Goldsmiths
UNIVERSITY OF LONDON

ALGORITHMS & DATA STRUCTURES

Breadth-First Traversal

DEPARTMENT OF COMPUTING

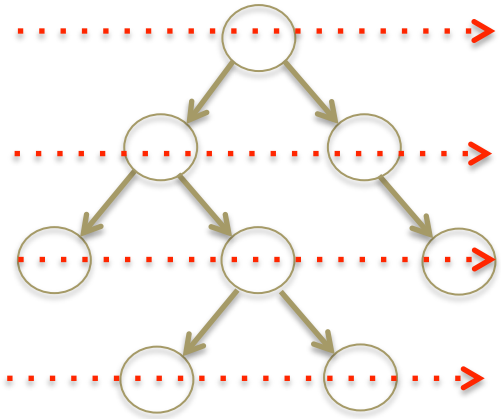
A. Beghelli

Exam revision – May 2019

Traversing a tree

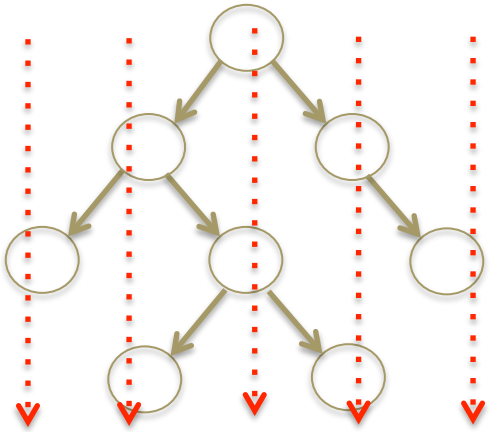
Traversal:
the process of visiting **all the nodes** of a tree

Breadth-First Traversal



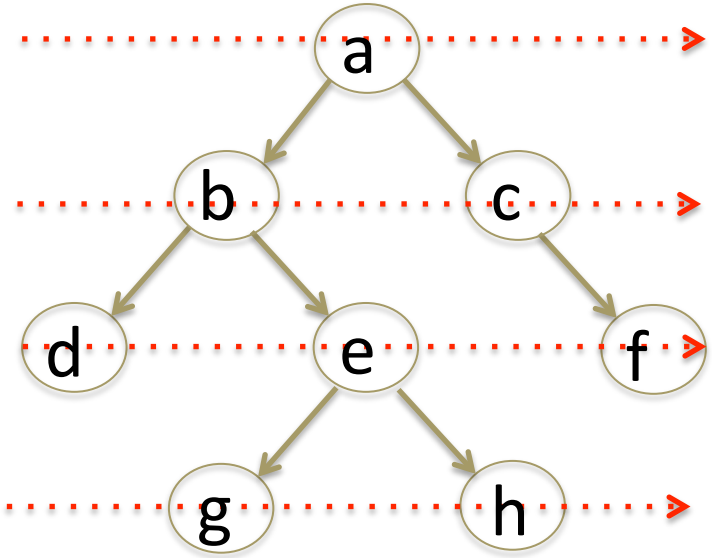
"reading the tree"
(in the western world)

Depth-First Traversal



"diving the tree"

Breadth-First Traversal



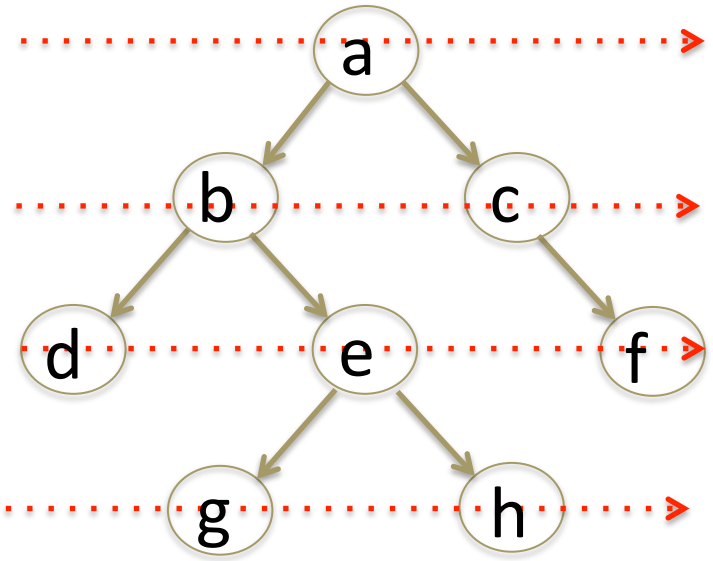
"reading the tree"
(in the western world)

[a,b,c,d,e,f,g,h]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



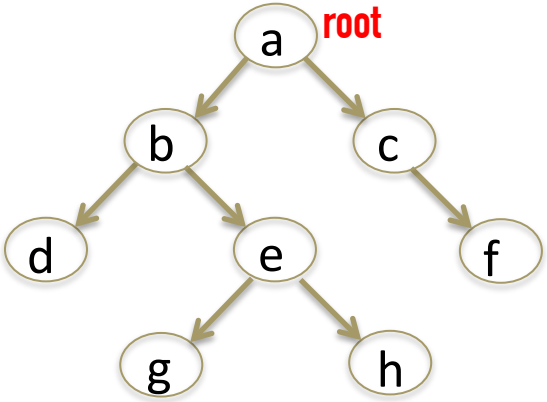
"reading the tree"
(in the western world)

