Finding the best order of insertion

=====
1 item(s) in stack: [[1, 2, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]]
Processing 15 items: [1, 2, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
Added 7. Split [1, 2, 2, 3, 4, 5, 6] & [8, 9, 10, 11, 12, 13, 14]. Stack size 2
Ordered list: [7]
=====
2 item(s) in stack: [[1, 2, 2, 3, 4, 5, 6], [8, 9, 10, 11, 12, 13, 14]]
Processing 7 items: [1, 2, 2, 3, 4, 5, 6]
Added 3. Split [1, 2, 2] & [4, 5, 6]. Stack size 3
Ordered list: [7, 3]
=====
3 item(s) in stack: [[8, 9, 10, 11, 12, 13, 14], [1, 2, 2], [4, 5, 6]]
Processing 7 items: [8, 9, 10, 11, 12, 13, 14]
Added 11. Split [8, 9, 10] & [12, 13, 14]. Stack size 4
Ordered list: [7, 3, 11]
=====
4 item(s) in stack: [[1, 2, 2], [4, 5, 6], [8, 9, 10], [12, 13, 14]]
Processing 3 items: [1, 2, 2]
Added 2. Split [1] & [2]. Stack size 5
Ordered list: [7, 3, 11, 2]
=====
5 item(s) in stack: [[4, 5, 6], [8, 9, 10], [12, 13, 14], [1], [2]]
Processing 3 items: [4, 5, 6]
Added 5. Split [4] & [6]. Stack size 6
Ordered list: [7, 3, 11, 2, 5]
=====
6 item(s) in stack: [[8, 9, 10], [12, 13, 14], [1], [2], [4], [6]]
Processing 3 items: [8, 9, 10]
Added 9. Split [8] & [10]. Stack size 7
Ordered list: [7, 3, 11, 2, 5, 9]
=====
7 item(s) in stack: [[12, 13, 14], [1], [2], [4], [6], [8], [10]]
Processing 3 items: [12, 13, 14]
Added 13. Split [12] & [14]. Stack size 8
Ordered list: [7, 3, 11, 2, 5, 9, 13]
=====
8 item(s) in stack: [[1], [2], [4], [6], [8], [10], [12], [14]]
Processing 1 items: [1]
Added 1. No more splitting. Stack size 7
Ordered list: [7, 3, 11, 2, 5, 9, 13, 1]
=====
7 item(s) in stack: [[2], [4], [6], [8], [10], [12], [14]]
Processing 1 items: [2]
Added 2. No more splitting. Stack size 6
Ordered list: [7, 3, 11, 2, 5, 9, 13, 1, 2]
=====
6 item(s) in stack: [[4], [6], [8], [10], [12], [14]]
Processing 1 items: [4]
Added 4. No more splitting. Stack size 5
Ordered list: [7, 3, 11, 2, 5, 9, 13, 1, 2, 4]
=====

5 item(s) in stack: [[6], [8], [10], [12], [14]]
Processing 1 items: [6]
Added 6. No more splitting. Stack size 4
Ordered list: [7, 3, 11, 2, 5, 9, 13, 1, 2, 4, 6]
=====
4 item(s) in stack: [[8], [10], [12], [14]]
Processing 1 items: [8]
Added 8. No more splitting. Stack size 3
Ordered list: [7, 3, 11, 2, 5, 9, 13, 1, 2, 4, 6, 8]
=====
3 item(s) in stack: [[10], [12], [14]]
Processing 1 items: [10]
Added 10. No more splitting. Stack size 2
Ordered list: [7, 3, 11, 2, 5, 9, 13, 1, 2, 4, 6, 8, 10]
=====
2 item(s) in stack: [[12], [14]]
Processing 1 items: [12]
Added 12. No more splitting. Stack size 1
Ordered list: [7, 3, 11, 2, 5, 9, 13, 1, 2, 4, 6, 8, 10, 12]
=====
1 item(s) in stack: [[14]]
Processing 1 items: [14]
Added 14. No more splitting. Stack size 0
Ordered list: [7, 3, 11, 2, 5, 9, 13, 1, 2, 4, 6, 8, 10, 12, 14]
Values sorted from [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 2] to [7, 3, 11, 2, 5, 9, 13, 1, 2, 4, 6, 8, 10, 12, 14]
{0: {'value': 7}}

