THE UNIVERSITY OF VERMONT
COLLEGE OF ENGINEERING & MATHEMATICAL SCIENCES

Binary Search Tree (BST)

CS 124 / Department of Computer Science

What is a binary search tree?

A *binary search tree* is a rooted, *binary tree* that is *sorted.*

A *binary tree* is a tree in which every node has either 0, 1 or 2 children.

Given an interior node in the tree, the node's left subtree contains only values less than that of the node, and the node's right subtree contains only values greater than that of the node.

What is a binary search tree?

In a binary search tree:

- any values that appear in the tree appear exactly once (no duplicate values), and

- in-order traversal yields a sorted list of all values in the tree.

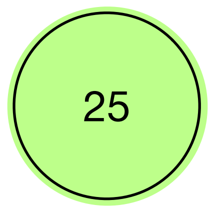
Constructing a binary search tree

Let's pick some numbers at random:

25, 21, 9, 17, 13, 36, 18, 2, 42, 28, 12, 45

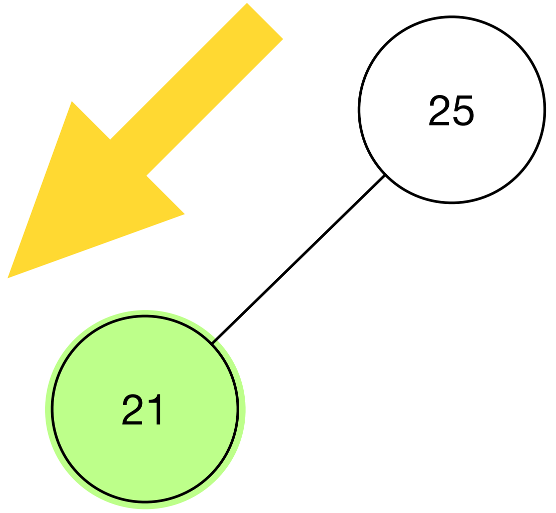
...and construct a tree

Constructing a binary search tree



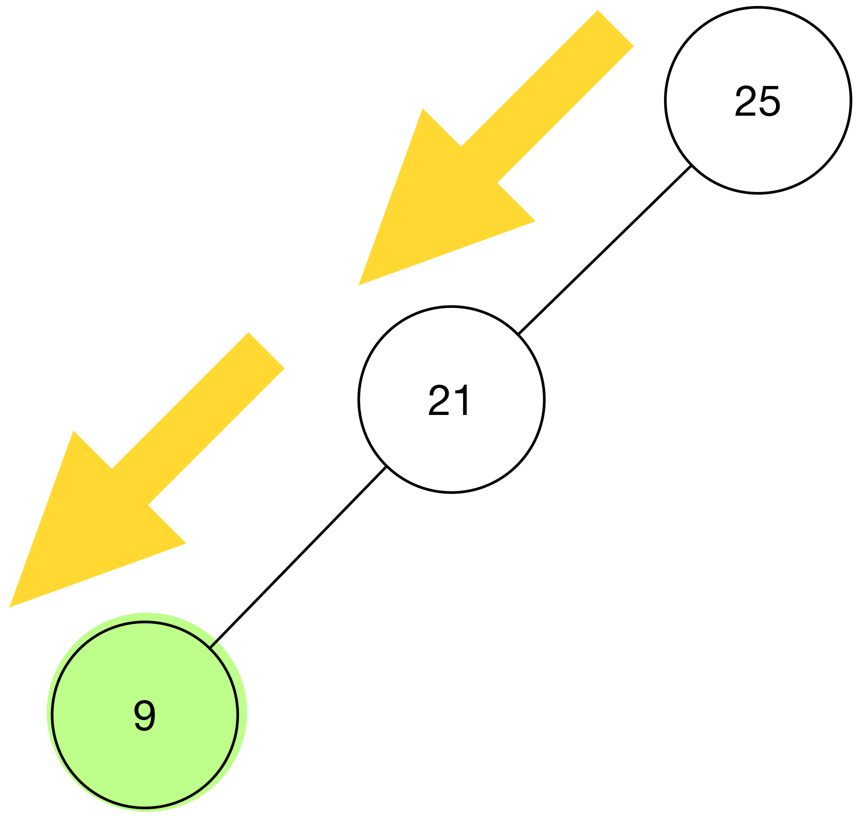
25

Constructing a binary search tree



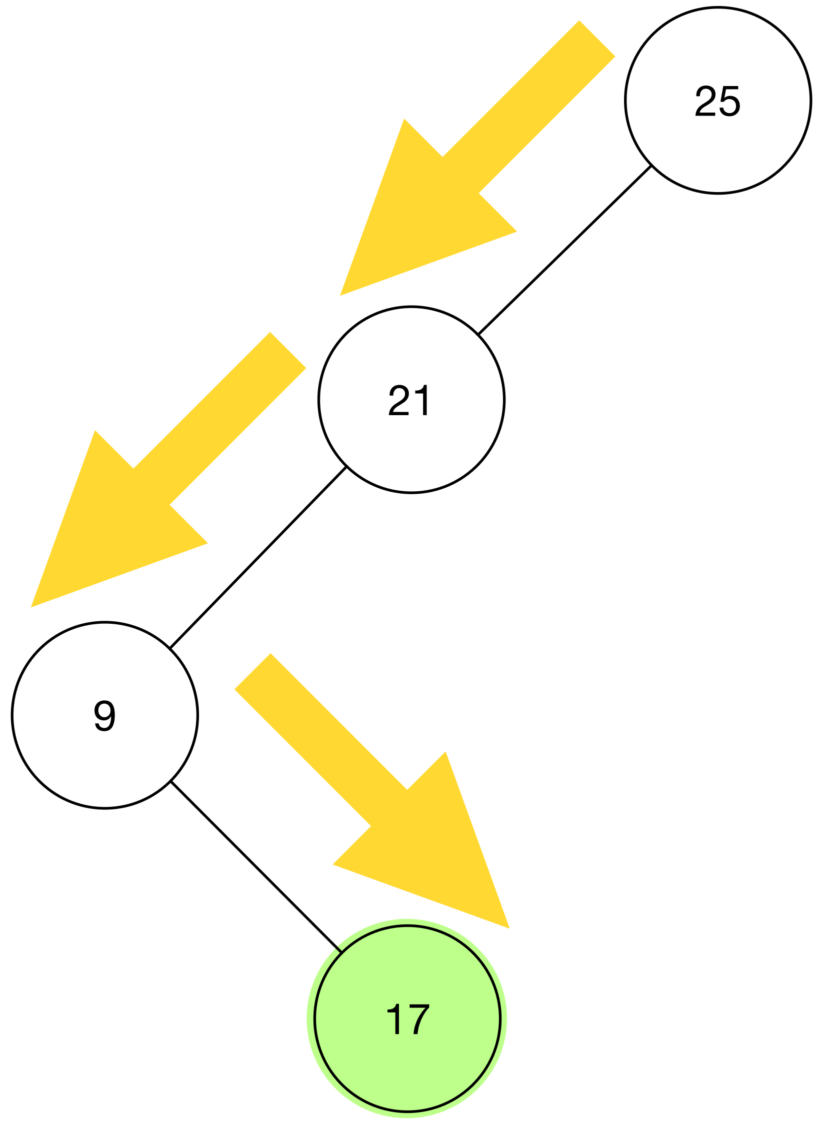
25, 21

Constructing a binary search tree



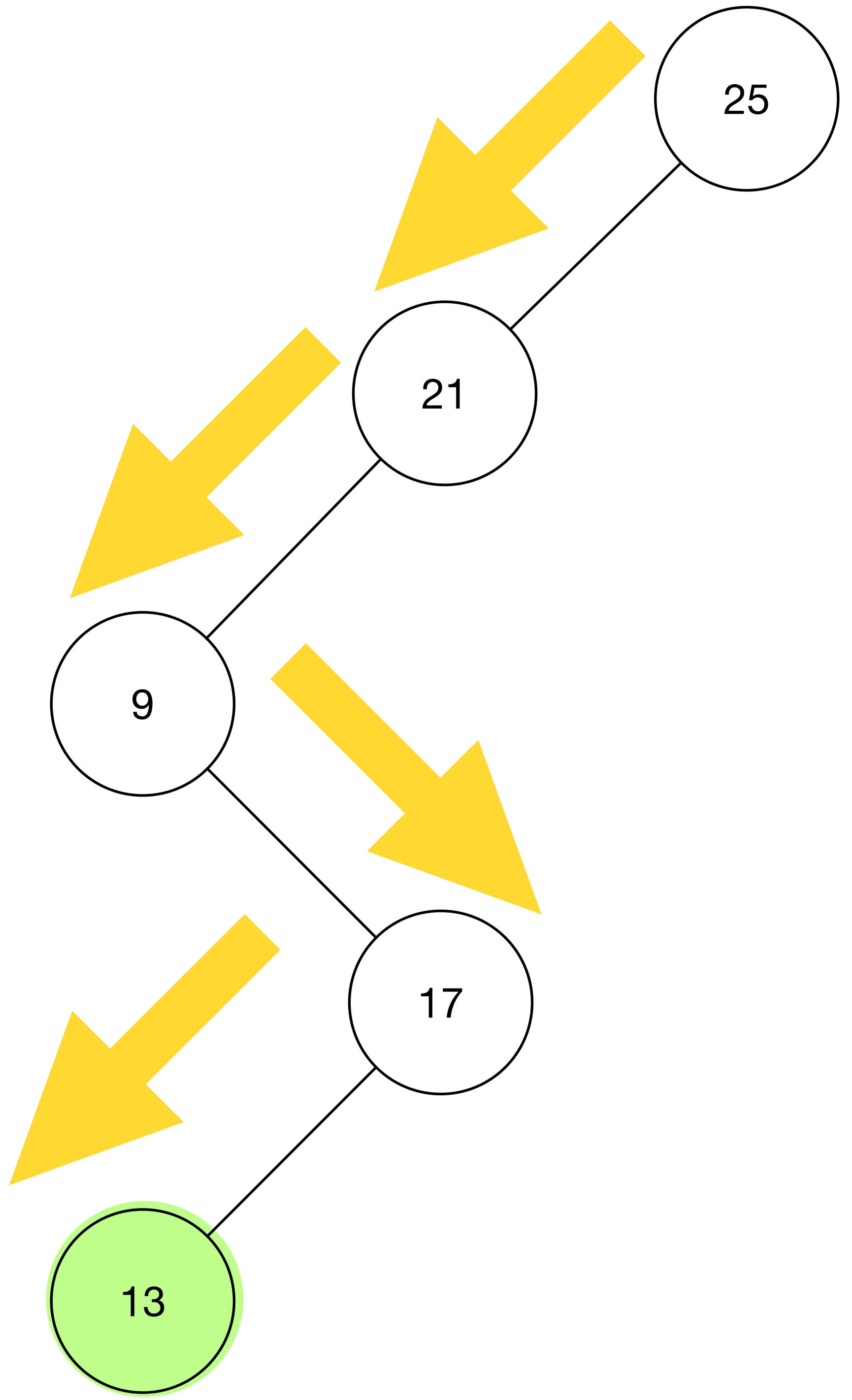
25, 21, 9

Constructing a binary search tree