[a,b,c,d,e,f,g,h]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



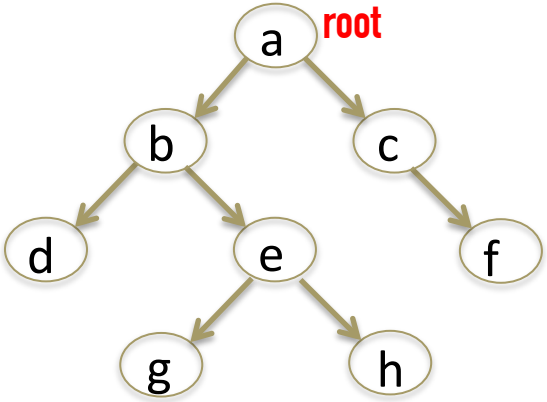
Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```

Q

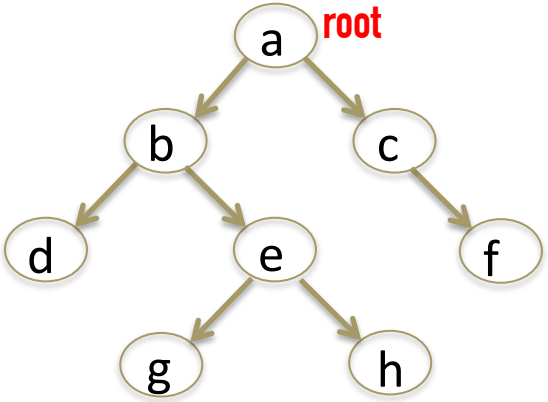
[]



Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



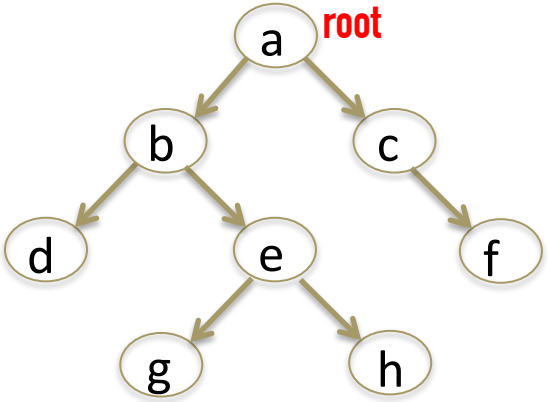
Q

[a]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[a]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

