Trees and Graphs

- Trees and Graphs
- Binary trees
  - definitions: size, height, levels, skinny, complete
- Trees, forests and orchards

Reading: Weiss, Chapter 18

Why Study Trees?

Wood...

"Trees are ubiquitous."

Examples...

genealogical trees	organisational trees
biological hierarchy trees	evolutionary trees
population trees	book classification trees
probability trees	decision trees
induction trees	design trees
graph spanning trees	search trees
planning trees	encoding trees
compression trees	program dependency trees
expression/syntax trees	gum trees
⋮	⋮

Also, *many other data structures are based on trees!*

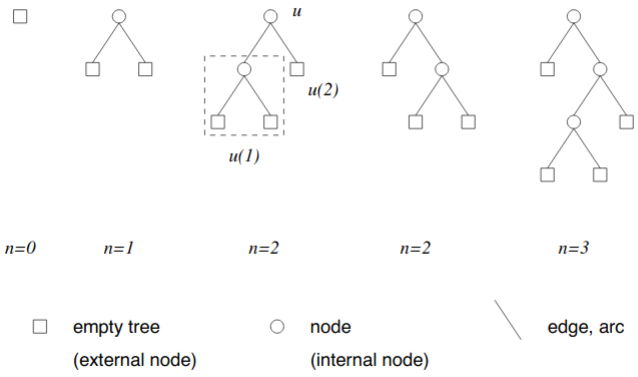
Binary Trees

Definition: A binary *(indexed)* tree T of n nodes, $n \geq 0$, either:

- is empty, if $n = 0$, or
- consists of a *root node* u and two binary trees $u(1)$ and $u(2)$ of n_1 and n_2 nodes respectively such that $n = 1 + n_1 + n_2$.
  - $u(1)$: *first* or *left subtree*
  - $u(2)$: *second* or *right subtree*

  The function u is called the *index*.

recursive definition!



will always draw u(1) to the left of u(2), and children below parents

We will often omit external nodes...

More terminology...
Definition: Let w_1, w_2 be the roots of the subtrees u_1, u_2 of u. Then:

- u is the parent of w_1 and w_2.
- w_1, w_2 are the (*left* and *right*) children of u. $u(i)$ is also called the i^{th} child.
- w_1 and w_2 are *siblings*.

Grandparent, *grandchild*, etc are defined as you would expect.

A leaf is an (internal) node whose left and right subtrees are both empty (external nodes).

The external nodes of a tree define its *frontier*.

In the following assume T is a tree with $n \geq 1$ nodes.
Definition: Node v is a descendant of node u in T if:

1. v is u, or
2. v is a child of some node w, where w is a descendant of u.

Note again recursive definition.
Proper descendant: $v \neq u$
Left descendant: u itself, or descendant of left child of u
Right descendant: u itself, or descendant of right child of u

Q: How would you define "v is *to the left of* u"?

If there is a node w such that v is a left descendant of w and u is a right descendant.

Q: How would you define descendant without using recursion?

Not easily! Have to use 'dots'. There exist nodes $w_1, \ldots, w_m$ such that w_2 is a child of w_1, ...

Size and Height of Binary Trees

The size of a binary tree is the number of (internal) nodes.

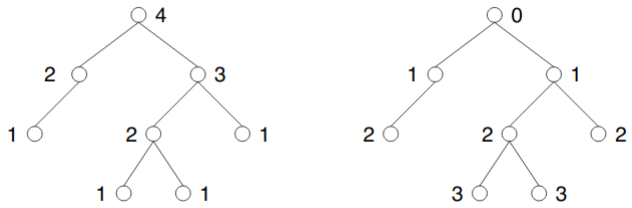
The height of a binary tree T is the length of the longest chain of descendants. That is:

- 0 if T is empty,
- $1 + max(height(T_1), height(T_2))$ otherwise, where T_1 and T_2 are subtrees of the root.

The height of a node u is the height of the subtree rooted at u.

The level of a node is the "distance" from the root. That is:

- 0 for the root node,
- 1 plus the level of the node's parent, otherwise.

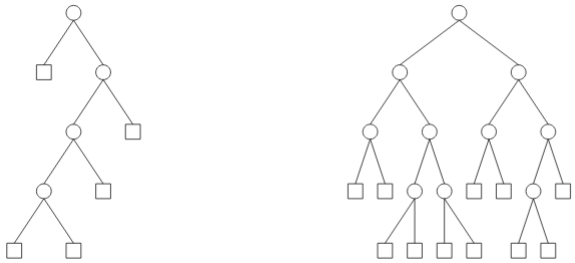


Skinny and Complete Trees

Since we will be doing performance analyses of tree representations, we will be interested in worst cases for height vs size.

skinny — every node has at most one child (internal) node

complete (fat) — external nodes (and hence leaves) appear on at most two adjacent levels

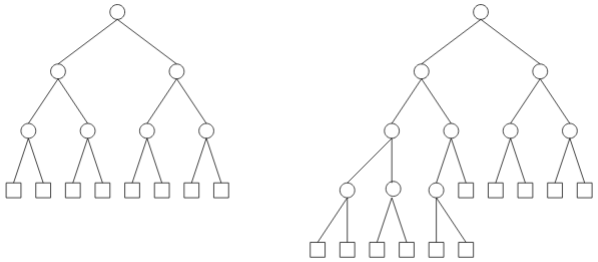


For a given size, skinny trees are the highest possible, and complete trees the lowest possible.