Creating the BST

=====
ID: 1 | Value: 3
Start search with origin node: {'value': 7}

Node 0: {'value': 7}
left_child node available
Filled left_child of ID 0 ({'value': 7, 'left_child': 1}), BST updated with id 1: {'value': 3}

=====
ID: 2 | Value: 11
Start search with origin node: {'value': 7, 'left_child': 1}

Node 0: {'value': 7, 'left_child': 1}
right_child node available
Filled right_child of ID 0 ({'value': 7, 'left_child': 1, 'right_child': 2}), BST updated with id 2: {'value': 11}

=====
ID: 3 | Value: 2
Start search with origin node: {'value': 7, 'left_child': 1, 'right_child': 2}

Node 0: {'value': 7, 'left_child': 1, 'right_child': 2}
Selected left_child for next iteration with ID: 1

Node 1: {'value': 3}

left_child node available
Filled left_child of ID 1 ({'value': 3, 'left_child': 3}), BST updated with id 3: {'value': 2}

=====
ID: 4 | Value: 5
Start search with origin node: {'value': 7, 'left_child': 1, 'right_child': 2}

Node 1: {'value': 7, 'left_child': 1, 'right_child': 2}
Selected left_child for next iteration with ID: 1

Node 1: {'value': 3, 'left_child': 3}
right_child node available
Filled right_child of ID 1 ({'value': 3, 'left_child': 3, 'right_child': 4}), BST updated with id 4: {'value': 5}

=====
ID: 5 | Value: 9
Start search with origin node: {'value': 7, 'left_child': 1, 'right_child': 2}

Node 1: {'value': 7, 'left_child': 1, 'right_child': 2}
Selected right_child for next iteration with ID: 2

Node 2: {'value': 11}
left_child node available
Filled left_child of ID 2 ({'value': 11, 'left_child': 5}), BST updated with id 5: {'value': 9}

=====
ID: 6 | Value: 13
Start search with origin node: {'value': 7, 'left_child': 1, 'right_child': 2}

Node 2: {'value': 7, 'left_child': 1, 'right_child': 2}
Selected right_child for next iteration with ID: 2

Node 2: {'value': 11, 'left_child': 5}
right_child node available
Filled right_child of ID 2 ({'value': 11, 'left_child': 5, 'right_child': 6}), BST updated with id 6: {'value': 13}

=====
ID: 7 | Value: 1
Start search with origin node: {'value': 7, 'left_child': 1, 'right_child': 2}

Node 2: {'value': 7, 'left_child': 1, 'right_child': 2}
Selected left_child for next iteration with ID: 1

Node 1: {'value': 3, 'left_child': 3, 'right_child': 4}
Selected left_child for next iteration with ID: 3

Node 3: {'value': 2}
left_child node available
Filled left_child of ID 3 ({'value': 2, 'left_child': 7}), BST updated with id 7: {'value': 1}