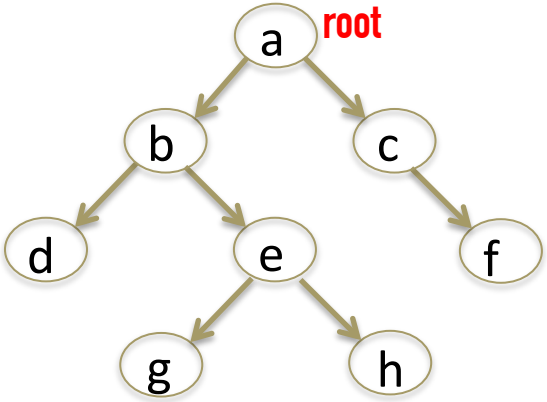
```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```

Q

[]

t

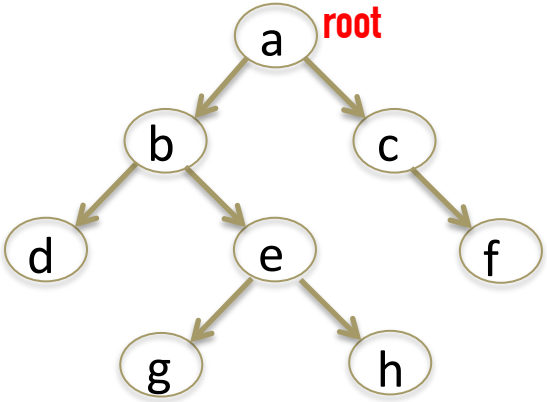
a



Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[ ]

t

a

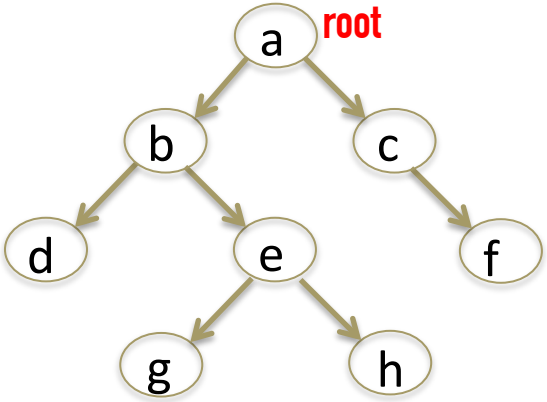
RESULT

[a]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[b]

t

a

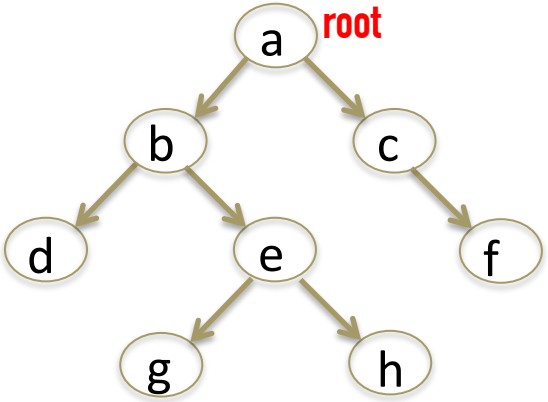
RESULT

[a]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[b, c]

t

a

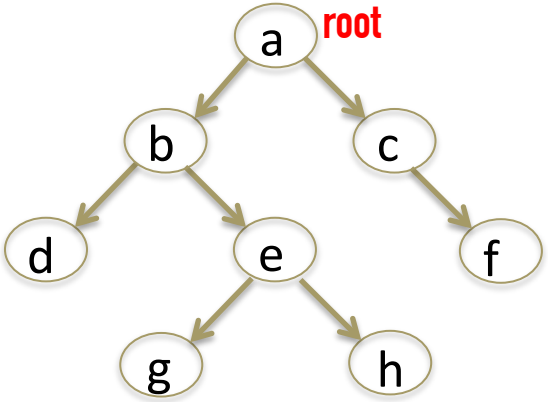
RESULT

[a]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[b, c]

t

a

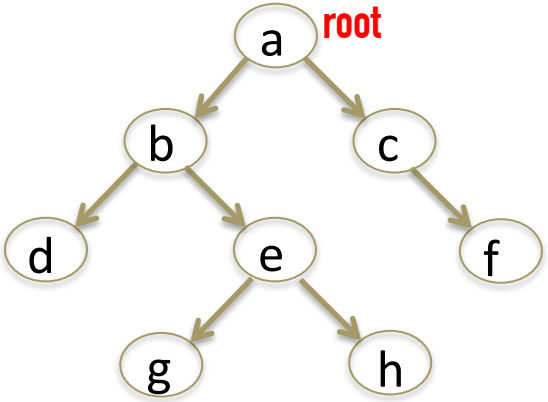
RESULT

[a]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[c]

t

b

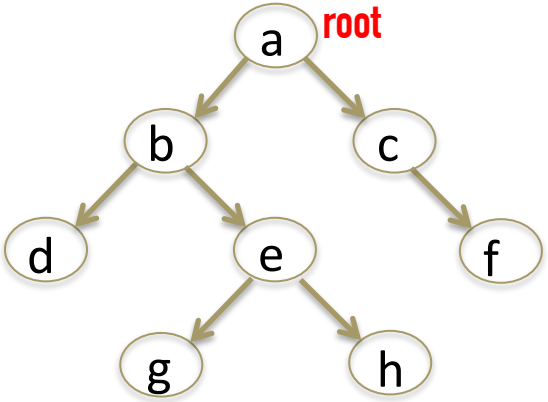
RESULT

[a]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[c]

t

b

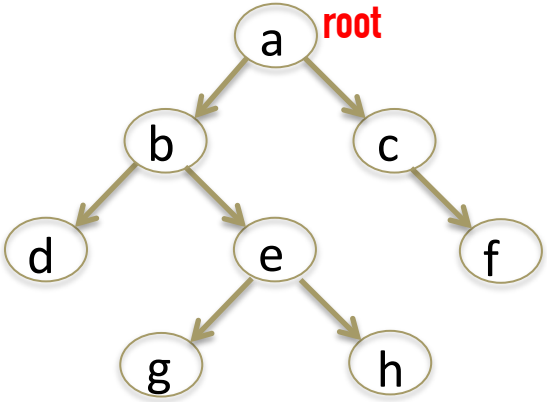
RESULT

[a, b]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[c, d]

t

b

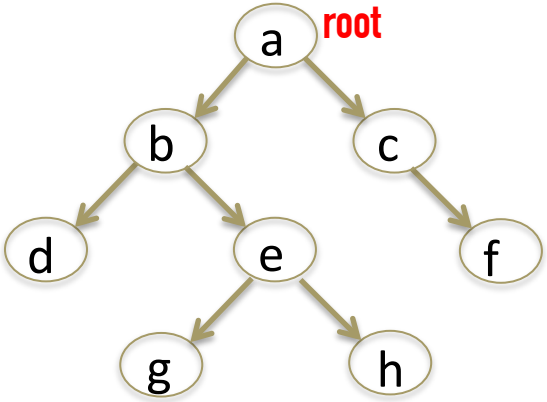
RESULT

