Lecture-14

Special Forms of Binary Trees

There are a few special forms of binary tree worth mentioning.

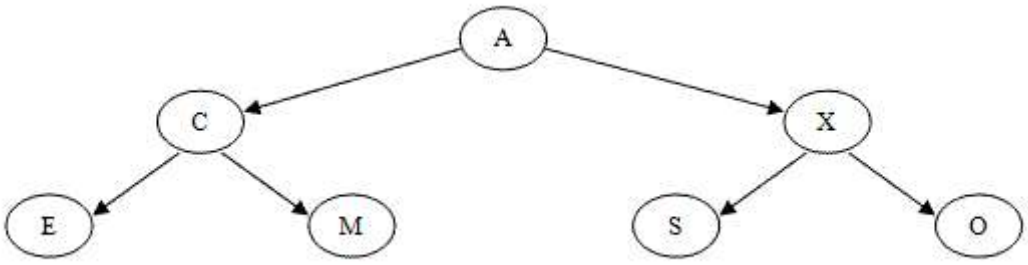
If every non-leaf node in a binary tree has nonempty left and right subtrees, the tree is termed a *strictly binary tree*. Or, to put it another way, all of the nodes in a strictly binary tree are of degree zero or two, never degree one. A strictly binary tree with N leaves always contains $2N - 1$ nodes.

Some texts call this a "full" binary tree.

A *complete binary tree* of depth d is the strictly binary tree all of whose leaves are at level d.

The total number of nodes in a complete binary tree of depth d equals $2^{d+1} - 1$. Since all leaves in such a tree are at level d, the tree contains 2^d leaves and, therefore, $2^d - 1$ internal nodes.

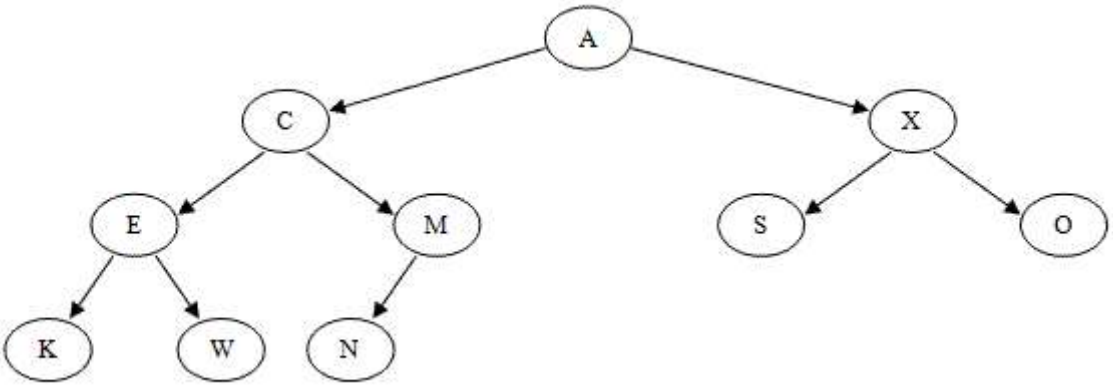
Diagram 2: A complete binary tree



A binary tree of depth d is an *almost complete binary tree* if:

- Each leaf in the tree is either at level d or at level $d - 1$.
- For any node n_d in the tree with a right descendant at level d, all the left descendants of n_d that are leaves are also at level d.

Diagram 3: An almost complete binary tree



An almost complete strictly binary tree with *N* leaves has 2*N* – 1 nodes (as does any other strictly binary tree). An almost complete binary tree with *N* leaves that is not strictly binary has 2*N* nodes. There are two distinct almost complete binary trees with *N* leaves, one of which is strictly binary and one of which is not.

There is only a single almost complete binary tree with *N* nodes. This tree is strictly binary if and only if *N* is odd.