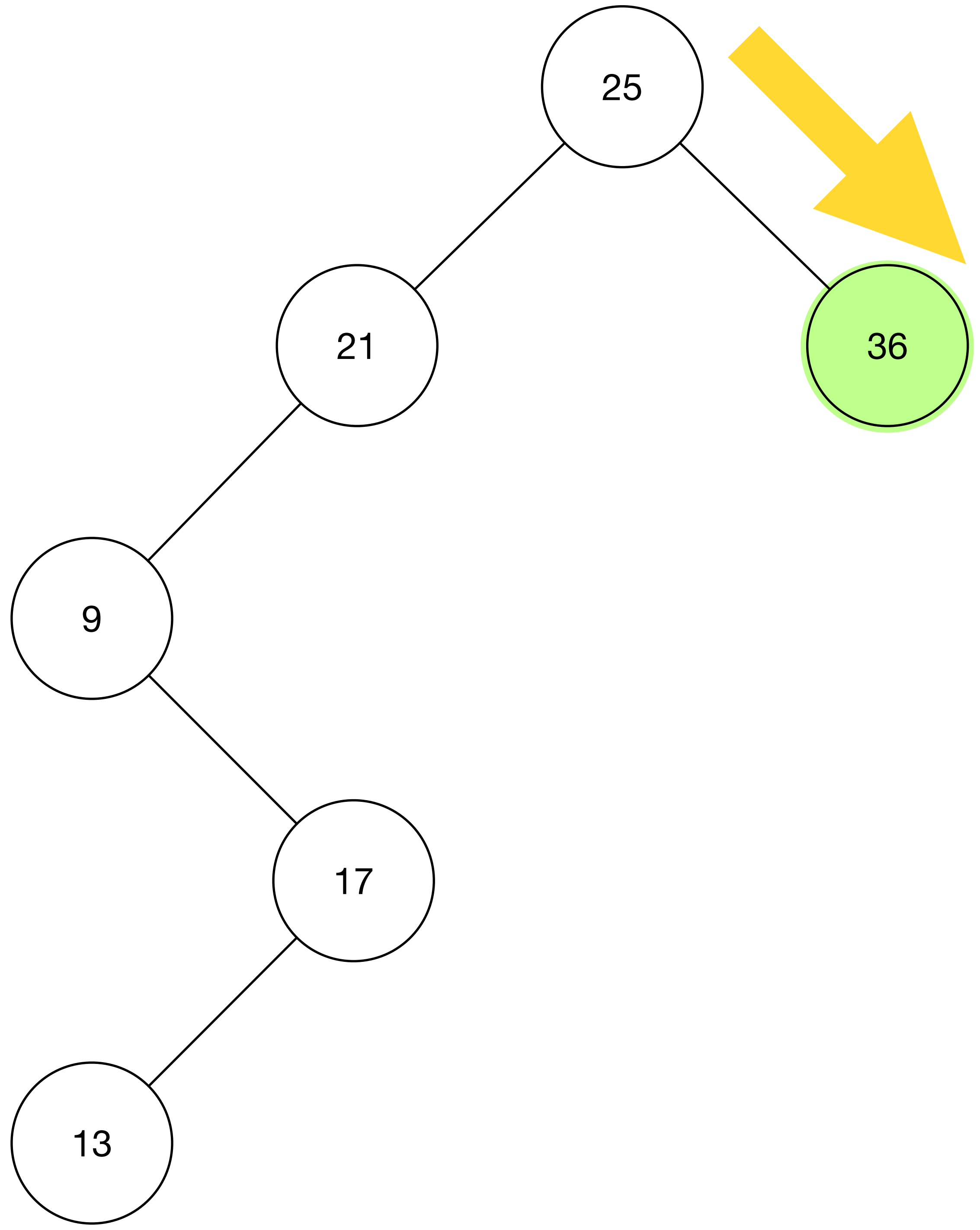
25, 21, 9, 17

Constructing a binary search tree



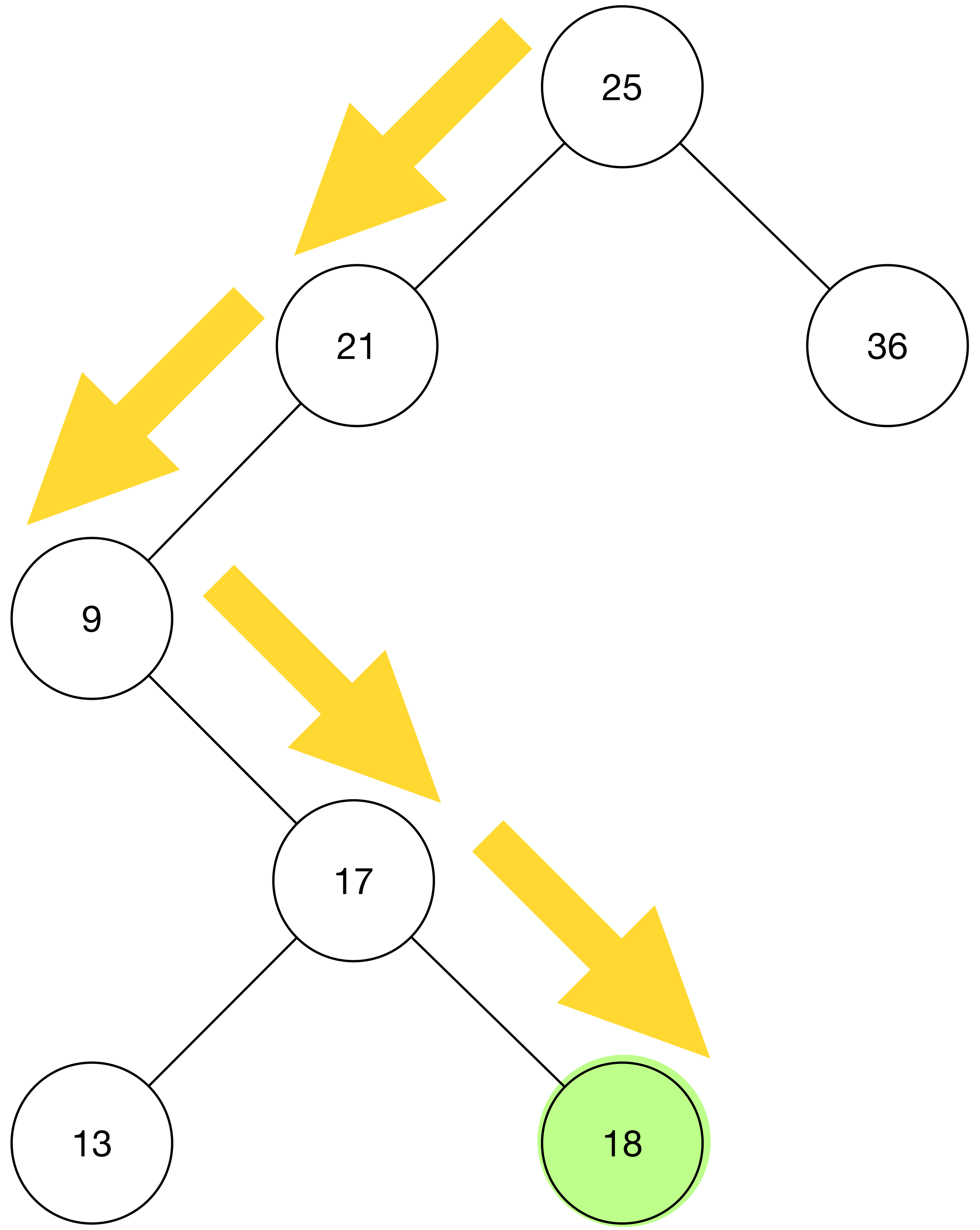
25, 21, 9, 17, 13

Constructing a binary search tree



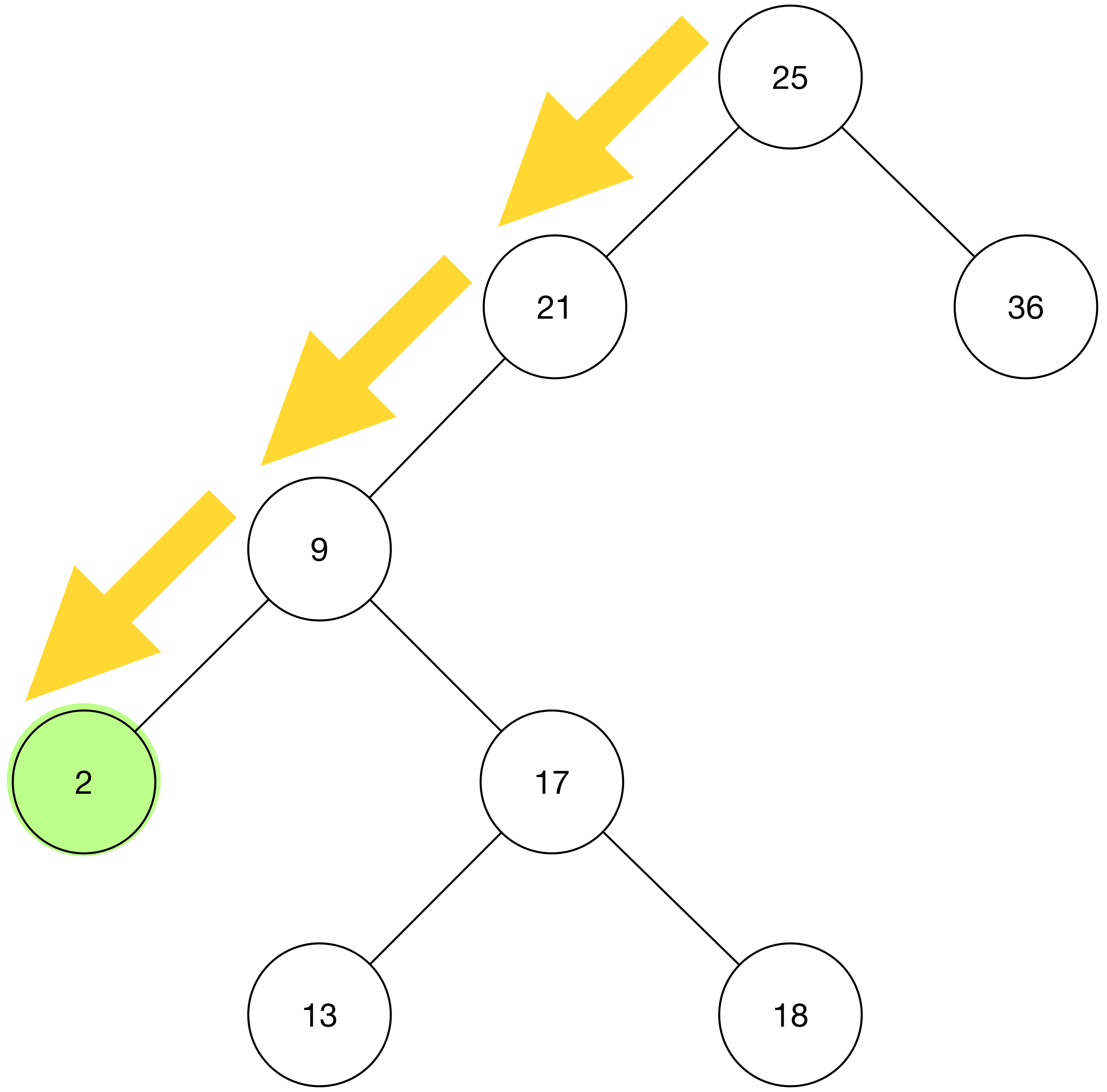
25, 21, 9, 17, 13, 36

Constructing a binary search tree



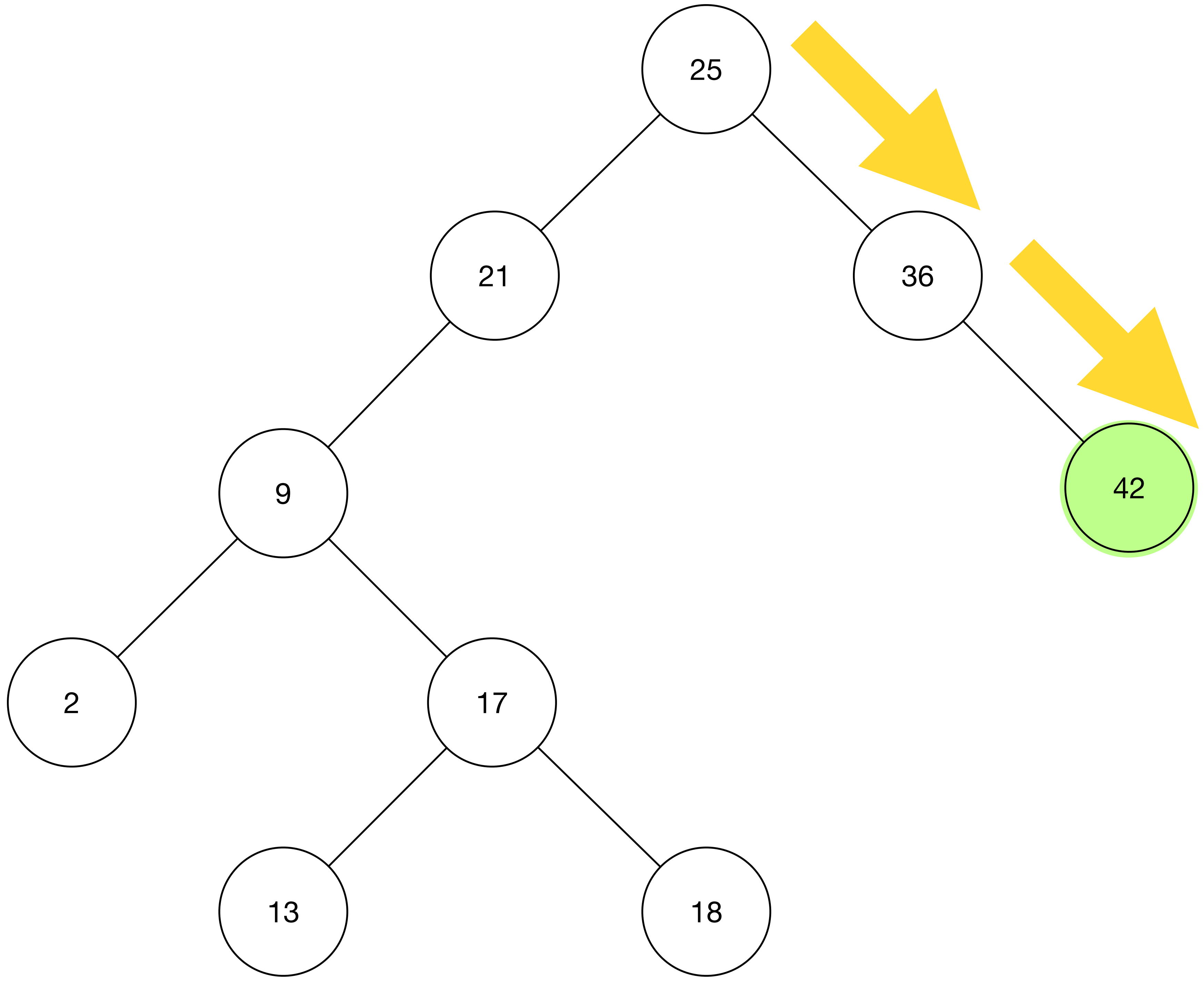
25, 21, 9, 17, 13, 36, 18

Constructing a binary search tree