We also identify the following subclasses of complete:

perfect — all external nodes (and leaves) on one level

left-complete — leaves at lowest level are in leftmost position



Relationships between Height and Size

The above relationships can be formalised/extended to the following:

1. A binary tree of height h has size at least h.
2. A binary tree of height h has size at most $2^h - 1$.
3. A binary tree of size n has height at most n.
4. A binary tree of size n has height at least $\lceil \log(n+1) \rceil$.

For each of the above, what class of binary tree represents an upper or lower bound?

1. lower bound is skinny tree.
2. upper bound is perfect tree.
3. upper bound is skinny tree.
4. lower bound is complete (and perfect for those cases where the two are the same, and ceiling has no effect)

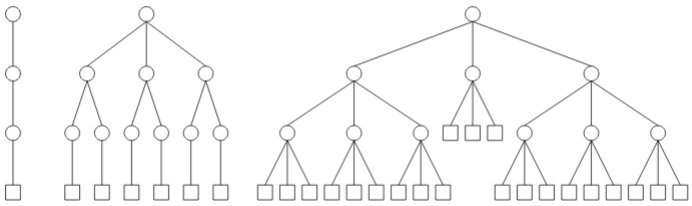
Exercise
Prove (2).
base easy, inductive $2(2^h - 1) + 1 = 2^{h+1} - 2 + 1 = 2^{h+1} - 1$

Trees, Forests, and Orchards

A general *tree* or *multiway (indexed) tree* is defined in a similar way to a binary tree except that a parent node does not need to have exactly two children.

Definition: A *multiway (indexed) tree* T of n nodes, $n \geq 0$, either:

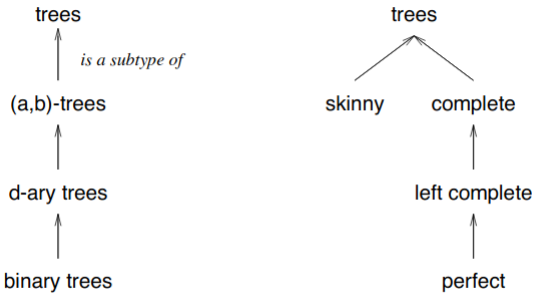
- is empty, if $n = 0$, or
- consists of a root node u, an integer $d \geq 1$ called the *degree* of u, and d multiway trees $u(1), u(2), \ldots, u(d)$ with sizes $n_1, n_2, \ldots, n_d$ respectively such that
$$n = 1 + n_1 + n_2 + \cdots + n_d.$$



A tree is a *d-ary tree* if $d_u = d$ for all (internal) nodes u. We have already looked at binary (2-ary) trees. Above is a unary (1-ary) tree and a ternary (3-ary) tree.

A tree is an (a, b)*-tree* if $a \leq d_u \leq b$, $(a, b \geq 1)$, for all u. Thus the above are all (1,3)-trees, and a binary tree is a (2,2)-tree.

Some trees of tree types!



Forests and Orchards

Removing the root of a tree leaves a collection of trees called a *forest*. An ordered forest is called an *orchard*. Thus:

forest — (possibly empty) set of trees

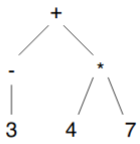
orchard — (possibly empty) queue or list of trees

Annotating Trees

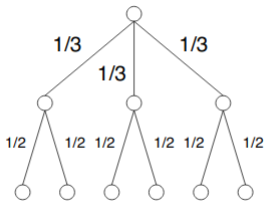
The trees defined so far have no values associated with nodes. In practice it is normally such values that make them useful.

We call these values *annotations* or *labels*.

a *syntax* or *formation* tree for the expression $-3 + 4 * 7$



The following is a probability tree for a problem like: "Of the students entering a language course, one half study French, one third Indonesian, and one sixth Warlpiri. In each stream, half the students choose project work and half choose work experience. What is the probability that Björk, a student on the course, is doing Warlpiri with work experience?"



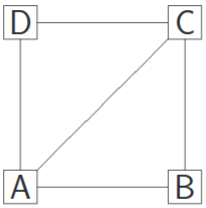
In examples such as this one, it often seems more natural to associate labels with the "arcs" joining nodes. However, this is equivalent to moving the values down to the nodes.

