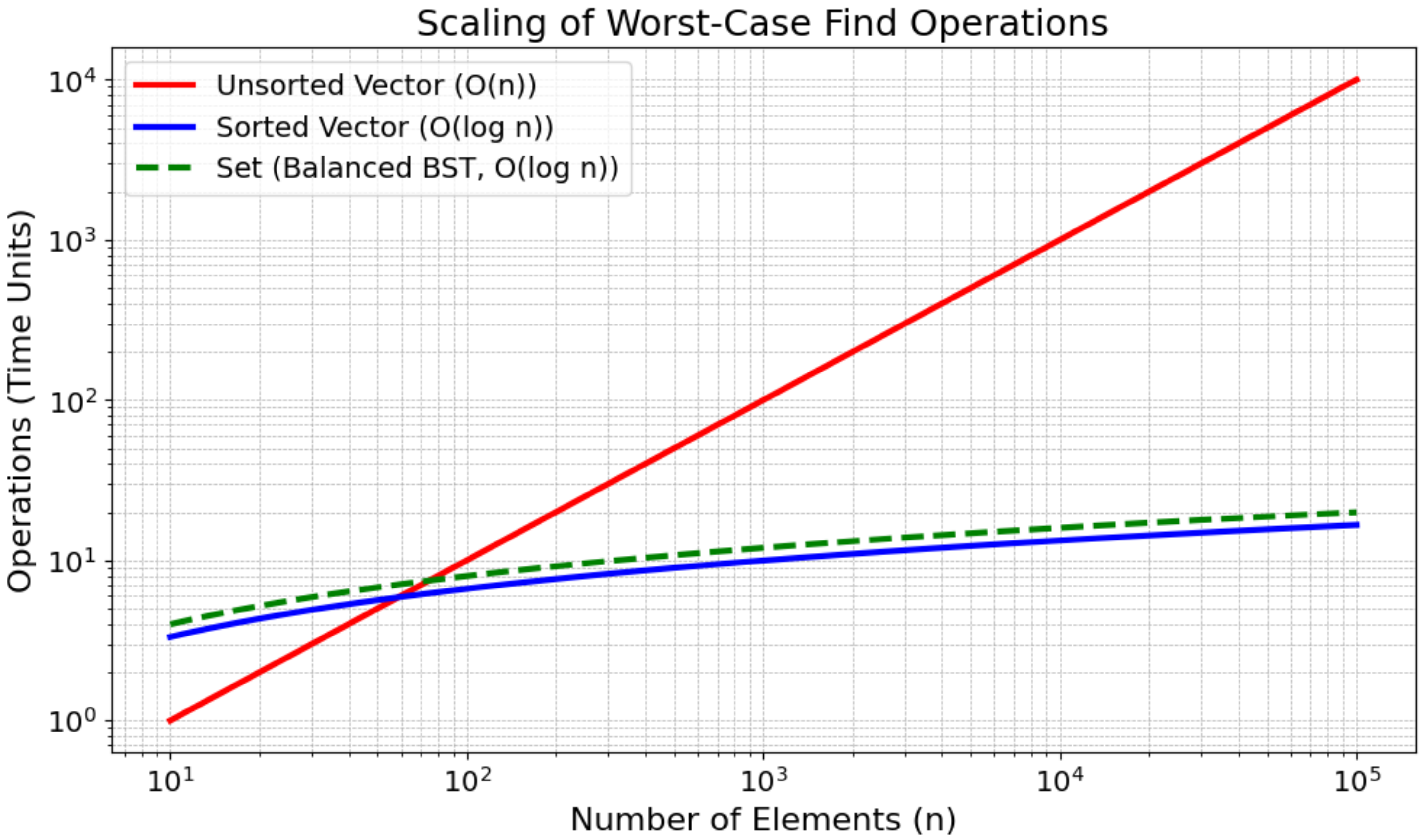
BINARY SEARCH TREES

Problem Solving with Computers-II



Scaling of Worst-Case Find Operations
Unsorted Vector (O(n))
Sorted Vector (O(log n))
Set (Balanced BST, O(log n))
Operations (Time Units)
Number of Elements (n)
10^0
10^1
10^2
10^3
10^4
10^1
10^2
10^3
10^4
10^5

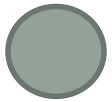
Binary Search Trees (std::set)

- What are the operations supported?
- What are the running times of these operations?
- How do you implement the BST i.e. operations supported by it?

