Jeudi 13 juillet 2023 — 11. Trees — 11.1 Introduction to trees

Definition 1: A tree is a connected undirected graph with no simple circuits.
Because a tree cannot have a simple circuit, a tree cannot contain multiple edges or loops. Therefore any tree must be a simple graph.

Example 1

G_1 and G_2 are trees because both are connected graphs with no simple circuits

G_3 is not a tree because e, b, a, d, e is a simple circuit in this graph

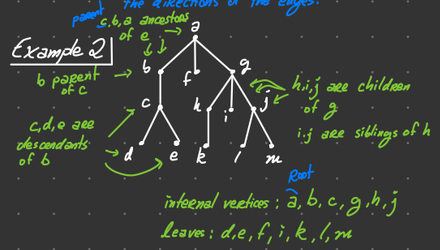
G_4 is not a tree because it is not connected.

These graphs are called forest and have the property that each of their connected components is a tree.

Theorem 1. An undirected graph is a tree if and only if there is a unique simple path between any two of its vertices

Definition 2. A rooted tree is a tree in which one vertex has been designated as the root and every edge is directed away from the root.
Thus, a tree together with its root produces a directed graph called a rooted tree. Rooted tree can also be defined recursively.
The arrow indicating the directions of the edges in a rooted tree can be omitted, because the choice of root determines the directions of the edges.

Example 2



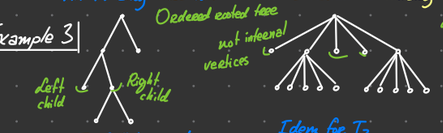
parent: c, b, a ancestors of e
b parent of c
c, d, e are descendants of b
h, i, j are children of g
i, j are siblings of h
root

internal vertices: a, b, c, g, h, j
leaves: d, e, f, i, k, l, m

Rooted trees with the property that all of their internal vertices have the same number of children are used in many different applications (searching, sorting, coding) — Full m-ary tree

Definition 3. A rooted tree is called an m-ary tree if every internal vertex has no more than m children.
The tree is called full m-ary tree if every internal vertex has exactly m children.
An m-ary tree with m = 2 is called a binary tree.

Example 3



T_1 full binary tree because each of its internal vertices has two children

Idem for T_3

T_4 is not a full m-ary tree for any m because some of its internal vertices have two children and others have three children.

Ordered rooted trees can be defined recursively

Trees as models

Example 5 | Saturated Hydrocarbons and Trees

Graphs can be used to represent molecules, where atoms are represented by vertices and bonds between them by edges
↳ Enumerate the isomers of compounds of the form $C_n H_{2n+2}$ (saturated hydrocarbons)
Compounds with the same chemical formula but different chemical properties.

In graph models of saturated hydrocarbons, each carbon atom is represented by a vertex of degree 4, and each hydrogen atom is represented by a vertex of deg. 1.
There are $3n+2$ vertices in a graph representing a compound of the form $C_n H_{2n+2}$.
The number of edges in such a graph is half the sum of degrees of the vertices. Hence, there are $\frac{4n+2n+2}{2} = 3n+1$ edges in this graph.

Because the graph is connected and the number of edges is one less than the number of vertices, it must be a tree.

The nonisomorphic trees with n vertices of deg 4 and $2n+2$ of deg. 1 represent different isomers of $C_n H_{2n+2}$. For instance, there are exactly 2 nonisomorphic trees of this type. Hence, there are exactly 2 different isomers of C_4H_{10}
butane. isobutane (methylpropane)

Properties of trees

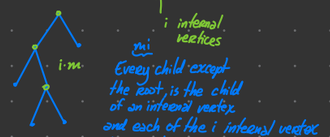
Theorem 2. A tree with n vertices has n−1 edges. (Use mathematical induction to prove this theorem, by removing a vertex v.)

(i) G is connected
(ii) G has no simple circuits
(iii) G has n−1 edges

When two of (i), (ii) and (iii) hold the third condition must also hold, and G must be a tree.

Counting vertices in Full m-ary trees.