=====
ID: 8 | Value: 2
Start search with origin node: {'value': 7, 'left_child': 1, 'right_child': 2}

```
Node 3: {'value': 7, 'left_child': 1, 'right_child': 2}
Selected left_child for next iteration with ID: 1
-----
Node 1: {'value': 3, 'left_child': 3, 'right_child': 4}
Selected left_child for next iteration with ID: 3
-----
Node 3: {'value': 2, 'left_child': 7}
Selected left_child for next iteration with ID: 7
-----
Node 7: {'value': 1}
right_child node available
Filled right_child of ID 7 ({'value': 1, 'right_child': 8}), BST updated with id 8: {'value': 2}

=====
ID: 9 | Value: 4
Start search with origin node: {'value': 7, 'left_child': 1, 'right_child': 2}
-----
Node 7: {'value': 7, 'left_child': 1, 'right_child': 2}
Selected left_child for next iteration with ID: 1
-----
Node 1: {'value': 3, 'left_child': 3, 'right_child': 4}
Selected right_child for next iteration with ID: 4
-----
Node 4: {'value': 5}
left_child node available
Filled left_child of ID 4 ({'value': 5, 'left_child': 9}), BST updated with id 9: {'value': 4}

=====
ID: 10 | Value: 6
Start search with origin node: {'value': 7, 'left_child': 1, 'right_child': 2}
-----
Node 4: {'value': 7, 'left_child': 1, 'right_child': 2}
Selected left_child for next iteration with ID: 1
-----
Node 1: {'value': 3, 'left_child': 3, 'right_child': 4}
Selected right_child for next iteration with ID: 4
-----
Node 4: {'value': 5, 'left_child': 9}
right_child node available
Filled right_child of ID 4 ({'value': 5, 'left_child': 9, 'right_child': 10}), BST updated with id 10: {'value':
6}

=====
ID: 11 | Value: 8
Start search with origin node: {'value': 7, 'left_child': 1, 'right_child': 2}
-----
Node 4: {'value': 7, 'left_child': 1, 'right_child': 2}
Selected right_child for next iteration with ID: 2
-----
Node 2: {'value': 11, 'left_child': 5, 'right_child': 6}
Selected left_child for next iteration with ID: 5
-----
Node 5: {'value': 9}
left_child node available
Filled left_child of ID 5 ({'value': 9, 'left_child': 11}), BST updated with id 11: {'value': 8}
```

=====
ID: 12 | Value: 10
Start search with origin node: {'value': 7, 'left_child': 1, 'right_child': 2}

Node 5: {'value': 7, 'left_child': 1, 'right_child': 2}
Selected right_child for next iteration with ID: 2

Node 2: {'value': 11, 'left_child': 5, 'right_child': 6}
Selected left_child for next iteration with ID: 5

Node 5: {'value': 9, 'left_child': 11}
right_child node available
Filled right_child of ID 5 ({'value': 9, 'left_child': 11, 'right_child': 12}), BST updated with id 12: {'value': 10}

=====
ID: 13 | Value: 12
Start search with origin node: {'value': 7, 'left_child': 1, 'right_child': 2}

Node 5: {'value': 7, 'left_child': 1, 'right_child': 2}
Selected right_child for next iteration with ID: 2

Node 2: {'value': 11, 'left_child': 5, 'right_child': 6}
Selected right_child for next iteration with ID: 6

Node 6: {'value': 13}
left_child node available
Filled left_child of ID 6 ({'value': 13, 'left_child': 13}), BST updated with id 13: {'value': 12}

=====
ID: 14 | Value: 14
Start search with origin node: {'value': 7, 'left_child': 1, 'right_child': 2}

Node 6: {'value': 7, 'left_child': 1, 'right_child': 2}
Selected right_child for next iteration with ID: 2

Node 2: {'value': 11, 'left_child': 5, 'right_child': 6}
Selected right_child for next iteration with ID: 6

Node 6: {'value': 13, 'left_child': 13}
right_child node available
Filled right_child of ID 6 ({'value': 13, 'left_child': 13, 'right_child': 14}), BST updated with id 14: {'value': 14}

BST Output

{0: {'value': 7, 'left_child': 1, 'right_child': 2},
 1: {'value': 3, 'left_child': 3, 'right_child': 4},
 2: {'value': 11, 'left_child': 5, 'right_child': 6},
 3: {'value': 2, 'left_child': 7},
 4: {'value': 5, 'left_child': 9, 'right_child': 10},
 5: {'value': 9, 'left_child': 11, 'right_child': 12},
 6: {'value': 13, 'left_child': 13, 'right_child': 14},
 7: {'value': 1, 'right_child': 8},
 8: {'value': 2},
 9: {'value': 4},
 10: {'value': 6},
 11: {'value': 8},
 12: {'value': 10},
 13: {'value': 12},
 14: {'value': 14}}

Verifying the BST