COMP108
Data Structures and Algorithms

Trees (Part II Binary Trees)

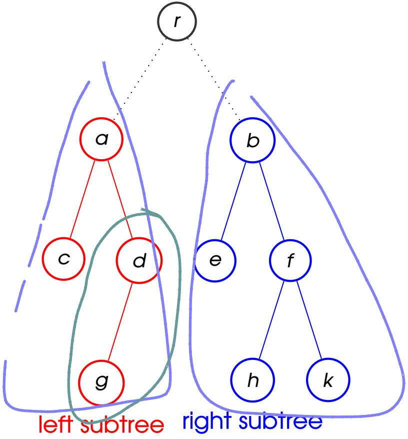
Professor Prudence Wong

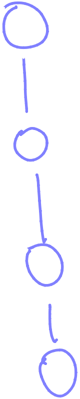
pwong@liverpool.ac.uk

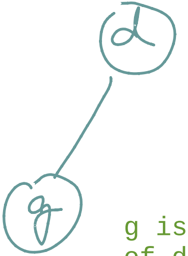
2022-23

Binary tree

- a tree of degree at most TWO
- the two subtrees are called left subtree and right subtree (may be empty)



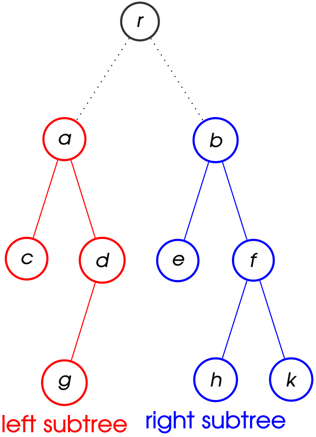




g is left child of d

Binary tree

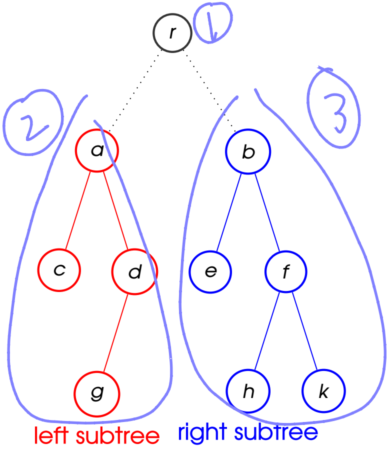
- a tree of degree at most TWO
- the two subtrees are called left subtree and right subtree (may be empty)



There are three common ways to traverse a binary tree.

Binary tree

- a tree of degree at most TWO
- the two subtrees are called left subtree and right subtree (may be empty)

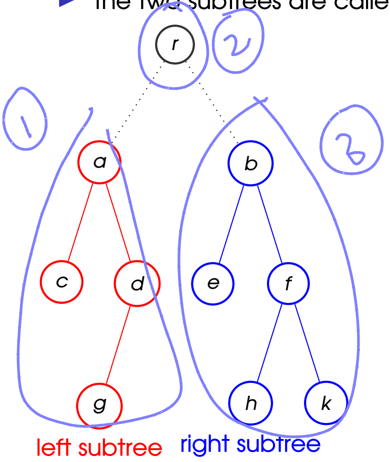


There are three common ways to traverse a binary tree.

- **preorder** traversal (VLR) - vertex V, left subtree L, right subtree R

Binary tree

- a tree of degree at most TWO
- the two subtrees are called left subtree and right subtree (may be empty)

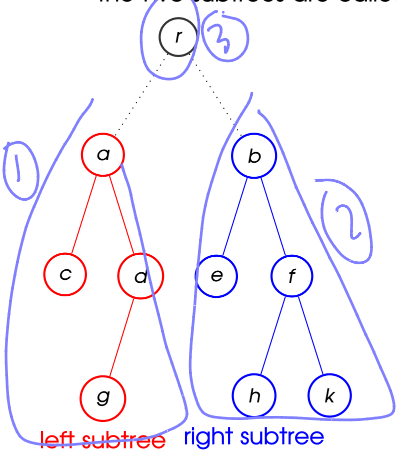


There are three common ways to traverse a binary tree.

