Tree Traversal

iterating through every element in a tree

3 choices at every step:

- go to left (NOT access left value - go there to make the next decision)
- go right (NOT access right value, - go there to make the next decision)
- visit current element (access the value)

when visiting child nodes, always visit left before right (by convention)

Depth First

visit a node's children before its siblings.

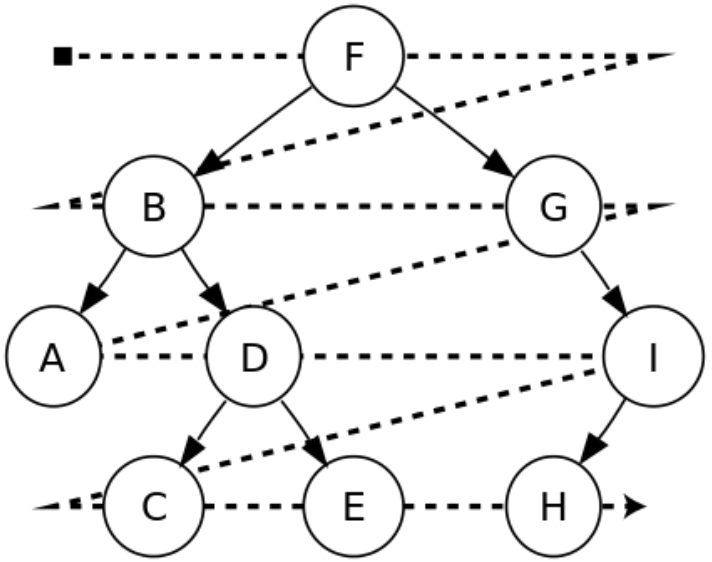
3 ways to do this:

- pre-order
- in-order
- post-order

Breadth First

visit a node's siblings before its children

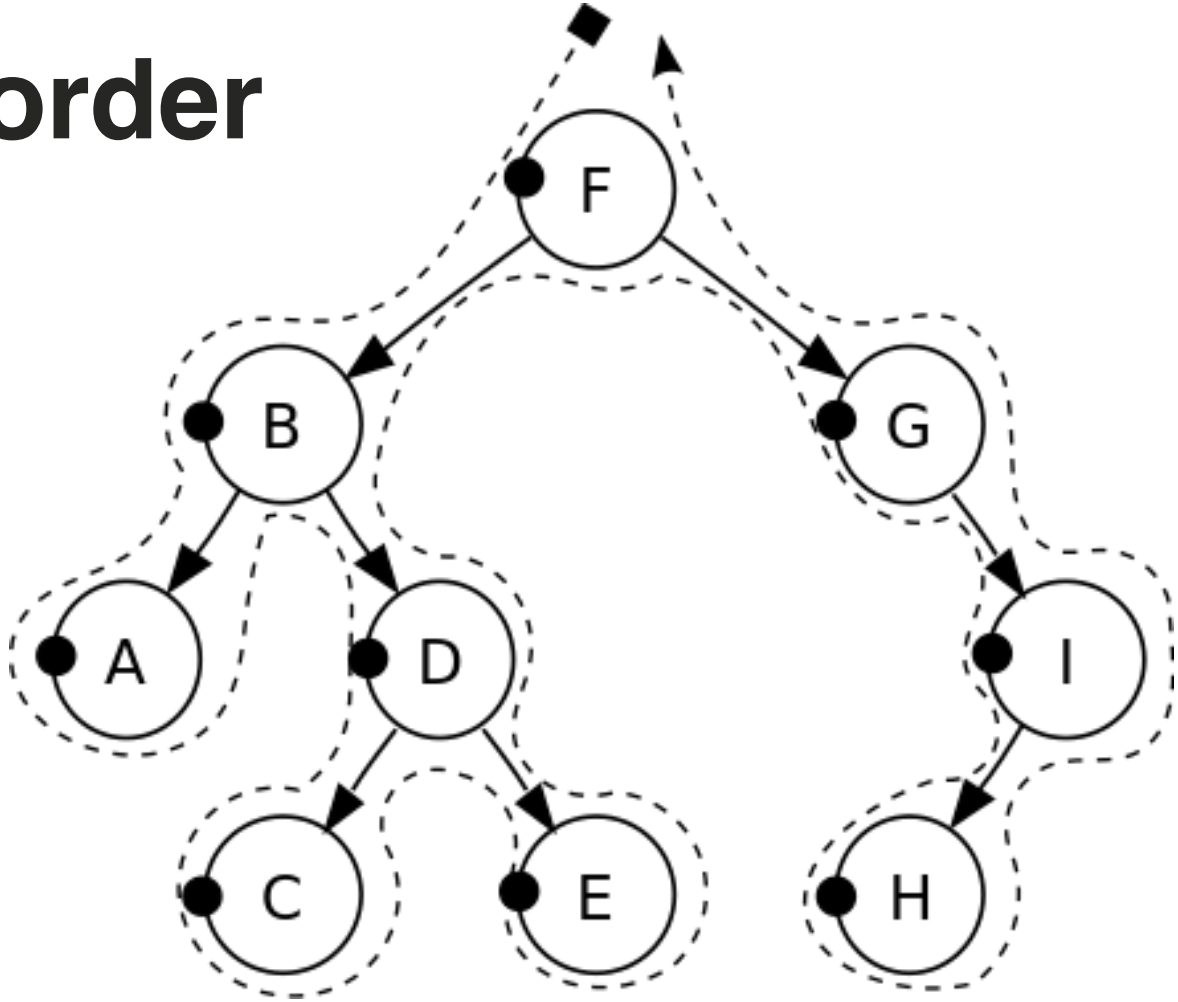
- level-order



Depth First: **pre-order**

parent then children

F, B, A, D, C, E, G, I, H



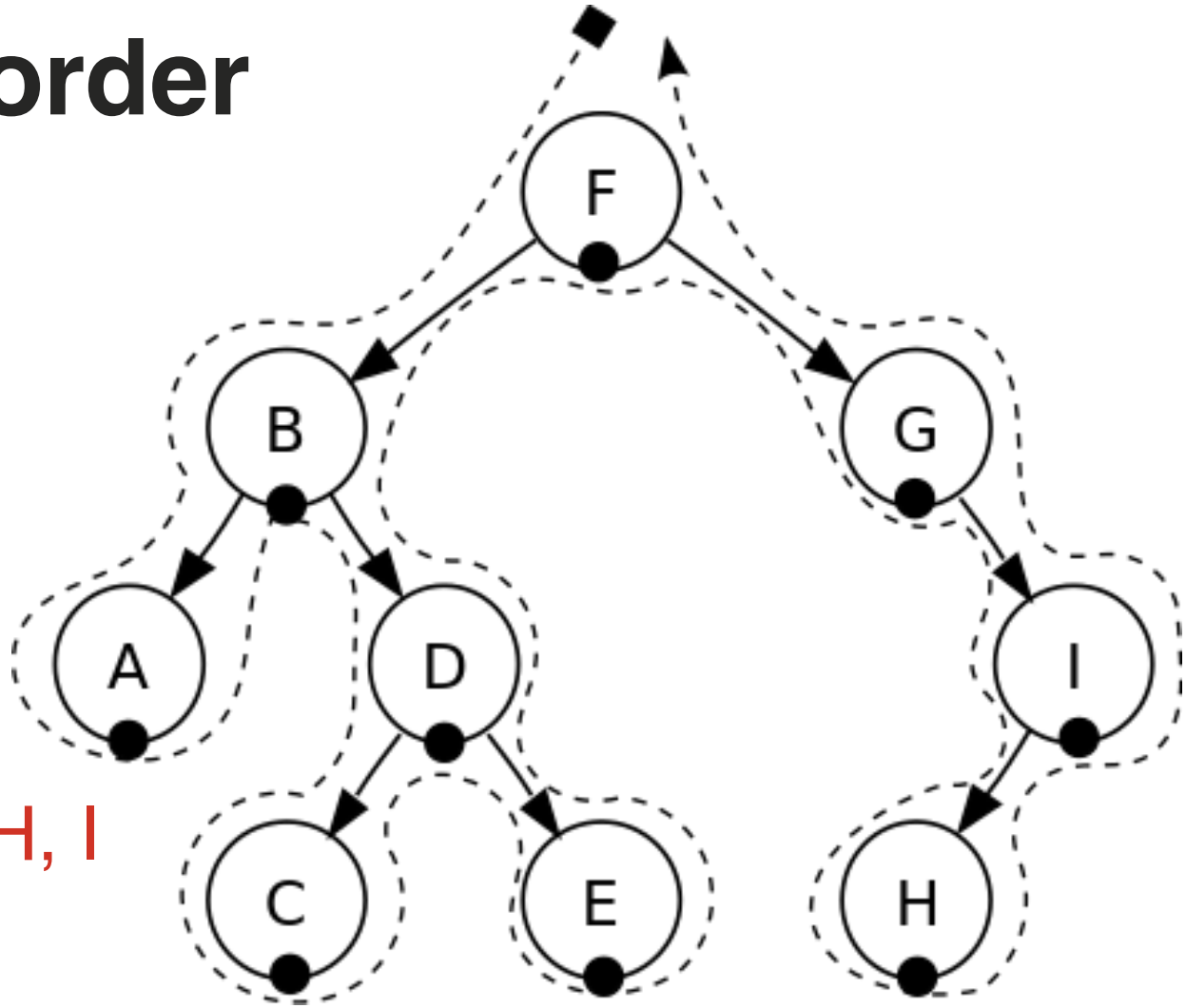
Depth First: **in-order**

left children,

then parent,

then right children

