Binary trees

Goldsmiths Computing

Motivation

- simplest form of tree data structure
- algorithms straightforward to understand
    - and (reasonably) simple to analyse
- generalise to practical applications
    - *e.g.* B-Trees for disk storage

Definition

A binary tree is an ordered collection of data

Operations

left return the left-child of the tree

right return the right-child of the tree

key return the data stored at this node of a tree

parent return the parent of the node

(and associated setters)

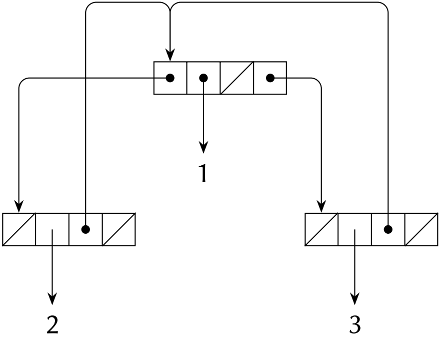
Collection operations

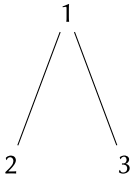
search[o] return `true` if o is in the collection

max return the maximum element (with respect to some ordering) of the collection

...

Implementation





Complexity analysis

left, right, key, parent

single pointer reads (or writes for setters)

$$\Longrightarrow \Theta(1)$$

Traversal

vector start at index zero, and visit elements in order of index until you reach the end

dynamic array as vector

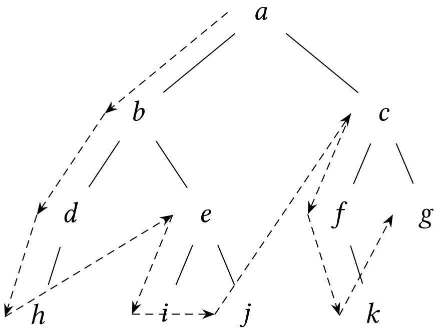
linked list start at the head of the list, and visit the FIRST of each successive REST

binary tree multiple possibilities!

Depth-first traversal

pre-order

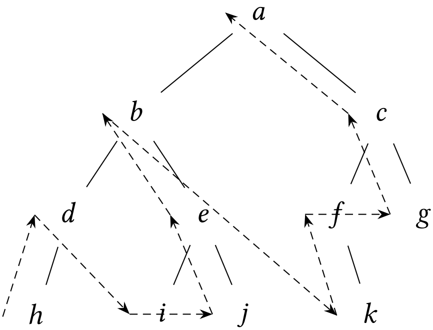
```
function PRE-ORDER(T)
    if ¬NULL?(T) then
        VISIT(T)
        PRE-ORDER(LEFT(T))
        PRE-ORDER(RIGHT(T))
    end if
end function
```



Depth-first traversal

post-order

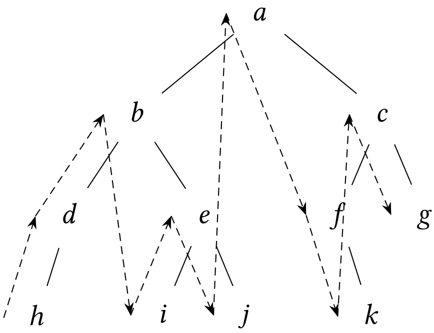
```
function POST-ORDER(T)
    if ¬NULL?(T) then
        POST-ORDER(LEFT(T))
        POST-ORDER(RIGHT(T))
        VISIT(T)
    end if
end function
```



Depth-first traversal

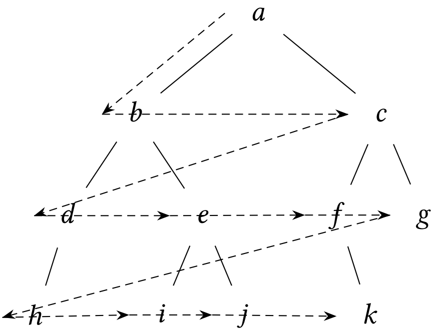
in-order

```
function IN-ORDER(T)
    if ¬NULL?(T) then
        IN-ORDER(LEFT(T))
        VISIT(T)
        IN-ORDER(RIGHT(T))
    end if
end function
```



Breadth-first traversal

```
function ENQUEUE-IF!(Q,T)
    if ¬NULL?(T) then
        ENQUEUE!(Q,T)
    end if
end function
function BREADTH-FIRST(T)
    Q ← new Queue()
    ENQUEUE-IF!(Q,T)
    while ¬EMPTY?(Q) do
        t ← DEQUEUE!(Q)
        VISIT(t)
        ENQUEUE-IF!(Q,LEFT(t))
        ENQUEUE-IF!(Q,RIGHT(t))
    end while
end function
```


a
b
c
d
e
f
g
h
i
j
k