- **preorder** traversal (VLR) - vertex V, left subtree L, right subtree R
- **inorder** traversal (LVR) - left subtree L, vertex V, right subtree R

Binary tree

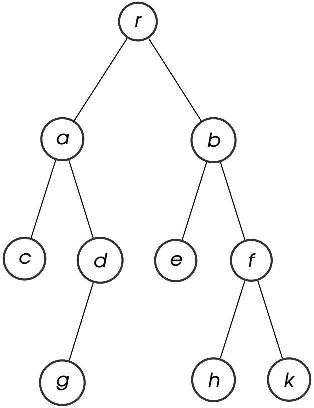
- a tree of degree at most TWO
- the two subtrees are called left subtree and right subtree (may be empty)



There are three common ways to traverse a binary tree.

- **preorder** traversal (VLR) - vertex V, left subtree L, right subtree R
- **inorder** traversal (LVR) - left subtree L, vertex V, right subtree R
- **postorder** traversal (LRV) - left subtree L, right subtree R, vertex V

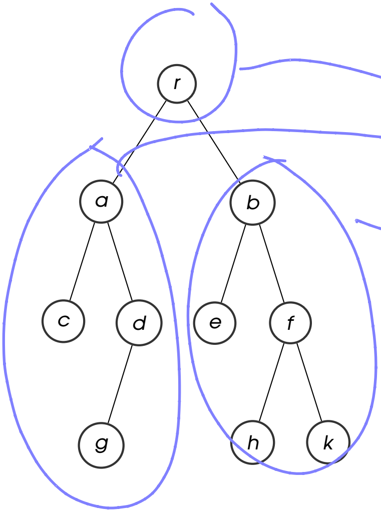
Traversing a binary tree



preorder traversal (VLR)

$r, a, c, d, g, b, e, f, h, k$

Traversing a binary tree

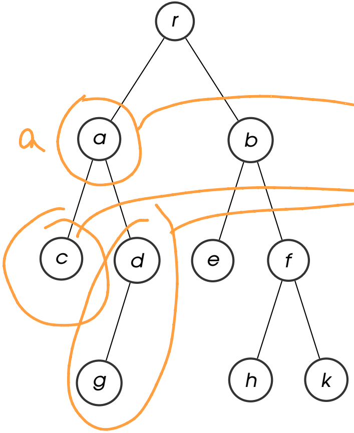


preorder traversal (VLR)

r, *a*, *c*, *d*, *g*, *b*, *e*, *f*, *h*, *k*

- ▶ *r*, (L of *r*), (R of *r*)

Traversing a binary tree

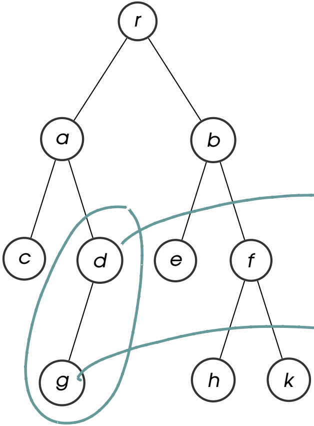


preorder traversal (VLR)

r, *a*, *c*, *d*, *g*, *b*, *e*, *f*, *h*, *k*

- *r*, (L of *r*), (R of *r*)
- L of *r*: *a*, (L of *a*), (R of *a*)
- R of *r*: *b*, (L of *b*), (R of *b*)

Traversing a binary tree



preorder traversal (VLR)

r, *a*, *c*, *d*, *g*, *b*, *e*, *f*, *h*, *k*

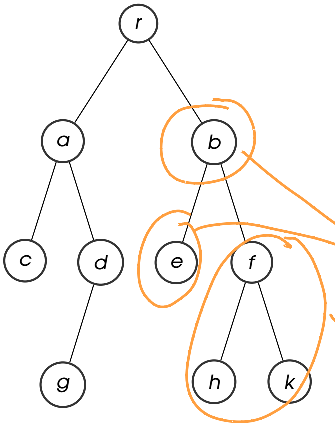
a, c, d, g

- *r*, (L of *r*), (R of *r*)
- L of *r*: *a*, (L of *a*), (R of *a*)
- L of *a*: *c*
- R of *a*: *d*, (L of *d*), (R of *d*) $\Longrightarrow$ *d*, *g*

empty

- R of *r*: *b*, (L of *b*), (R of *b*)

