Binary Search Trees

Binary Search Tree

A binary search tree has the following properties for **each** node, n

- n's value is greater than all values in its left subtree T_L
- n's value is less than all values in its right subtree T_R
- Both T_L and T_R are binary search trees

Clicker

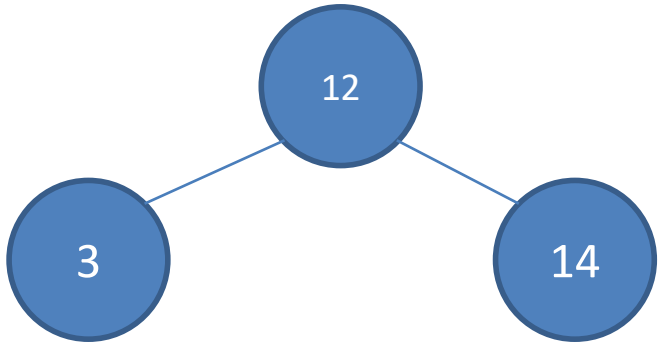
A binary search tree has the following properties for **each** node, n

- n's value is greater than all values in its left subtree T_L
- n's value is less than all values in its right subtree T_R
- Both T_L and T_R are binary search trees

- Is this a valid Binary Search Tree?

A. Yes

B. No



Clicker

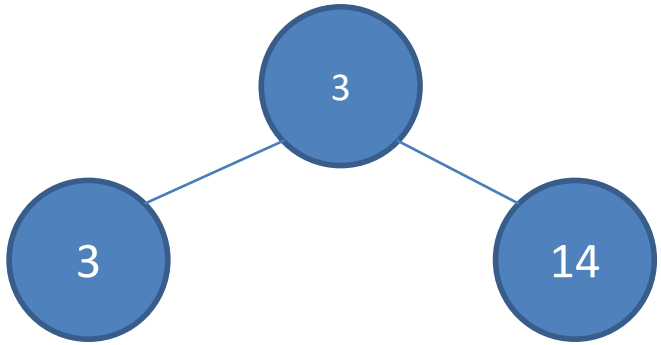
A binary search tree has the following properties for **each** node, n

- n's value is greater than all values in its left subtree T_L
- n's value is less than all values in its right subtree T_R
- Both T_L and T_R are binary search trees

- Is this a valid Binary Search Tree?

A. Yes

B. No



Clicker

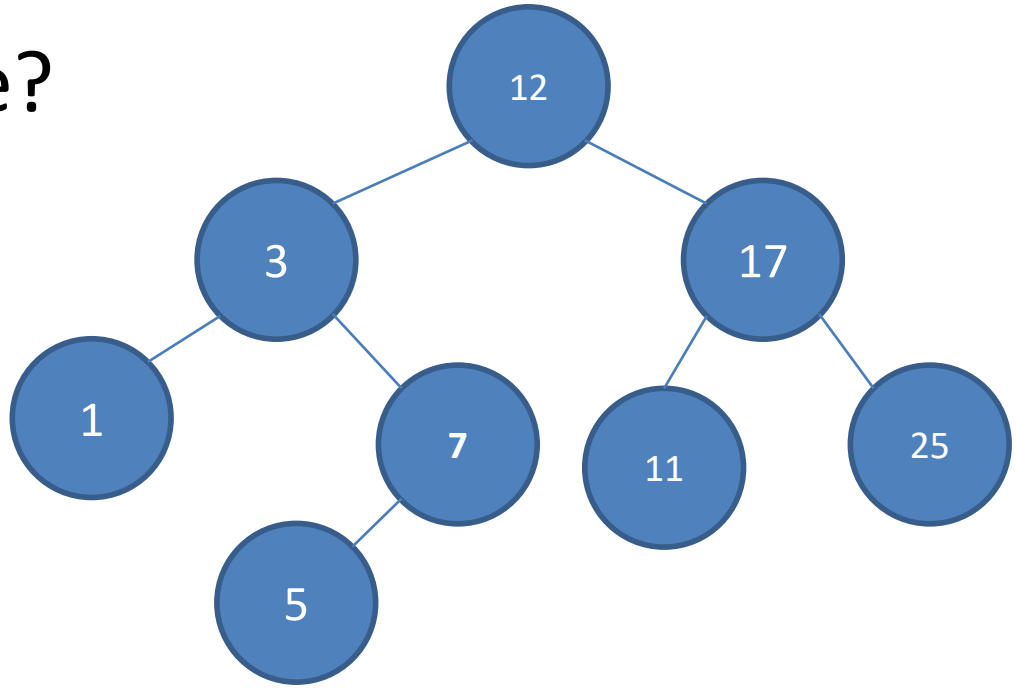
A binary search tree has the following properties for **each** node, n

- n's value is greater than all values in its left subtree T_L
- n's value is less than all values in its right subtree T_R
- Both T_L and T_R are binary search trees

- Is this a valid Binary Search Tree?