[a, b]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[c, d, e]

t

b

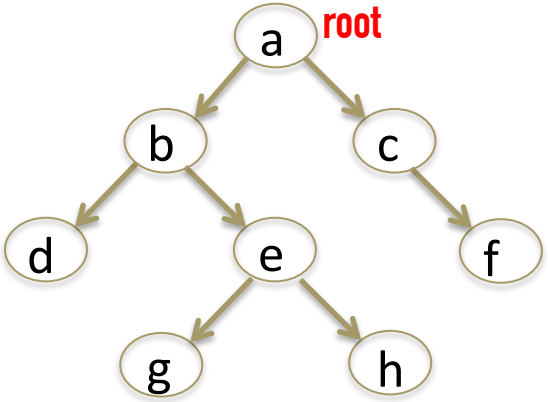
RESULT

[a, b]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[c, d, e]

t

b

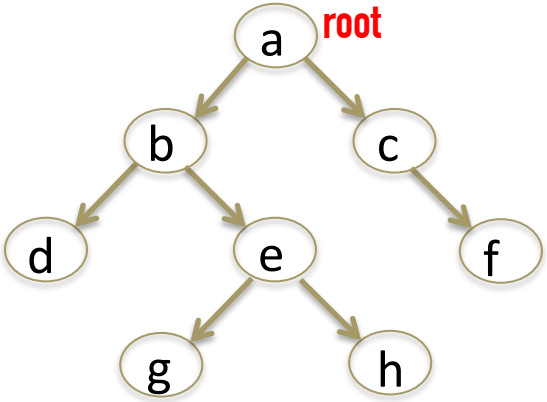
RESULT

[a, b]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[d, e]

t

c

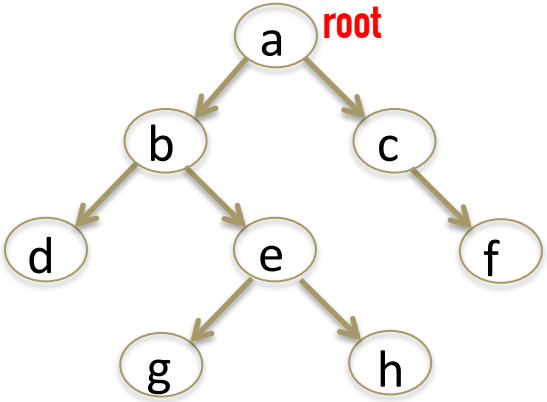
RESULT

[a, b]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[d, e]

t

c

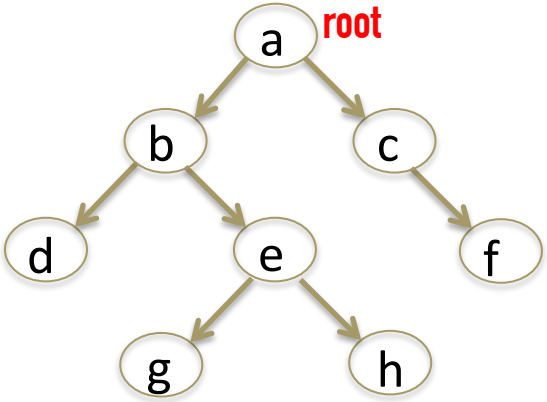
RESULT

[a, b, c]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[d, e]

t

c

NOTHING TO ENQUEUE

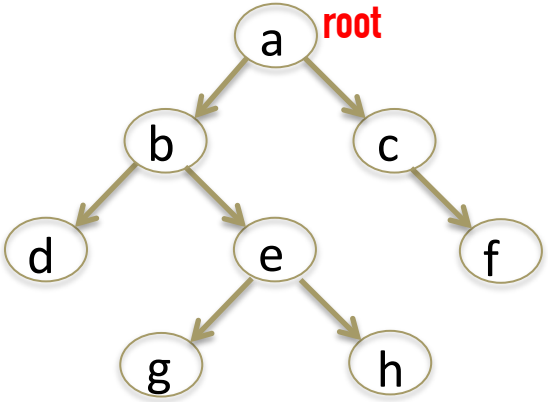
RESULT

[a, b, c]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[d, e, f]

t

c

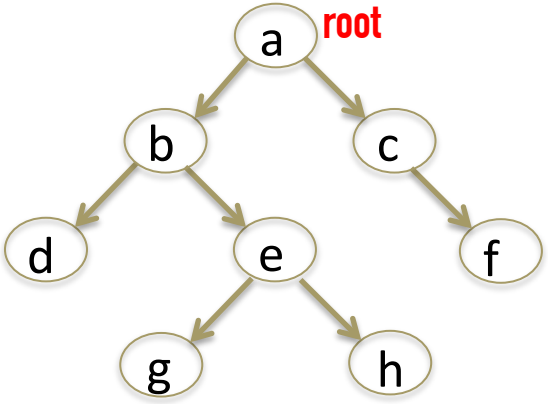
RESULT

[a, b, c]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[d, e, f]

t

c

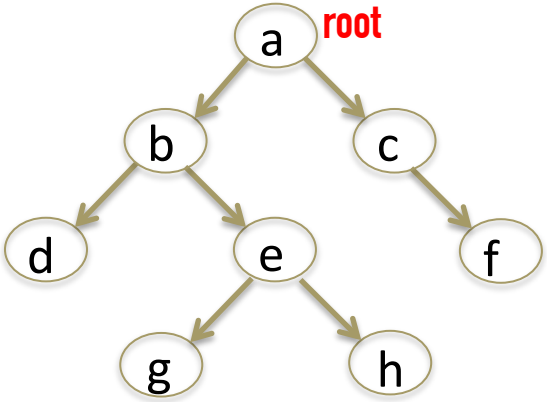
RESULT

[a, b, c]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[e, f]

t

d