Traversing a binary tree

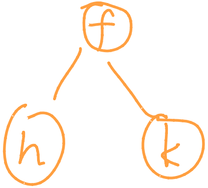


preorder traversal (VLR)

$r, a, c, d, g, b, e, f, h, k$

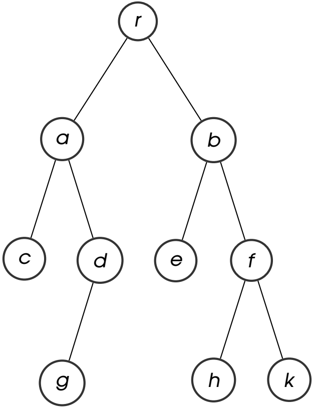
- r, (L of r), (R of r)
- L of r: a, (L of a), (R of a)
- L of a: c
- R of a: d, (L of d), (R of d) $\Longrightarrow$ d, g
- $\Longrightarrow$ L of r: a, c, d, g
- R of r: b, (L of b), (R of b)

e



f h k

Traversing a binary tree

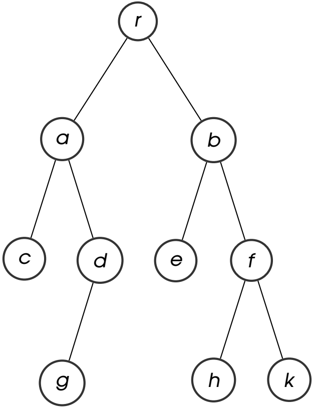


preorder traversal (VLR)

$r, a, c, d, g, b, e, f, h, k$

- r, (L of r), (R of r)
- L of r: a, (L of a), (R of a)
- L of a: c
- R of a: d, (L of d), (R of d) $\Longrightarrow$ d, g
- $\Longrightarrow$ L of r: a, c, d, g
- R of r: b, (L of b), (R of b)
- L of b: e
- R of b: f, (L of f), (R of f) $\Longrightarrow$ f, h, k

Traversing a binary tree

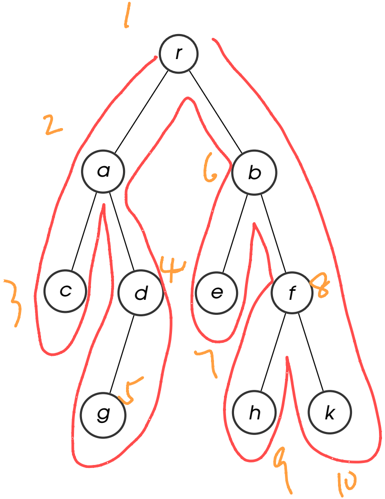


preorder traversal (VLR)

$r, a, c, d, g, b, e, f, h, k$

- r, (L of r), (R of r)
- L of r: a, (L of a), (R of a)
- L of a: c
- R of a: d, (L of d), (R of d) $\implies$ d, g
- $\implies$ L of r: a, c, d, g
- R of r: b, (L of b), (R of b)
- L of b: e
- R of b: f, (L of f), (R of f) $\implies$ f, h, k
- $\implies$ R of r: b, e, f, h, k

