

CSE 21: Math for Algorithm & System Analysis Summer II 2006

Lecture 14

Wenjing Rao
wrao@cs.ucsd.edu

Material covered

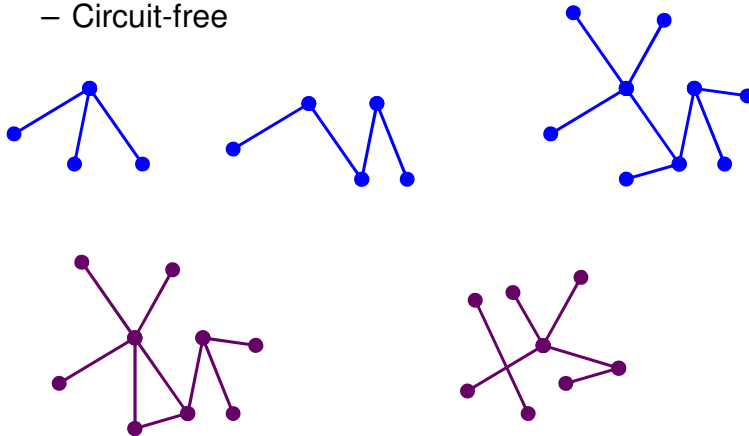
- Graph theory
 - Euler circuit / path
 - Hamiltonian circuit

Today's Topic

- Tree
- Spanning Trees

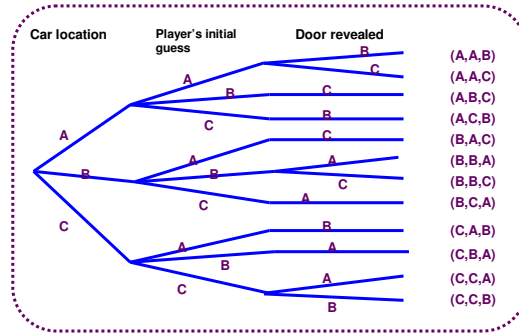
Tree: definition

- *A Tree is:*
 - A connected graph
 - Circuit-free



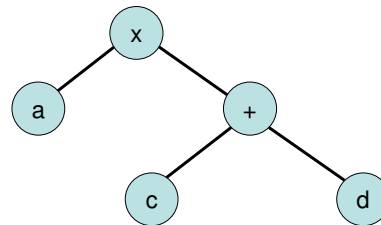
Tree Applications

- *Decision tree*



- Expression representation

$$- a \times (c + d)$$



Circuit / No circuit

- What is the minimum # of edges you need to connect n vertices?

Tree or not?

- A graph has 5 vertices and 6 edges, can it be a tree?

Paths connecting tree vertices

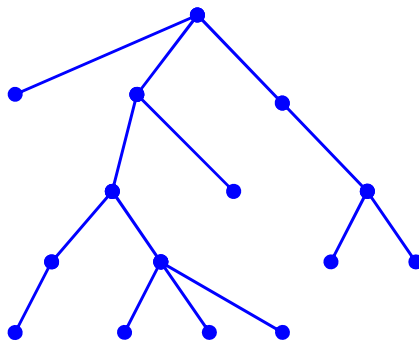
- How many possible paths connecting two vertices in a tree ?

Equivalent definitions

- If G is a connected graph, then the following are equivalent:
 - G is a tree
 - G has no circuits
 - For every pair of distinct vertices v and w in G , there is exactly one path from v to w
 - Removing any edge from G disconnects the graph

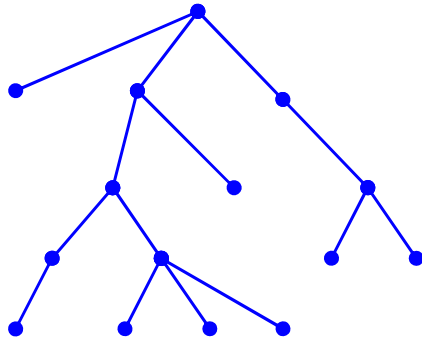
Rooted Trees

- One vertex distinguished from others as the *root*



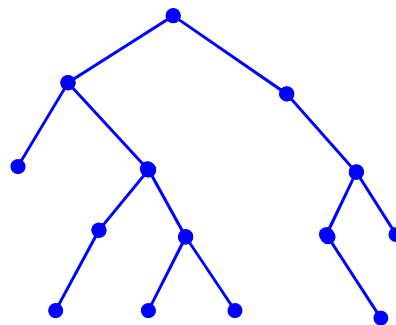
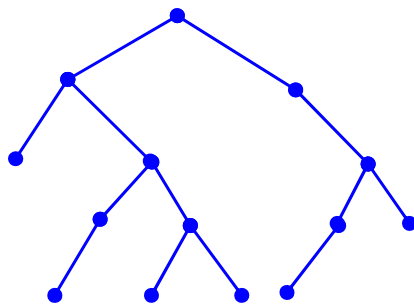
Rooted Trees

- Pick a different root?