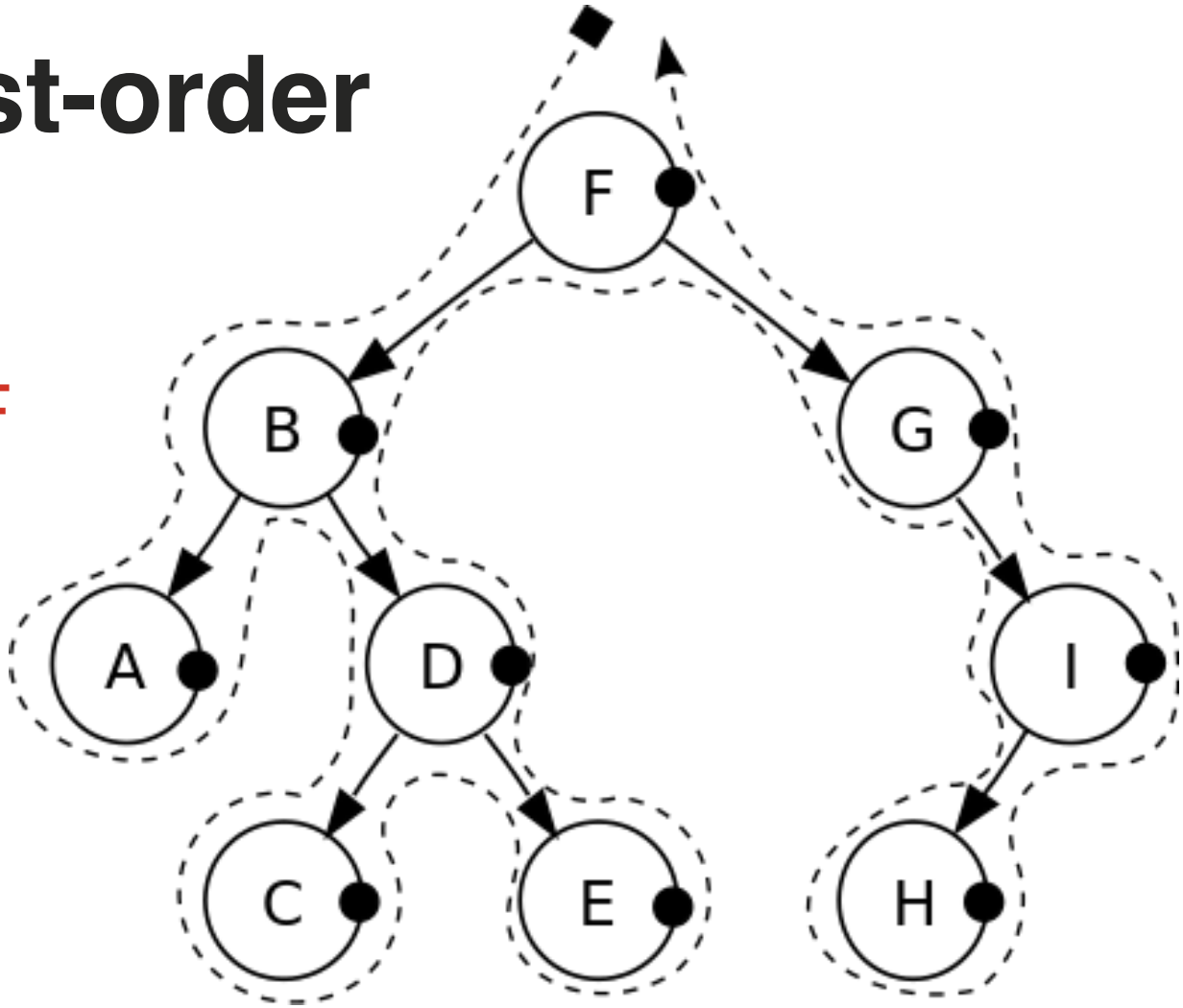
A, B, C, D, E, F, G, H, I



Depth First: **post-order**

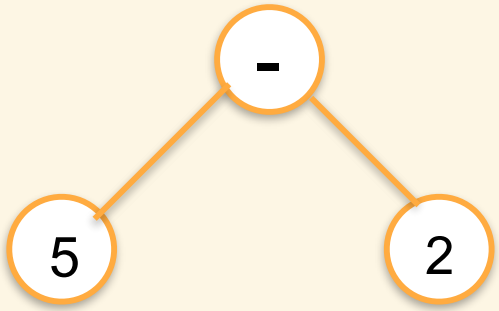
parent after children

A, C, E, D, B, H, I, G, F



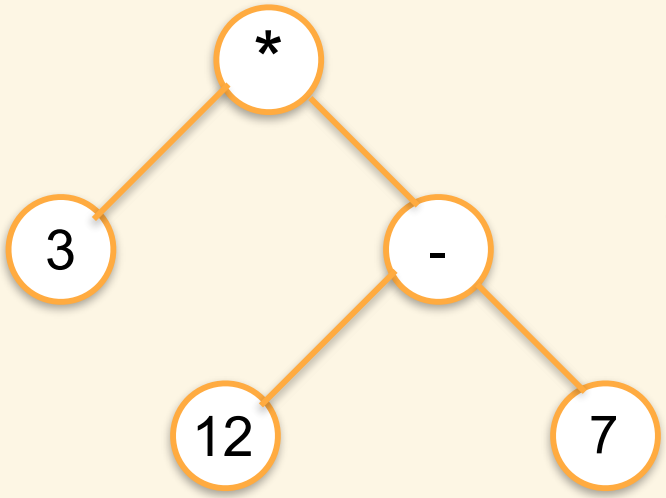
Use case: mathematics

Each operator can only “operate” on 2 elements, so those can be represented as child nodes.



example - infix notation: 5 - 2