Theorem 3. A full m-ary tree with i internal vertices contains $n = mi+1$ vertices (n vertices, m-ary, root; i internal vertices)



mi: Every child except the root, is the child of an internal vertex and each of the i internal vertex has m child

Example 6 | Representing Organizations

The structure of a large organization can be modeled using a rooted tree.
Each vertex in this tree represents a position in the organization.
An edge from one vertex to another indicates that the person represented by the initial vertex is the direct boss of the person represented by the terminal vertex.

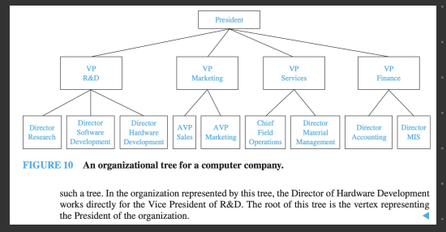


FIGURE 10 An organizational tree for a computer company.

such a tree. In the organization represented by this tree, the Director of Hardware Development works directly for the Vice President of R&D. The root of this tree is the vertex representing the President of the organization.

Example 7 | Computer File Systems

Files in computer memory can be organized into directories.
A directory can contain both files and subdirectories.
The root directory contains the entire file system.
Thus, a file system may be represented by a rooted tree where the root represents the root directory, internal vertices represent subdirectories, and leaves represent ordinary files or empty directories

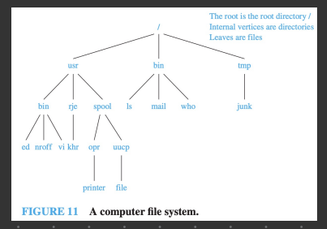


FIGURE 11 A computer file system.

Example 8 | Tree-Connected Parallel Processors

A tree-connected network is another important way to interconnect processors.
The graph representing such a network is a complete binary tree (full binary tree) where every leaf is at the same level. Such network interconnects $n = 2^k - 1$ processors, where k is a positive integer.

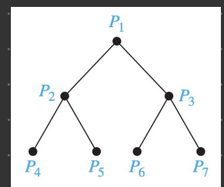


FIGURE 12 A tree-connected network of seven processors.

Tree-connected network can be used for parallel computation
Add eight numbers using three steps.
In the first step, we add x_1 and x_2 using P_4; x_3 and x_4 using P_5; x_5 and x_6 using P_6; x_7 and x_8 using P_7
In the second step, we add x_1+x_2 and x_3+x_4 using P_2; x_5+x_6 and x_7+x_8 using P_3
Finally, in the third step, we add $x_1+x_2+x_3+x_4$ and $x_5+x_6+x_7+x_8$ using P_1

The three steps used to add eight numbers compares favorably to seven step required to add eight numbers serially, where the steps are the addition of one number of the sum of the previous numbers in the list.

Theorem 4. Once one of n, i and l is known, the other two quantities are determined

A full m-ary tree with edge (Th.2)

(I.) n vertices ⟶ $\begin{cases} i = \frac{(n-1)}{m} \iff n = im+1 \text{ (Th.3)} \\ l = \frac{[(m-1)\cdot n + 1]}{m} \end{cases}$ m-ary

$i(n,m) = \frac{(n-1)}{m}$, $l(n,m) = \frac{[(m-1)n+1]}{m}$

(II.) i internal vertices ⟶ $\begin{cases} n(i,m) = im+1 \text{ (Th. 3)} \\ l(i,m) = (m-1)\cdot i + 1 \end{cases}$

(III.) l leaves ⟶ $\begin{cases} n = \frac{(ml-1)}{m-1} \\ i = \frac{(l-1)}{m-1} \end{cases}$

The three part theorem can all be proved using $n = im+1$ (Th 3) and $n = l+i$ (each vertex is either a leaf or an internal vertex)

Proofs (I.) n vertices given: $n = im+1 \iff i = \frac{n-1}{m}$ ∎
$n = l+i \iff l = n-i = n-\frac{(n-1)}{m} = \frac{m\cdot n-(n-1)}{m} = \frac{m\cdot n-n+1}{m} = \frac{(m-1)n+1}{m}$

(II.) i internal vertices given: $n = i\cdot m+1$ ∎
$n = l+i \iff l = n-i = im+1-i = im-i+1 = (m-1)i+1$