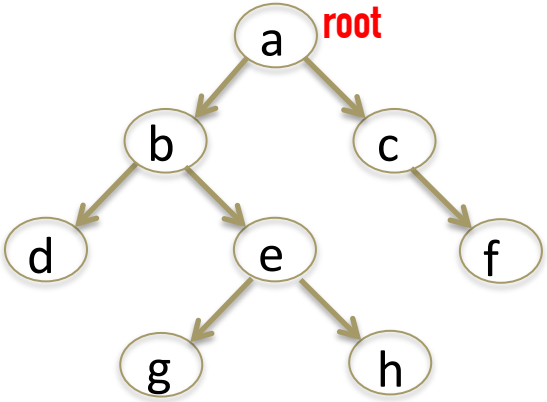
RESULT

[a, b, c]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[e, f]

t

d

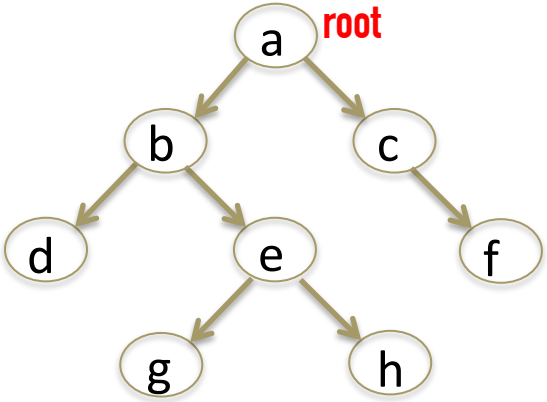
RESULT

[a, b, c, d]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[e, f]

t

d

NOTHING TO ENQUEUE

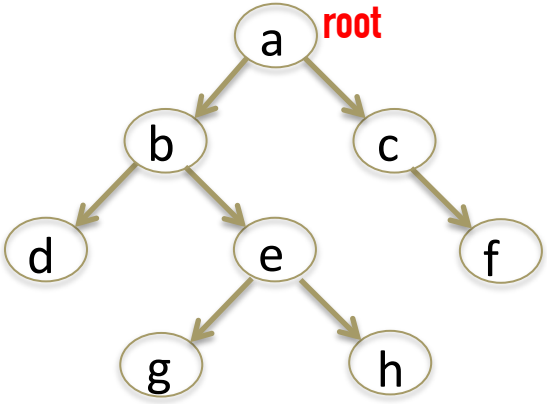
RESULT

[a, b, c, d]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[e, f]

t

d

RESULT

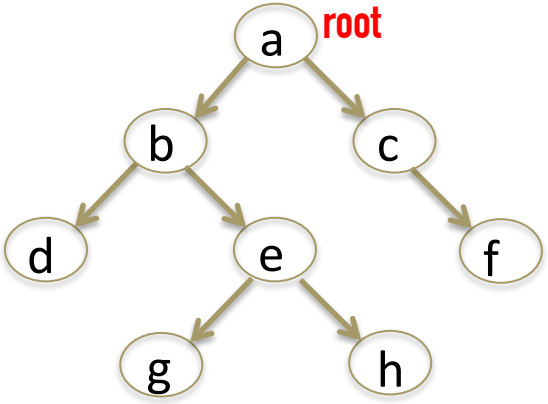
[a, b, c, d]

NOTHING TO ENQUEUE

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[e, f]

t

d

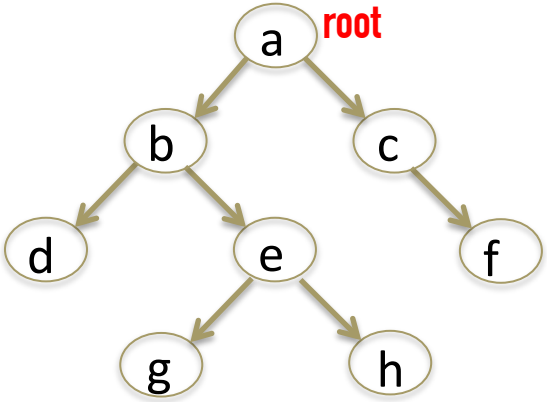
RESULT

[a, b, c, d]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[f]

t

e

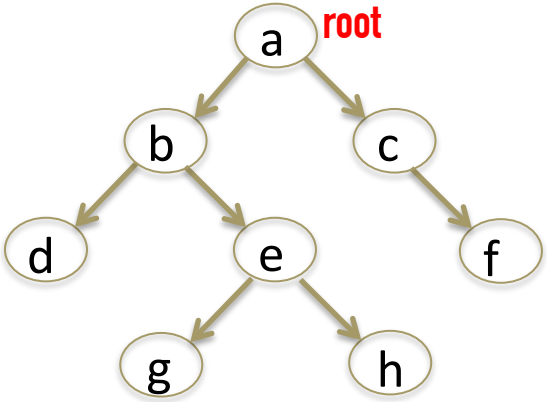
RESULT

[a, b, c, d]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[f]

t

e

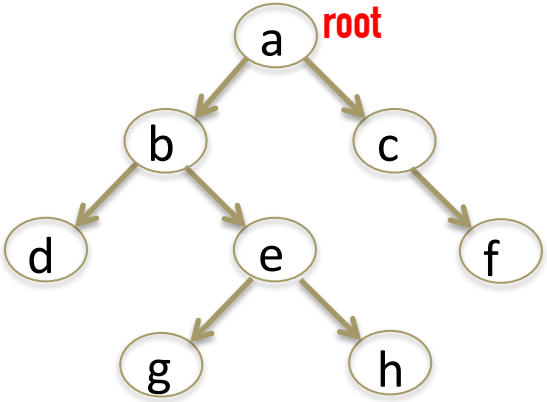
RESULT

[a, b, c, d, e]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[f, g]

t

e

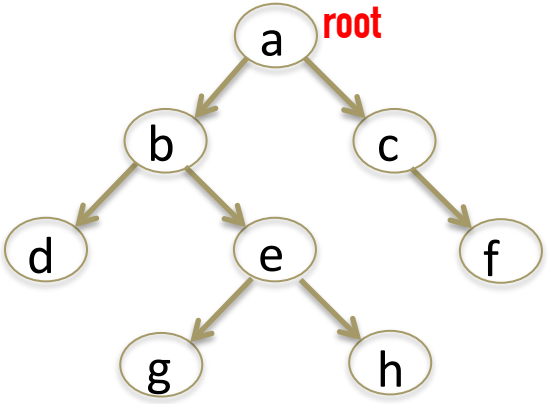
RESULT

