Cheatsheet - Trees

Fabio Lama – fabio.lama@pm.me

NOTE It's recommended to read the cheatsheets on Graphs, first.

1. Definition

In computer science, **trees** are used in a wide range of algorithms, such as data structures for efficient lookups or making decisions. A tree is a **connected acyclic undirected graph**.

1.1. Acyclic Graphs

A graph G is called an **acyclic** graph if and only if G has **no cycles**.

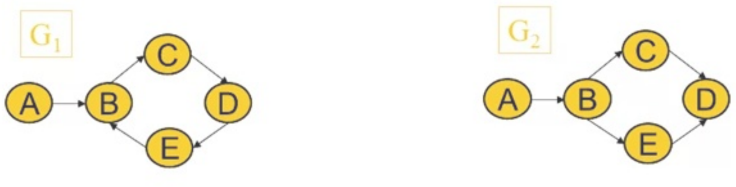
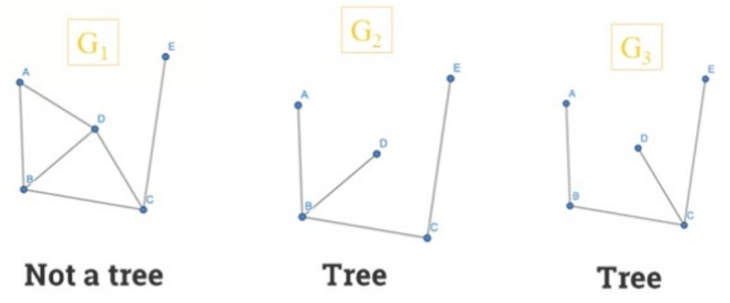


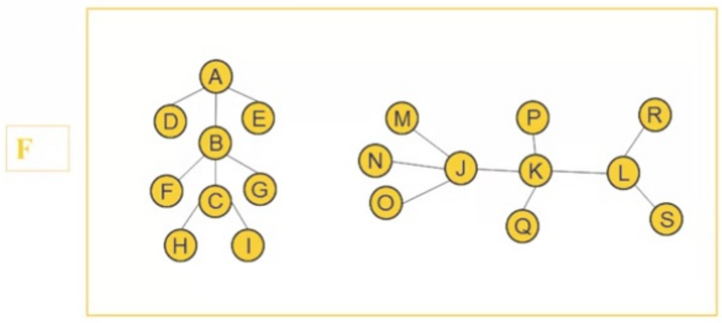
Figure 1. G_1 is NOT an acyclic graph, while G_2 is.

Not all acyclic graphs are trees.



1.2. Forest

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



Theorem 1

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

Theorem 2

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

1.3. Rooted Tree

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

2. Spanning Tree

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

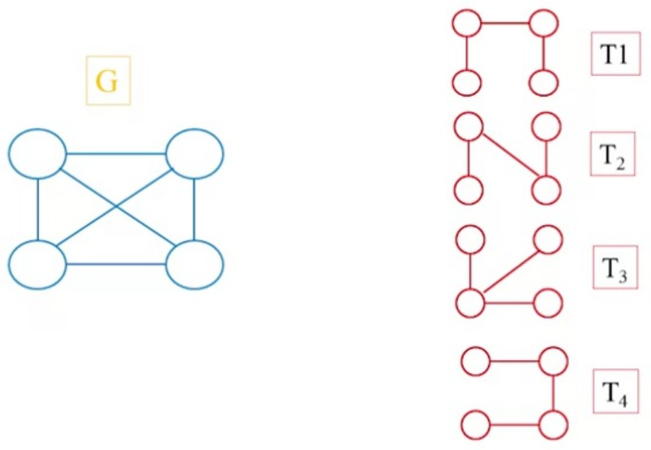
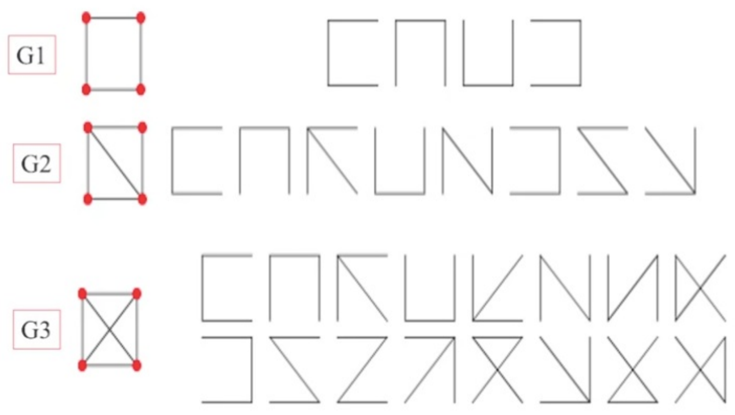


Figure 2. All T_n are spanning sub graphs of graph G.