(III.) l leaves given:
$n = im+1 \longrightarrow n = (n-l)m+1 \iff n = nm-lm+1$
$n = l+i \iff i = n-l$
$\iff n-nm = -lm+1$
$\iff n(1-m) = -lm+1$
$\iff n = \frac{-lm+1}{(1-m)} = \frac{ml-1}{m-1}$ ∎
$n = im+1$, $n = l+i$ ⟶ $l+i = im+1 \iff i-im = 1-l$
$\iff i(1-m) = 1-l$
$\iff i = \frac{1-l}{1-m} = \frac{l-1}{m-1}$ ∎

Example 9 Suppose that someone starts a chain letter. Each person who receives the letter is asked to send it on to four other people. Some people do this, but others do not send send any letters to anyone. A total of 100 people receive the letter, including the first person, if no one receives more than one letter and if the chain letter ends after there have been 100 people who read it but did not send it out?

(I.) How many people have seen the letter, including the first person, if no one receives more than one letter and if the chain letter ends after there have been 100 people who read it but did not send it out?

The chain letter can be represented using a 4-ary tree.
The internal vertices correspond to people who sent out the letter, and the leaves correspond to people who did not sent it out.
Because 100 people did not sent out the letter, the number of leaves in this rooted tree is $l = 100$.
Hence, the number of people who have seen the letter is

$$\begin{cases} n = im + 1 \\ n = l + i \end{cases} \Leftrightarrow i = n - l$$

$$\Rightarrow n = (n-l)m + 1$$
$$\Leftrightarrow n = nm - lm + 1$$
$$\Leftrightarrow n - nm = -lm + 1$$
$$\Leftrightarrow n(1-m) = -lm + 1$$
$$\Leftrightarrow n = \frac{-lm+1}{(1-m)} = \frac{ml-1}{m-1} = \frac{4 \cdot 100 - 1}{100 - 1} = \frac{399}{99} = 133$$

(II.) How many people sent out the letter?
The internal vertices correspond to the people who send out the letter

So, $n = l + i \Leftrightarrow i = n - l = 133 - 100 = 33$

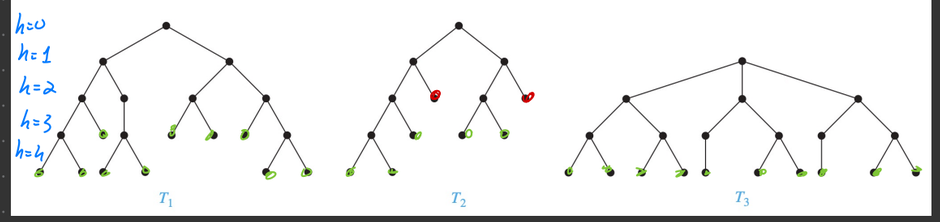
Balanced m-ary trees:

Level: The level of a vertex v in a rooted tree is the length of the unique path from the root to this vertex. The level of the root is defined to be zero.

Height: The height of a rooted tree is the maximum of levels of vertices.
In other words, the height of a rooted tree is the length of the longest path from the root any vertex.

~~**Def**~~ A rooted m-ary tree of height h is balanced if all leaves are at levels h or $h-1$

EXAMPLE 11 Which of the rooted trees shown in Figure 14 are balanced?



T_1 is balanced because all its leaves are at levels 3 and 4

T_2 is not balanced, because it has leaves at levels 2, 3 and 4

T_3 balanced, because all its leaves are at level 3

A bound for the number of leaves in an m-ary tree

an upper bound

Theorem 5. There are at most m^h leaves in an m-ary tree of height h

Proof with mathematical induction on the height and subtrees.