[a, b, c, d, e]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[f, g, h]

t

e

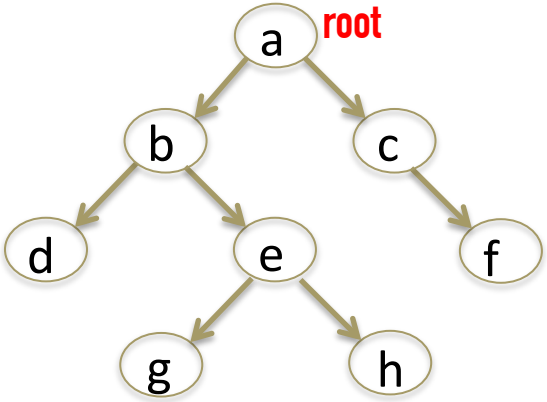
RESULT

[a, b, c, d, e]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[f, g, h]

t

e

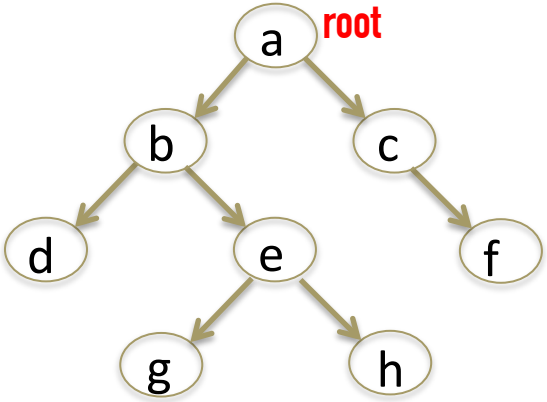
RESULT

[a, b, c, d, e]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[g, h]

t

f

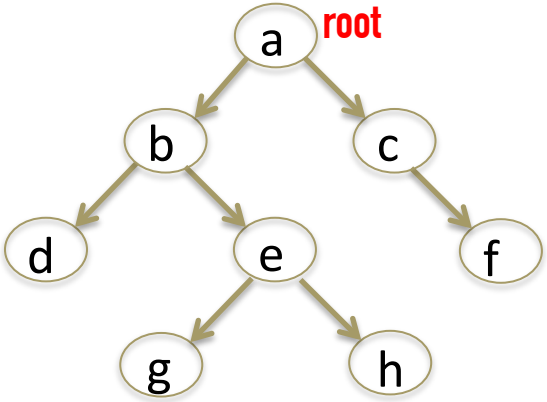
RESULT

[a, b, c, d, e]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[g, h]

t

f

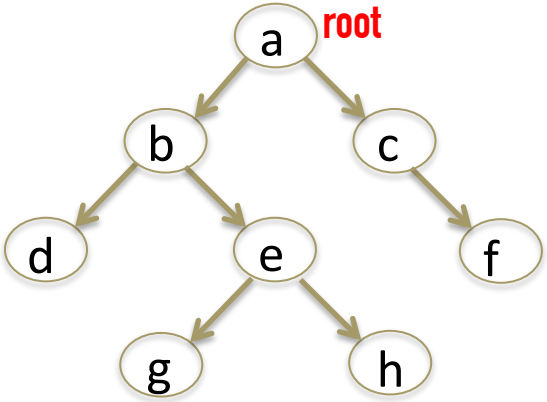
RESULT

[a, b, c, d, e, f]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[g, h]

t

f

NOTHING TO ENQUEUE

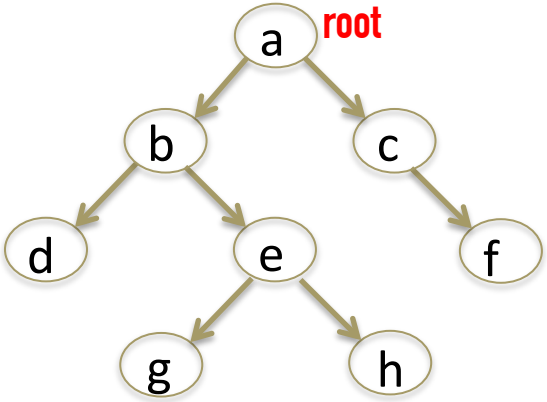
RESULT

[a, b, c, d, e, f]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[g, h]

t

f

NOTHING TO ENQUEUE

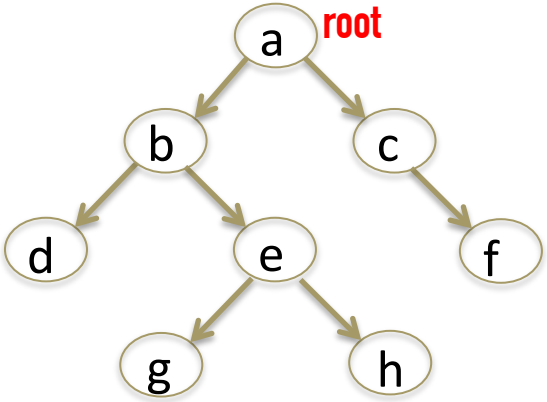
RESULT

[a, b, c, d, e, f]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[g, h]

t

f

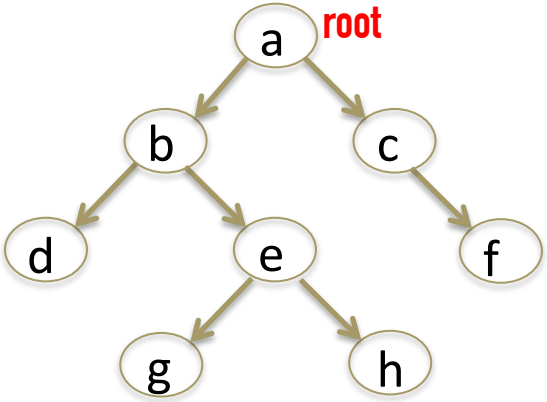
RESULT

