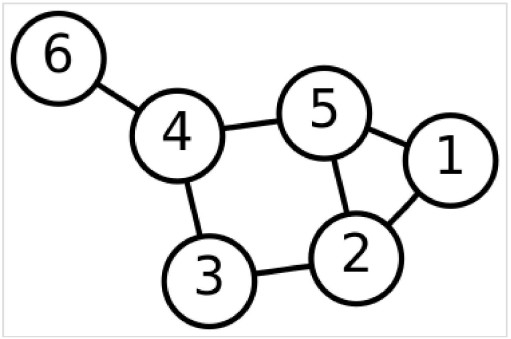
WIKIPEDIA
The Free Encyclopedia

Node (computer science)

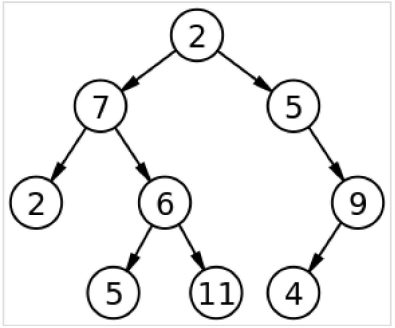
A **node** is a basic unit of a data structure, such as a linked list or tree data structure. Nodes contain data and also may link to other nodes. Links between nodes are often implemented by pointers.

Nodes and trees



In graph theory, the image provides a simplified view of a network, where each of the numbers represents a different node.

Nodes are often arranged into tree structures. A node represents the information contained in a single data structure. These nodes may contain a value or condition, or possibly serve as another independent data structure. Nodes are represented by a single parent node. The highest point on a tree structure is called a root node, which does not have a parent node, but serves as the parent or 'grandparent' of all of the nodes below it in the tree. The height of a node is determined by the total number of edges on the path from that node to the furthest leaf node, and the height of the tree is equal to the height of the root node.[1] Node depth is determined by the distance between that particular node and the root node. The root node is said to have a depth of zero.[2] Data can be discovered along these network paths.[3] An IP address uses this kind of system of nodes to define its location in a network.



A simple binary tree of size 9 and height 3, with a root node whose value is 2. The above tree is unbalanced and not sorted.

Definitions

- **Child**: A child node is a node extending from another node. For example, a computer with internet access could be considered a child node of a node representing the internet. The inverse relationship is that of a **parent node**. If node *C* is a child of node *A*, then *A* is the parent node of *C*.
- **Degree**: the degree of a node is the number of children of the node.
- **Depth**: the depth of node *A* is the length of the path from *A* to the root node. The root node is said to have depth 0.
- **Edge**: the connection between nodes.
- **Forest**: a set of trees.
- **Height**: the height of node *A* is the length of the longest path through children to a leaf node.
- **Internal node**: a node with at least one child.
- **Leaf node**: a node with no children.
- **Root node**: a node distinguished from the rest of the tree nodes. Usually, it is depicted as the highest node of the tree.
- **Sibling nodes**: these are nodes connected to the same parent node.

Markup languages

Another common use of node trees is in web development. In programming, XML is used to communicate information between computer programmers and computers alike. For this reason XML is used to create common communication protocols used in office productivity software, and serves as the base for the development of modern web markup languages like XHTML. Though similar in how it is approached by a programmer, HTML and CSS is typically the language used to develop website text and design. While XML, HTML and XHTML provide the language and expression, the DOM serves as a translator.[4]

Node type

Different types of nodes in a tree are represented by specific interfaces. In other words, the node type is defined by how it communicates with other nodes. Each node has a node type property, which specifies the type of node, such as sibling or leaf. For example, if the node type property is the constant properties for a node, this property specifies the type of the node. So if a node type property is the constant node ELEMENT_NODE, one can know that this node object is an object Element. This object uses the Element interface to define all the methods and properties of that particular node.

Different W3C World Wide Web Consortium node types and descriptions:

- **Document** represents the entire document (the root-node of the DOM tree)
- **DocumentFragment** represents a "lightweight" Document object, which can hold a portion of a document
- **DocumentType** provides an interface to the entities defined for the document
- **ProcessingInstruction** represents a processing instruction
- **EntityReference** represents an entity reference
- **Element** represents an element
- **Attr** represents an attribute
- **Text** represents textual content in an element or attribute
- **CDATASection** represents a CDATA section in a document (text that will NOT be parsed by a parser)
- **Comment** represents a comment
- **Entity** represents an entity
- **Notation** represents a notation declared in the DTD

NodeType	Named constant
1	ELEMENT_NODE
2	ATTRIBUTE_NODE
3	TEXT_NODE
4	CDATA_SECTION_NODE
5	ENTITY_REFERENCE_NODE
6	ENTITY_NODE
7	PROCESSING_INSTRUCTION_NODE
8	COMMENT_NODE
9	DOCUMENT_NODE
10	DOCUMENT_TYPE_NODE
11	DOCUMENT_FRAGMENT_NODE
12	NOTATION_NODE

Node object

A node object is represented by a single node in a tree. It can be an element node, attribute node, text node, or any type that is described in section "node type". All objects can inherit properties and methods for dealing with parent and child nodes, but not all of the objects have parent or child nodes. For example, with text nodes that cannot have child nodes, trying to add child nodes results in a DOM error.

Objects in the DOM tree may be addressed and manipulated by using methods on the objects. The public interface of a DOM is specified in its application programming interface (API). The history of the Document Object Model is intertwined with the history of the "browser wars" of the late 1990s between Netscape Navigator and Microsoft Internet Explorer, as well as with that of JavaScript and JScript, the first scripting languages to be widely implemented in the layout engines of web browsers.

See also

- Vertex (graph theory)

References

1. "tree (data structure)" (http://xlinux.nist.gov/dads//HTML/tree.html). National Institute of Standards and Technology. Archived (https://web.archive.org/web/20141124223717/http://xlinux.nist.gov/dads/HTML/tree.html) from the original on 2014-11-24.
2. Teukolsky, Roselyn (2013). *Barron's AP Computer Science A* (https://archive.org/details/apcomputerscienc0006teuk). Barron's. ISBN 978-1-4380-0152-4.
3. "Simply Scheme: Introducing Computer Science ch 18: Trees" (http://www.eecs.berkeley.edu/~bh/ssch18/trees.html). College Of Engineering, University of California, Berkeley. Archived (https://web.archive.org/web/20131222183836/http://www.eecs.berkeley.edu/~bh/ssch18/trees.html) from the original on 2013-12-22.
4. "XML DOM Introduction" (https://web.archive.org/web/20140611052725/https://www.w3schools.com/dom/dom_intro.asp). W3Schools. Archived from the original (https://www.w3schools.co

m/dom/dom_intro.asp) on 2014-06-11. Retrieved 2018-04-07.

External links

- Data Trees as a Means of Presenting Complex Data Analysis (http://www.community-of-knowledge.de/beitrag/data-trees-as-a-means-of-presenting-complex-data-analysis/) by Sally Knipe
- STL-like C++ tree class (http://tree.phi-sci.com) Archived (https://web.archive.org/web/20201126210313/http://tree.phi-sci.com/) 2020-11-26 at the Wayback Machine
- Description of tree data structures from ideainfo.8m.com (https://web.archive.org/web/20180306223544/http://ideainfo.8m.com/)
- WormWeb.org: Interactive Visualization of the *C. elegans* Cell Tree (http://wormweb.org/celllineage) - Visualize the entire cell lineage tree of the nematode *C. elegans* (javascript)