Tree-walking algorithms: Incrementally performing an inorder walk of a binary tree

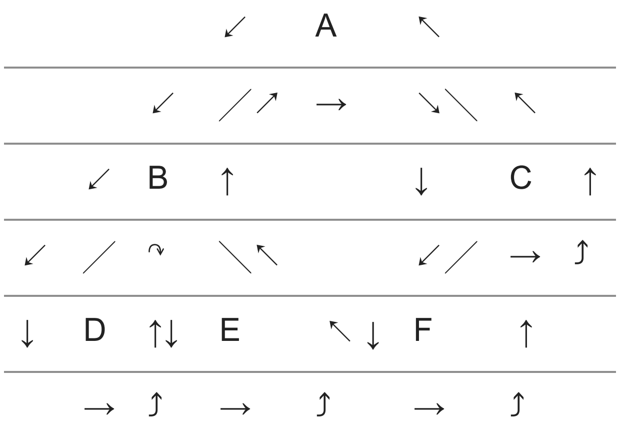
devblogs.microsoft.com/oldnewthing/20200109-00

January 9, 2020

Raymond Chen

We continue our exploration of algorithms for walking incrementally through a tree by perform an inorder walk through a binary tree. (Note that inorder traversal is meaningful only for binary trees.)

Recall that our goal is to follow the red arrows through the tree, as if we are walking along the outside of the tree with our left hand touching it.



Since we are doing an inorder traversal, the fact that we stopped on a node means that we have finished enumerating the left child and need to start enumerating the right child, if there is one.

If there is no right child, then we have finished enumerating the entire subtree, and we need to walk up the tree looking for another subtree to the right. This step is a little tricky: When we move to the parent, we need to check whether we moved up from the left child or from the right child. If we moved up from the left child, then the right child is the subtree to the right. If we moved up from the right child, then there is no subtree to the right at this level, and we need to go up another level.

```
class InorderWalker
{
  private TreeCursor cursor;

  public InorderWalker(TreeNode node)
  {
    cursor = new TreeCursor(node);
    GoDeepLeft();
  }

  public bool MoveNext()
  {
    if (cursor.TryMoveToRightChild()) {
      GoDeepLeft();
      return true;
    }

    TreeNode start;
    do {
      start = cursor.Current;
      if (!cursor.TryMoveToParent()) {
        return false;
      }
    } while (start != cursor.Current.LeftChild);
    return true;
  }

  public TreeNode Current => cursor.Current;

  private void GoDeepLeft()
  {
    while (cursor.TryMoveToLeftChild()) { }
  }
}
```

I think that's enough incremental tree traversal algorithms for now.

Raymond Chen

Follow