Example:
Loading a Tree From a File

At each step:

1. Get the next line of the file and separate into its data, left-bit and right-bit components.
2. Build a new node for the line and insert the data into it.
3. If the right-bit is 1 pop the stack to get the node's right child.
4. If the left-bit is 1 pop the stack to get the node's left child.
5. Push the node onto the stack

When you reach the end of the file there should be 1 item on the stack --- the entire tree.

Here is an example.

Example: Consider this file

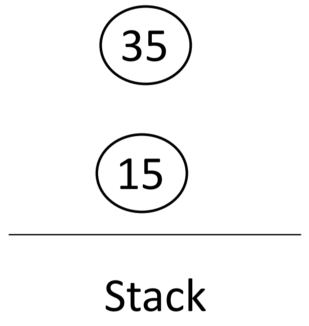
```
15   0  0
35   0  0
40   1  0
30   1  1
70   0  0
150 0  0
100 1  1
50   1   1
```

The first two steps are easy. We read data 15 with both bits 0, so we have no children. We push a single node 15 onto the stack.

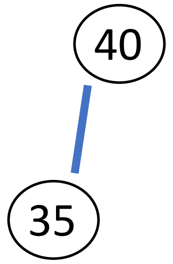
Then we read data 35 0 0 and we push a node with just 35 onto the stack.

~~15~~	~~0~~	~~0~~
~~35~~	~~0~~	~~0~~
40	1	0
30	1	1
70	0	0
150	0	0
100	1	1
50	1	1

Here is the stack after we read the first two lines:



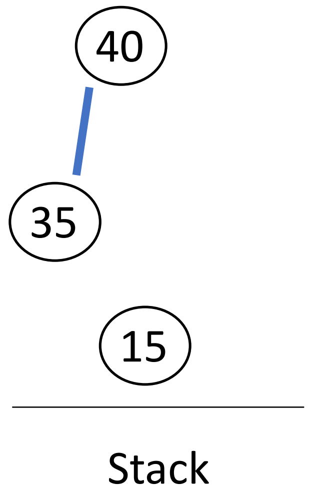
For the next line of the file we get data of 40 and our left-bit is 1. That means we pop the 35 node off the stack and use it as 40's left child. That makes



and we push this onto the stack.

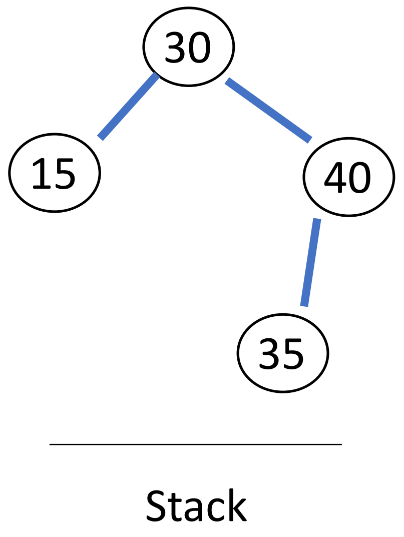
~~15~~	~~0~~	~~0~~
~~35~~	~~0~~	~~0~~
~~40~~	~~1~~	~~0~~
30	1	1
70	0	0
150	0	0
100	1	1
50	1	1

Here is the stack after the first three lines. Note that it contains to separate trees:



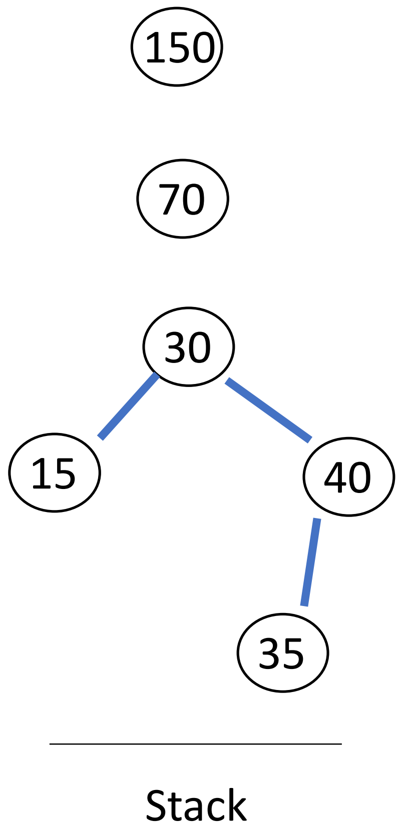
The next step is important; if you understand that the rest of the algorithm is easy. Note that we have two 1-bits with the 30, so it has both children. We pop the stack into the *right* child first, and then the left child. We push this onto the now empty stack.

