Euler Tours

→ generally defined as a "walk" around T, where we start by going from the root towards its leftmost child, viewing edges of T as being walls which we always keep to left

→ complexity of walk is O(n), because it progresses exactly 2 times along each of the n-1 edges of the tree

- a pre-visit occurs when first reaching the position
- a post-visit occurs when the walk later proceeds upward from that position

→ Euler tour can be performed recursively:

```
Algorithm eulertour(T, p):
    perform pre visit for position p
    for each child c in T.children(c) do
        eulertour(T, c)                     // recursively tour subtree
    perform post visit for position p
```

→ Python Implementation of a Euler Tour (algorithm only)

```
def execute(self)
    if len(self._tree) > 0
        return self._tour(self._tree.root(), 0, [])    // start recursion

def _tour(self, p, d, path):
    self._hook_previsit(p, d, path)                   // previsit p
    results = []
    path.append(0)                                    // ~~[illegible]~~ add new index to end of path b4 recursion
    for c in self._tree.children(p)                   //
        results.append(self._tour(c, d+1, path))      // recur on subtree
        path[-1] += 1                                 // increment index
path.pop()                                            // remove extraneous index from end of path
answer = self._hook_postvisit(p, d, path, results)    // post visit p
return answer
```