reverse polish notation: 5 2 -

Use case: mathematics



example - infix notation: 3 * (12- 7)
reverse polish notation: 3 12 7 - *
(no need for parens with reverse polish notation!)

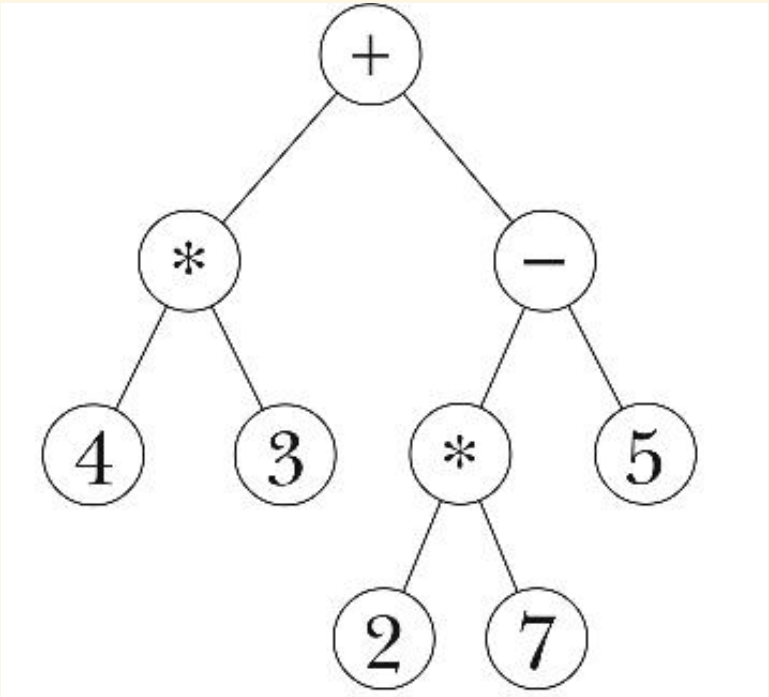
Write a function that take a binary tree (node) like the one to the left and produces the string representation as presented above.

