Data Structures and Algorithms (5)

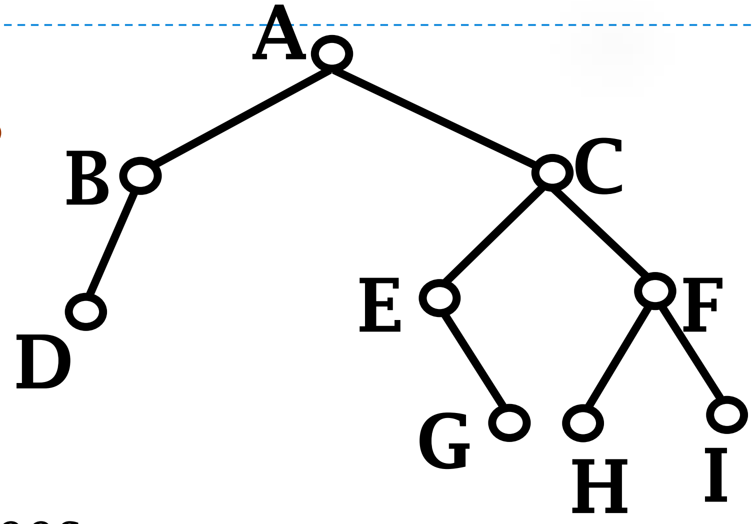
Instructor: Ming Zhang

Textbook Authors: Ming Zhang, Tengjiao Wang and Haiyan Zhao

Higher Education Press, 2008.6 (the "Eleventh Five-Year" national planning textbook)

https://courses.edx.org/courses/PekingX/04830050x/2T2014/

Chapter 5 Binary Trees



- Concepts of Binary Trees
- Abstract Data Types
    - Depth-First Search
    - Breadth-First Search
- Storage Structures of Binary Trees
- Binary Search Tree
- Heap and Priority Queue
- Huffman Tree and its Application

Linked Storage Structure of Binary Trees

Each node of a Binary Tree is stored in memory randomly, and the logical relationships among the nodes are linked by pointers.

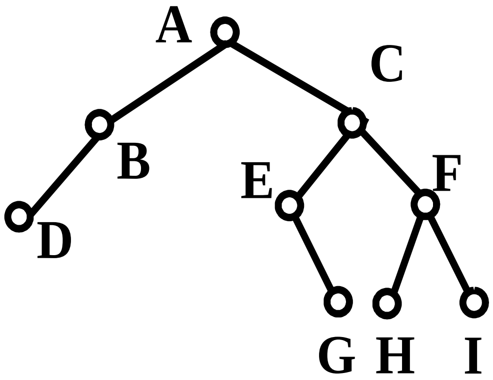
- Binary Linked List
  - Left and right pointers, point to its left child and its right child respectively