A. Yes

B. No



Clicker

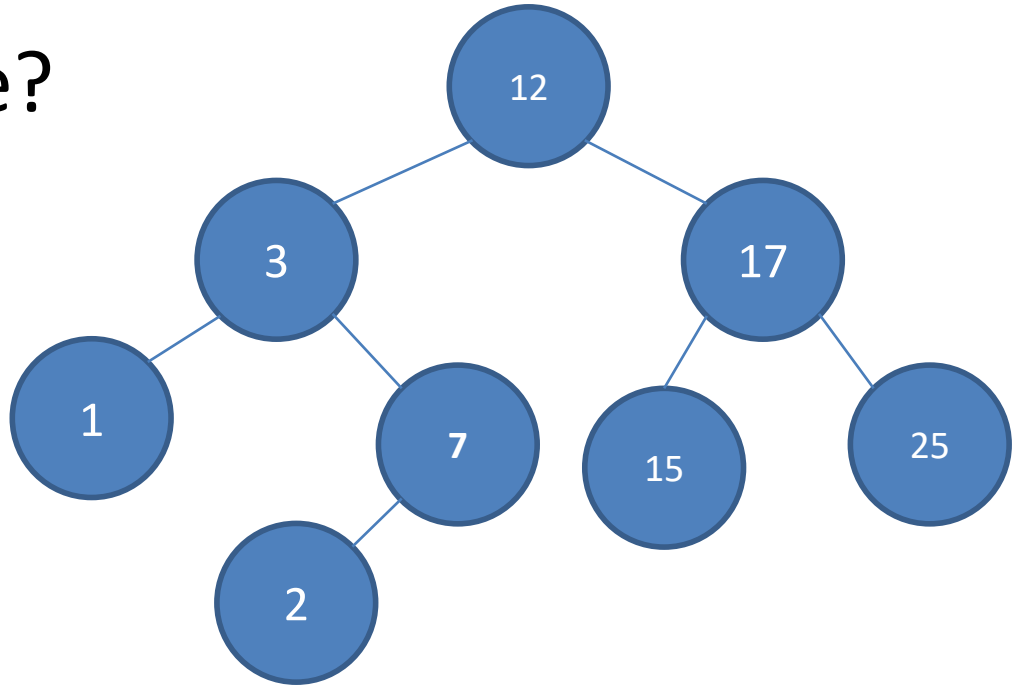
A binary search tree has the following properties for **each** node, n

- n's value is greater than all values in its left subtree T_L
- n's value is less than all values in its right subtree T_R
- Both T_L and T_R are binary search trees

- Is this a valid Binary Search Tree?

A. Yes

B. No



Clicker

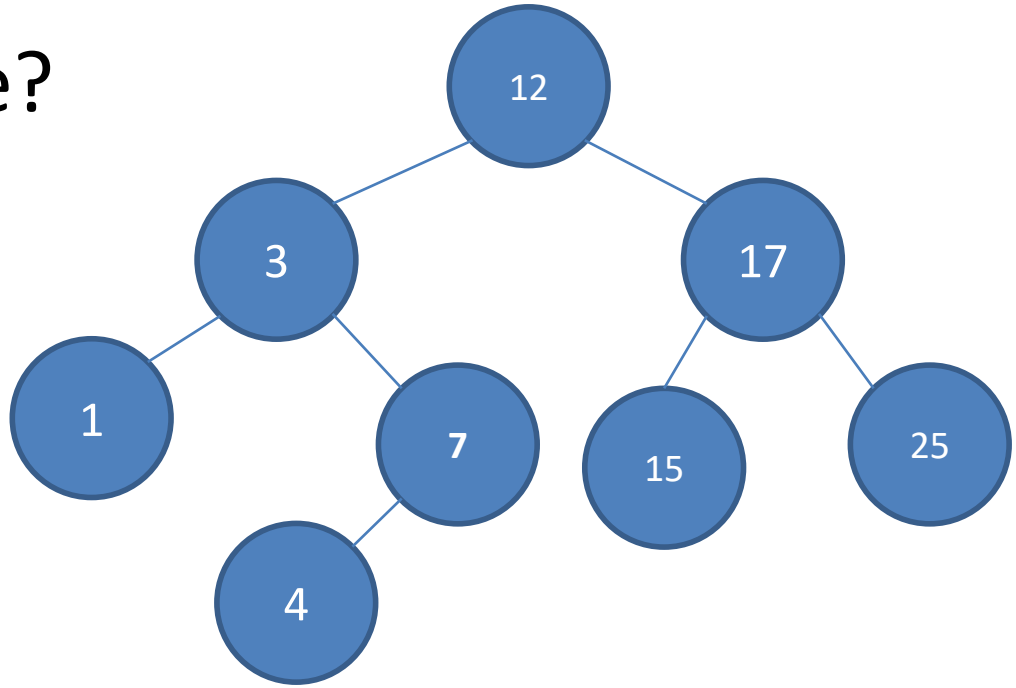
A binary search tree has the following properties for **each** node, n

- n's value is greater than all values in its left subtree T_L
- n's value is less than all values in its right subtree T_R
- Both T_L and T_R are binary search trees

- Is this a valid Binary Search Tree?

