8.1 Introduction to trees: basic concepts

Notebook:	Discrete Mathematics [CM1020]		
Created:	2019-10-07 2:31 PM	**Updated:**	2019-12-26 1:47 PM
Author:	SUKHJIT MANN		

Cornell Notes	**Topic:** 8.1 Introduction to trees: basic concepts	Course: BSc Computer Science Class: Discrete Mathematics-Lecture Date: December 23, 2019

Essential Question:

What is a tree?

Questions/Cues:

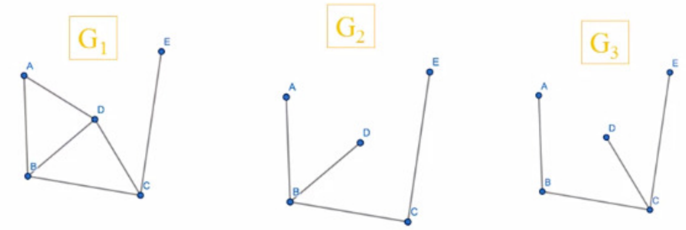
- What is acyclic graph?
- What is a tree?
- What is a forest?
- What are some properties of trees?
- What is a rooted tree?
- What is a spanning tree?
- How we construct a spanning tree?
- What are non-isomorphic spanning trees?
- What is cost mean in terms of trees?
- What is a minimum-cost spanning tree?
- What are the two greedy algorithms for finding minimum-cost spanning trees?
- What is Kruskal's Algorithm?
- What is Prim's Algorithm?

Notes

- Acyclic graph = A graph G is an acyclic graph if and only if G has no cycles. This includes no loops and no parallel edges.

- Tree = An undirected graph G is a tree if and only if it is connected and acyclic. This means there exists a path between any two vertices of G & G is cycle-free.

Hence, a tree can have neither loops nor multiple edges (parallel edges)

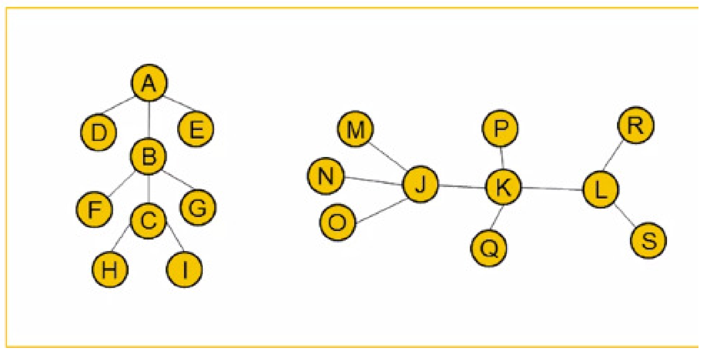


Not a tree **Tree** **Tree**

Definition of a forest

A **disconnected graph** containing no cycles is called a **forest**.

F

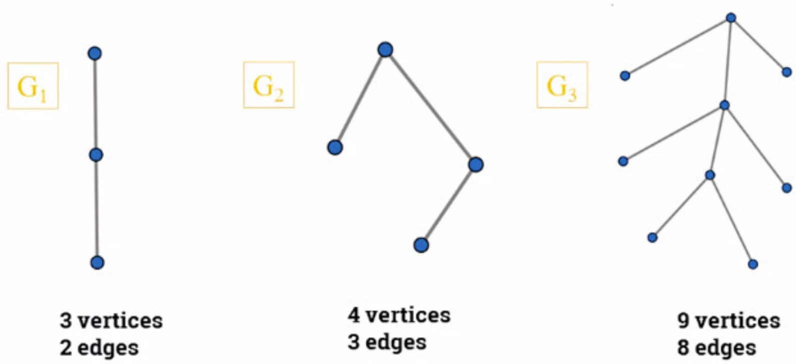


Theorem 1

An undirected graph is a **tree** if and only if there is **unique simple path** between **any two** of its vertices.

Theorem 2

A tree with **n vertices** has **n-1 edges**.