~~15~~	~~0~~	~~0~~
~~35~~	~~0~~	~~0~~
~~40~~	~~1~~	~~0~~
~~30~~	~~1~~	~~1~~
70	0	0
150	0	0
100	1	1
50	1	1



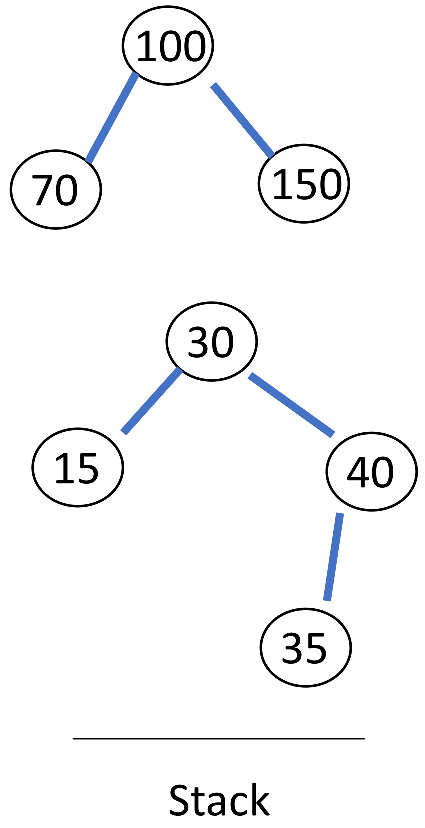
The next two steps make single nodes holding 70 and 150 and push them onto the stack:

~~15~~	~~0~~	~~0~~
~~35~~	~~0~~	~~0~~
~~40~~	~~1~~	~~0~~
~~30~~	~~1~~	~~1~~
~~70~~	~~0~~	~~0~~
~~150~~	~~0~~	~~0~~
100	1	1
50	1	1



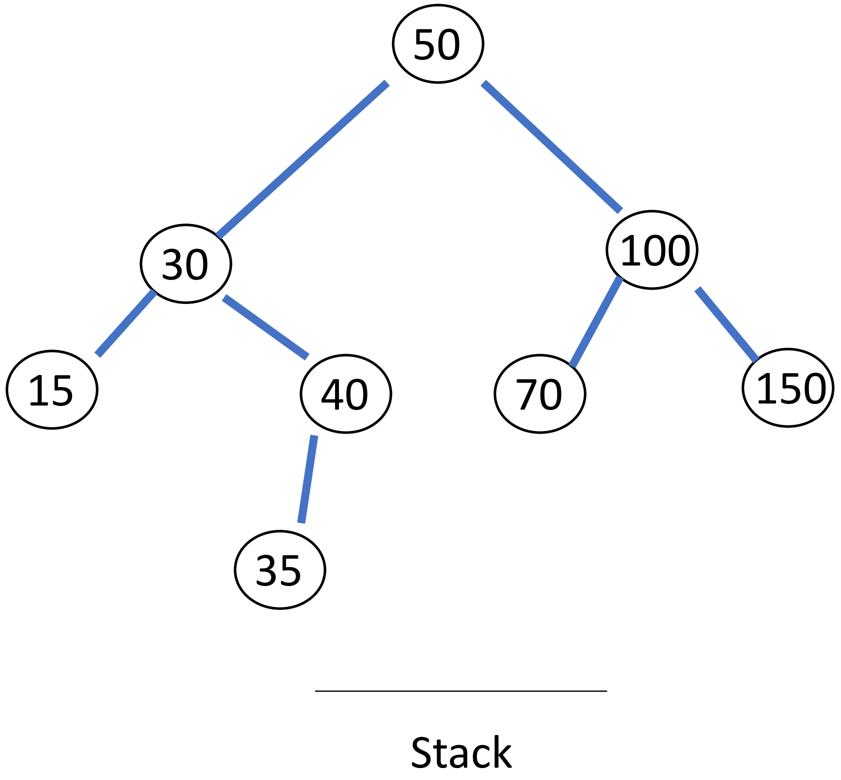
The next line makes a node with data 100; the 150 node is its right child and the 70 node its left.

~~15~~	~~0~~	~~0~~
~~35~~	~~0~~	~~0~~
~~40~~	~~1~~	~~0~~
~~30~~	~~1~~	~~1~~
~~70~~	~~0~~	~~0~~
~~150~~	~~0~~	~~0~~
~~100~~	~~1~~	~~1~~
50	1	1



Finally, the last line makes a node with data 50 and two children. The tree with root 100 is popped first, so that becomes the right child

~~15~~	~~0~~	~~0~~
~~35~~	~~0~~	~~0~~
~~40~~	~~1~~	~~0~~
~~30~~	~~1~~	~~1~~
~~70~~	~~0~~	~~0~~
~~150~~	~~0~~	~~0~~
~~100~~	~~1~~	~~1~~
~~50~~	~~1~~	~~1~~



We have read the entire file. Pop this tree from the stack. Since the stack is now empty our algorithm was successful and we can return this tree as the result.