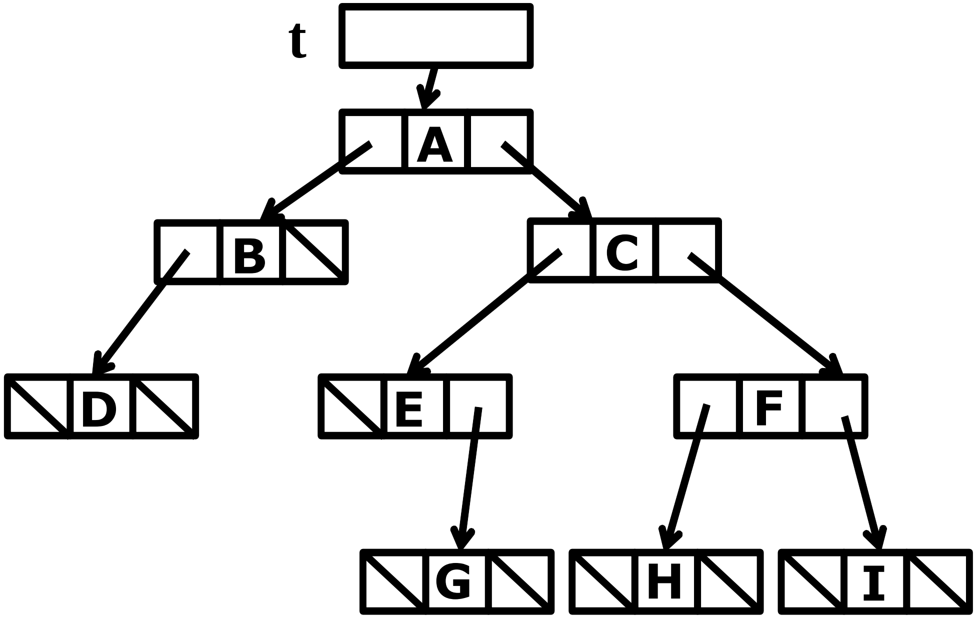
left	info	right

- Trinomial Linked List
  - Left and right pointers, point to its left child and its right child respectively
  - Add a parent pointer

left	info	parent	right

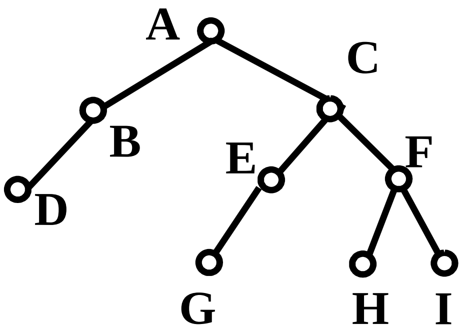
Binary Linked List

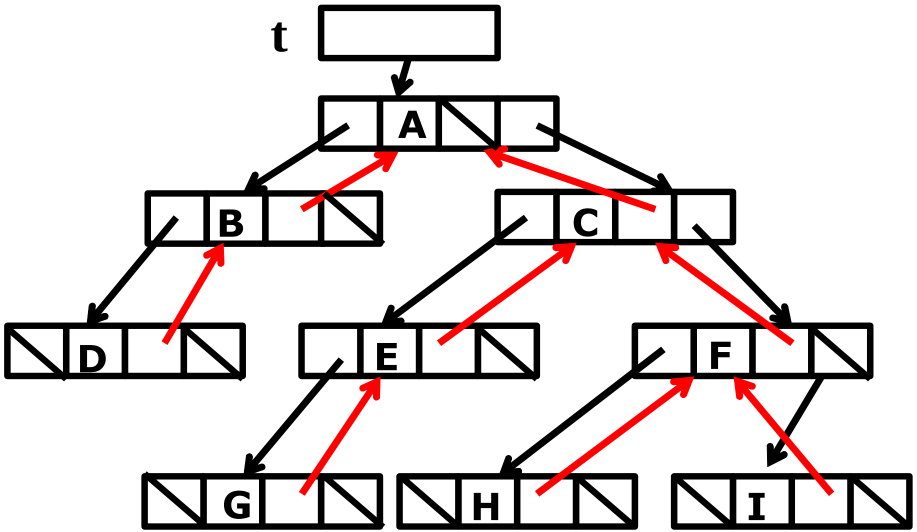




"Trinomial Linked List"

- Pointer parent points to its parent, the ability of "upward"





Add Two Private Data Members into Class BinaryTreeNode

```
private:
    BinaryTreeNode<T> *left;     // The pointer pointing to the left subtree
    BinaryTreeNode<T> *right;    // The pointer pointing to the right subtree

template <class T> class BinaryTreeNode  {
friend class BinaryTree<T>;           // Declare the class of Binary Trees as a friend class
private:
   T  info;                           // Data domain of nodes in a binary tree
public:
   BinaryTreeNode();                  // Default constructor function
   BinaryTreeNode(const T& ele);      // Construction with given data
   BinaryTreeNode(const T& ele, BinaryTreeNode<T> *l,
              BinaryTreeNode<T> *r);  // Construction of nodes in a subtree
.... }
```

A Recursive Structure to Find out the Father Node —— Be Careful of its Return

```
template<class T>
BinaryTreeNode<T>* BinaryTree<T>::
Parent(BinaryTreeNode<T> *rt, BinaryTreeNode<T> *current)  {
      BinaryTreeNode<T> *tmp,
      if (rt == NULL) return(NULL);
      if (current == rt ->leftchild() || current == rt->rightchild())
            return rt; // If the child is current, then return parent
      if ((tmp =Parent(rt- >leftchild(), current) != NULL)
            return tmp;
      if ((tmp =Parent(rt- > rightchild(), current) != NULL)
            return tmp;
      return NULL;
}
```

Questions

- What structure does the algorithm use?
- Which order of traversal does the algorithm belong to?
  - Could we use non-recursive methods?
  - Could we use BFS ?
- How to find out brother nodes starting from this algorithm?