Traversing a binary tree

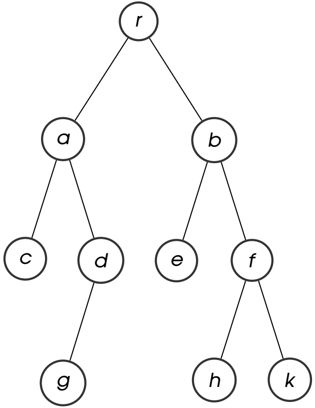


preorder traversal (VLR)

r, a, c, d, g, b, e, f, h, k

- *r*, (L of *r*), (R of *r*)
- L of *r*: *a*, (L of *a*), (R of *a*)
- L of *a*: *c*
- R of *a*: *d*, (L of *d*), (R of *d*) $\Longrightarrow$ *d*, *g*
- $\Longrightarrow$ L of *r*: *a*, *c*, *d*, *g*
- R of *r*: *b*, (L of *b*), (R of *b*)
- L of *b*: *e*
- R of *b*: *f*, (L of *f*), (R of *f*) $\Longrightarrow$ *f*, *h*, *k*
- $\Longrightarrow$ R of *r*: *b*, *e*, *f*, *h*, *k*
- Final: *r*, *a*, *c*, *d*, *g*, *b*, *e*, *f*, *h*, *k*

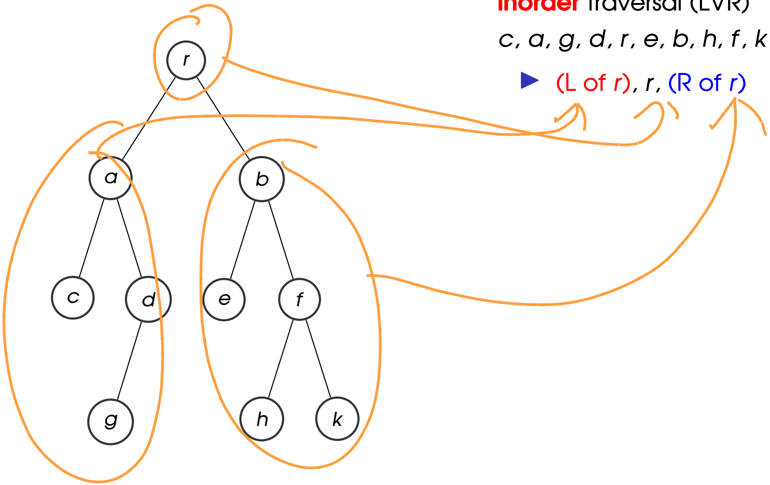
Traversing a binary tree



inorder traversal (LVR)

$c, a, g, d, r, e, b, h, f, k$

Traversing a binary tree

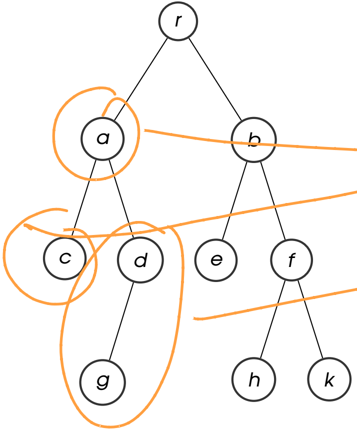


inorder traversal (LVR)

c, *a*, *g*, *d*, *r*, *e*, *b*, *h*, *f*, *k*

- (L of *r*), *r*, (R of *r*)

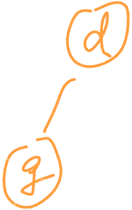
Traversing a binary tree



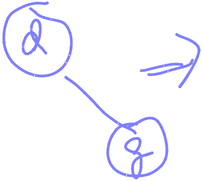
inorder traversal (LVR)

c, *a*, *g*, *d*, *r*, *e*, *b*, *h*, *f*, *k*

- (L of *r*), *r*, (R of *r*)
- L of *r*: (L of *a*), *a*, (R of *a*)
- R of *r*: (L of *b*), *b*, (R of *b*)