Corollary 1. If an m-ary tree of height h has l leaves, then $h \geq \lceil \log_m l \rceil$
If the m-ary tree is full and balanced, then $h = \lceil \log_m l \rceil$

if balanced, each leaf is at level h or $h-1$
$m^{h-1} < l \leq m^h \Leftrightarrow h-1 < \log_m l \leq h$

$m^h = l \Leftrightarrow h = \log_m l$ → upper bound so $h \geq \lceil \log_m l \rceil$

11.2 Applications of trees

Three problems that can be studied using trees:

(I.) How should items in a list be stored so that an item can easily be located? } Binary Search Trees

(II.) What series of decisions should be made to find an object with a certain property in a collection of objects of a certain type? } Decision Trees / Game Trees

(III.) How should a set of characters be efficiently coded by bit strings? } Prefix code ↳ Huffman coding

(I.) Binary Search Trees

Searching for items in a list is one of the most important tasks that arise in computer science.
find items efficiently when the items are totally ordered

This can be accomplished through the use of a binary search tree
If a binary search tree is balanced, locating or adding an item requires no more than $\lceil \log_2(n+1) \rceil$ (Corollary 1)
algorithm have been devised
that rebalance binary search trees as items are added

(II.) Decision Trees

A rooted tree in wich each internal vertex corresponds to a decision, with a subtree at these vertices for each possible outcome of the decision, is called a decision tree.
The possible solutions of the problem correspond to the paths to the leaves of this rooted tree.

Example 3 Suppose there are seven coins, all with the same weight, and a counterfeit coin that weight less than the others. How many weighings are necessary using a balance scale to determine which of the eight coins is the counterfeit one?

There are three possibilities for each weighing on a balance scale:
(I.) The two pans can have equal weight
(II.) The first pan can be heavier
(III.) The second pan can be heavier

Consequently, the decision tree for the sequence of weighings is a 3-ary tree.
There are at least 8 leaves in the decision tree because there are eight possible outcomes. (because each of the eight coins can be the counterfeit lightee coin)
and each possible outcome must be represented by at least one leaf.
It follows that the height of the decision tree is at least $\lceil \log_3 8 \rceil = 2$.
Hence, at least two weighings are needed.

EXAMPLE 4 We display in Figure 4 a decision tree that orders the elements of the list a, b, c.

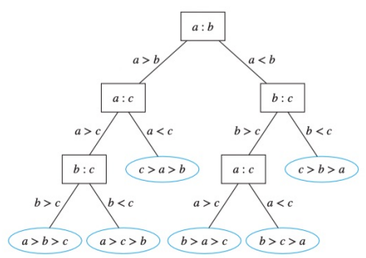
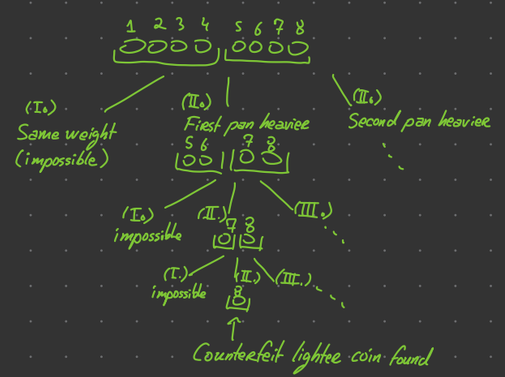


FIGURE 4 A decision tree for sorting three distinct elements.



Theorem 1. A sorting algorithm based on a binary comparisons requires at least $\lceil \log_2 n! \rceil$ comparisons.

Corollary 1. The number of comparisons used by a sorting algorithm to sort n elements based on binary comparisons is $\Omega(n \log n)$

No other algorithm has better worst-case complexity

bonne inf. ou égale → we know that $\lceil \log n! \rceil$ is $\Theta(n \log n)$

A function that is $\Omega(\log n!)$ is also $\Omega(n \log n)$ because $\log n!$ is $\Theta(n \log n)$

Theorem 2. The average number of comparisons used by a sorting algorithm to sort n elements based on binary comparisons is $\Omega(n \log n)$.

Prefix Codes

Consider the problem of using bit strings to encode the letters of the English alphabet (No distinction between lower and upper case)

Is it possible to find a coding scheme of these letters such that, when data are coded, fewer bits are used? (We can save memory and reduce transmittal time if this can be done)

Letters that occur more frequently should be encoded using short bit strings, and longer bit strings should be used to encode rarely occuring letters.