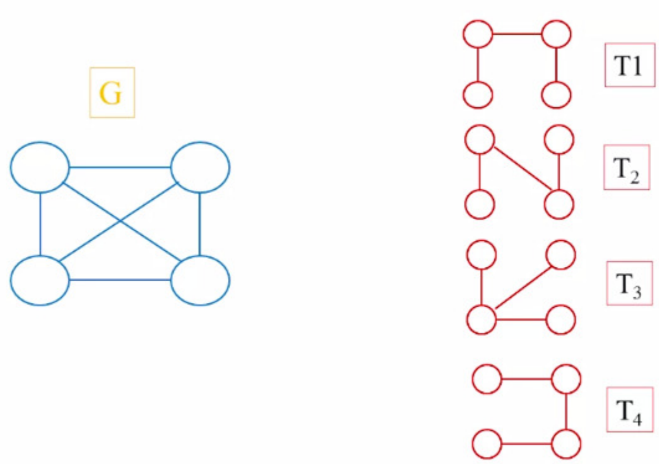
Rooted trees

A **rooted tree** is when one vertex has been **designated as the root** and every edge is **directed away from the root**.

Definition of spanning trees

A **spanning tree** of a graph G is a **connected** sub graph of G which **contains all vertices of G**, but with **no cycles**.

Example

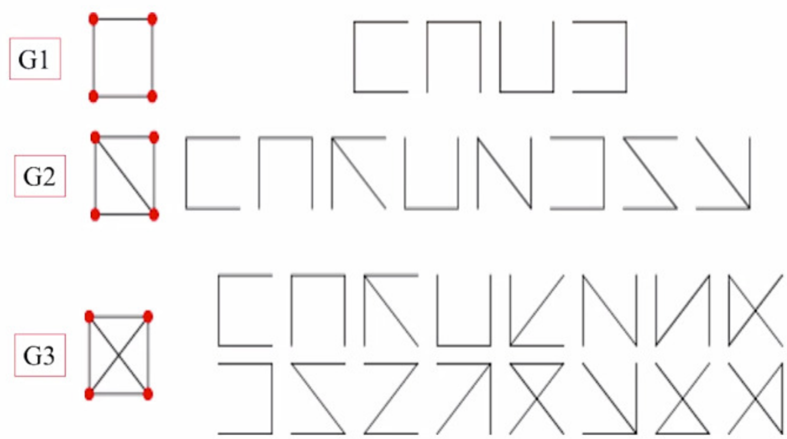


Constructing a spanning tree

To get a spanning tree of a graph G

1. Keep all vertices of G
2. Break all the cycles but keep the tree connected.

Example of spanning trees

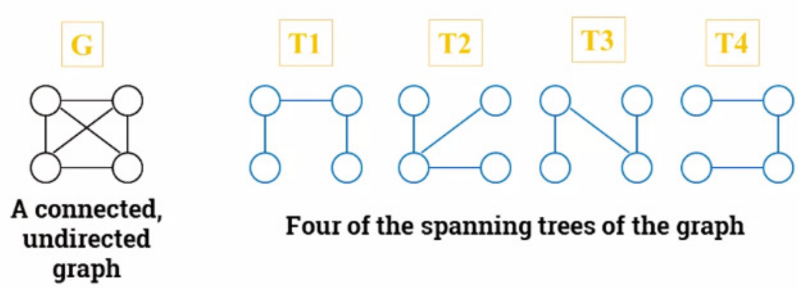


Non-isomorphic spanning trees

Two spanning trees are said **isomorphic** if there is a **bijection preserving adjacency** between the two trees.

*****NOTE** if we asked to draw all the spanning trees of a graph, we are only interested in drawing the non-isomorphic ones

Example



T_1, T_3 and T_4 are all **isomorphic** to each others

T_1, T_3 and T_4 are all **non-isomorphic** to T_2.

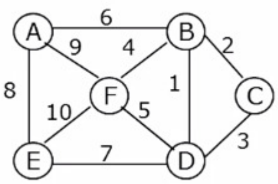
Spanning trees cost

Suppose you have a connected undirected graph with a **weight** (or cost) **associated** with **each** edge.

The **cost** of a spanning tree would be the **sum of the costs** of its edges.

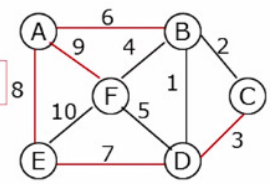
The weight of spanning trees

G



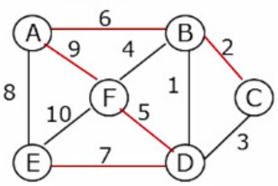
A connected, undirected graph

T1



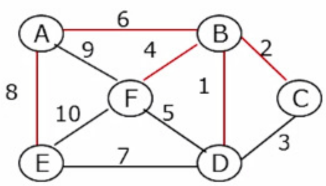
w=8+9+6+7+3 = 33

T2



w=6+9+5+2+7 =29

T3

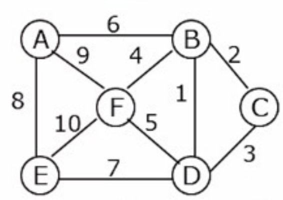


w=8+6+4+1+2 = 21

Minimum-cost spanning trees

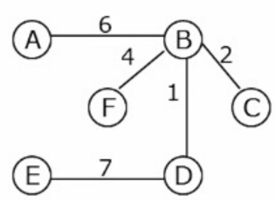
A **minimum-cost** spanning tree is a spanning tree that has the **lowest weight** (lowest cost).