As with the list ADT, we will associate elements with the nodes.

What is a graph?

Definition: A *graph* G consists of a set $V(G)$ called *vertices* together with a collection $E(G)$ of pairs of vertices. Each pair $\{x, y\} \in E(G)$ is called an *edge* of G.

Example If $V(G) = \{A, B, C, D\}$ and
$E(G) = \{\{A, B\}, \{C, D\}, \{A, D\}, \{B, C\}, \{A, C\}\}$ then G is a graph with 4 vertices and 5 edges.



What are graphs used for?

Graphs are used to model a wide range of commonly occurring situations, enabling questions about a particular problem to be reduced to certain well-studied "standard" graph theory questions.

For examples consider the three graphs G_1, G_2 and G_3 defined as follows:

$V(G_1)$ = all the telephone exchanges in Australia, and $\{x, y\} \in E(G_1)$ if exchanges x and y are physically connected by fibre-optic cable.

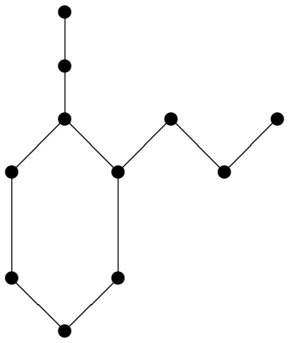
$V(G_2)$ = all the airstrips in the world, and $\{x, y\} \in E(G_2)$ if there is a direct passenger flight from x to y.

$V(G_3)$ = all the people who have ever published a paper in a refereed journal in the world, and $\{x, y\} \in E(G_3)$ if x and y have been joint authors on a paper.

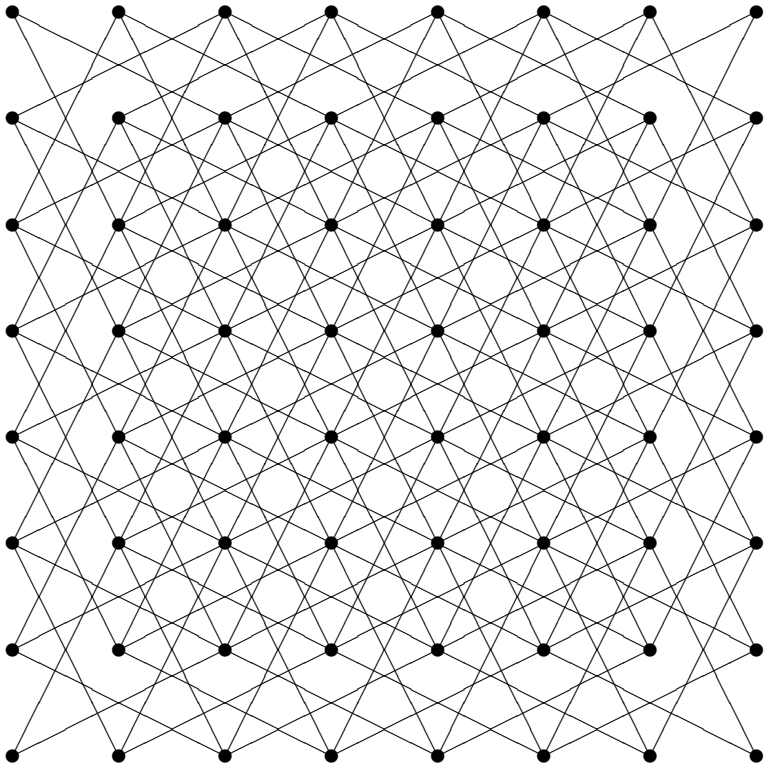
More examples

In computing: A graph can be used to represent processors that are connected via a communication link in a parallel computer system.

In chemistry: The vertices of a graph can be used to represent the carbon atoms in a molecule, and an edge between two vertices represents the bond between the corresponding atoms.

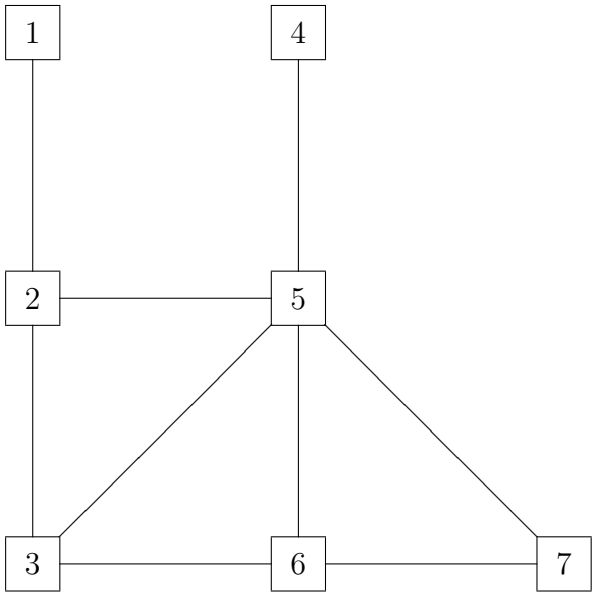


In games: The vertices can be the 64 squares on a chessboard, and the edge that joins two squares can be used to denote the valid movement of a knight from one square to the other.