We can represent each letter with a bit string of length five, because there are only 26 letters and there are 32 bit strings of length 5

For instance, if e were encoded with 0,
a with 1,
t with 01,
then the string 0101 could correspond to eat, tea, eaea, or tt.

One way to ensure that no bit string correspond to more than one sequence of letters is to encode letters so that the bit string for a letter never occures as the first part of the bit string for another letter.

Codes with this property are called **prefix codes**.

- e as 0
- a as 10
- t as 11

1011D is the encoding for ate (1011D: a t e)

A prefix code can be represented using a **binary tree**, where the characters are the labels of the leaves in the tree.

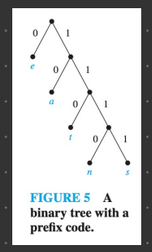


FIGURE 5 A binary tree with a prefix code.

11111011100 (S a n e)

Huffman Coding

probabilities of occurrences

An algorith that takes as input the frequencies of symbols in a string and produces, as output a prefix code that encodes the string using the fewest possible bits among all possible binary prefix codes for these symbols.

Huffman coding is a fundamental algorith in data compression, the subject devoted to reducing the number of bits required to represent information. (text, audio, image files)

Given symbols and their frequencies, our goal is to construct a rooted binary tree whose the symbols are the labels of the leaves.

Huffman coding is a greedy algorithm.

(I.) The algorith begins with a forest of trees each containing one vertex.
(II.) At each step, we combine two trees having the least total weight into a single tree
(III.) The algorithm is finished when it has constructed a tree, that is: when the forest is reduced to a single tree.

Game Trees

Trees can be used to analyze certain types of games such as tic-tac-toe, nim, checkers and chess

In each of these games, (I.) two players take turns making moves
(II.) Each player knows the moves made by the other player
(III.) No element of chance enters into the game.

The vertices represent the positions that a game can be in as it progresses
The edges represent legal moves between these positions.
The root is the starting position
Even level: it is the first player's move (rep. by boxes)
Odd: it is the second player's move (rep. by circles)
The leaves represent the final position of a game.
- 1 win by the first player
- -1 win by the second
- 0 draw

Nim

The player removing the last stone loses

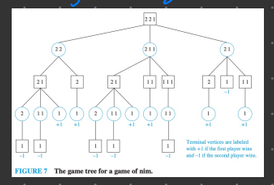


FIGURE 7 The game tree for a game of nim.

Tic-tac-toe

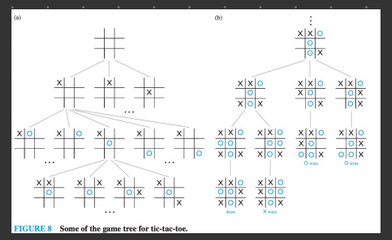


FIGURE 8 Some of the game tree for tic-tac-toe.

Chess

10^{100} vertices

Combinatorial game theory

11.3 Tree traversal

Ordered rooted trees are often used to store information.

We need procedures for visiting each vertex of an ordered rooted tree to access data

Universal Address Systems

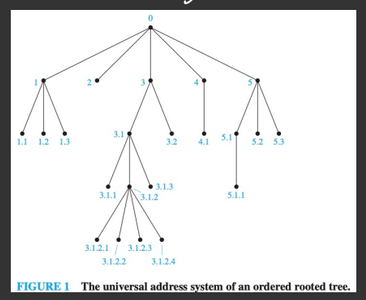


FIGURE 1 The universal address system of an ordered rooted tree.

Traversal Algorithm

Procedures for systematically visiting every vertex of an ordered rooted tree are called traversal algorithms

Three of the most commonly used algorithms: (I.) Preorder traversal
(II.) Inorder traversal
(III.) Postorder traversal

Each of these algorithms can be defined recursively

(I.) Preorder (Same order as the ordering obtained using a universal adress system)