2.1. 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.



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

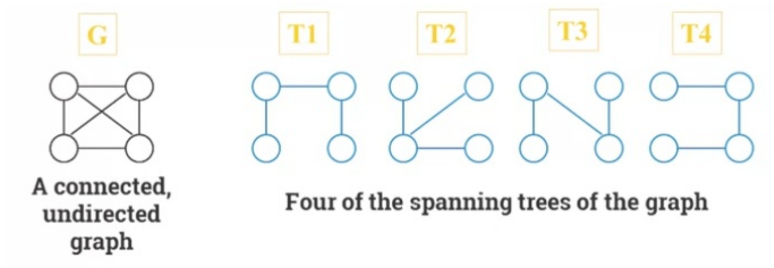

G
T1
T2
T3
T4
A connected, undirected graph
Four of the spanning trees of the graph

Figure 3. 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.*

2.2. Minimum-cost Spanning Tree

The **cost** (or **weight**) of a spanning tree is the **sum of the costs** of its edges. A **minimum-cost** spanning tree is a spanning tree that has the **lowest weight** (lowest cost).

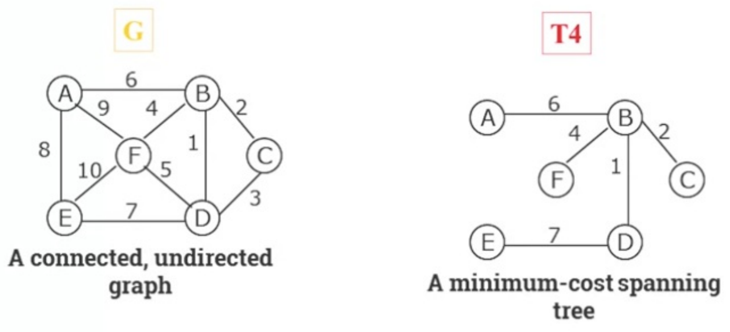


Figure 4. The weight of T_2 is $w = 6 + 4 + 1 + 2 + 7 = 20$.

There are two basic algorithms for finding minimum-cost spanning trees:

Kruskal's algorithm

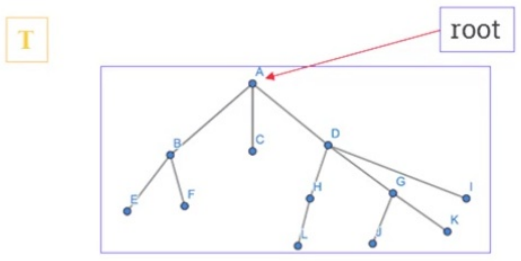
Start with the cheapest edges in the spanning tree, then repeatedly add the cheapest edge that does not create a cycle.

Prim's algorithm

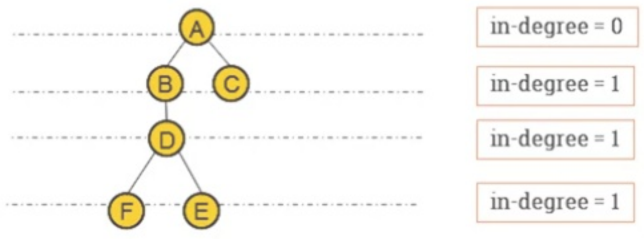
Start with any one node in the spanning tree, then repeatedly add the cheapest edge, and the node it leads to, for which the node is not already in the spanning tree.

3. Rooted Tree

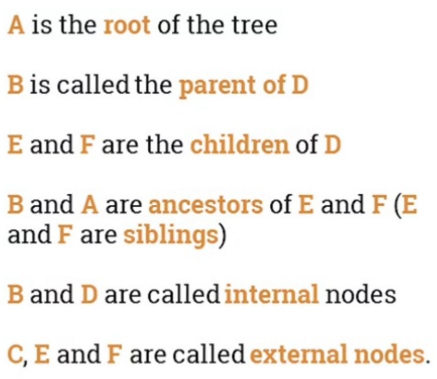
A rooted tree is a **directed tree** having one **distinguished** vertex r, called the root, such that for every vertex v there is a **directed path** from r to v



A directed tree is represented as a rooted tree **if and only if one vertex** has in-degree 0 whereas **all** other **vertices** have in-degree 1.