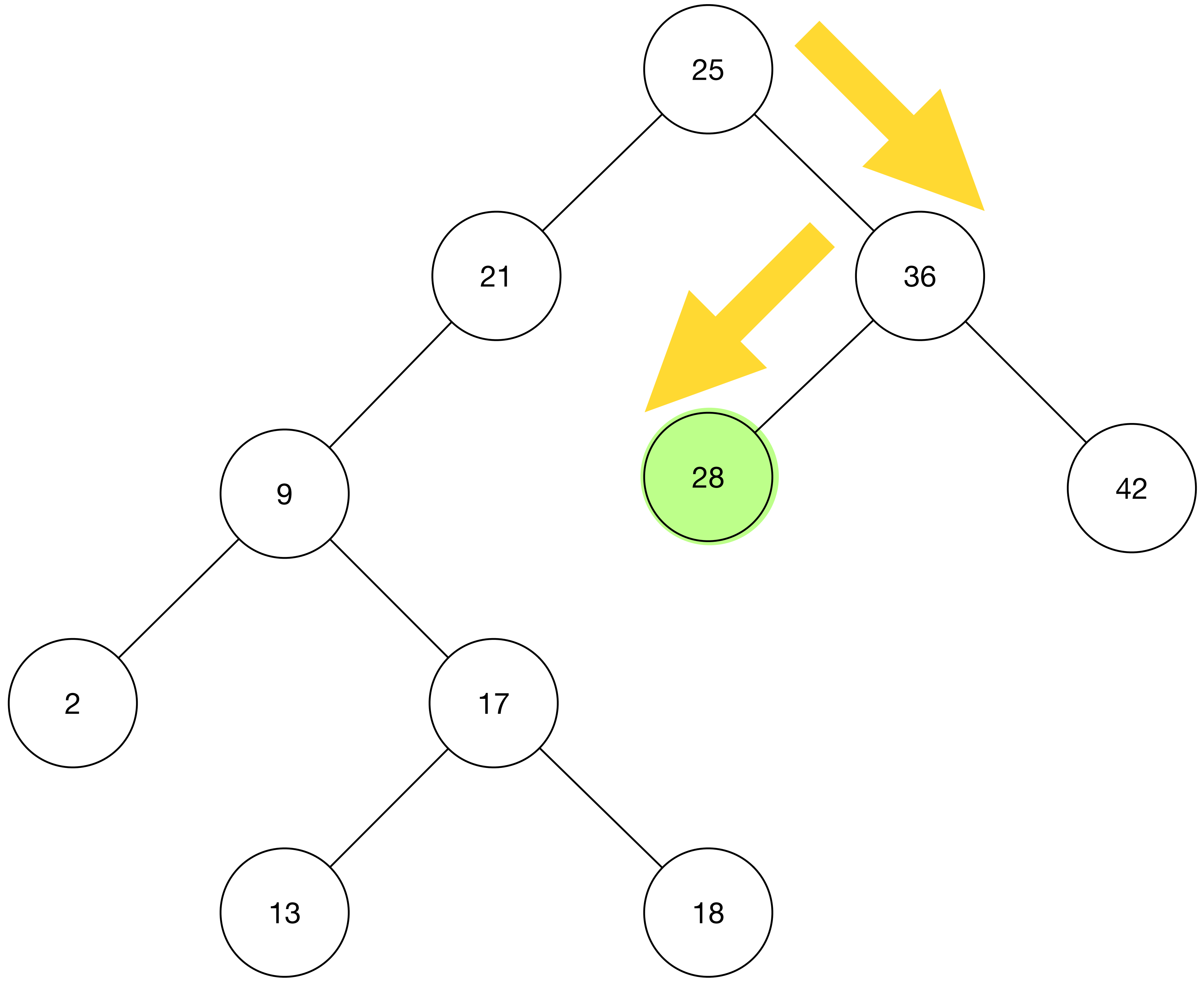
25, 21, 9, 17, 13, 36, 18, 2

Constructing a binary search tree



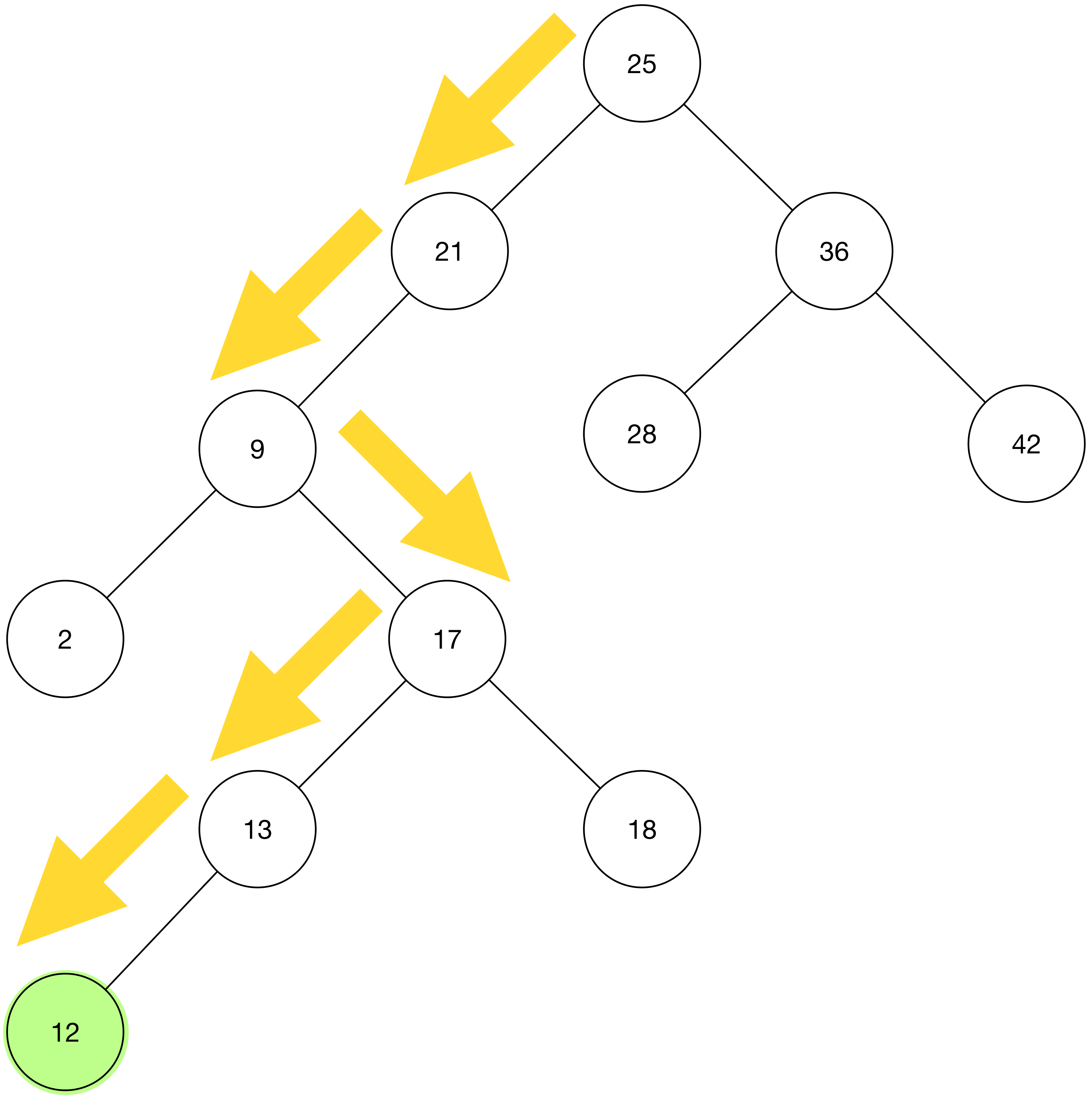
25, 21, 9, 17, 13, 36, 18, 2, 42

Constructing a binary search tree



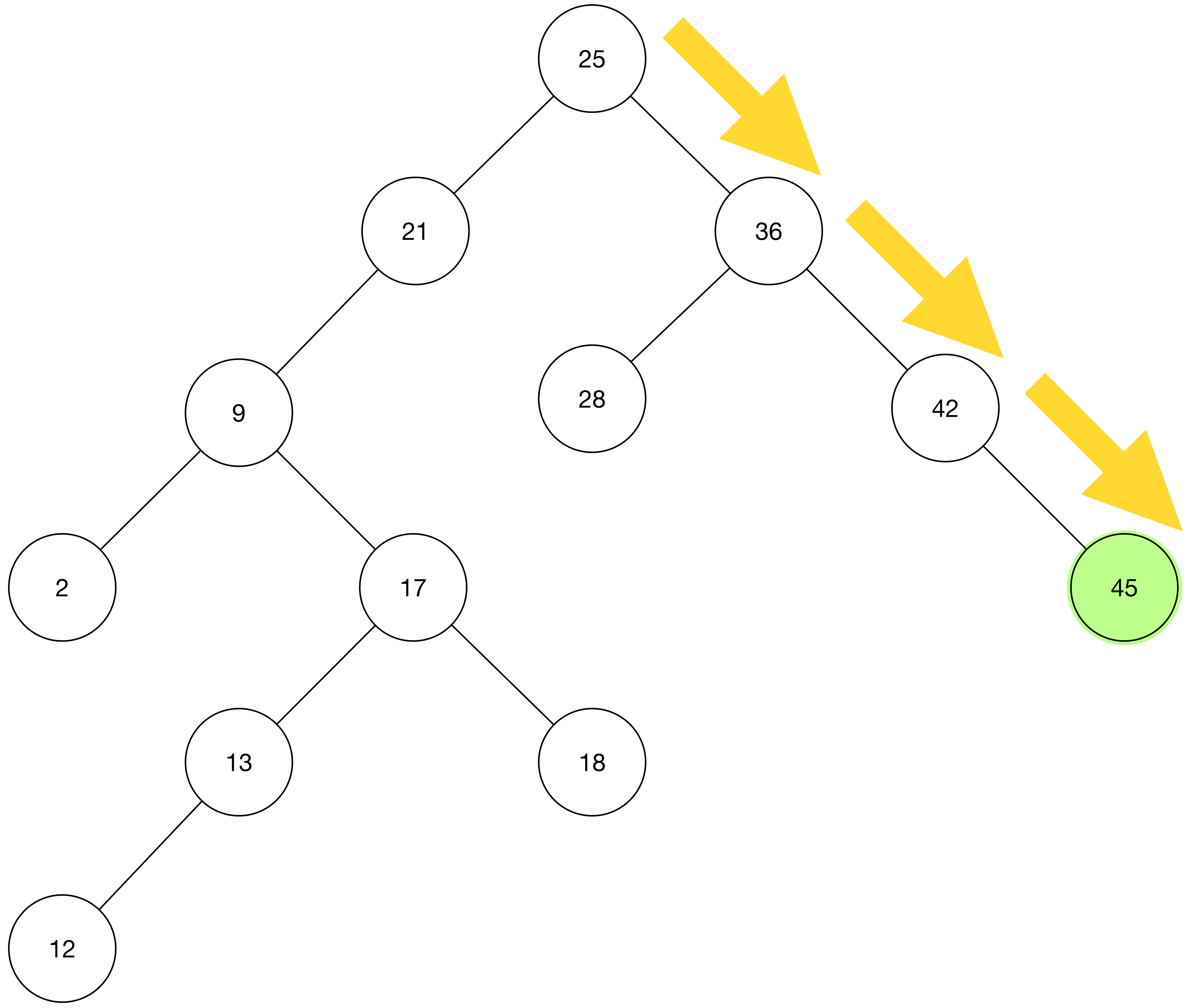
25, 21, 9, 17, 13, 36, 18, 2, 42, 28

Constructing a binary search tree



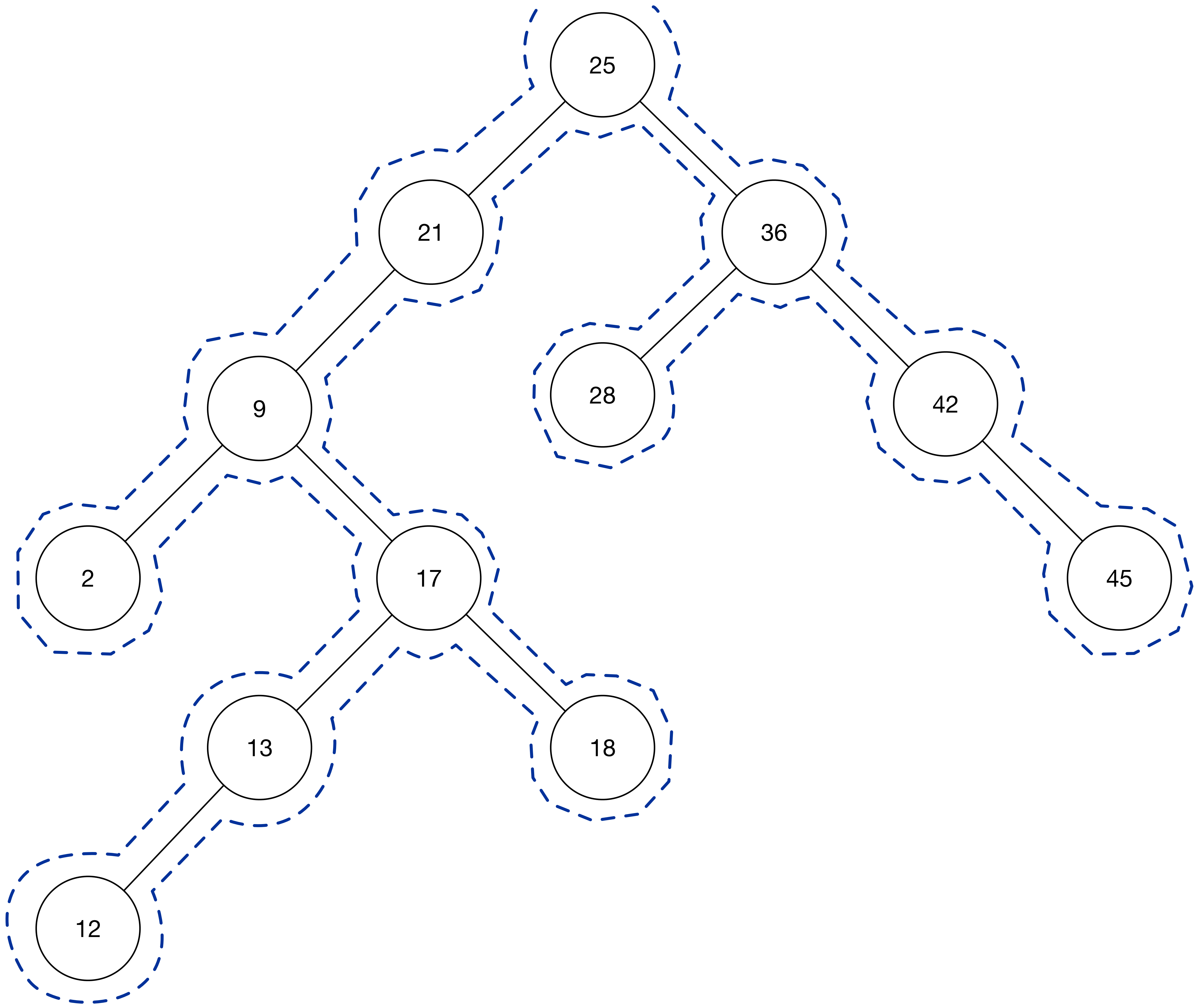
25, 21, 9, 17, 13, 36, 18, 2, 42, 28, 12