[a, b, c, d, e, f]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[h]

t

g

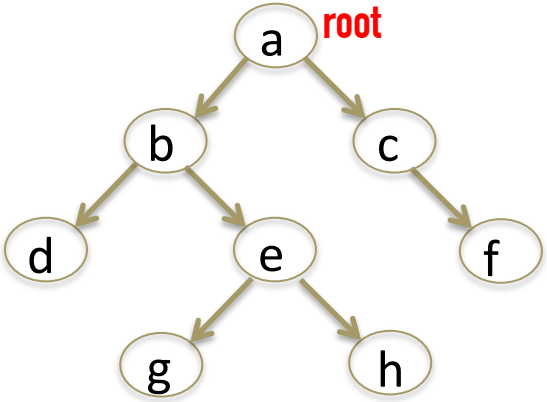
RESULT

[a, b, c, d, e, f]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[h]

t

g

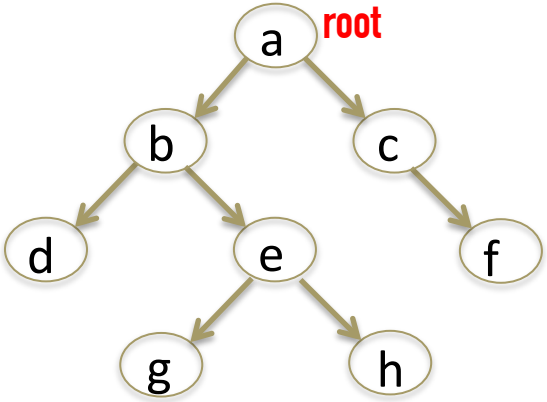
RESULT

[a, b, c, d, e, f, g]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[h]

t

g

NOTHING TO ENQUEUE

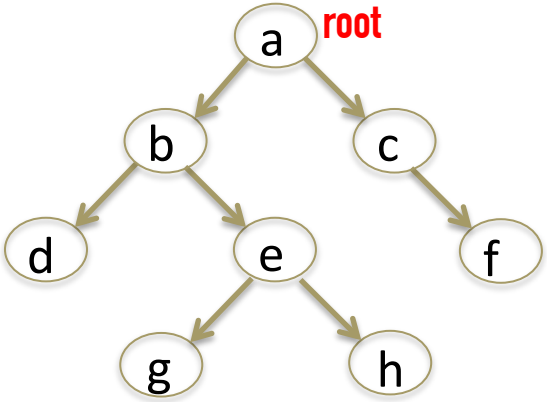
RESULT

[a, b, c, d, e, f, g]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[h]

t

g

NOTHING TO ENQUEUE

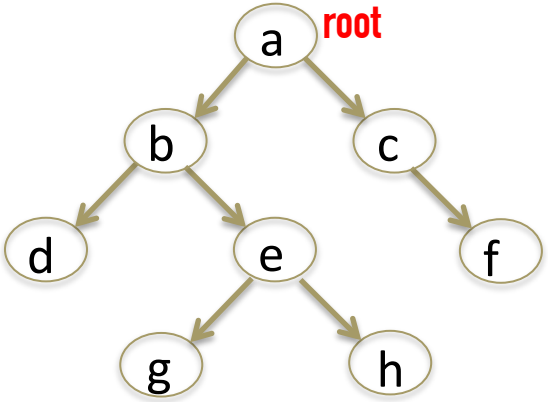
RESULT

[a, b, c, d, e, f, g]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[h]

t

g

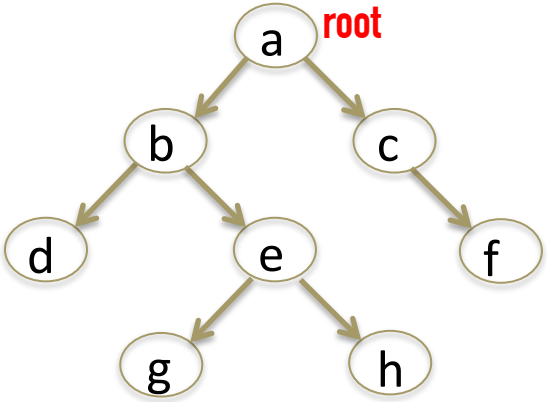
RESULT

[a, b, c, d, e, f, g]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[ ]

t

h

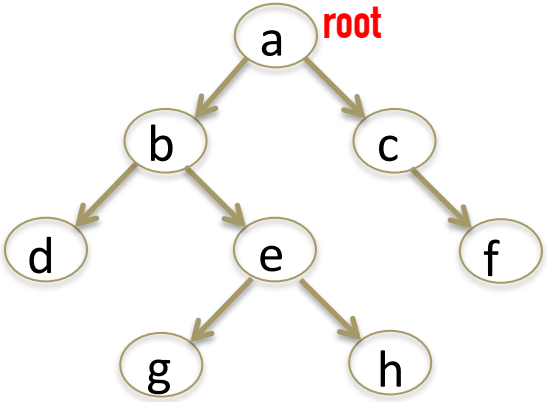
RESULT

[a, b, c, d, e, f, g]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[ ]

t

h

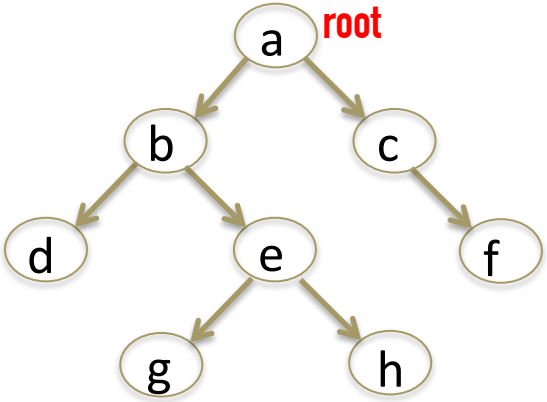
RESULT

[a, b, c, d, e, f, g, h]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[ ]