Binary Trees

- A rooted tree with every parent has at most 2 children
 - Left children – left subtree
 - Right children – right subtree



Binary Tree Application

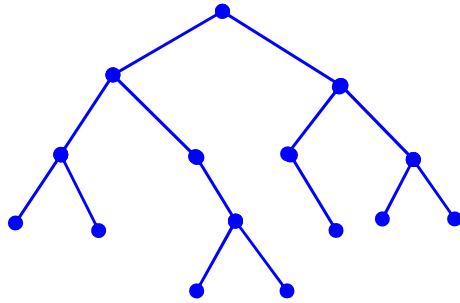
- Represent $((a-b) \times c) + (d - e) / f$

Binary Tree Application

- Use binary tree structure to keep data sorted
 - Incoming data: 53, 69, 55, 12, 7, 45, 98, 120, 67, 63, 8, 3,
 - Search for 67?

Binary Tree: Height vs Terminal node

- T is a binary tree with height h , how many terminal vertices can T have?

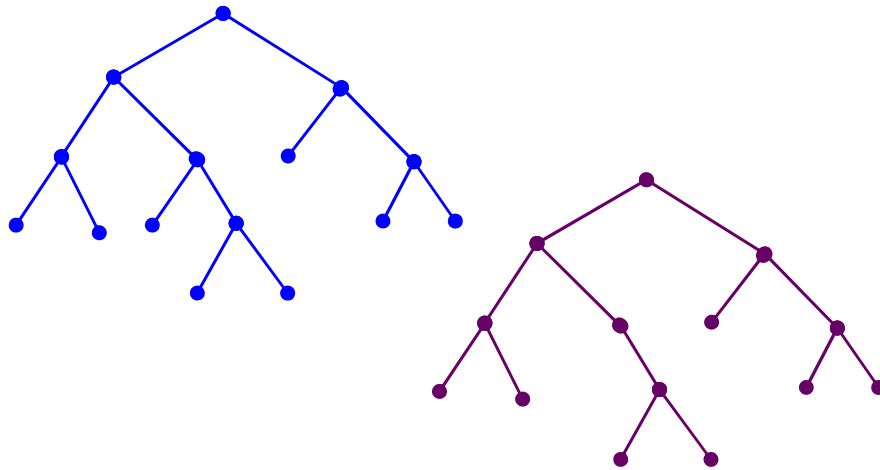


Example

- Can you draw a binary tree of height 3 with 9 terminal vertices?
- How about a binary tree of height 4 with 9 terminal vertices?

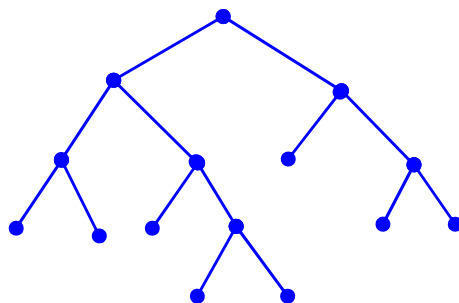
Full Binary Trees

- A binary tree with each parent has exactly 2 children



Full Binary Tree: Internal vs Terminal nodes

- T is a full binary tree with k internal nodes. Can you deduct the # of terminal nodes?



Example

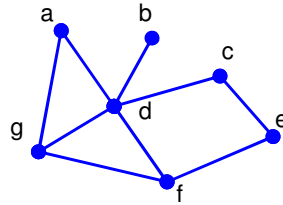
- Can you draw a **full binary tree** of 5 internal vertices and 4 terminal vertices?
- How about a **binary tree** of 5 internal vertices and 4 terminal vertices?

Example

- Can you draw a **full binary tree** of 5 internal vertices and 7 terminal vertices?
- How about a **binary tree** of 5 internal vertices and 7 terminal vertices?

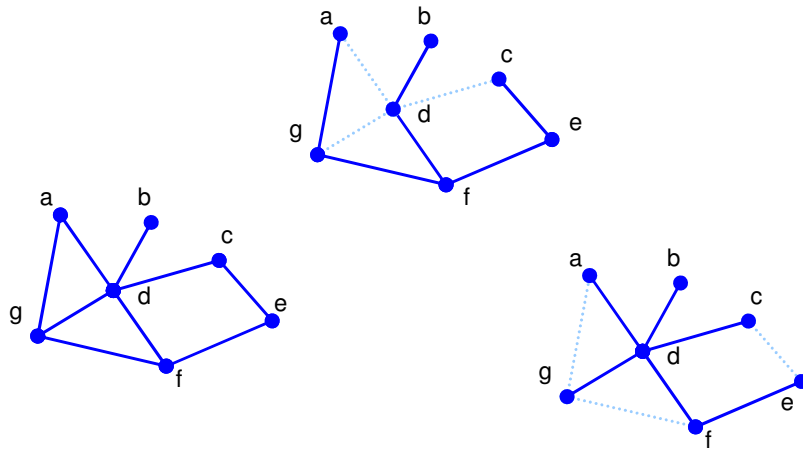
Spanning Tree of a Graph

- A **spanning tree** for a graph G is
 - A subgraph of G
 - Contains every vertex of G
 - Is a tree



