Lecture 12.1

Topics:

1. Binary Trees – Properties & Theorems
2. Complete Binary Tree – Introduction

---

1. Binary Trees – Properties & Theorems

The main distinctions between a binary tree and a tree are:

- Each element in a binary tree has exactly two subtrees (one or both of these subtrees may be empty. Each element in a tree can have any number of subtrees.
- The subtrees of each element in a binary tree are ordered as left and right subtrees. The subtrees in a tree are unordered.
- A binary tree can be empty whereas a tree cannot (some may allow an empty tree just as a matter of definition.

Let's look at some properties of binary trees.

Binary Properties & Theorems

Digression

- A binary tree with depth one can hold one node, which must be a leaf.
- A full binary tree with depth two (2) holds three nodes, two of which are leaves.
- A full binary tree of depth three (3) holds seven nodes, four of which are leaves.

Property 1

The drawing of every binary tree with n elements, $n > 0$, has exactly $n - 1$ edges.

Property 2

A binary tree of depth (height) d, $d >= 0$, has at least d and at most $2^d - 1$ elements.

Property 3

The depth of a binary tree that contains n, $n >= 0$, elements is at most n and at least $U[\log_2(n + 1)]$ (i.e., the smallest integer that is larger or equal to $\log_2(n + 1)$).

Theorem 1

A **full** binary tree of depth n will have $2^{(n-1)}$ leaves.

Theorem 2

The number of nodes in a **full** binary tree of depth n is $(2^n - 1)$.

Theorem 3

A binary tree containing n nodes must have at least one path of length $U[\log_2 n]$.

2. Complete Binary Trees – Introduction

Definition

A binary tree that is completely filled, with the possibly exception of the bottom level, and is filled from left to right, is called a **complete binary tree**.

2.1 Theorems & Properties

Theorem 4

In a complete binary tree containing `n` nodes, the longest path traverses $U[\log_2 n]$ nodes.

Property 4

The height (depth) of a complete binary tree containing `n` elements is $U[\log_2(n + 1)]$.

Theorem 5

Let `i` where `1 <= i <= n`, be the number assigned to an element of a complete binary tree. The following are true:

(i) If `i = 1`, then this element is the root of the binary tree. If `i > 1`, then the parent of this element has been assigned the number `L[i/2]` (i.e., the largest integer that is less than or equal to `(i/2)`

(ii) If `2i > n`, then this element has no left child. Otherwise, its left child has been assigned the number `2i`.

(iii) If `2i + 1 > n`, then this element has no right child. Otherwise, its right child has been assigned the number `2i + 1`.

2.2 Examples

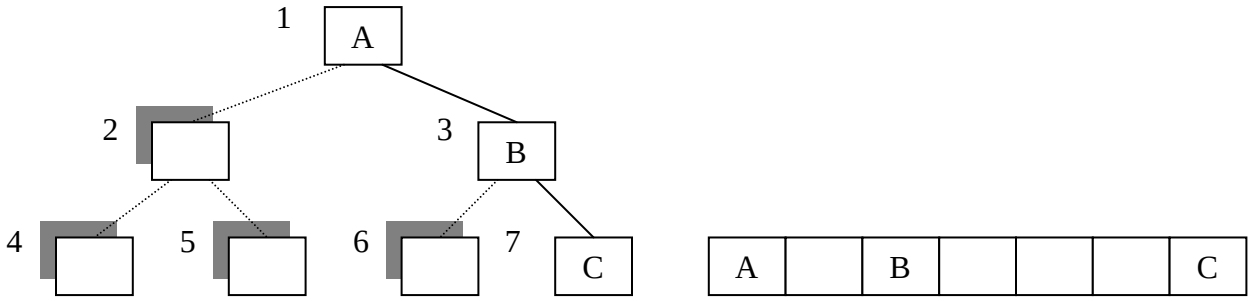
Recall that arrays and lists can be used as the bases to implement other data structures. Previous examples showed a linked-list based implementation for a binary tree. Let's look again at the binary trees and their array representations.