A. Yes

B. No

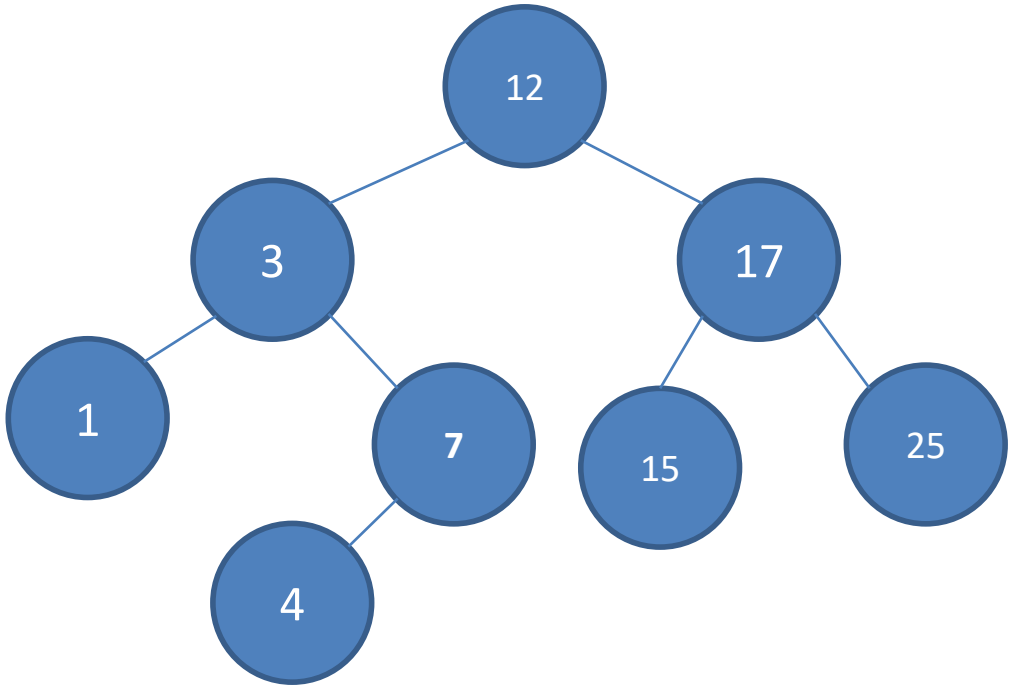


Clicker

- Is this tree balanced?

A. Yes

B. No



Clicker

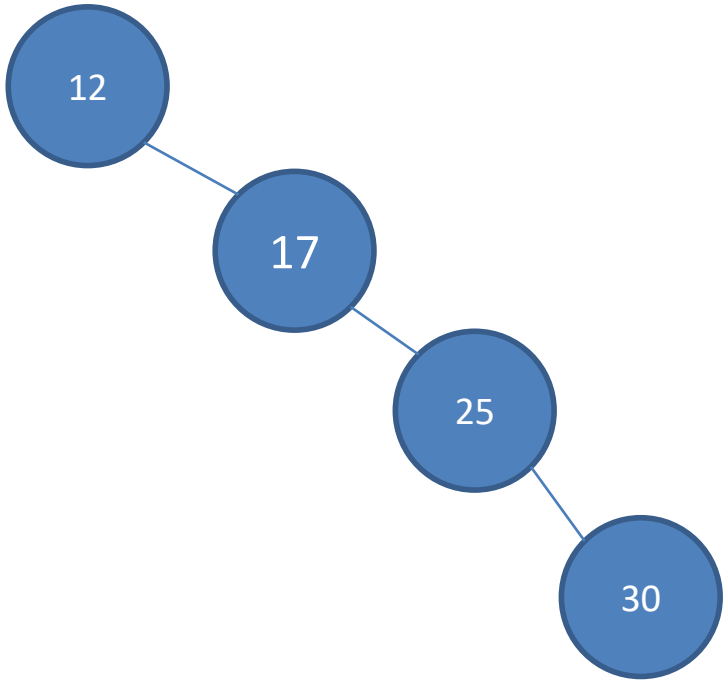
A binary search tree has the following properties for **each** node, n

- n's value is greater than all values in its left subtree T_L
- n's value is less than all values in its right subtree T_R
- Both T_L and T_R are binary search trees

- Is this an empty tree a valid Binary Search Tree?

