CSCI 1900
Discrete Structures

Labeled Trees
Reading: Kolman, Section 7.2

Giving Meaning to Vertices and Edges

- Our discussion of trees implied that a vertex is simply an entity with parents and offspring much like a family tree.
- What if the position of a vertex relative to its siblings or the vertex itself represented an operation. Examples:
  - Edges from a vertex represent cases from a switch statement in software
  - Vertex represented a mathematical operation

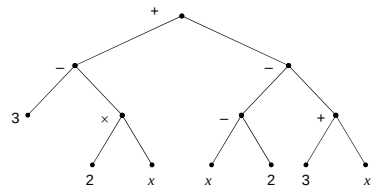
Mathematical Order of Precedence Represented with Trees

- Consider the equation:

$$(3 - (2 \times x)) + ((x - 2) - (3 + x))$$

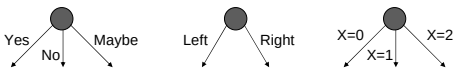
- Each element is combined with another using an operator, i.e., this expression can be broken down into a hierarchy of (a ° b) where "°" represents an operation used to combine two elements.
- We can use a binary tree to represent this equation with the elements as the leaves.

Precedence Example Tree



Positional Tree

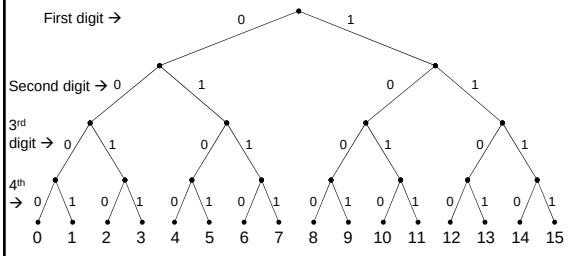
- A positional tree is an n-tree that relates the direction/angle an edge comes out of a vertex to a characteristic of that vertex. For example:



- When n=2, then we have a positional binary tree.

Tree to Convert Base-2 to Base-10

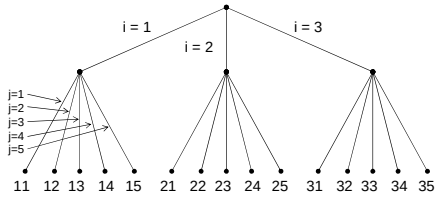
Starting with the first digit, take the left or right edge to follow the path to the base-10 value.



For-Loop Represented with Tree

```
for i = 1 to 3
    for j = 1 to 5
        array[i,j] = 10*i + j
    next j
next i
```


For Loop Positional Tree



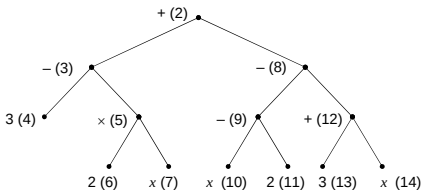
Storing Binary Trees in Memory

- Section 4.6 introduced us to "linked lists". Each item in the list was comprised of two components:
  - Data
  - Pointer to next item in list
- Positional binary trees require two links, one following the right edge and one following the left edge. This is referred to as a "doubly linked list."

Doubly Linked List

Index	Left	Data	Right
1	2	-------	0
2 (root)	3	+	8
3	4	–	5
4	0	3	0
5	6	×	7
6	0	2	0
7	0	*x*	0
8	9	–	12
9	10	–	11
10	0	*x*	0
11	0	2	0
12	13	+	14
13	0	3	0
14	0	*x*	0

Precedence Example Derived from the Doubly Linked List



The numbers in parenthesis represent the index from which they were derived in the linked list on the previous slide.

Huffman Code

- Depending on the frequency of the letters occurring in a string, the Huffman Code assigns patterns of varying lengths of 1's and 0's to different letters.
- These patterns are based on the paths taken in a binary tree.
- A Huffman Code Generator can be found at:
  http://www.inf.puc-rio.br/~sardinha/Huffman/Huffman.html