graph that is still connected?
Spanning Trees

Motivation

How can we find a minimal subgraph of a connected graph that is still connected?

An Example

We should find a subgraph which is a tree.
Spanning Trees

Motivation
How can we find a minimal subgraph of a connected graph that is still connected?

An Example

Answer
We should find a subgraph which is a tree.
Spanning Trees

Definition

- **G**: a simple graph

A **spanning tree** of **G** is a subgraph of **G** that is a tree containing every vertex of **G**.
Definition

- $G$: a simple graph

A spanning tree of $G$ is a subgraph of $G$ that is a tree containing every vertex of $G$.

Theorem

A simple graph $G$ is connected iff it has a spanning tree.
**Definition**

- $G$: a simple graph

A **spanning tree** of $G$ is a subgraph of $G$ that is a tree containing every vertex of $G$.

**Theorem**

A simple graph $G$ is connected iff it has a spanning tree.

- **proof**: A spanning tree can be obtained by repeatedly removing edges from simple circuits.
Construction of Spanning Trees

- depth-first search
- breadth-first search
Spanning Trees

Depth-First Search  （深度优先搜索）

1. Fix an arbitrary starting vertex.
2. Continuously augment a path until no edge is augmentable.
3. Trace back to the deepest vertex from which some edge is augmentable and go back to Step 2.

An Example
Spanning Trees

Depth-First Search （深度优先搜索）

1. Fix an arbitrary starting vertex.
Spanning Trees

Depth-First Search （深度优先搜索）

1. Fix an arbitrary starting vertex.
2. Continuously augment a path until no edge is augmentable.
Spanning Trees

Depth-First Search (深度优先搜索)

1. Fix an arbitrary starting vertex.
2. Continuously augment a path until no edge is augmentable.
3. Trace back to the deepest vertex from which some edge is augmentable and go back to Step 2.
Spanning Trees

Depth-First Search （深度优先搜索）

1. Fix an arbitrary starting vertex.
2. Continuously augment a path until no edge is augmentable.
3. Trace back to the deepest vertex from which some edge is augmentable and go back to Step 2.

An Example
Spanning Trees

Breadth-First Search

1. Fix an arbitrary starting vertex.
2. Continuously augment all possible edges from the vertices at the current level.
Spanning Trees

Breadth-First Search (广度优先搜索)

1. Fix an arbitrary starting vertex.
Breadth-First Search （广度优先搜索）

1. Fix an arbitrary starting vertex.
2. Continuously augment all possible edges from the vertices at the current level.
Spanning Trees

Breadth-First Search (广度优先搜索)

1. Fix an arbitrary starting vertex.
2. Continuously augment all possible edges from the vertices at the current level.

An Example

![Graph diagram]

- Vertex 'a' is connected to 'b', 'd', and 'h'.
- Vertex 'b' is connected to 'c' and 'e'.
- Vertex 'c' is connected to 'l' and 'f'.
- Vertex 'l' is connected to 'g'.
- Vertex 'd' is connected to 'e' and 'f'.
- Vertex 'e' is connected to 'f' and 'i'.
- Vertex 'f' is connected to 'g' and 'i'.
- Vertex 'g' is connected to 'j'.
- Vertex 'i' is connected to 'j' and 'j'.
- Vertex 'j' is connected to 'g'.
- Vertex 'k' is connected to 'm'.
- Vertex 'm' is connected to 'k'.

The graph shows a network of vertices connected by edges, illustrating a spanning tree.
Minimum Spanning Trees (最小生成树)

main textbook, Page 797 – 802
Minimum Spanning Trees

Weighted Simple Graphs

source: from internet
Minimum Spanning Trees

Weighted Simple Graphs

A weighted graph is a simple graph \( G = (V, E) \) with a function that assigns to each edge a non-negative real number.
Minimum Spanning Trees

Weighted Simple Graphs

A **weighted** graph is a simple graph \( G = (V, E) \) with a function that assigns to each edge a non-negative real number.

Minimum Spanning Trees

- **\( G = (V, E) \):** a connected weighted graph

A **minimal spanning tree** of \( G \) is a spanning tree whose sum of all weights in the tree is the minimum among all spanning trees.
Trees
- definitions and properties
- spanning trees

reference material: 《图论与代数结构》，戴一奇等，清华大学出版社

reference material: Reinhard Diestel, *Graph Theory*, 5th edition [RD]
http://diestel-graph-theory.com/basic.html?
• [R], Page 745 – 755, Page 785 – 795,
• (optional) 《图论与代数结构》，戴一奇等，清华大学出版社
Homeworks

- [R], Page 755, Exercise 2(a)(b)(c)(d)(e)(f)
- [R], Page 756, Exercise 14, 15(a)
Homework Submission

- **submission time**: the start of the class on Nov. 26th
- **teaching assistant**:
  - Peixin Wang: peter007008@qq.com
  - Jinyi Wang: jinyi.wang@sjtu.edu.cn
  - Luhua Jin: 1097795310@qq.com
- **submission**:
  - written version (preferred): submit on the desk
  - electronic version: word or pdf version, send email with title

“离散数学+姓名+学号+第十一周周五”

to the teaching assistants:
- Students from F1903001, F1903003 and F1903004, send to Luhua Jin.
- Students from F1903801 and F1903802, send to Jinyi Wang.
- All other students please send to Peixin Wang.