A. Yes

B. No

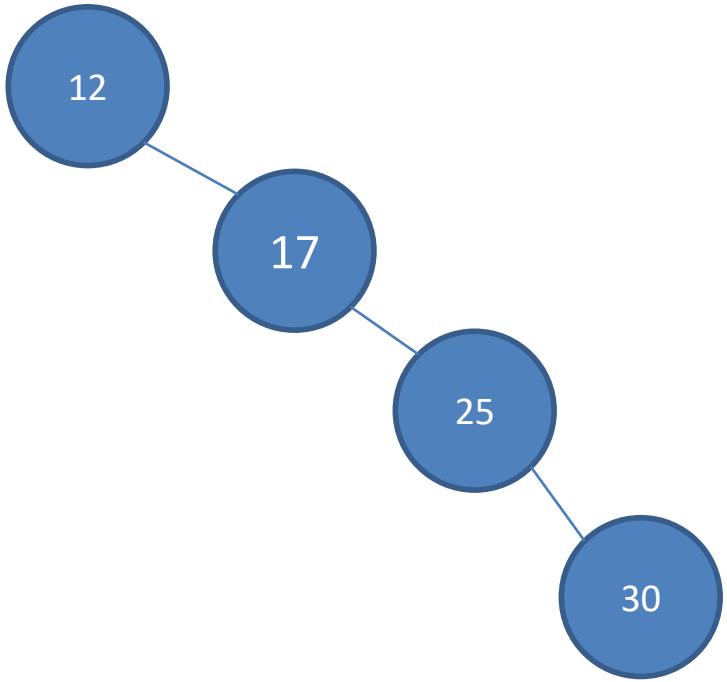


Clicker

- Is this tree balanced?

A. Yes

B. No



Clicker

A binary search tree has the following properties for **each** node, n

- n's value is greater than all values in its left subtree T_L
- n's value is less than all values in its right subtree T_R
- Both T_L and T_R are binary search trees

- Is this an empty tree a valid Binary Search Tree?

A. Yes

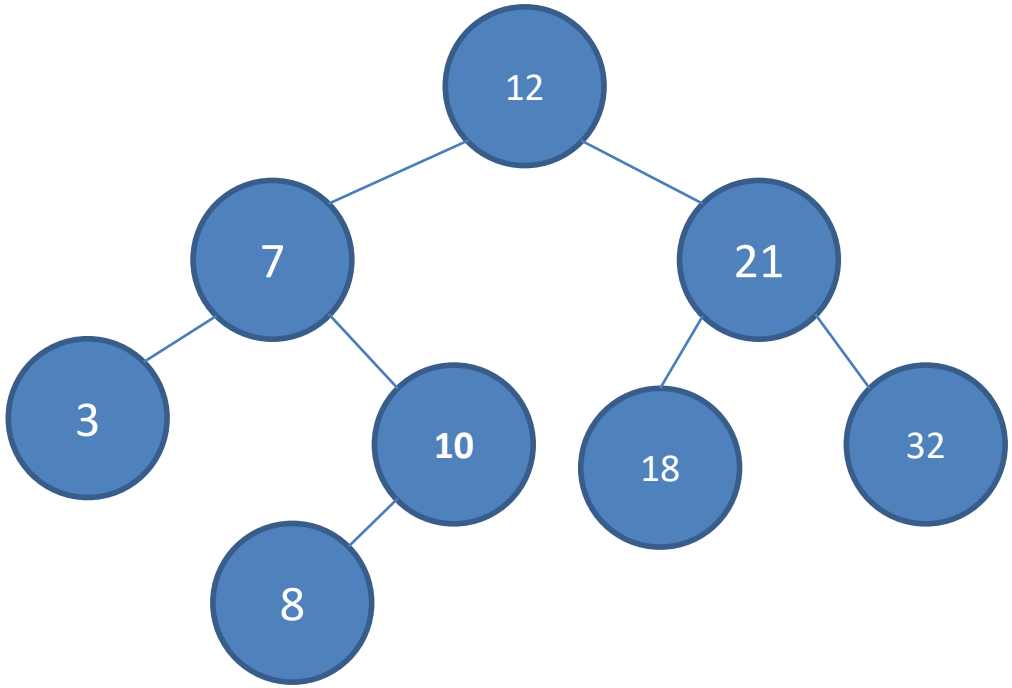
B. No

Binary Search Tree

A binary search tree has the following properties for **each** node, n

- n's value is greater than all values in its left subtree T_L
- n's value is less than all values in its right subtree T_R
- Both T_L and T_R are binary search trees

