Breadth-First Traversal

→ recall that BFS is not recursive; it relies on a queue of positions to manage traversal

```
def breadthfirst(self):
  if not self.is_empty()
    fringe = LinkedQueue()
    fringe.enqueue(self.root())          // known positions not yet yielded, start with root
    while not fringe.is_empty()
      p = fringe.dequeue()               // remove from front of queue
      yield p                            // report position
      for c in self.children(p)
        fringe.enqueue(c)                // add children to back of queue
```

Inorder Traversal

→ only applies to binary trees because it relies on the notion of a left & right child node

```
def inorder(self):
  if not self.is_empty()
    for p in self._subtree_inorder(self.root()):
      yield p

def _subtree_inorder(self, p):
  if self.left(p) not None:                  // if left child exists traverse subtree
    for other in self._subtree_inorder(self.left(p))
      yield other
  yield p                                    // visit p between subtrees
  if self.right(p) not None:                 // if right child exists traverse subtree
    for other in self._subtree_inorder(self.right(p))
      yield other
```

→ for many applications of binary trees, inorder traversal provides natural iteration

```
def positions(self):
  return self.inorder()      // make inorder default
```