tree traversals

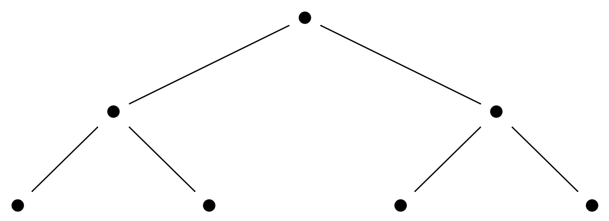
Canonical name	TreeTraversals
Date of creation	2013-03-22 12:29:00
Last modified on	2013-03-22 12:29:00
Owner	mps (409)
Last modified by	mps (409)
Numerical id	10
Author	mps (409)
Entry type	Algorithm
Classification	msc 05C05
Synonym	inorder traversal
Related topic	Tree
Defines	preorder traversal
Defines	postorder traversal
Defines	in-order traversal

A *tree traversal* is an algorithm for visiting all the `http://planetmath.org/Graph`nodes in a rooted tree exactly once. The constraint is on rooted trees, because the root is taken to be the starting point of the traversal. A traversal is also defined on a forest in the sense that each tree in the forest can be iteratively traversed (provided one knows the roots of every tree beforehand). This entry presents a few common and simple tree traversals.

In the description of a `http://planetmath.org/forest`tree, the notion of rooted-subtrees was presented. Full understanding of this notion is necessary to understand the traversals presented here, as each of these traversals depends heavily upon this notion.

In a traversal, there is the notion of *visiting* a node. Visiting a node often consists of doing some computation with that node. The traversals are defined here without any notion of what is being done to visit a node, and simply indicate where the visit occurs (and most importantly, in what order).

Examples of each traversal will be illustrated on the following binary tree.

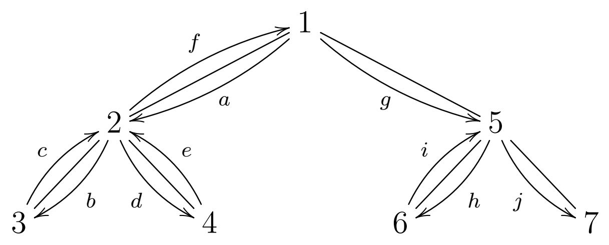


Vertices will be numbered in the order they are visited, and edges will be drawn with arrows indicating the path of the traversal.

Preorder Traversal

Given a rooted tree, a *preorder traversal* consists of first visiting the root, and then executing a preorder traversal on each of the root's children (if any).

For example



The term *preorder* refers to the fact that a node is visited *before* any of its descendents. A preorder traversal is defined for any rooted tree. As pseudocode, the preorder traversal is

```
PreorderTraversal($x$, Visit)
**Input**: A node $x$ of a binary tree, with children left($x$) and right($x$), and some computation Visit
**Output**: Visits nodes of subtree rooted at $x$ in a preorder traversal
**Procedure**:
Visit($x$)
PreorderTraversal(left($x$), Visit)
PreorderTraversal(right($x$), Visit)
```

Postorder Traversal

Given a rooted tree, a *postorder traversal* consists of first executing a postorder traversal on each of the root's children (if any), and then visiting the root.

For example

$$\begin{tikzcd}
 & & & 7 \arrow[dll, bend left, "a"] \arrow[drr, "g"] & & & \\
 & 3 \arrow[dl, "b"] \arrow[dr, "d"] \arrow[urr, bend left, "f"] & & & & 6 \arrow[dl, "h"] \arrow[dr, "j"] & \\
1 \arrow[ur, bend left, "c"] & & 2 \arrow[ul, bend right, "e"'] & & 4 \arrow[ur, bend left, "i"] & & 5
\end{tikzcd}$$

As with the preorder traversal, the term *postorder* here refers to the fact that a node is visited *after* all of its descendents. A postorder traversal is defined for any rooted tree. As pseudocode, the postorder traversal is

```
PostorderTraversal($x$, Visit)
**Input**: A node $x$ of a binary tree, with children left($x$) and right($x$), and some computation Visit
**Output**: Visits nodes of subtree rooted at $x$ in a postorder traversal
**Procedure**:
PostorderTraversal(left($x$), Visit)
PostorderTraversal(right($x$), Visit)
Visit($x$)
```

In-order Traversal

Given a binary tree, an *in-order traversal* consists of executing an in-order traversal on the root's left child (if present), then visiting the root, then executing an in-order traversal on the root's right child (if present). Thus all of a root's left descendents are visited before the root, and the root is visited before any of its right descendents.

For example

$$\begin{tikzcd}
 & & & 4 \arrow[dll, "a"] \arrow[drr, "g"] & & & \\
 & 2 \arrow[dl, "b"] \arrow[dr, "d"] \arrow[urr, bend left, "f"] & & & & 6 \arrow[dl, "h"] \arrow[dr, "j"] & \\
1 \arrow[ur, bend left, "c"] & & 3 \arrow[ul, bend right, "e"'] & & 5 \arrow[ur, bend left, "i"] & & 7
\end{tikzcd}$$

As can be seen, the in-order traversal has the wonderful property of traversing a tree from left to right (if the tree is visualized as it has been drawn here). The term *in-order* comes from the fact that an in-order traversal of a binary search tree visits the data associated with the nodes in sorted order. As pseudocode, the in-order traversal is

```
InOrderTraversal($x$, Visit)
Input: A node $x$ of a binary tree, with children left($x$) and right($x$), and some computation Visit
Output: Visits nodes of subtree rooted at $x$ in an in-order traversal
Procedure:
InOrderTraversal(left($x$), Visit)
Visit($x$)
InOrderTraversal(right($x$), Visit)
```