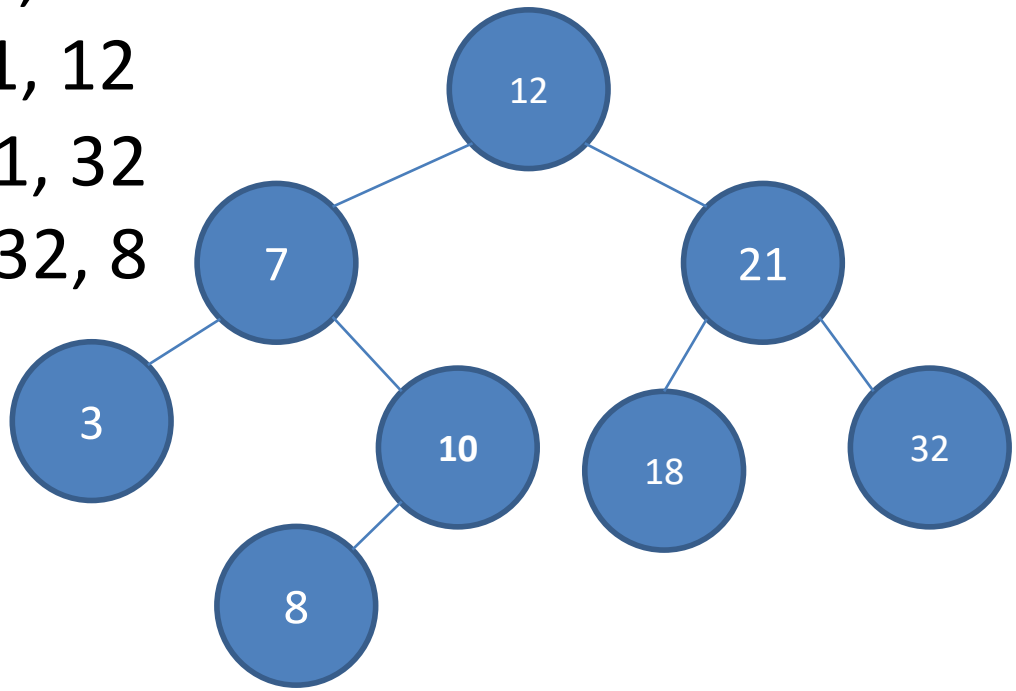
- add the following values one at a time:
  - 7
  - 21
  - 10
  - 8
  - 32
  - 18
  - 3



Binary Search Tree

Which is a preorder traversal of this tree?

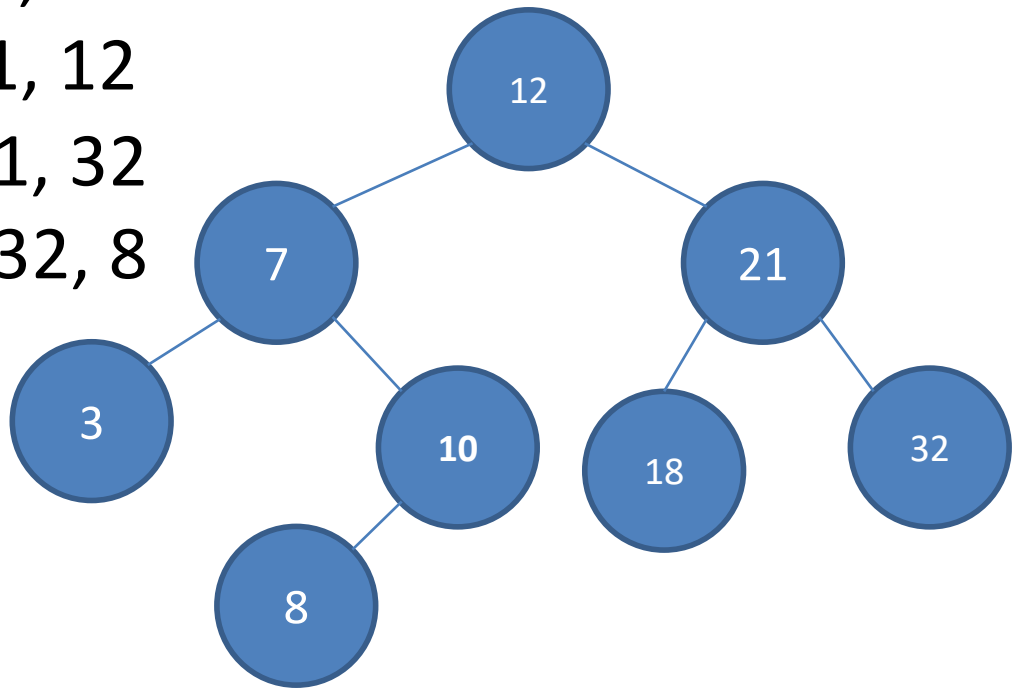
A) 12, 7, 3, 10, 8, 21, 18, 32
B) 3, 8, 10, 7, 18, 32, 21, 12
C) 3, 7, 8, 10, 12, 18, 21, 32
D) 12, 7, 21, 3, 10, 18, 32, 8
E) none of the above



Binary Search Tree

Which is a postorder traversal of this tree?