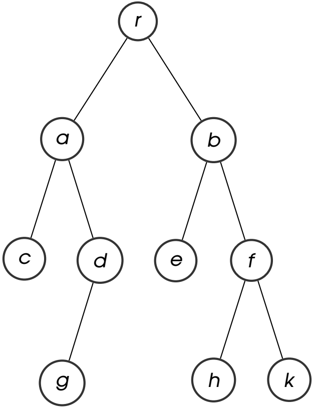


g , d



→ d, g

Traversing a binary tree

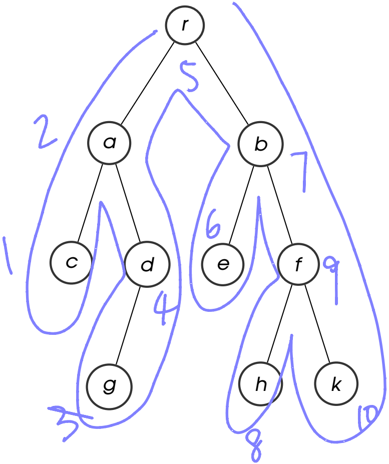


inorder traversal (LVR)

$c, a, g, d, r, e, b, h, f, k$

- (L of r), r, (R of r)
- L of r: (L of a), a, (R of a)
- L of a: c
- R of a: (L of d), d, (R of d) $\Longrightarrow$ g, d
- R of r: (L of b), b, (R of b)

Traversing a binary tree

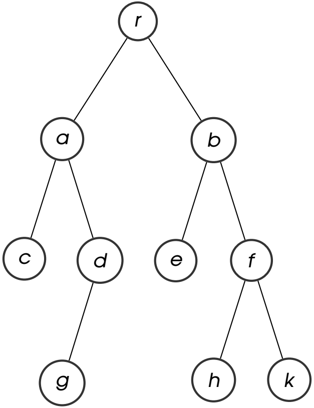


inorder traversal (LVR)

c, a, g, d, r, e, b, h, f, k

- (L of *r*), *r*, (R of *r*)
- L of *r*: (L of *a*), *a*, (R of *a*)
- L of *a*: *c*
- R of *a*: (L of *d*), *d*, (R of *d*) ⟹ *g*, *d*
- ⟹ L of *r*: *c*, *a*, *g*, *d*
- R of *r*: (L of *b*), *b*, (R of *b*)

Traversing a binary tree

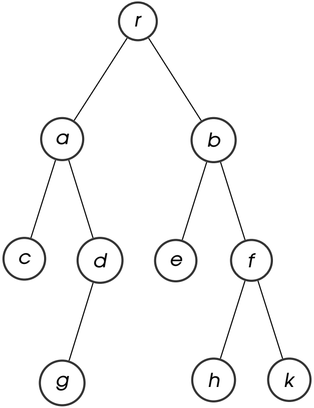


inorder traversal (LVR)

$c, a, g, d, r, e, b, h, f, k$

- (L of r), r, (R of r)
- L of r: (L of a), a, (R of a)
- L of a: c
- R of a: (L of d), d, (R of d) $\Longrightarrow$ g, d
- $\Longrightarrow$ L of r: c, a, g, d
- R of r: (L of b), b, (R of b)
- L of b: e
- R of b: (L of f), f, (R of f) $\Longrightarrow$ h, f, k

Traversing a binary tree

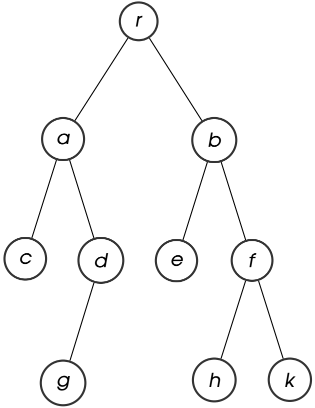


inorder traversal (LVR)

$c, a, g, d, r, e, b, h, f, k$

- (L of r), r, (R of r)
- L of r: (L of a), a, (R of a)
- L of a: c
- R of a: (L of d), d, (R of d) $\Longrightarrow$ g, d
- $\Longrightarrow$ L of r: c, a, g, d
- R of r: (L of b), b, (R of b)
- L of b: e
- R of b: (L of f), f, (R of f) $\Longrightarrow$ h, f, k
- $\Longrightarrow$ R of r: e, b, h, f, k

Traversing a binary tree

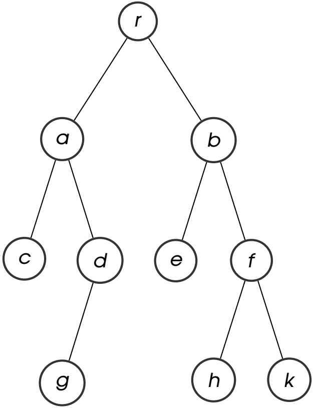


inorder traversal (LVR)