Constructing a binary search tree

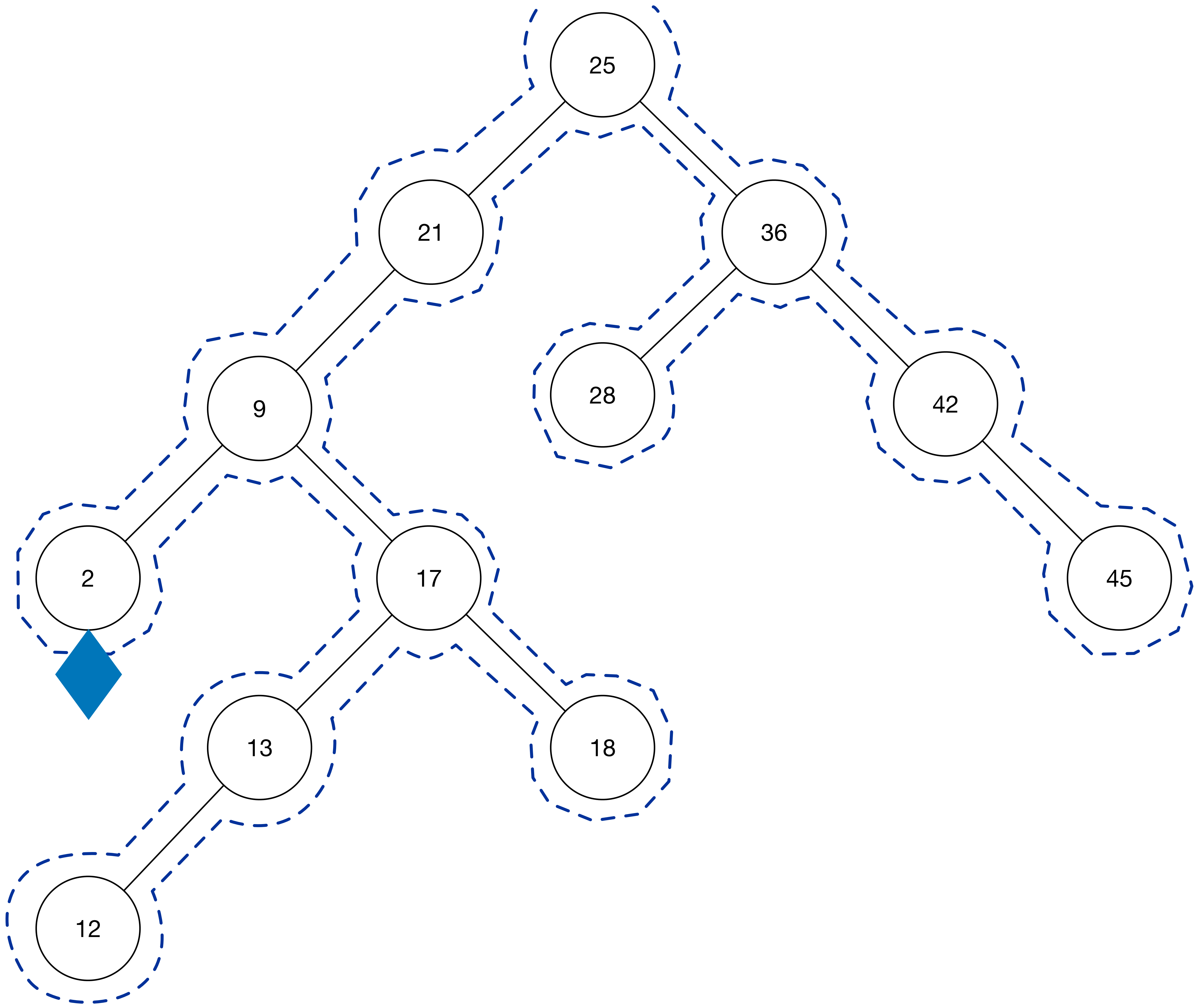


25, 21, 9, 17, 13, 36, 18, 2, 42, 28, 12, 45

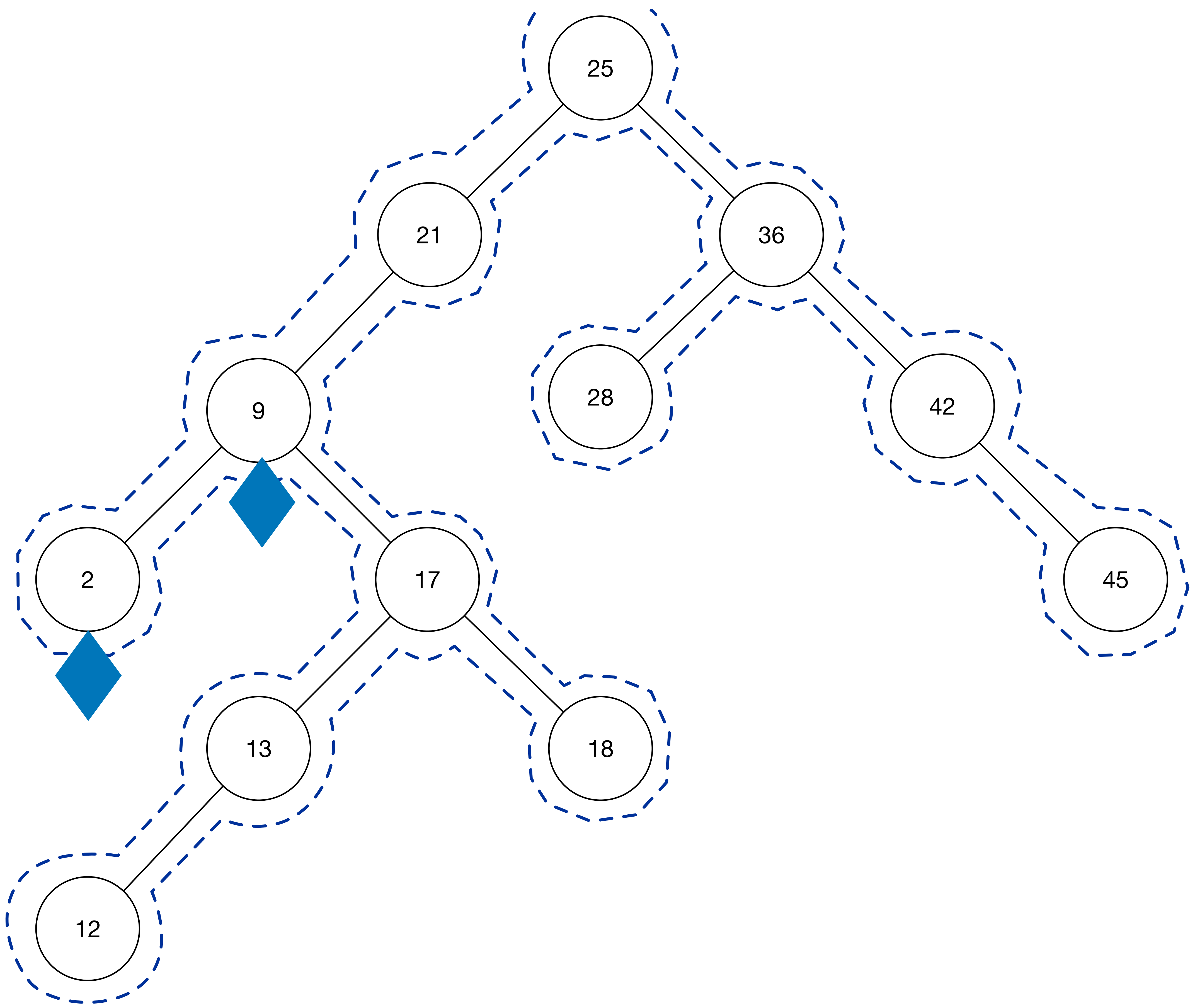
Constructing a binary search tree