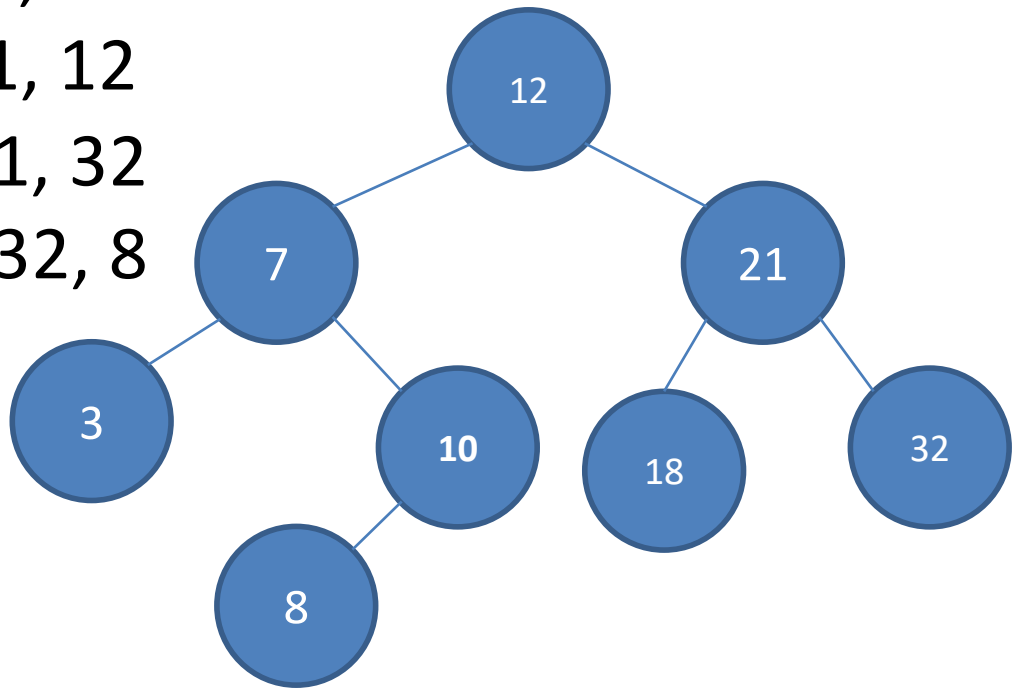
A) 12, 7, 3, 10, 8, 21, 18, 32
B) 3, 8, 10, 7, 18, 32, 21, 12
C) 3, 7, 8, 10, 12, 18, 21, 32
D) 12, 7, 21, 3, 10, 18, 32, 8
E) none of the above



Binary Search Tree

Which is an level order traversal of this tree?

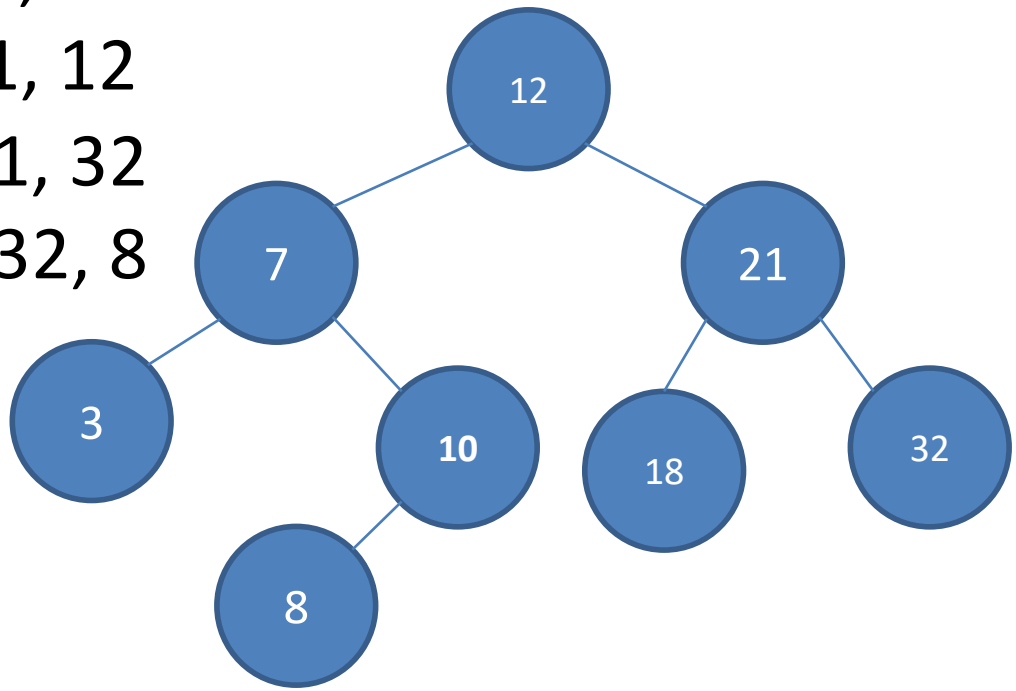
A) 12, 7, 3, 10, 8, 21, 18, 32
B) 3, 8, 10, 7, 18, 32, 21, 12
C) 3, 7, 8, 10, 12, 18, 21, 32
D) 12, 7, 21, 3, 10, 18, 32, 8
E) none of the above



Binary Search Tree

Which is an inorder traversal of this tree?