5.3 Storage Structure of Binary Trees

A Non-Recursive Structure to Find out the Father Node

```
BinaryTreeNode<T>* BinaryTree<T>::Parent(BinaryTreeNode<T> *current)  {
    using std::stack;                              // Use the stack in STL
    stack<BinaryTreeNode<T>* > aStack;
    BinaryTreeNode<T> *pointer = root;
    aStack.push(NULL);                             // The lookout at the bottom of the stack
    while (pointer) {                              // Or !aStack.empty()
        if (current == pointer->leftchild() || current == pointer->rightchild())
            return pointer;              // if the child of pointer is current, then return parent
        if (pointer->rightchild() != NULL)   // Push the non-empty right subtree into the stack
            aStack.push(pointer->rightchild());
        if (pointer->leftchild() != NULL)
            pointer = pointer->leftchild();        // Go down along the left side
        else {               // After visiting the left subtree, it is turn to visit the right subtree
            pointer=aStack.top(); aStack.pop(); // Get the element on the top of the stack and pop it
}   }  }
```

Analysis of the Space Cost

- Storage density α (≤ 1) means the efficiency of the structure of data storage

$$\alpha(\text{storage density}) = \frac{\text{Storage size of data itself}}{\text{Storage size of the whole structure}}$$

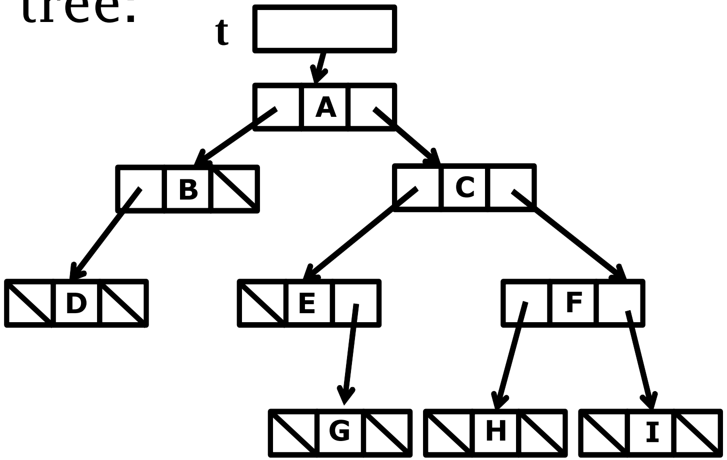
- Overhead of the structure $\gamma = 1 - \alpha$

Analysis of the Space Cost

According to the property of full binary tree:

Half of the pointers are null

- Each node stores two pointers and one data domain
  - Total space $(2p + d)n$
  - Cost of the structure : $2pn$
  - If $p = d$, then the cost of the structure is $2p/(2p + d) = 2/3$



Analysis of the Space Cost

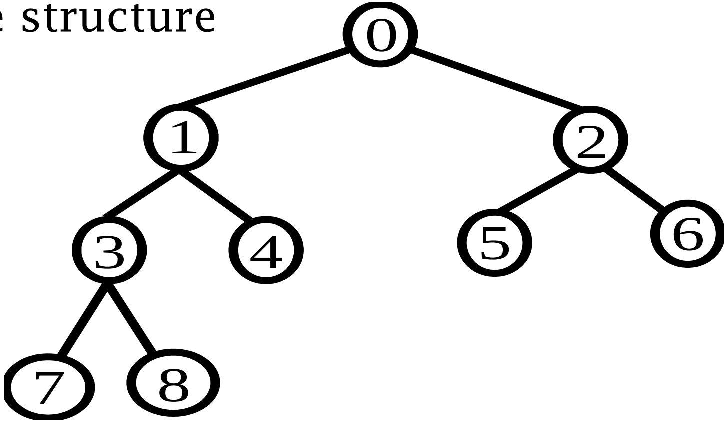
- C++ has two methods to implement different branch and leaf nodes:
  - Use union type to define
  - Use the subclass in C++ to implement branch nodes and leaf nodes respectively,

