Data Structures and Algorithms Design

A.Baskar

BITS Pilani, K. K. Birla Goa Campus

12 September 2015

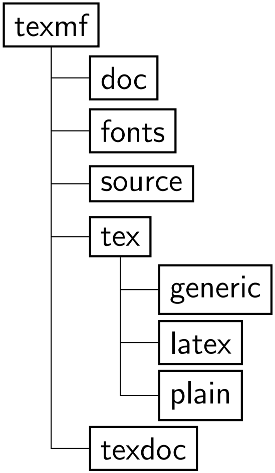
Recap

- We have studied linear data structures so far
- Arrays, Stacks, Queues, Linked lists
- These all have properties that their elements can be adequately displayed in a straight line

Recap

- We have studied linear data structures so far
- Arrays, Stacks, Queues, Linked lists
- These all have properties that their elements can be adequately displayed in a straight line
- How to obtain data structures for data that have nonlinear relationships
- Unix file system, Company organization, Table of contents

Trees



Tree terminology

- Trees have nodes(vertices) and edges
- A path in a tree is a list of distinct nodes in which successive nodes are connected by edges in the tree
- Between any two nodes there will be exactly one path

Tree terminology

- Trees that we consider are rooted. Once the root is defined (by the user) all nodes have a specific level
- Nodes with no children are called leaves (terminal, external nodes), n_e denote the number of external nodes
- Nodes which are not leaves are called internal nodes. n_i to denote the number of internal nodes

Tree terminology

- Trees that we consider are rooted. Once the root is defined (by the user) all nodes have a specific level
- Nodes with no children are called leaves (terminal, external nodes), n_e denote the number of external nodes
- Nodes which are not leaves are called internal nodes. n_i to denote the number of internal nodes
- Size, n, of a tree is the number of nodes in it ($n = n_i + n_e$)

Tree properties

Lemma

Let T be a tree with n nodes and let c_v denote the number of children of node v in T. Then

$$\sum_{v \in T} c_v = n - 1$$

Relationship among nodes

- Child of a node u :- Any node reachable from u by 1 edge.
- Parent node :- If b is a child of a, then a is the parent of b.
- All nodes except root have exactly one parent.
- Nodes that share parents are called siblings

Relationship among nodes

- An ancestor of a node is either node itself or an ancestor of the parent of the node
- A node u is descendant of a node v if v is ancestor of u

Relationship among nodes

- An ancestor of a node is either node itself or an ancestor of the parent of the node
- A node u is descendant of a node v if v is ancestor of u
- The subtree of T rooted at a node v is the tree consisting of all the descendants of v in T

Depth and Height

- Depth of root node is 0.
- Depth of any other node is 1 greater than depth of its parent.

Depth and Height

- Depth of root node is 0.
- Depth of any other node is 1 greater than depth of its parent.
- Height of a tree is maximum of depth of all nodes in it

Binary Trees

- A tree is ordered if there is a linear ordering defined for the children of each node
- A binary tree is a ordered tree in which each node has at most two children
- So each node might have a left child and a right child (so left subtree and right subtree)

Binary Trees

- A tree is ordered if there is a linear ordering defined for the children of each node
- A binary tree is a ordered tree in which each node has at most two children
- So each node might have a left child and a right child (so left subtree and right subtree)
- A binary tree is proper if every internal node has exactly two children

Binary Trees

- A tree is ordered if there is a linear ordering defined for the children of each node
- A binary tree is a ordered tree in which each node has at most two children
- So each node might have a left child and a right child (so left subtree and right subtree)
- A binary tree is proper if every internal node has exactly two children

Properties of proper binary tree

- $h + 1 \leq n_e \leq 2^h$
- $h \leq n_i \leq 2^h - 1$
- $h + 1 \leq n \leq 2^{h+1} - 1$
- $log(n+1) - 1 \leq h \leq (n-1)/2$

Properties of proper binary tree

- $h + 1 \leq n_e \leq 2^h$
- $h \leq n_i \leq 2^h - 1$
- $h + 1 \leq n \leq 2^{h+1} - 1$
- $log(n + 1) - 1 \leq h \leq (n - 1)/2$
- The number of external nodes is one more than the number of internal nodes.

Complete binary tree

- A binary tree is a complete binary tree if in every level, except possibly the deepest, is completely filled. At depth n, the height of the tree, all nodes must be as far left as possible.

Conventions

- All the trees are rooted and ordered
- Edges have direction, from above to below
- All the binary trees are proper

Tree Abstract Data Type

- Tree ADT stores elements at nodes
- element(v) returns the object stored at the node v
- size(), root(), parent(v)
- children(v)
- isInternal(v), isExternal(v), IsRoot(v)
- elements(), positions()
- swapElements(v,w), replaceElements(v,e)

Tree Abstract Data Type

- Tree ADT stores elements at nodes
- element(v) returns the object stored at the node v O(1)
- size(), root(), parent(v)O(1)
- children(v) O(c_v)
- isInternal(v), isExternal(v), IsRoot(v) O(1)
- elements(), positions() O(n)
- swapElements(v,w), replaceElements(v,e) O(1)

Algorithms

- Finding depth of node v in a tree T

Algorithms

- Finding depth of node v in a tree T
- Finding height of the tree T

Algorithms

- Finding depth of node v in a tree T
- Finding height of the tree T
- Visiting all the nodes of T in a systematic way

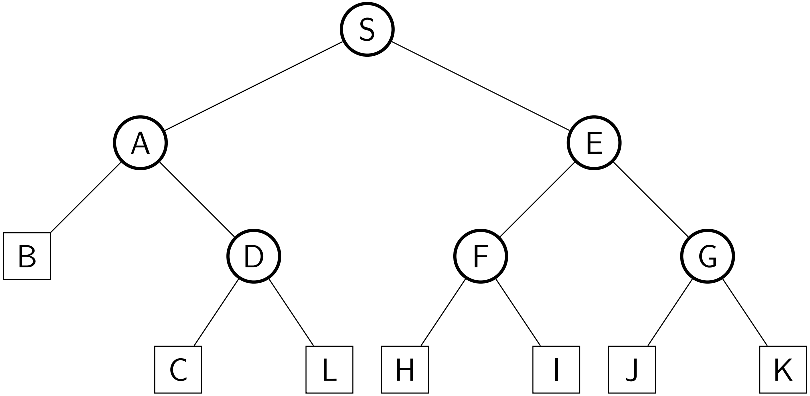
Algorithms

- Finding depth of node v in a tree T
- Finding height of the tree T
- Visiting all the nodes of T in a systematic way
    - preorder
    - postorder
    - inorder

Binary Trees: Array implementation

- Binary Trees can be represented using arrays so that all nodes can be accessed in O(1) time:
- Label nodes sequentially top-to-bottom and left-to-right
- Left child of A[i] is at position A[2i]
- Right child of A[i] is at position A[2i + 1]
- Parent of A[i] is at A[i/2]

Example



Summary

- Non linear data structures: Trees
- Tree terminology
- Binary trees
- Tree ADT, and array implementation