A) 12, 7, 3, 10, 8, 21, 18, 32
B) 3, 8, 10, 7, 18, 32, 21, 12
C) 3, 7, 8, 10, 12, 18, 21, 32
D) 12, 7, 21, 3, 10, 18, 32, 8
E) none of the above



Recall…

If I have an array of numbers that are NOT in sorted order…

What does the algorithm to search that array for a particular value look like? Say, I am looking for 32…

BigO of that algorithm?

0	1	2	3	4	5	6	7
21	10	8	7	12	18	3	32

Recall…

If I have an array of numbers that ARE in sorted order…

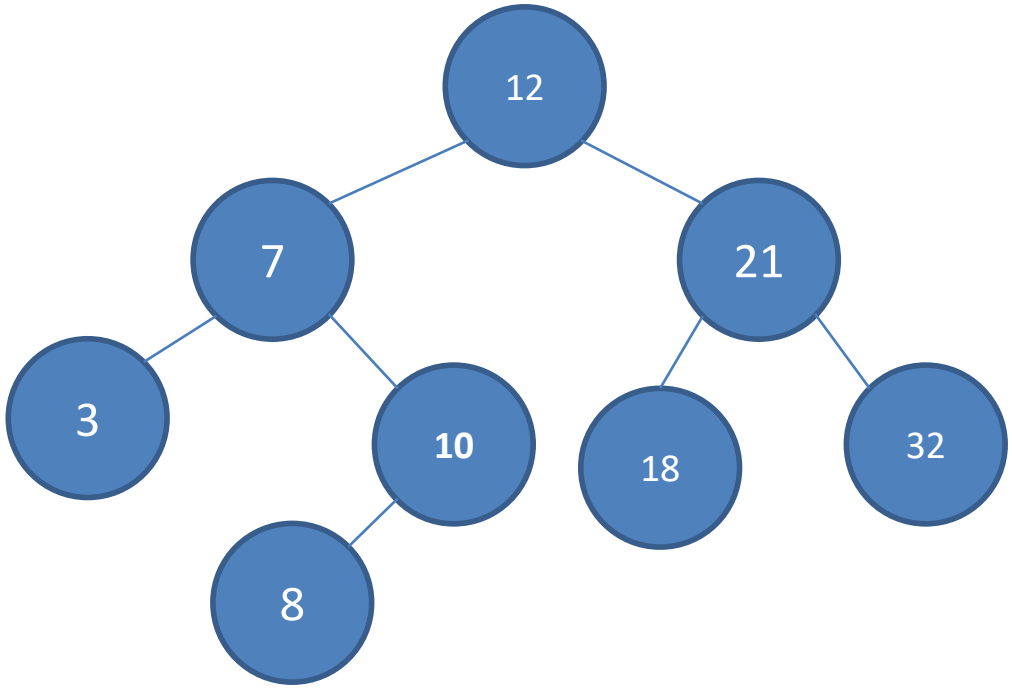
What does the algorithm to search that array for a particular value look like? Say, I am looking for 32…

BigO of that algorithm?

0	1	2	3	4	5	6	7
3	7	8	10	12	18	21	32

Binary Search Tree

What if I was searching for a value in a Binary Search Tree?
Say, I am looking for 32 – what would the algorithm look like?
How many comparisons will I have to do – worst case?



Exercise

- Write the pseudocode for a recursive function that will take a binary search tree and a number and will return true if that number is found in the tree and false otherwise