G



A connected, undirected graph

T4



A minimum-cost spanning tree

Finding spanning trees

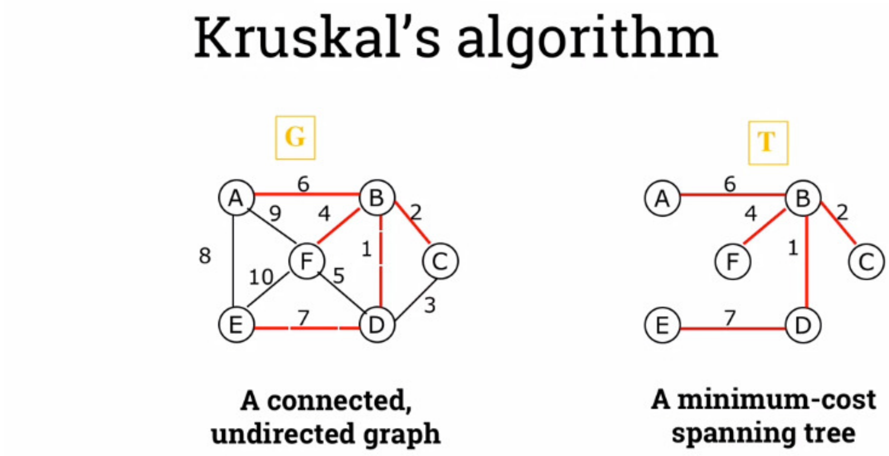
There are two basic algorithms for finding minimum-cost spanning trees, and both are greedy algorithms:

Kruskal's algorithm

Prim's algorithm

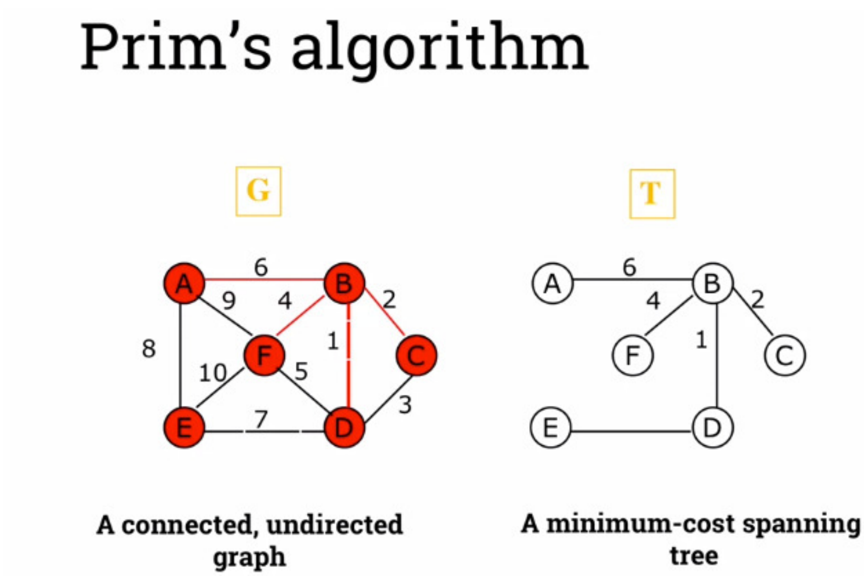
Kruskal's Algorithm

1. Start with the cheapest edge in the spanning tree.
2. Repeatedly add the cheapest edge that keeps the tree connected but free from any cycles.



Prim's Algorithm

1. Start with any one node in the spanning tree.
2. Repeatedly add the cheapest edge incident to that node and the node it leads to, for which the node is not already in the spanning tree.



Summary

In this week, we learned what tree is, what a forest is, what a rooted tree is? Alongside this, we explored some properties of trees, what a spanning tree and minimum-cost spanning trees are and the two greedy algorithms for finding minimum cost spanning trees; Kruskal's and Prim's Algorithm.