An example graph

Consider the following graph G_4.



The graph G_4 has 7 vertices and 9 edges.

Basic properties of graphs

Let us consider some of the basic terminology of graphs:

Adjacency If $\{x, y\} \in E(G)$, we say that x and y are *adjacent* to each other, and sometimes write $x \sim y$. The number of vertices adjacent to v is called the *degree* or *valency* of v. The sum of the degrees of the vertices of a graph is even.

Paths A *path* of *length* n in a graph is a sequence of vertices $v_1 \sim v_2 \sim \cdots \sim v_{n+1}$ such that $(v_i, v_{i+1}) \in E(G)$ and vertices $\{v_1, v_2, \ldots, v_{n+1}\}$ are distinct.

Cycles A *cycle* of length n is a sequence of vertices $v_1 \sim v_2 \sim \cdots v_n \sim v_{n+1}$ such that $v_1 = v_{n+1}$, $(v_i, v_{i+1}) \in E(G)$ and therefore only vertices $\{v_1, v_2, \ldots, v_n\}$ are distinct.

Distance The *distance* between two vertices x and y in a graph is the length of the shortest path between them.

Subgraphs

If G is a graph, then a subgraph H is a graph such that

$$V(H) \subseteq V(G)$$

and

$$E(H) \subseteq E(G)$$

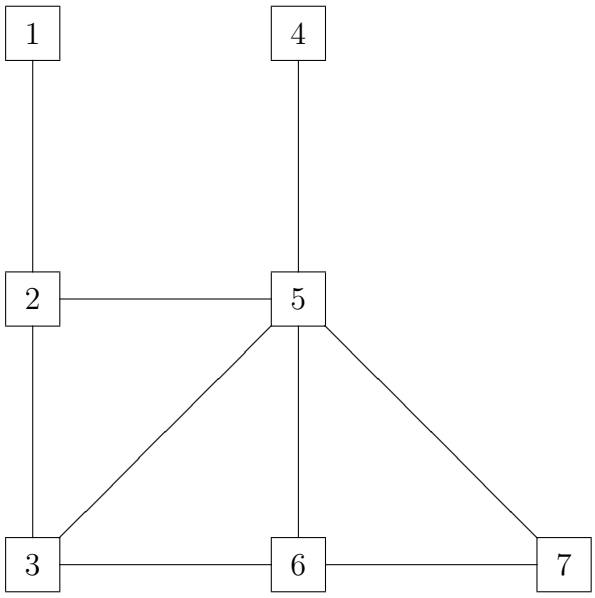
A *spanning* subgraph H has the property that $V(H) = V(G)$ — in other words H has been obtained from G only by removing edges.

An *induced* subgraph H must contain every edge of G whose endpoints lie in $V(H)$ — in other words H has been obtained from G by removing vertices and their adjoining edges.

Counting Exercises

In the graph G_4:

- How many paths are there from 1 to 7?
- How many cycles are there?
- How many spanning subgraphs are there?
- How many induced subgraphs are there?



Connectivity, forests and trees

Connected A graph G is *connected* if there is a path between any two vertices. If the graph is not connected then its connected components are the maximal induced subgraphs that are connected.

Forests A forest is a graph that has no cycles.

Trees A tree is a forest with only one connected component. It is easy to see that a tree with n vertices must have exactly $n - 1$ edges.

The vertices of degree 1 in a tree are called the *leaves* of the tree.

Directed and weighted graphs

There are two important extensions to the basic definition of a graph.

Directed graphs In a directed graph, an edge is an ordered pair of vertices, and hence has a direction. In directed graphs, edges are often called *arcs*.

Directed Tree Each vertex has at most one directed edge leading into it, and there is one vertex (the root) which has a path to every other vertex.

Weighted graphs In a weighted graph, each of the edges is assigned a weight (usually a non-negative integer). More formally we say that a weighted graph is a graph G together with a weight function $w : E(G) \to \mathbf{R}$ (then $w(e)$ represents the weight of the edge e).

Distance in weighted graphs

When talking about weighted graphs, we need to extend the concept of distance.

Definition: In a weighted graph X a path

$$x = x_0 \sim x_1 \sim \cdots \sim x_n = y$$

has *weight*

$$\sum_{i=0}^{i=n-1} w(x_i, x_{i+1}).$$

The *shortest path* between two vertices x and y is the path of minimum weight.