Note that any binary tree may or may not be a complete binary tree. However, one can certainly retain the non-existing elements through empty node representations. In the following examples, let's denote nodes with numbers that starting 1 at the root and incremented by one at each level going from left to right.

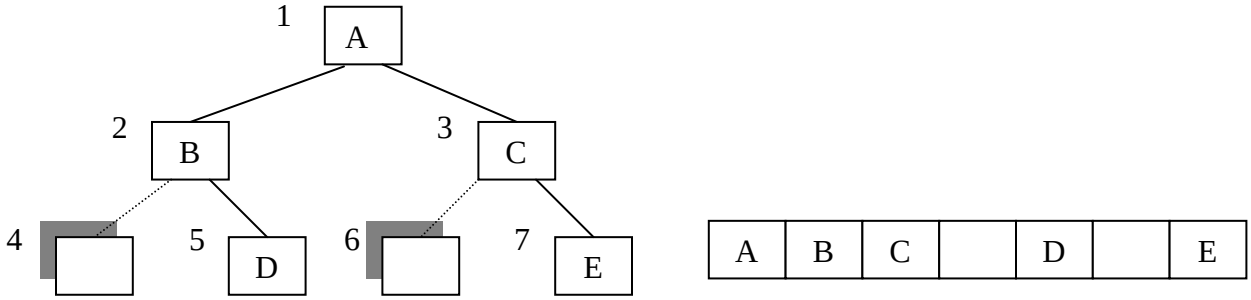
In the array representation, storage of an incomplete binary tree will be wasteful because of the unused elements between the occupied locations in the array. On the other hand, a complete binary tree uses up a block of adjacent elements from an array.

This is one of the reasons why an array-based implementation is attractive for "**max (or min) heaps**", which are data structures that represent trees with stricter and more specific order than just a binary tree. Using this understanding and observation, several interesting and important operations can be devised with heap structures.

We will look into these structures next.



(a)



(b)

Figure 1. Incomplete binary trees

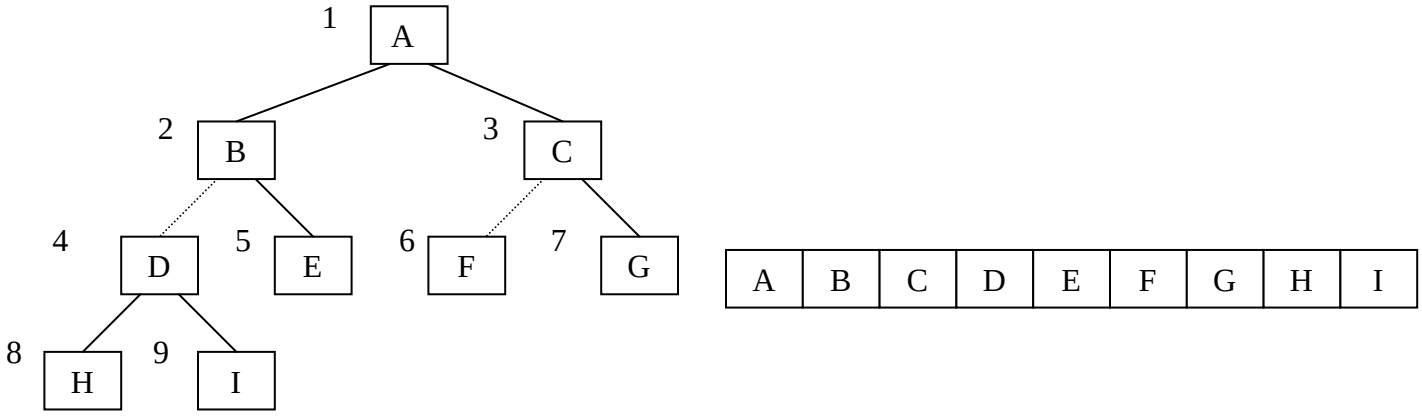


Figure 2. Complete binary tree