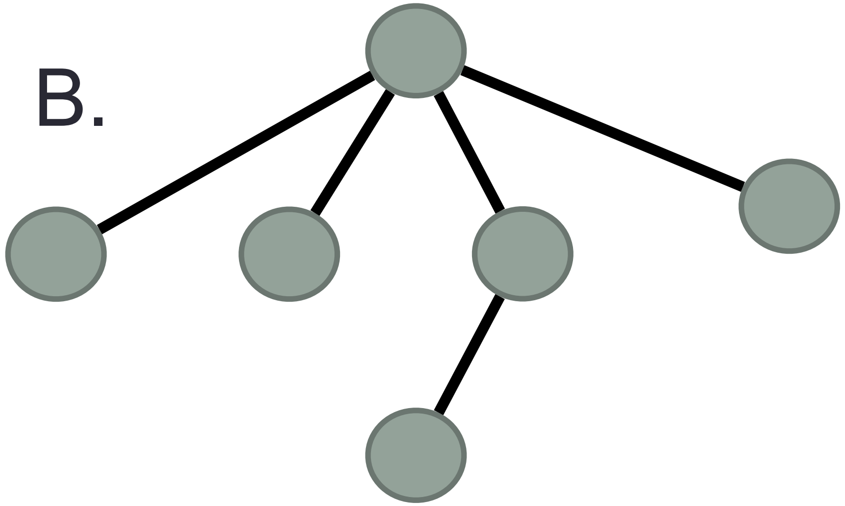
https://cplusplus.com/reference/set/set/?kw=set

Which of the following is/are a tree?

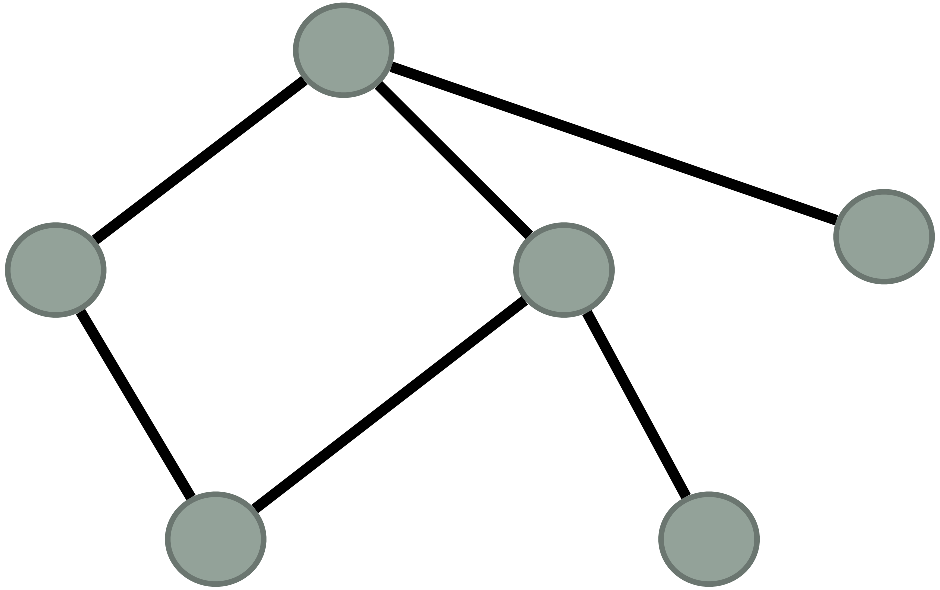
A.



B.