$c, a, g, d, r, e, b, h, f, k$

- (L of r), r, (R of r)
- L of r: (L of a), a, (R of a)
- L of a: c
- R of a: (L of d), d, (R of d) $\implies$ g, d
- $\implies$ L of r: c, a, g, d
- R of r: (L of b), b, (R of b)
- L of b: e
- R of b: (L of f), f, (R of f) $\implies$ h, f, k
- $\implies$ R of r: e, b, h, f, k
- Final: c, a, g, d, r, e, b, h, f, k

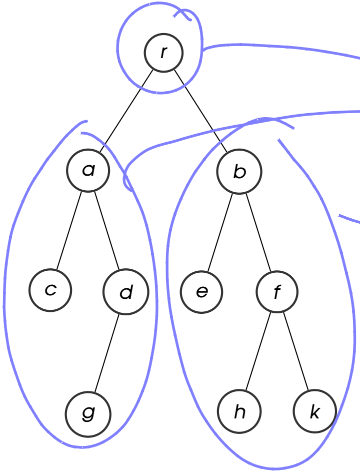
Traversing a binary tree



postorder traversal (LRV)

$c, g, d, a, e, h, k, f, b, r$

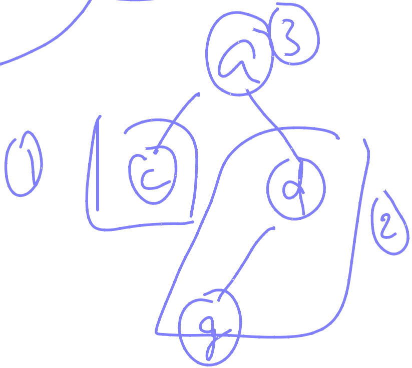
Traversing a binary tree



postorder traversal (LRV)

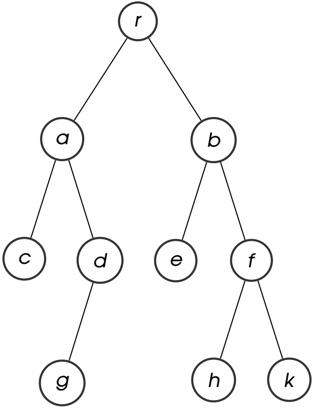
c, *g*, *d*, *a*, *e*, *h*, *k*, *f*, *b*, *r*

▶ (L of *r*), (R of *r*), *r*



c, d, g, a

Traversing a binary tree

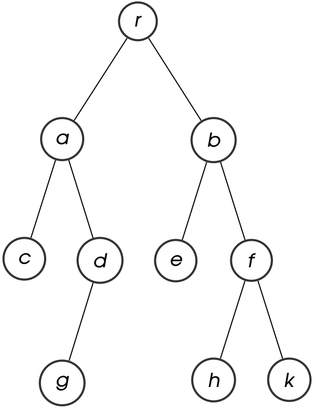


postorder traversal (LRV)

c, *g*, *d*, *a*, *e*, *h*, *k*, *f*, *b*, *r*

- (L of *r*), (R of *r*), *r*
- L of *r*: (L of *a*), (R of *a*), *a*
- R of *r*: (L of *b*), (R of *b*), *b*

Traversing a binary tree

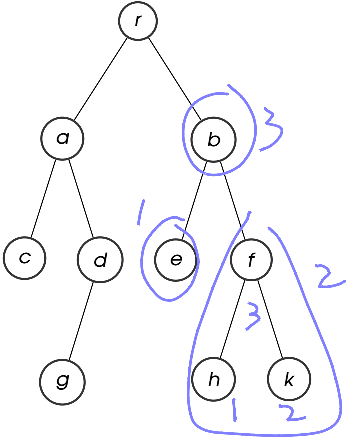


postorder traversal (LRV)