Constructing a binary search tree

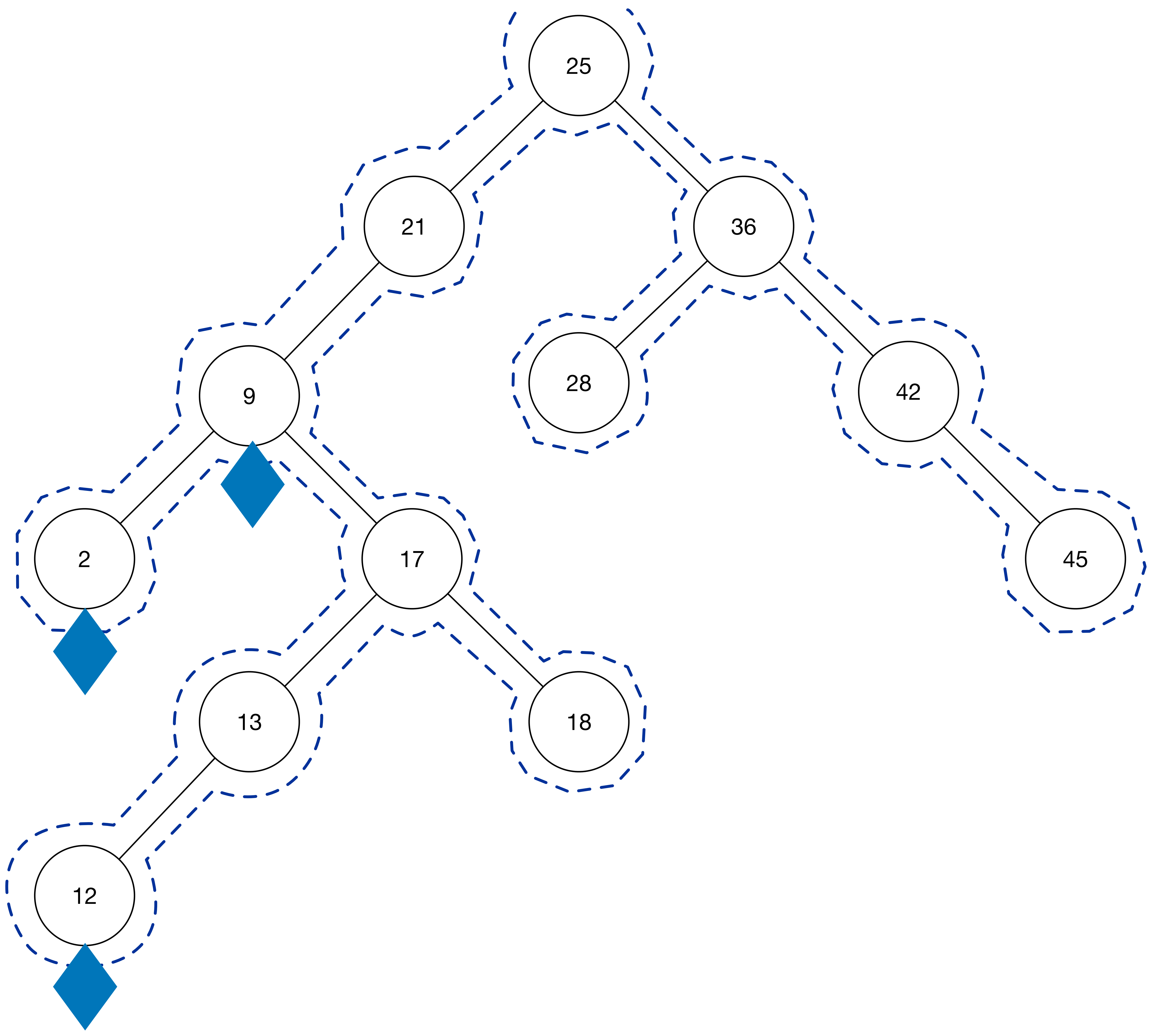


Constructing a binary search tree



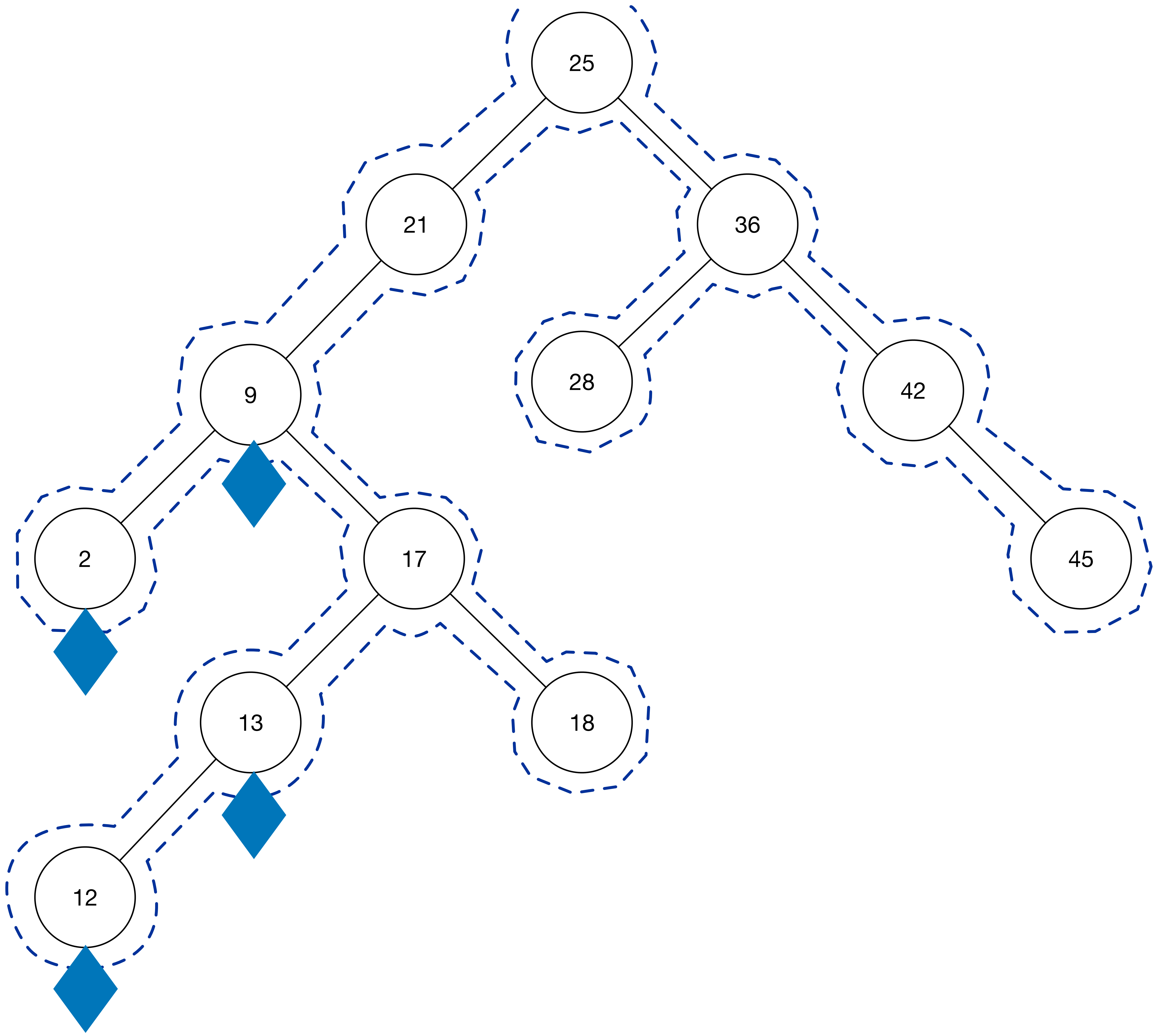
2, 9

Constructing a binary search tree