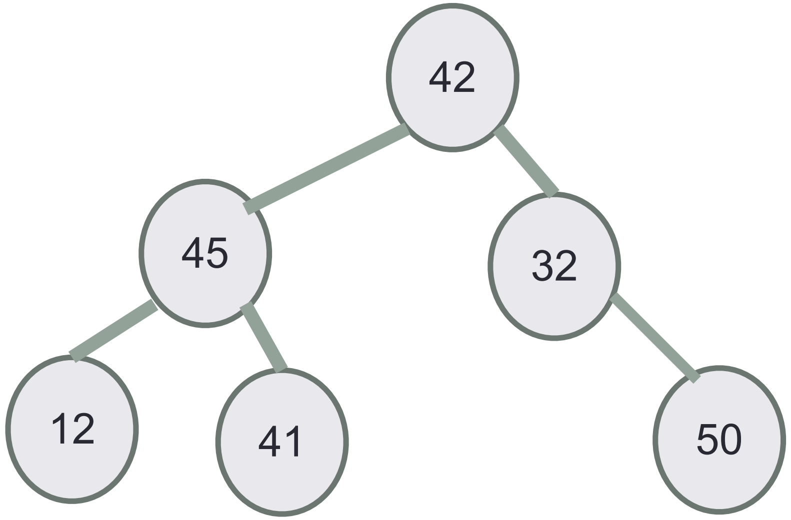
C.



D. A & B

E. All of A-C

Binary Trees



In a tree, nodes are arranged in a hierarchy

- One node is distinguished as the root
- Each node:
  - stores a key
  - has a pointer to child nodes and parent (optional)
- Unique path between any two nodes
- Leaf nodes have no children

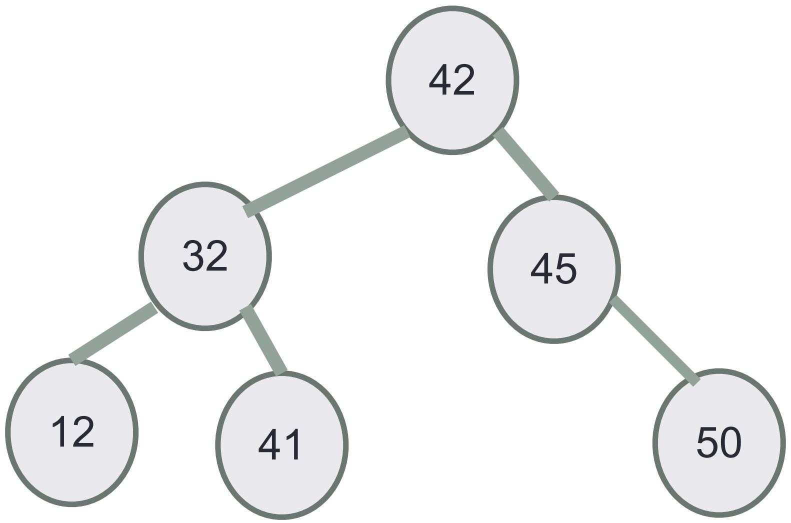
In a binary tree, each node has at most ____________ children

```
struct TreeNode {

  TreeNode* left;
  TreeNode* right;
  TreeNode* parent;
  int const data;

  TreeNode(int d) : data(d) {
    left = right = parent = nullptr;
  }
};
```

Binary Search Tree – What is it?



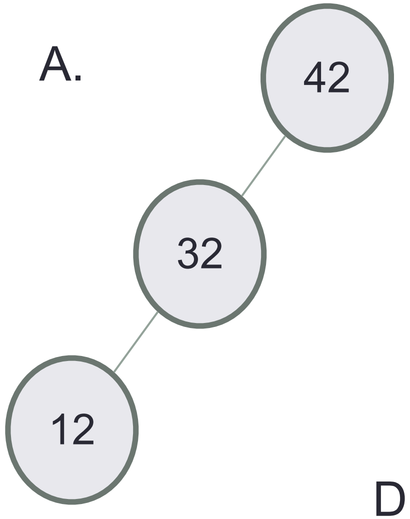
BST is a binary tree where each node satisfies the Search Tree Property

For any node,
Keys in node’s left subtree < Node’s key < Keys in node’s right subtree