$c, g, d, a, e, h, k, f, b, r$

- (L of r), (R of r), r
- L of r: (L of a), (R of a), a
- L of a: c
- R of a: (L of d), (R of d), d $\Longrightarrow$ g, d
- R of r: (L of b), (R of b), b

Traversing a binary tree



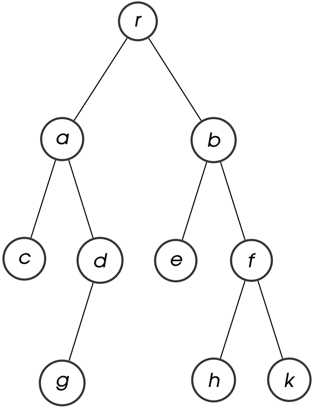
postorder traversal (LRV)

$c, g, d, a, e, h, k, f, b, r$

- (L of r), (R of r), r
- L of r: (L of a), (R of a), a
- L of a: c
- R of a: (L of d), (R of d), d $\Longrightarrow$ g, d
- $\Longrightarrow$ L of r: c, g, d, a
- R of r: (L of b), (R of b), b

e h k f b

Traversing a binary tree

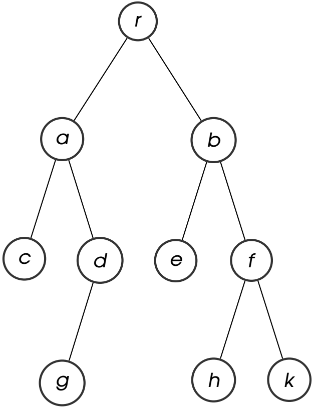


postorder traversal (LRV)

$c, g, d, a, e, h, k, f, b, r$

- (L of r), (R of r), r
- L of r: (L of a), (R of a), a
- L of a: c
- R of a: (L of d), (R of d), d $\Longrightarrow$ g, d
- $\Longrightarrow$ L of r: c, g, d, a
- R of r: (L of b), (R of b), b
- L of b: e
- R of b: (L of f), (R of f), f $\Longrightarrow$ h, k, f

Traversing a binary tree

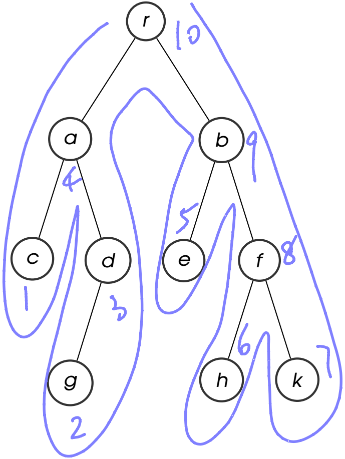


postorder traversal (LRV)

$c, g, d, a, e, h, k, f, b, r$

- (L of r), (R of r), r
- L of r: (L of a), (R of a), a
- L of a: c
- R of a: (L of d), (R of d), d $\Longrightarrow$ g, d
- $\Longrightarrow$ L of r: c, g, d, a
- R of r: (L of b), (R of b), b
- L of b: e
- R of b: (L of f), (R of f), f $\Longrightarrow$ h, k, f
- $\Longrightarrow$ R of r: e, h, k, f, b

Traversing a binary tree



postorder traversal (LRV)

$c, g, d, a, e, h, k, f, b, r$

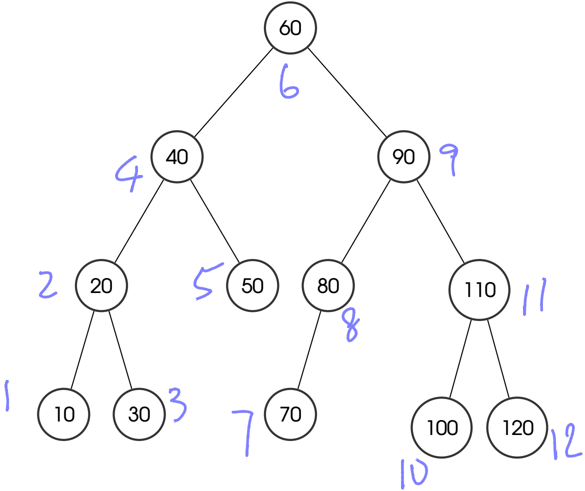
- (L of r), (R of r), r
- L of r: (L of a), (R of a), a
- L of a: c
- R of a: (L of d), (R of d), d ⟹ g, d
- ⟹ L of r: c, g, d, a
- R of r: (L of b), (R of b), b
- L of b: e
- R of b: (L of f), (R of f), f ⟹ h, k, f
- ⟹ R of r: e, h, k, f, b
- Final: $c, g, d, a, e, h, k, f, b, r$