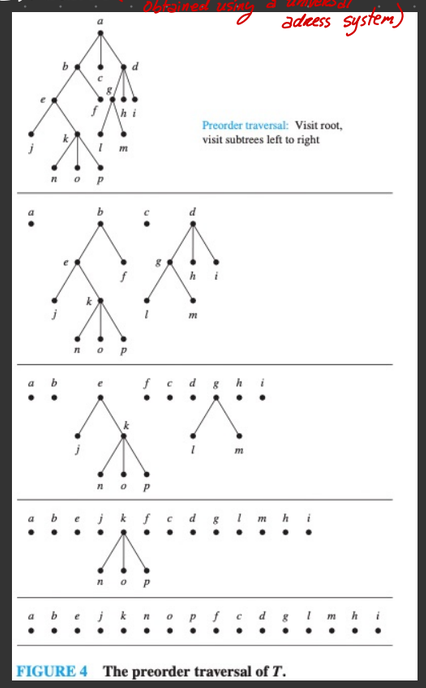


FIGURE 4 The preorder traversal of T.

(II.) Inorder

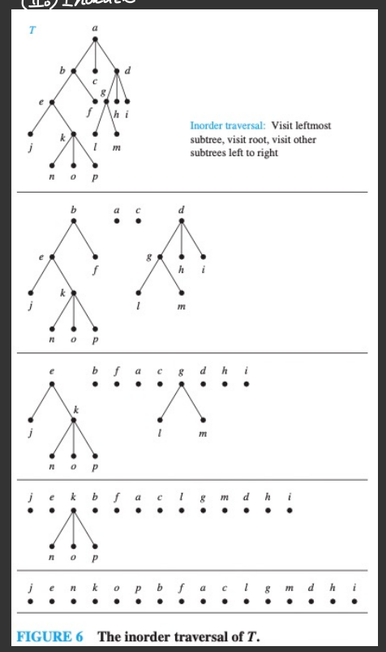


FIGURE 6 The inorder traversal of T.

(III.) Postorder

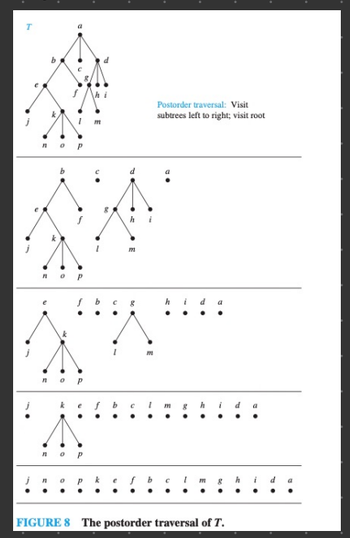


FIGURE 8 The postorder traversal of T.

structure of a full ordered m-ary tree

Both the preorder traversal and the postorder traversal encode the structure of an ordered rooted tree when the number of children of each vertex is specified

if not specified, neither a preorder traversal nor a postorder traversal encodes the structure of an ordered rooted tree.

Uses of Inorder, Preorder, and Postorder Traversals

Tree traversals have many applications and they play a key role in the implementation of many algorithms

Guidance about the practical use of tree traversals.

A general peincipal for deciding the teaversal to use is to explore the vertices of interest as soon as possible

(I.) Preorder traversal is the best choice for applications where internal vertices must be explored before leaves.

Preorder traversals are also used to make copies of a binary search tree.

Preorder traversal of the vertices of the relevent family tree produces the order of succession to the throne, once deceased members are removed.

(II.) Postorder traversal is the best choice for applications where leaves need to be explored before internal vertices.

Postorder traversal explores leaves before internal vertices, so it is the best choice for deleting a tree because the vertices below the root of a subtree can be removed before the root of the subtree

*Topological sorting is an example of an algorithm that can be efficiently implemented using postorder traversal.

(III.) An inorder traversal of a binary search tree visits the vertices in ascending order of their key values.

Such a traversal creates a sorted list of the data in a binary tree.

11.4 Spanning Trees

Definition 1 Let G be a simple graph. A spanning tree of G is a subgraph of G that is a tree containing every vertex of G

A simple graph with a spanning tree must be connected.

The converse is also true, that is, every connected simple graph has a spanning tree.

Theorem 1. A simple graph is connected iff it has a spanning tree.