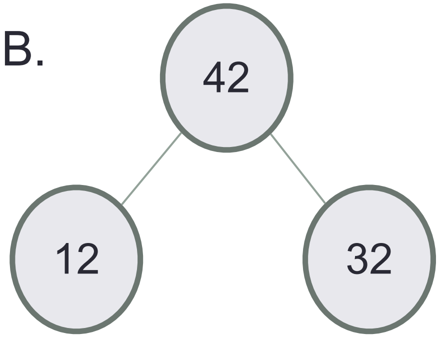
std::set does not store duplicate values
Do the keys have to be integers?

Which of the following is/are a binary search tree?

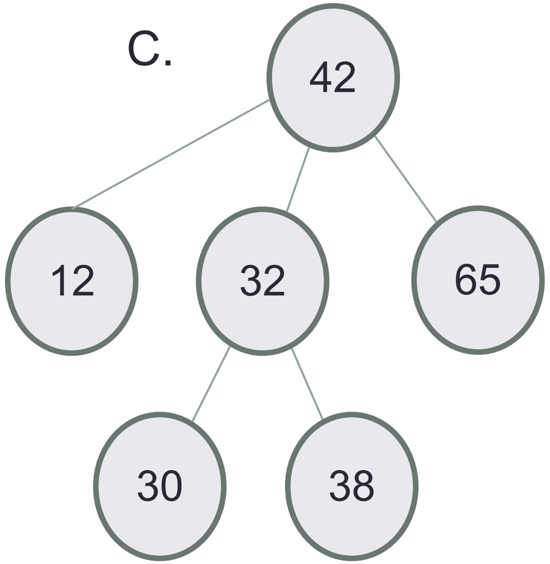
A.



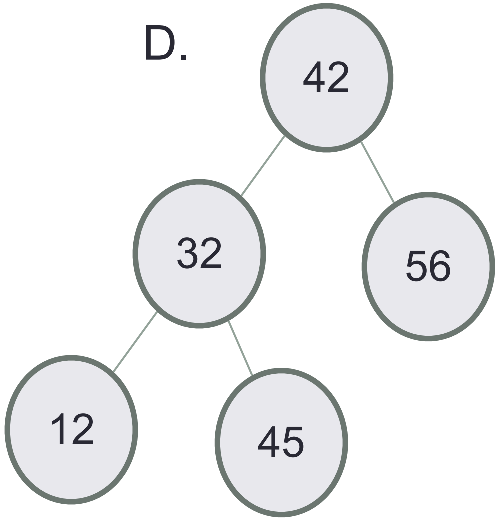
B.



C.



D.

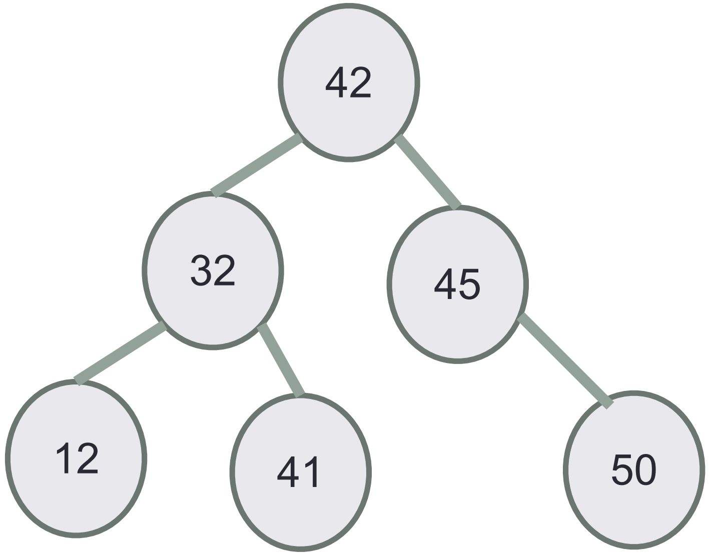


E. More than one of these

- Path – a sequence of (zero or more) connected nodes.
- Length of a path - number of edges traversed on the path
- Height of node – Length of the longest path from the node to a leaf node.
- **Height of the tree** - Length of the longest path from the **root** to a leaf node.

