CS 135 — L13: Binary Tree

Luke Lu • 2025-11-11

Tree

Tree is an **abstraction** data type and is represented as lists where order matters

Terminologies

1. A **root** node has no parent and is an internal node
2. A **leaf** node has no children
3. Internal node is not a **leaf**
4. A line connecting nodes are **edges**
5. A **label** of a node is the value it carries
6. There are at most 2 **children nodes** each internal node can have

Binary Tres

Binary Tree Helpers

```
;; mk-node consumes a label, a left child, and a right child
;; mk-node produces a BT
;; mk-node: Any BT BT -> BT
(define (mk-node label left right)
  (list label left right))
;; get the label from a binary tree
;; get-label: BT -> Any
(define (get-label bt)
  (first bt))
;; get the left child from a binary tree
;; get-left: BT -> BT
(define (get-left bt)
  (second bt))
;; get the right child from a binary tree
;; get-right: BT -> BT
(define (get-right bt)
  (third bt))
```

Searching

$O(n \log n)$ complexity

```
;; Does the tree contain the label
;; bt-contains?: Num BT -> Bool
;; Requires: labels are numbers
(define (bt-contains? label bt)
  (cond
    [(empty? bt) false]
    [(= (get-label bt) label) true]
    [else (or (bt-contains? label (get-left bt))(bt-contains? label (get-right
bt)))]))
```

Finding Path

```
;; Produce a path to a label
;; bt-path: Num BT -> (listof (anyof 'left 'right 'found))
;; Requires: labels are numbers
```

```
(define (bt-path label bt)
  (cond
    [(empty? bt) empty]
    [(= (get-label bt) label) (list 'found)]
    [(bt-contains? label (get-left bt)) (cons 'left (bt-path label (get-left bt)))]
    [(bt-contains? label (get-right bt)) (cons 'right (bt-path label (get-right
bt)))]
    [else empty]
    ))
```

Binary Search Tree

Binary Search Tree property:

- Order matters
- Left Child node < Parent node
- Right Child node > Parent node

Searching Binary Search Tree

$O(n)$ complexity

```
;; bst-contains?: Nat BST -> Bool
(define (bst-contains? n bst)
  (cond
    [(empty? bst) false]
    [(= (get-label bst) n) true]
    [(> (get-label bst) n) (bst-contains? n (get-left bst))]
    [else (bst-contains? n (get-right bst))]))
```

Adding Label to Binary Search Tree

```
;; Add a label to a binary search tree
;; bst-add: Nat BST -> BST
(define (bst-add n bst)
  (cond
    [(empty? bst) (mk-node n empty empty)]
    [(= (get-label bst) n) bst]
    [(> (get-label bst) n) (mk-node (get-label bst) (bst-add n (get-left bst)) (get-
right bst))]
    [else (mk-node (get-label bst) (get-left bst) (bst-add n (get-right bst)))]))
```