Example 2 | IP Multicasting

Spanning trees play an important role in multicasting over Internet Protocol (IP) networks. To send data from a source computer to multiple receiving computers, each of which is a subnetwork, data could be sent seperately to each computer. This type of networking, called unicasting, is inefficient, because many copies of the same data are transmitted over the network. To make the transmission of data to multiple receiving computers more efficient, IP multicasting is used. With IP multicasting, a computer sends a single copy of data over the network, and as data reaches intermediate routers, the data are forwarded to one or more other routers so that ultimately all receiving computers in their various subnetworks receive these data. For data to reach receiving computers as quickly as possible, there should be no loops in the path that data take through the network. That is, once data have reached a particular router, data should never return to this router. To avoid loops, the multicast routers use algorithms to construct a spanning tree in the graph that has the multicast source, the routers, and the subnetworks containing receiving computers as vertices, with edges representing the links between computers and/or routers. The root of this spanning tree is the multicast source. The subnetworks containing receiving computers are leaves of the trees.

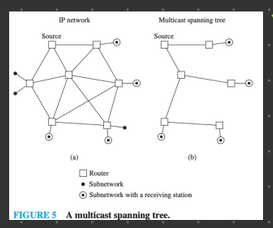


FIGURE 5 A multicast spanning tree.

Instead of constructing spanning trees by removing edges, spanning trees can be built up by successively adding edges.

Two algorithms based on this principle: (I.) Depth-First Search (Backtracking) $O(n^2)$ n vertices or $O(e)$ e edges — Use when graph is dense to quickly reach vertices far away from the root.

Chapter 10 Graph...

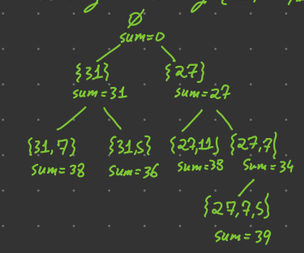
(II.) Breadth-first search $O(n^2)$ or $O(e)$

Example 8 | Sum of Subsets

Consider this proble. Given a set of positive integers $x_1, x_2, \ldots, x_n$, find a subset of this set of integers that has M as its sum. How can backtracking be used to solve this problem?

We start with a sum with no terms. We build up the sum by successively adding terms. An integer in the sequence is included if the sum remains less than M when this integer is added to the sum. If a sum is reached such that the addition of any term is greater than M, backtrack by dropping the last term of the sum.

E.g. Find a sum equal to 39 using backtracking $\{31, 27, 15, 11, 7, 5\}$



We can easily modify both DFS and BFS so that they can both run given a directed graph as input.

However, the output will not necessarily be a spanning tree, but rather a spanning forest.

Example 10 | Web Crawlers

Search engines have developed sophisticated strategies for web crawlers that take advantage of BFS and DFS (Google bot).

The use BFS, starting on high quality web pages and explore all links on these pages. However, the quality of pages reached decreases as the web crawl continues.

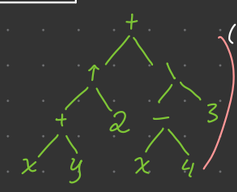
DFS can be used to find candidates for high quality pages. Also, DFS can be used to reach parts of the web not reached by BFS when it is restricted to a particular number of levels.

Infix, Prefix, and Postfix Notation

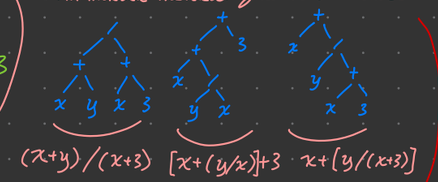
We can represent complicated expressions, such as compound propositions, combinations of set, and arithmetic expressions using ordered rooted trees

An ordered rooted tree can be used to represent such expressions, where the internal vertices represent operations, and the leaves represent the variables or numbers

Example 5 | What is the ordered rooted tree that represents the expression $((x+y)\uparrow 2)+((x-4)/3)$?



(I.) An inorder traversal gives the same order as they originally occureed, except for unary operations



All lead to the infix expressions $x+y/x+3$

To make such expression unambiguous it is necessary to include parentheses in the inorder traversal whenever we encounter an operation.

The fully parenthesized expression obtained in this way is said to be in infix form

(II.) We obtain the prefix form of an expression when we traverse its rooted tree in preorder. Expression written in prefix form are said to be in Polish notation. An expression in prefix notation is unambiguous, so no parentheses are needed in such an expression.