BSTs of different heights are possible with the same set of keys
Examples for keys: 12, 32, 41, 42, 45

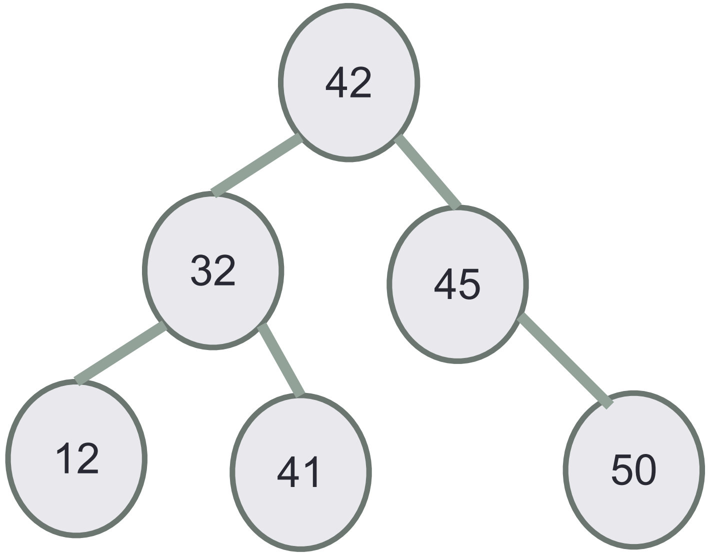
search in a BST



- Start at the root;
- Trace down a path by comparing **k** with the key of the current node x:
  - If the keys are equal: we have found the key
  - If **k** < key[x] search in the left subtree of x
  - If **k** > key[x] search in the right subtree of x
- What is the running time of search?

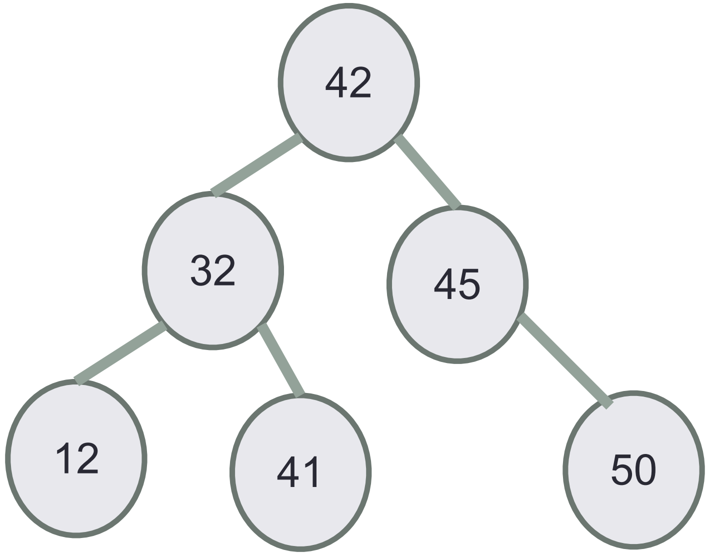
Search for 41, then search for 53

Insert



- Insert 40
- Search for the key
- Insert at the spot you expected to find it
- What is the running time of insert?

Min/Max



Which of the following describes the algorithm to find the maximum value in the BST?