Binary Search Tree

For a vertex with value X

- left child has value $\leq X$
- right child has value $> X$

```
balanced tree
depth (number of level)
is O(log n)
```

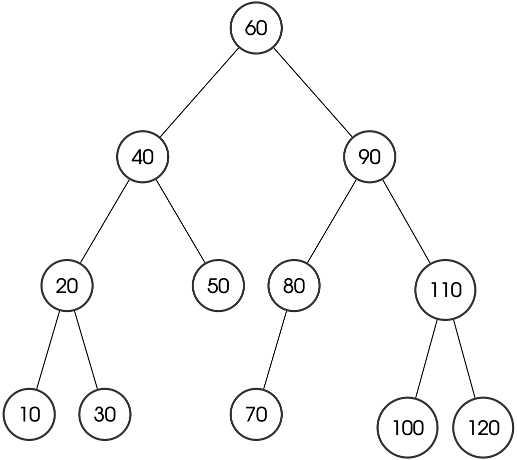


which traversal gives numbers in ascending order?

Binary Search Tree

For a vertex with value X

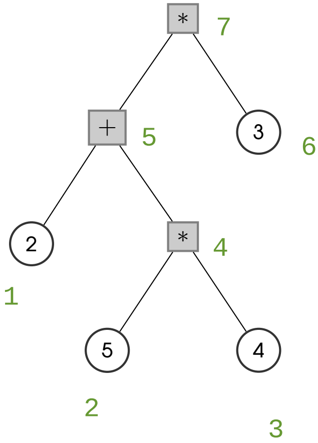
- left child has value $\leq X$
- right child has value $> X$



inorder traversal

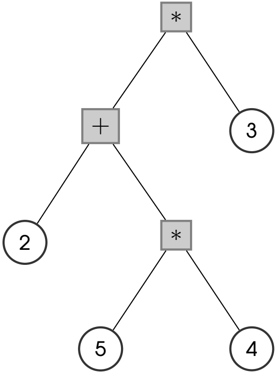
which traversal gives numbers in ascending order?

Expression Tree



- infix: (2+5*4)*3
- postfix: 2 5 4 * + 3 *

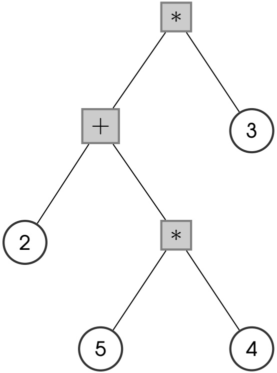
Expression Tree



- infix: (2+5*4)*3
- postfix: 2 5 4 * + 3 *

which traversal gives postfix representation?

Expression Tree

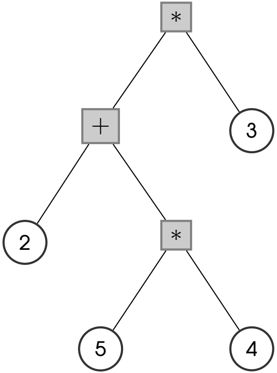


- infix: (2+5*4)*3
- postfix: 2 5 4 * + 3 *

which traversal gives postfix representation?

postorder traversal

Expression Tree



- infix: (2+5*4)*3
- postfix: 2 5 4 * + 3 *

which traversal gives postfix representation?

postorder traversal

Recall that we can use a stack to evaluate a postfix expression

Summary

Summary: Trees

Next: Graphs

For note taking