3.1. Terminology


A is the root of the tree

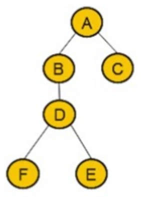
B is called the parent of D

E and F are the children of D

B and A are ancestors of E and F (E and F are siblings)

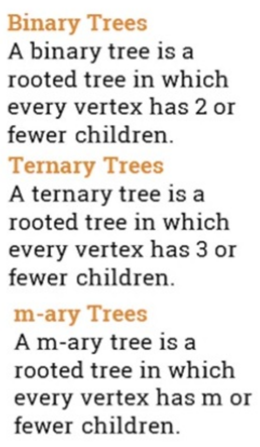
B and D are called internal nodes

C, E and F are called external nodes.



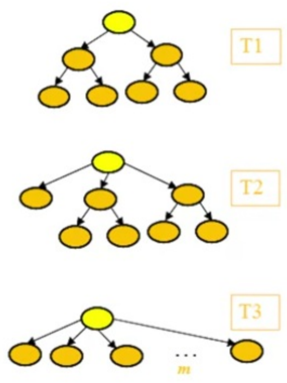
The **depth** or **path length** of a node in a tree is the number of edges from the root to that node. The **height** of a node in a tree is the longest path from that node to a leaf. The **depth or the height** of a tree is the maximum path length across all its nodes.

3.2. Special Trees


Binary Trees
A binary tree is a rooted tree in which every vertex has 2 or fewer children.

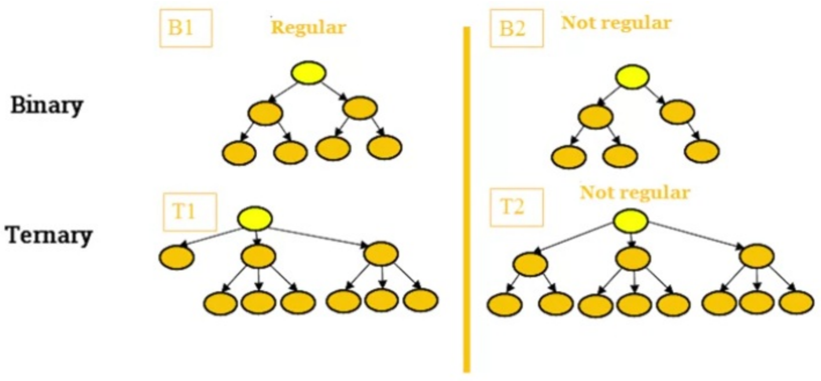
Ternary Trees
A ternary tree is a rooted tree in which every vertex has 3 or fewer children.

m-ary Trees
A m-ary tree is a rooted tree in which every vertex has m or fewer children.

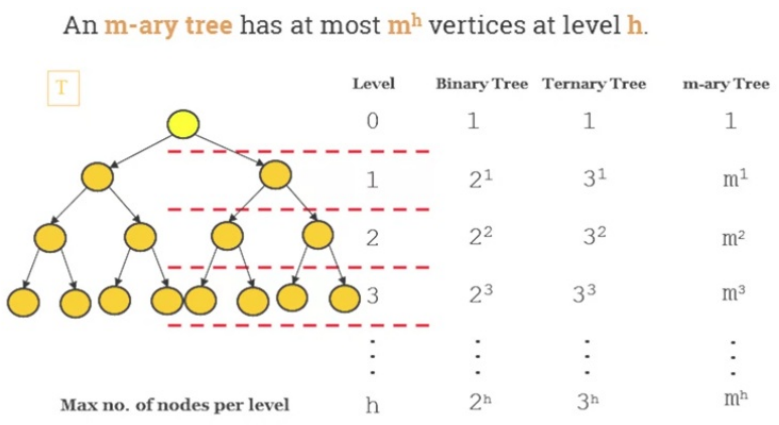


3.3. Regular Rooted Trees

An m-ary tree is **regular** if every one of its **internal** notes **has exactly** m children.



An m**-ary tree** has at most m^h vertices at level h.


An m-ary tree has at most m^h vertices at level h.