Example 6 | What is the prefix form for $((x+y)\uparrow 2)+((x-4)/3)$?

$+,\uparrow,+,x,y,2,/,-,x,4,3$

Example 7 | What is the value of $+-*235/\uparrow 234$?

The steps used to evaluate this expression by working right to left, and performing operations using the operands on the right.

$+-*235/\uparrow 234$

$2\uparrow 3=8$

$+-*235/84$

$8/4=2$

$+-*2352$

$2*3=6$

$+-652$

$6-5=1$

$+12$

$1+2=3$

Value of expression: 3

(III.) We obtain the postfix form of an expression by traversing its binary tree in postorder.

Expression written in postfix form are said to be in Reverse Polish notation.

Expression in reverse Polish notation are unambiguous. Reverse polish notation was extensively used in electronic calculators in the 1970s and 1980s.

Example 8 | What is the postfix form of $((x+y)\uparrow 2)+((x-4)/3)$?

The postfix form is obtained by carrying out a postorder traversal of the binary tree.

$x,y,+,2,\uparrow,x,4,-,3,/,+$

Example 9 | What is the value of the postfix expression $723*-4\uparrow 93/+$?

The steps used to evaluate this expression by starting at the left and carrying out operations when two operands are followed by an operator.

$723*-4\uparrow 93/+$

$2*3=6$

$76-4\uparrow 93/+$

$7-6=1$

$14\uparrow 93/+$

$1\uparrow 4=1$

$193/+$

$9/3=3$

$13+$

$1+3=4$

Value of expression: 4

Example 10 | Find the ordered rooted tree representing the compound proposition $(\neg(p\wedge q))\leftrightarrow(\neg p\vee\neg q)$. Then use this rooted tree to find the prefix, postfix, and infix form of the expression.

(I.) Prefix: $\leftrightarrow\neg\wedge pq\vee\neg p\neg q$

(II.) Postfix: $pq\wedge\neg p\neg q\vee\leftrightarrow$

(III.) Infix: $(\neg(p\wedge q))\leftrightarrow((\neg p)\vee(\neg q))$

Such expressions are especially useful in the construction of compilers

11.5 Minimum Spanning Trees

Definition 1 A minimum spanning tree in a connected weighted graph is a spanning tree that has the smallest possible sum of weights of its edges

Two algorithms for constructing minimum spanning trees are greedy algorithms that produce optimal solutions:

(I.) Prim's algorithm: 1. Begin by choosing any edge with smallest weight, putting it into the spanning tree.
2. Successively add to the tree edges of minimum weight that are incident to a vertex already in the tree, never forming a simple circuit with those edges already in the tree.
3. Stop when $n-1$ edges have been added.

$O(m \log n)$

(II.) Kruskal's algorithm: 1. List in order all the edges
2, 3 same as Prim's algorithm

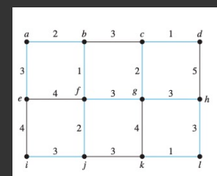
$O(m \log m)$

m edges
n vertices

Used for sparse graphs, that is,
where m is very small compared to $C(n,2) = \frac{n(n-1)}{2}$
the total number of possible edges in an undirected graph with n vertices.
Otherwise, there is little difference in the complexity of these two algorithms.

Choice	Edge	Cost
1	{Chicago, Atlanta}	$ 700
2	{Atlanta, New York}	$ 800
3	{Chicago, San Francisco}	$1200
4	{San Francisco, Denver}	$ 900
	Total:	$3600

FIGURE 2 **A minimum spanning tree for the weighted graph in Figure 1.**



(a)

Choice	Edge	Weight
1	{b, f}	1
2	{a, b}	2
3	{f, j}	2
4	{a, e}	3
5	{i, j}	3
6	{f, g}	3
7	{c, g}	2
8	{c, d}	1
9	{g, h}	3
10	{h, l}	3
11	{k, l}	1
	Total:	24

(b)

FIGURE 4 **A minimum spanning tree produced using Prim's algorithm.**