    and use virtual function isLeaf to distinguish branch nodes and leaf nodes
- Save memory resource at the early stage
  - Use a idle bit (one bit) of the pointer of a node to mark the type of the node
  - Use the pointer pointing to a leaf or the pointer domain in a leaf to store the value of this leaf node

Sequential Storage Structure of Complete Binary Tree

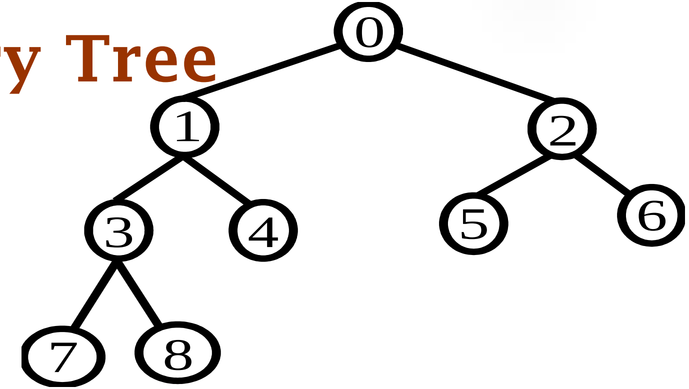
- Sequential method to store a binary tree
  - Store nodes into a piece of continuous space according to a particular order
  - Make the position of nodes in the sequence able to reflect corresponding information of the structure
- The storage structure is linear

0	1	2	3	4	5	6	7	8



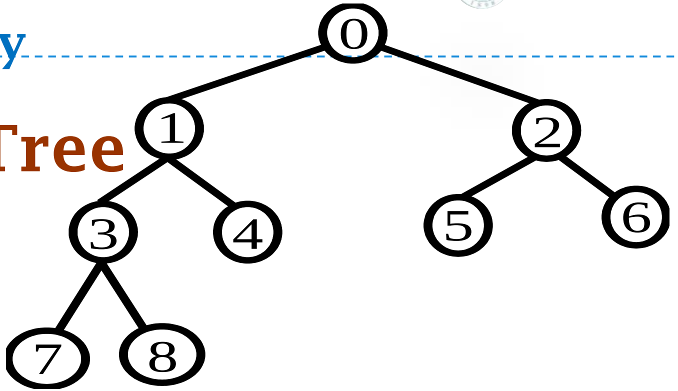
- It is still a structure of a binary tree on its logical structure

Index Formula of Complete Binary Tree



- We could know the indexes of a node's parent, children and brothers according to its index
  - When 2i+1<n, the left child of node i is node 2i+1, otherwise node i doesn't have a left child
  - When 2i+2<n, the right child of node i is node 2i+2, otherwise node i doesn't have a right child

Index Formula of Complete Binary Tree



- When 0<i<n, the parent of node i is node $\lfloor (i-1)/2 \rfloor$
- When i is even and 0<i<n, the left brother of node i is node i-1, otherwise node i doesn't have a left brother
- When i is odd and i+1<n, the right brother of node i is node i+1, otherwise node i doesn't have a right brother

Questions

- What changes should we make to the algorithm of binary tree if we use the storage structure of trinomial linked list? We should pay more attention to maintain the father pointer when inserting and deleting nodes.
- What is the index formula of complete trinomial tree ?

Data Structures and Algorithms

Thanks!

the National Elaborate Course (Only available for IPs in China)
http://www.jpk.pku.edu.cn/pkujpk/course/sjjg/

Ming Zhang, Tengjiao Wang and Haiyan Zhao
Higher Education Press, 2008.6 (awarded as the "Eleventh Five-Year" national planning textbook)