Level	Binary Tree	Ternary Tree	m-ary Tree
0	1	1	1
1	2^1	3^1	m^1
2	2^2	3^2	m^2
3	2^3	3^3	m^3
⋮	⋮	⋮	⋮
h	2^h	3^h	m^h

4. Isomorphic Trees

Two trees T_1 and T_2 are isomorphic if there is a **bijection**:

$$f : V(T_1) \to V(T_2)$$

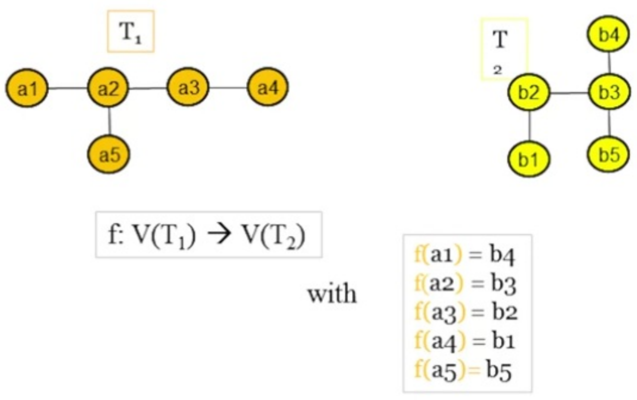
which **preserves adjacency** and **non-adjacency**. That is, if uv is in $E(T_1)$ and $f(u)f(v)$ is in $E(T_2)$.

Respectively:

$$T_1 \cong T_2$$

Two trees with **different degree sequences** are **not isomorphic**. Two trees with the **same degree** sequence **are not necessarily isomorphic**.

4.1. Example



$f: V(T_1) \to V(T_2)$ with

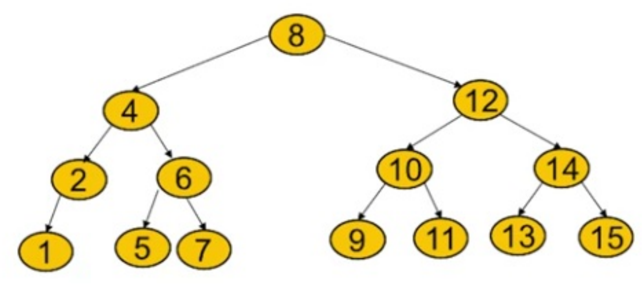
$f(a1) = b4$
$f(a2) = b3$
$f(a3) = b2$
$f(a4) = b1$
$f(a5) = b5$

4.2. Isomorphic Rooted Trees

Two isomorphic trees are **isomorphic as rooted trees** if and only if there is a **bijection** that maps the **root** of one tree to the root of the other. Isomorphic trees **may** or **may not** be isomorphic as **rooted trees**.

5. Binary Search Trees

A binary search tree is a **binary tree** in which the vertices are **labelled** with items so that a **label of a vertex is greater than** the labels of all vertices in the **left subtree** of this vertex and **is less than** the labels of all vertices in the **right subtree** of this vertex.



5.1. Application

The usage of binary search trees apply in the case where we want to **store a modifiable collection** in a **computer's memory** and be able to **search, insert** or **remove** elements from the collection in an efficient way.

5.2. Height of the Binary Search Tree

There are two methods that one can use to find the height of a binary search tree, where N is number of nodes in the tree and h is the height (the following formula must be satisfied):

$$2^{h-1} < N + 1 \le 2^h$$

$$\equiv$$

$$h - 1 < \log_2(N+1) \le h$$

or the second method (hint: $\lceil \dots \rceil$ means *ceiling*, i.e round up):

$$\dots$$

$$\equiv$$

$$h = \lceil \log_2(N+1) \rceil$$

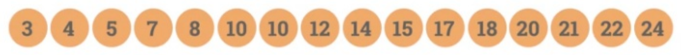
For example, if $N = 15$, then $h = 4$.

$$h = \lceil \log_2(15+1) = \lceil \log_2(16) \rceil = 4$$

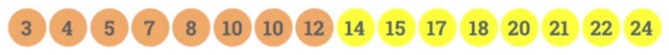
5.3. Binary Search Algorithm

The algorithm starts by comparing the searched element to the middle term of the list. The list is then split into two smaller sub-lists of the same size, or where one of these smaller lists has one fewer term than the other. The search continues by restricting the search to the appropriate sub-list based on the comparison of the searched element and term in the middle.

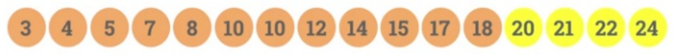
For example, when searching for 21 in the list of:



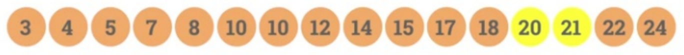
$$\downarrow$$



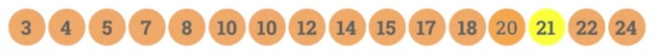
$$\downarrow$$



$$\downarrow$$



$$\downarrow$$



Last updated 2023-03-06 08:18:12 UTC