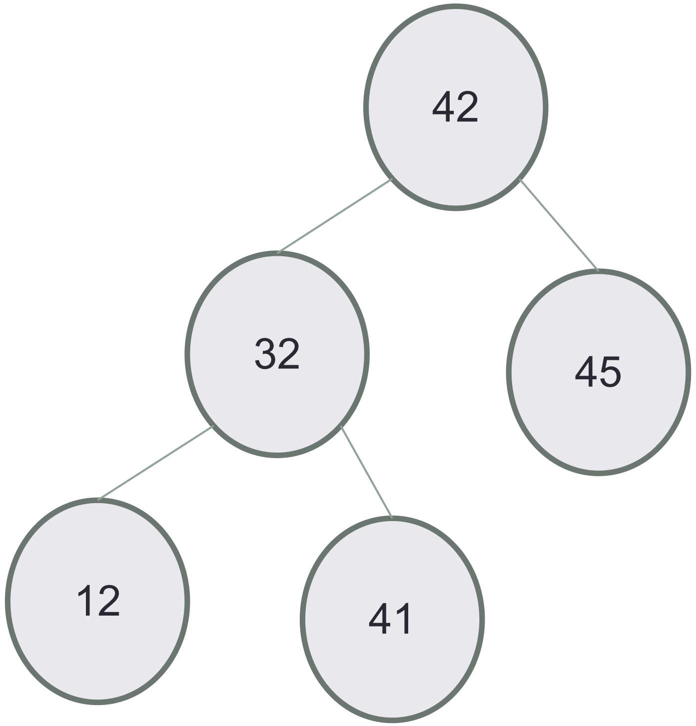
A. Return the root node's value

B. Follow right child pointers from the root, until a node with no right child is encountered, return that node's key

C. Follow left child pointers from the root, until a node with no left child is encountered, return that node's key

Define the BST ADT

Operations
Search
Insert
Min
Max
Successor (next largest key)
Predecessor (next smaller key)
Delete
Print elements (3 variations)

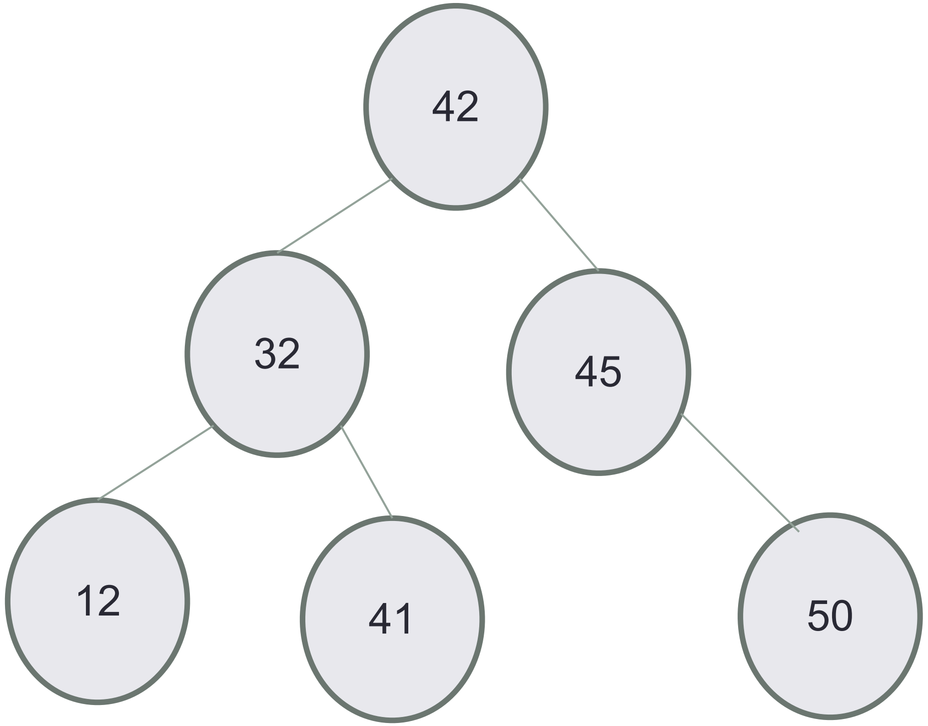


Traversing down the tree

- Suppose n is a pointer to the root. What is the output of the following code:

```
n = n->left;
n = n->right;
cout<<n->data<<endl;
```

A. 42
B. 32
C. 12
D. 41
E. Segfault



Quiz Time!

- This is a closed book, closed notes quiz
- No calculators, phones, or notes are allowed
- Write all answers clearly in pen or dark pencil
- Show your work for partial credit
- Take the rest of class time.
- You can leave when you are done