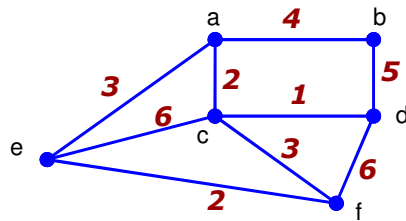
Spanning Tree

- Every connected graph has a spanning tree (maybe more)
- The spanning trees of a same graph have the same number of edges



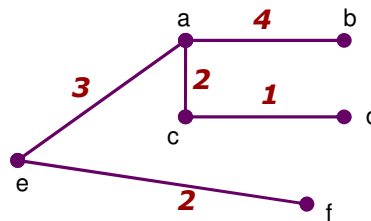
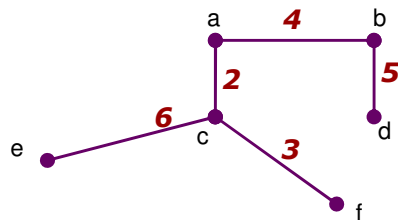
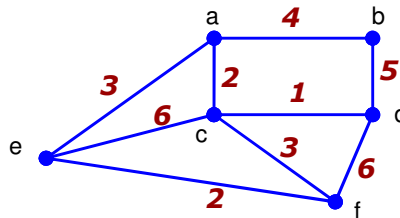
Minimal Spanning Tree

- A **minimal spanning tree** for a **weighted** graph G is
 - A spanning tree
 - Has the least total weight (comparing to other possible spanning trees)
- Application:
 - Cost of building roads between cities



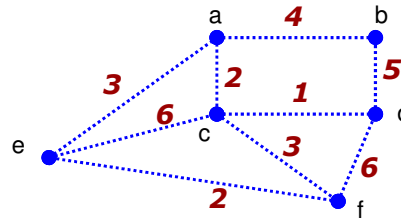
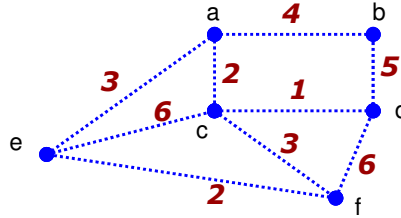
Minimal Spanning Tree

- Cost of building roads between cities



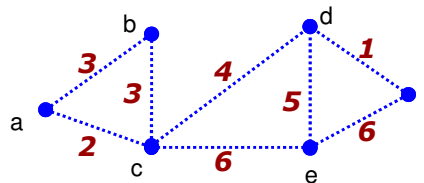
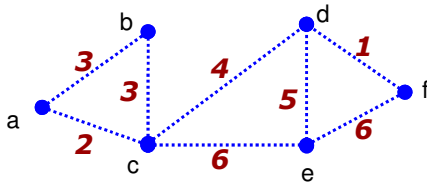
Minimal Spanning Tree

- Kruskal's algorithm
 - Start with all the vertices, with no edges
 - Repeat:
 - Pick up the edge with minimum weight among the "haven't check" set
 - Check: adding this edge introduces circuit?
 - If not, add the edge to the spanning tree



Example

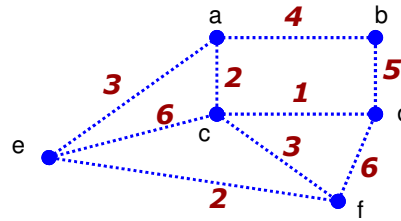
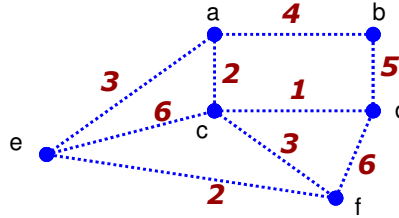
- Use Kruskal's algorithm to find the minimal spanning tree



Minimal Spanning Tree

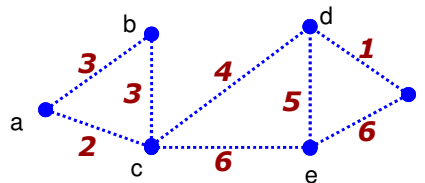
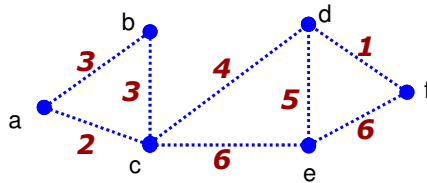
- Prim's algorithm

- Start with one vertex as the initial tree T, and no edges
- Repeat for n-1 times:
 - Pick up the edge with minimum weight that can connect an outside vertex to T
 - Add the edge and the new vertex to T



Example

- Use Prim's algorithm to find the minimal spanning tree, starting with vertex a



The two MST algorithms

- Kruskal's algo
 - Do not start with any specific vertex
 - Pick min-weight edge and check for circuits
- Prim's algo
 - Start with a specific vertex
 - Grow the existing tree by adding the min-weight edge connecting to the outside vertices