Found Possible Solutions.

Python Solution with text file
https://stackoverflow.com/questions/6075974/python-file-parsing-build-tree-from-text-file

A recursive python implementation which takes a file with a similar root structure and parses it into an in memory data structure (a stack) and prints this stack.

```
ROOT
    Node1
        Node2
            Node3
                Node4
    Node5
    Node6
```

—>

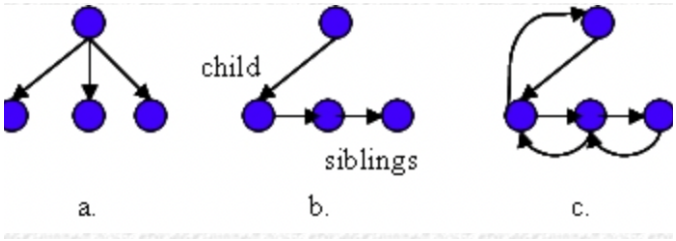
```
['ROOT']
['ROOT', 'Node1']
['ROOT', 'Node1', 'Node2']
['ROOT', 'Node1', 'Node2', 'Node3']
['ROOT', 'Node1', 'Node2', 'Node3', 'Node4']
['ROOT', 'Node5']
['ROOT', 'Node6']
```

Things which are similar to my solution
See points 1,2,3

Cons
See points 1,3,4

Java Solution with XML File
http://bearcave.com/software/java/xml/treebuilder.html

Takes a complex XML document and processes it in to an in-memory tree. The in-memory tree can then be traversed and/or modified to produce a result. The tree is constructed using the solutions TreeBuilder object as processing is easier when the document is represented as an in-memory tree.



Things which are similar to my solution
See points 1,3

Cons
See points 2

C++ Text file json-esque structure into in memory Tree
https://stackoverflow.com/questions/52782045/parsing-text-file-into-tree-like-data-structure

Can take a simple text file with its data structure to a similar way as a JSON file and parse its elements into an in memory data structure (n-ary tree), this allows for easy processing and navigation down branches.

```
{
    Element_A (3)
    Element_B (3,4)

    {
        Element_B (6,24)
        Element_A (1)
    }

    {
        Element_A (3)

        {
            Element_A (4)
            Element_B (12,6)
        }

        Element_B (1,4)
    }
}
```



```
Node: a(0, 0), b(0) {
  Node: a(3, 0), b(3) {
    Node: a(1, 0), b(6) {
    }
    Node: a(3, 0), b(1) {
      Node: a(4, 0), b(12) {
      }
    }
  }
}
```

Things which are similar to my solution
See point 3

Cons
See points 1,6

Java Directories Solution
https://stackoverflow.com/questions/3154488/how-do-i-iterate-through-the-files-in-a-directory-and-its-sub-directories-in-ja

Uses the inbuilt java.io.File functionality to transverse though an existing file structure to find all path roots and it sub-directories. Uses the use Files#walk() function which utilises tail recursion.

Things which are similar to my solution
None

Cons
See point 5

Things which are similar to my solution

1. Utilises some type of recursive element, in my case which is tail recursion.
2. Reads from a file in a similar format to my structure which is using indentation to determine when a file/folder is the child of a parent folder. This is a simpler way of formatting the file structure which makes reading and creating the text file.
3. Utilises some sort of in-memory data structure which contain pointers to child nodes and to parent nodes. Allows easy traversing down different branches.

Cons

1. My solution can process multiple types of end nodes i.e folders and files. This solution only works with single element data types.
2. Very complex solution which can be difficulty to maintain and isn't appropriate to this use case.
3. Doesn't utilise a single in-memory data structure which allows traversal such as a linked list or n-ary tree. Can become unorganised and inefficient when processing larger amounts of data.

4. Doesn't use an object based allocation of data which reduces the eases that functionality can be expanded and its adaptability to other use cases.
5. Isn't an appropriate solution for this use case as having the flexibility of reading from a file makes the systems more flexible across multiple systems.
6. Overly complicated file structure by using {,} instead of a simpler but more effective method of using indentation.