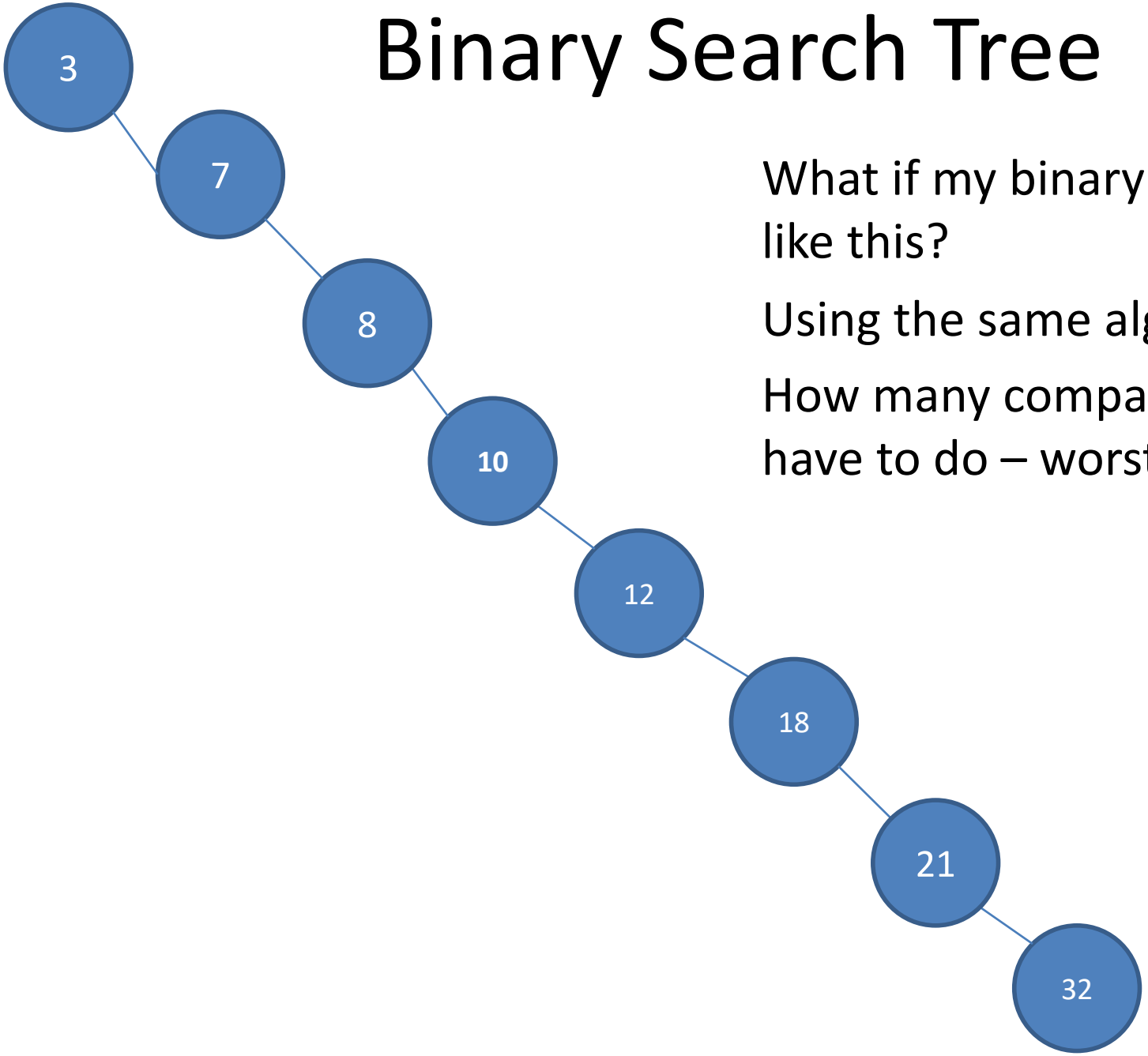
Recall our process:

- Input/output?
- Examples:
  - Simplest example of calling the function
  - A more complex example
- Edit the template
  - Rename function & data
  - Edit the basecase
  - Deal with one data piece
  - Update to smaller problem for recursive call

HINT: - if you have more than one smaller problem, you can make a recursive call on each of them

```
function(data)
    if (smallestPossibleProblem? data)
        the simple answer
        return ...
    else
        first part of data ...
        function(smallerProblem(data))
        return ...
```

Binary Search Tree



What if my binary tree looked like this?

Using the same algorithm

How many comparisons will I have to do – worst case?

Exercise

- Write the pseudocode for a recursive function that will take a tree and will return true if the tree is full and false otherwise

Recall our process:

- Input/output?
- Examples:
  - Simplest example of calling the function
  - A more complex example
- Edit the template
  - Rename function & data
  - Edit the basecase
  - Deal with one data piece
  - Update to smaller problem for recursive call

HINT: - if you have more than one smaller problem, you can make a recursive call on each of them

```
function(data)
    if (smallestPossibleProblem? data)
        the simple answer
        return ...
    else
        first part of data ...
        function(smallerProblem(data))
        return ...
```

Exercise

- Write the pseudocode for a recursive function that will take a tree and will return true if the tree is balanced and false otherwise
- You may call the height function we wrote previously that takes a tree and returns its height

Recall our process:

- Input/output?
- Examples:
  - Simplest example of calling the function
  - A more complex example
- Edit the template
  - Rename function & data
  - Edit the basecase
  - Deal with one data piece
  - Update to smaller problem for recursive call

HINT: - if you have more than one smaller problem, you can make a recursive call on each of them

```
function(data)
    if (smallestPossibleProblem? data)
        the simple answer
        return ...
    else
        first part of data ...
        function(smallerProblem(data))
        return ...
```