Choose either infix or postfix

Tell your neighbor:

1.Write the string for this math problem in infix (e.g. x + y) and reverse polish notations (e.g. x y +)

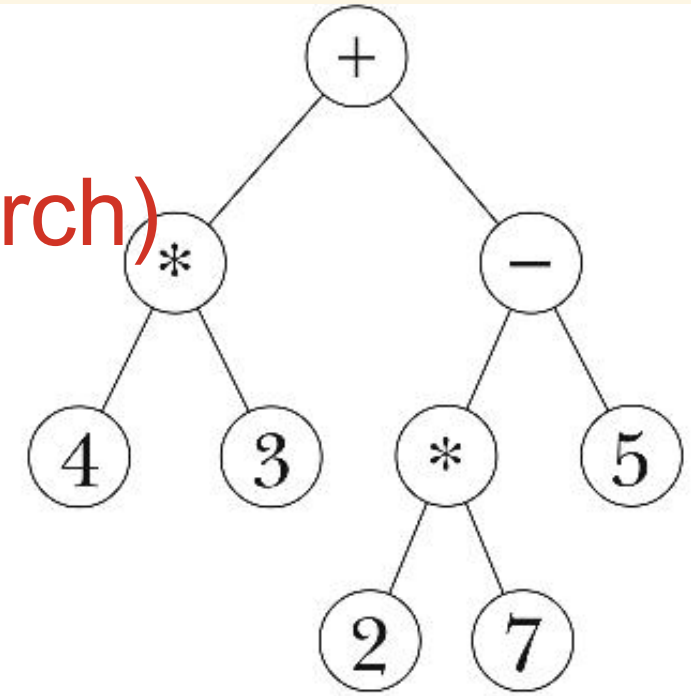
2. What kind of traversal would you use to turn the tree into the into each string?



Tell your neighbor:

1.Write the string for this math problem in infix (e.g. x + y) and reverse polish notations (e.g. x y +)

2. What kind of traversal would you use to turn the tree into the into each string?



infix: “4*3+2*7-5” (in-order search)

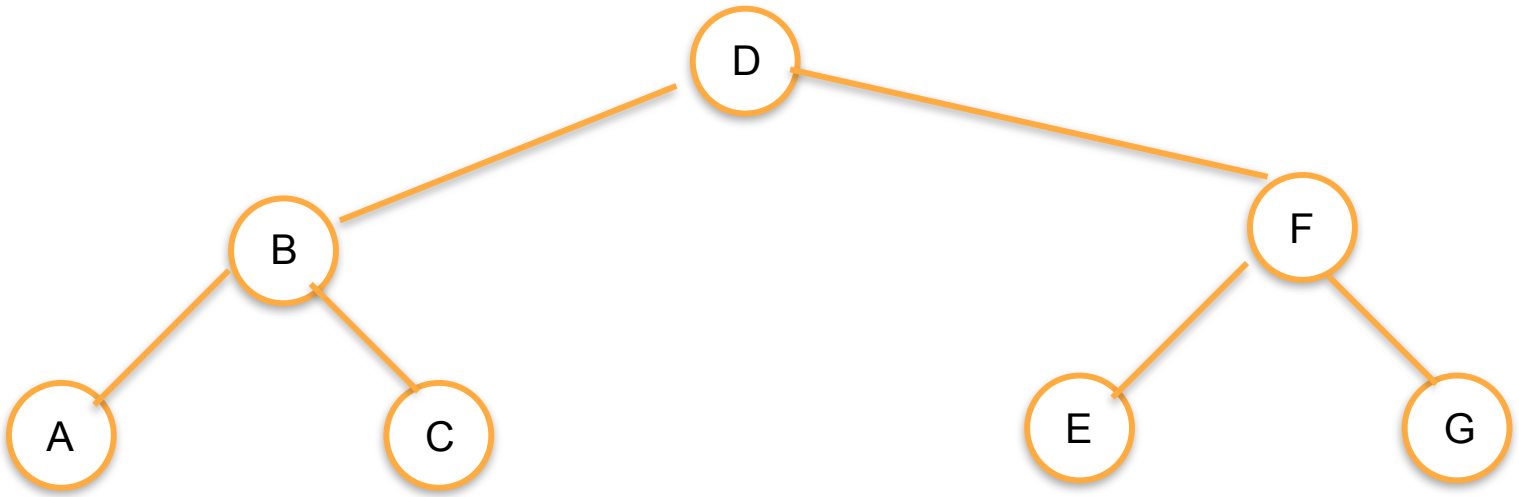
postfix: “4 3 * 2 7 * 5 - +
(post-order search)

Whiteboarding: **Depth-Frist**

Implement these traversal functions for a binary tree

Remember: trees are recursive data structures… hint hint

- **inOrder()**
- **preOrder()**
- **postOrder()**



If you finish early, try to come up with a Tree.toString method that prints out a string for an in-order search

Whiteboarding: **Breadth-First**

implement a breadth-first search function for this tree

- **levelOrder()**