Representing Binary Trees in Memory

Array Representation

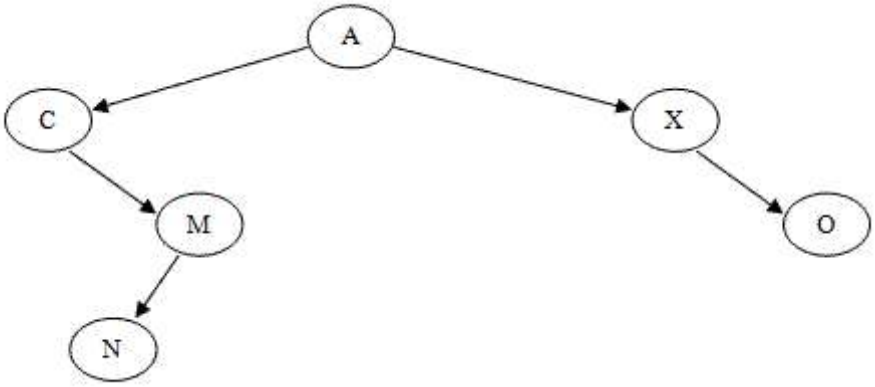
For a complete or almost complete binary tree, storing the binary tree as an array may be a good choice.

One way to do this is to store the root of the tree in the first element of the array. Then, for each node in the tree that is stored at subscript *k*, the node's left child can be stored at subscript 2*k*+1 and the right child can be stored at subscript 2*k*+2. For example, the almost complete binary tree shown in **Diagram 2** can be stored in an array like so:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
A	C	X	E	M	S	O	K	W	N

However, if this scheme is used to store a binary tree that is not complete or almost complete, we can end up with a great deal of wasted space in the array.

For example, the following binary tree

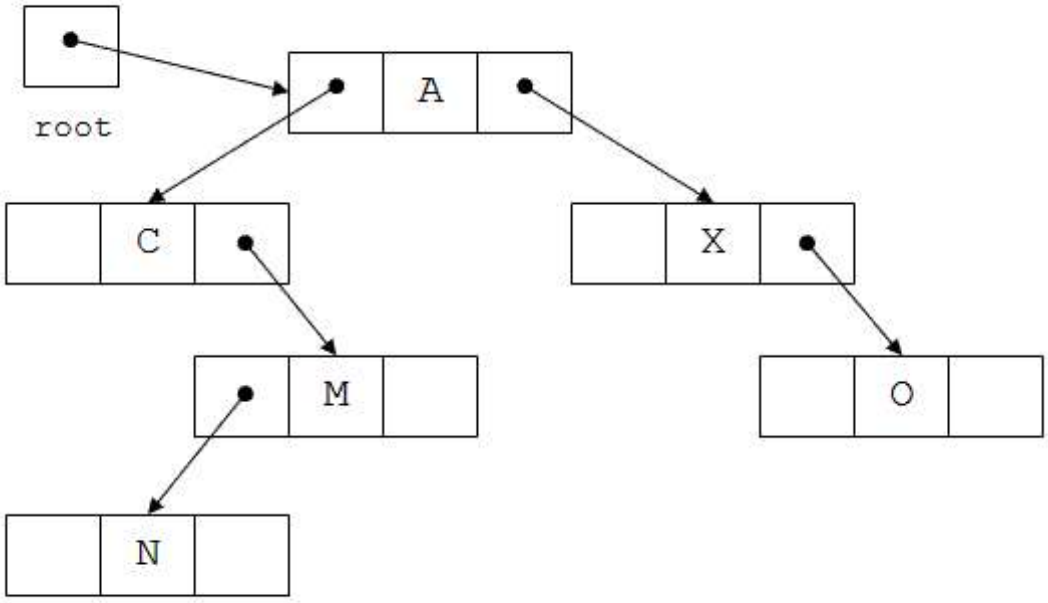


would be stored using this techinque like so:

[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
A	C	X		M		O			N

Linked Representation

If a binary tree is not complete or almost complete, a better choice for storing it is to use a linked representation similar to the linked list structures covered earlier in the semester:



Each tree node has two pointers (usually named left and right). The tree class has a pointer to the root node of the tree (labeled root in the diagram above).

Any pointer in the tree structure that does not point to a node will normally contain the value NULL. A linked tree with *N* nodes will always contain *N* + 1 null links.