t

h

NOTHING TO ENQUEUE

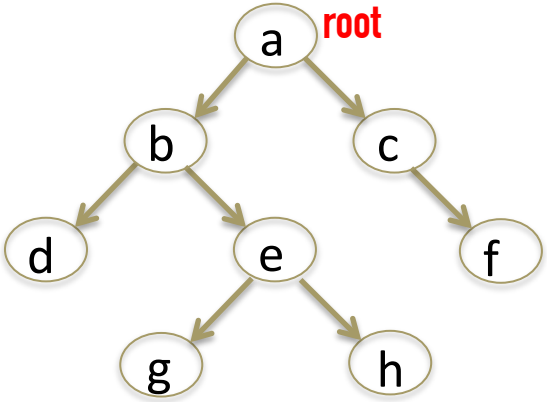
RESULT

[a, b, c, d, e, f, g, h]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[ ]

t

h

NOTHING TO ENQUEUE

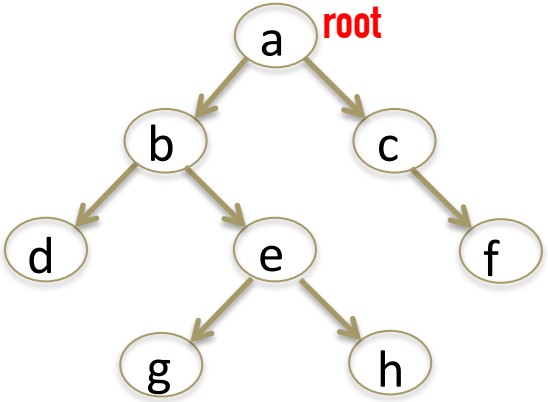
RESULT

[a, b, c, d, e, f, g, h]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[ ]

t

h

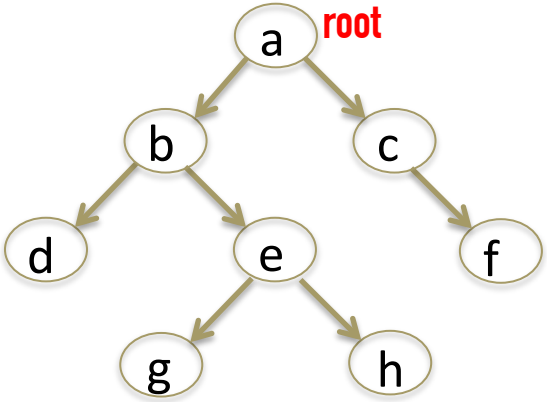
RESULT

[a, b, c, d, e, f, g, h]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[ ]

t

h

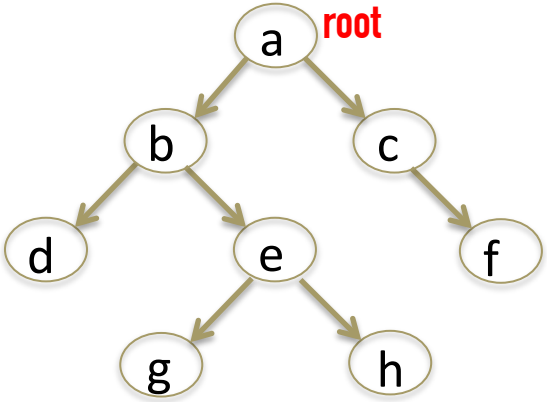
RESULT

[a, b, c, d, e, f, g, h]

Breadth-First Traversal

```
function breadth-first(root)
    Q ← new Queue()
    enqueue-if!(Q,root)
    while ¬empty?(Q)  do
        t ← dequeue!(Q)
        visit(t)
        enqueue-if!(Q,left(t))
        enqueue-if!(Q,right(t))
    end while
end function
```

```
function enqueue-if!(Q,T)
    if ¬null?(T)  then
        enqueue!(Q,T)
    end if
end function
```



Q

[ ]

t

h

RESULT

[a, b, c, d, e, f, g, h]