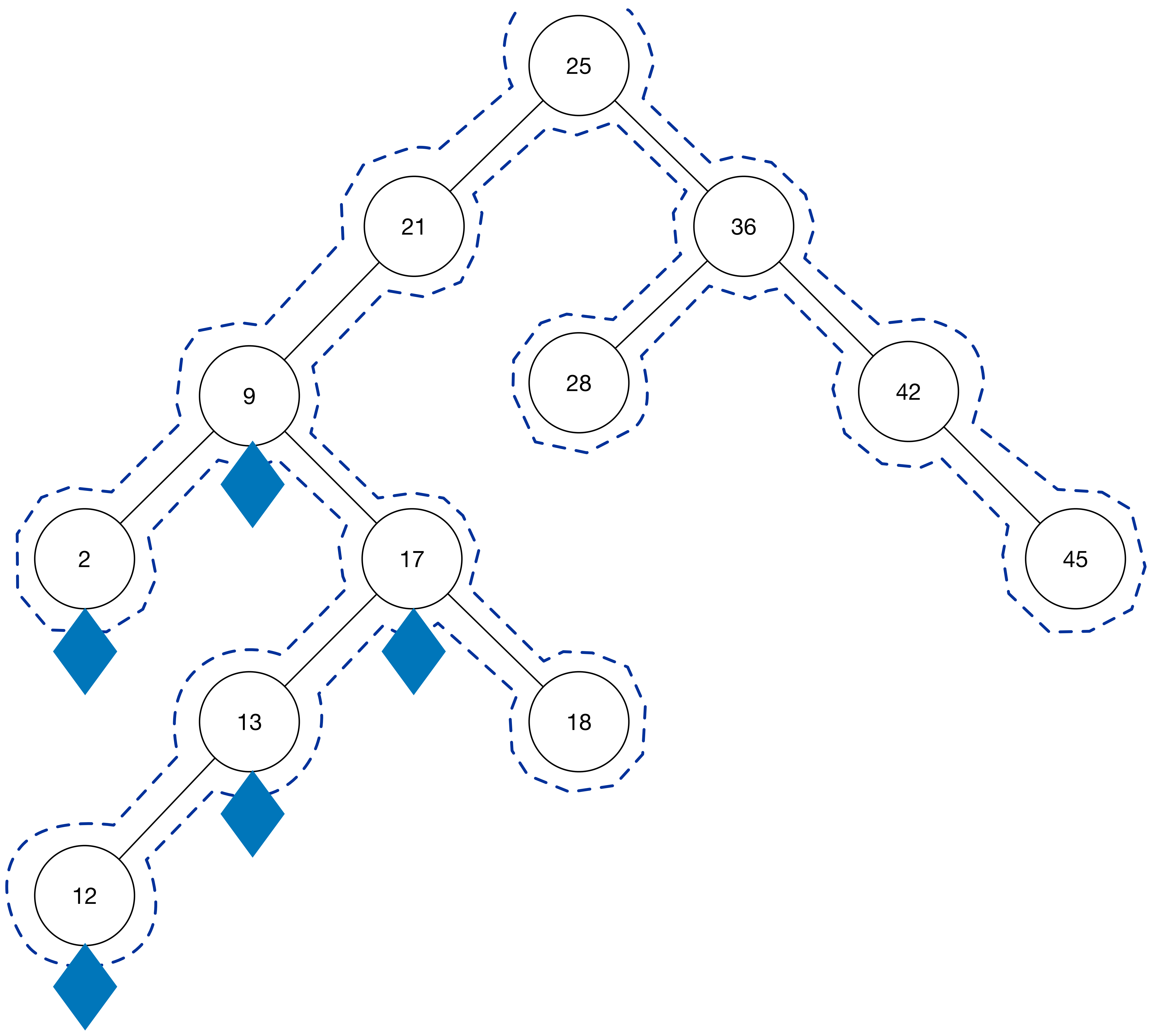
2, 9, 12

Constructing a binary search tree



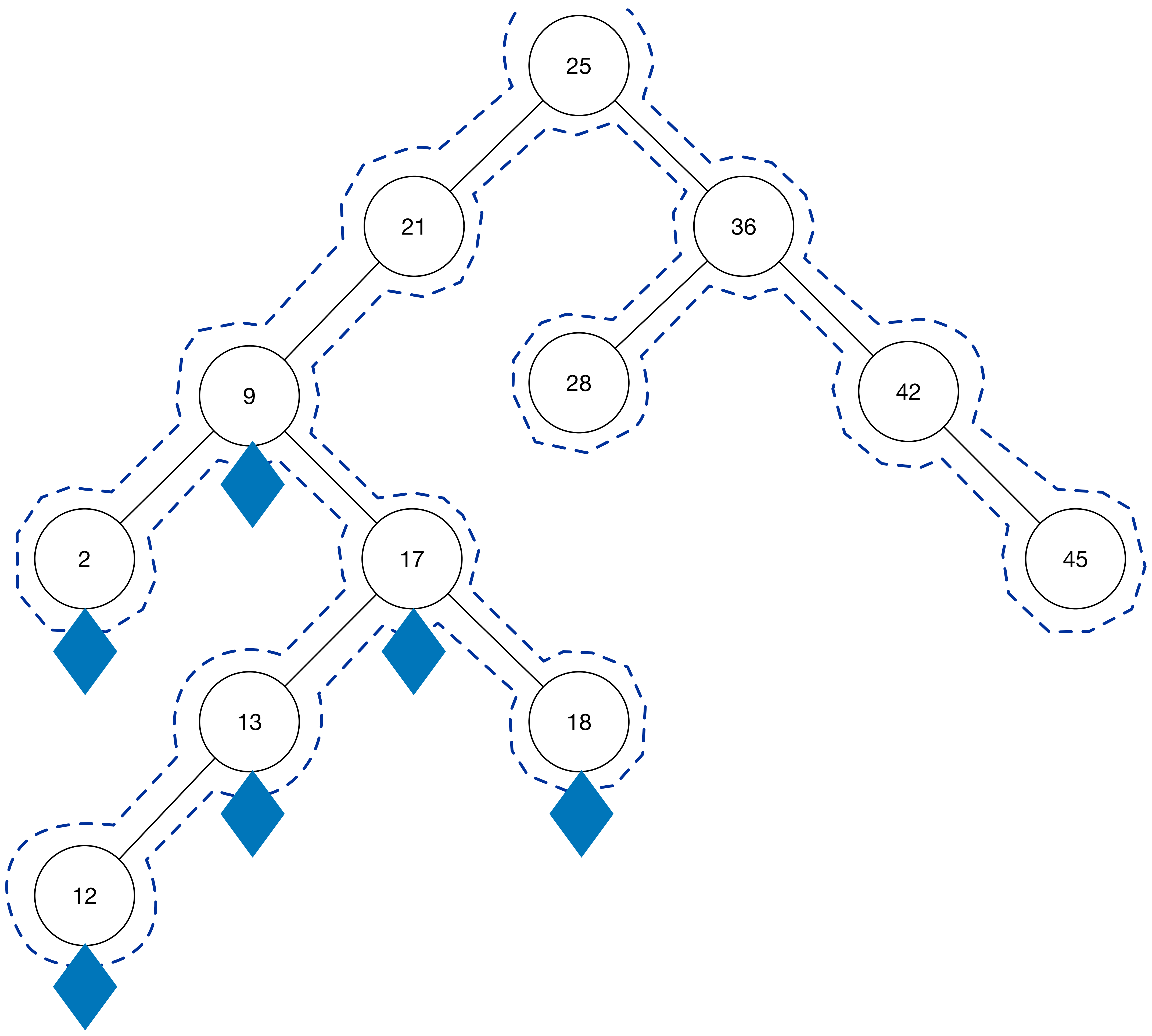
2, 9, 12, 13

Constructing a binary search tree



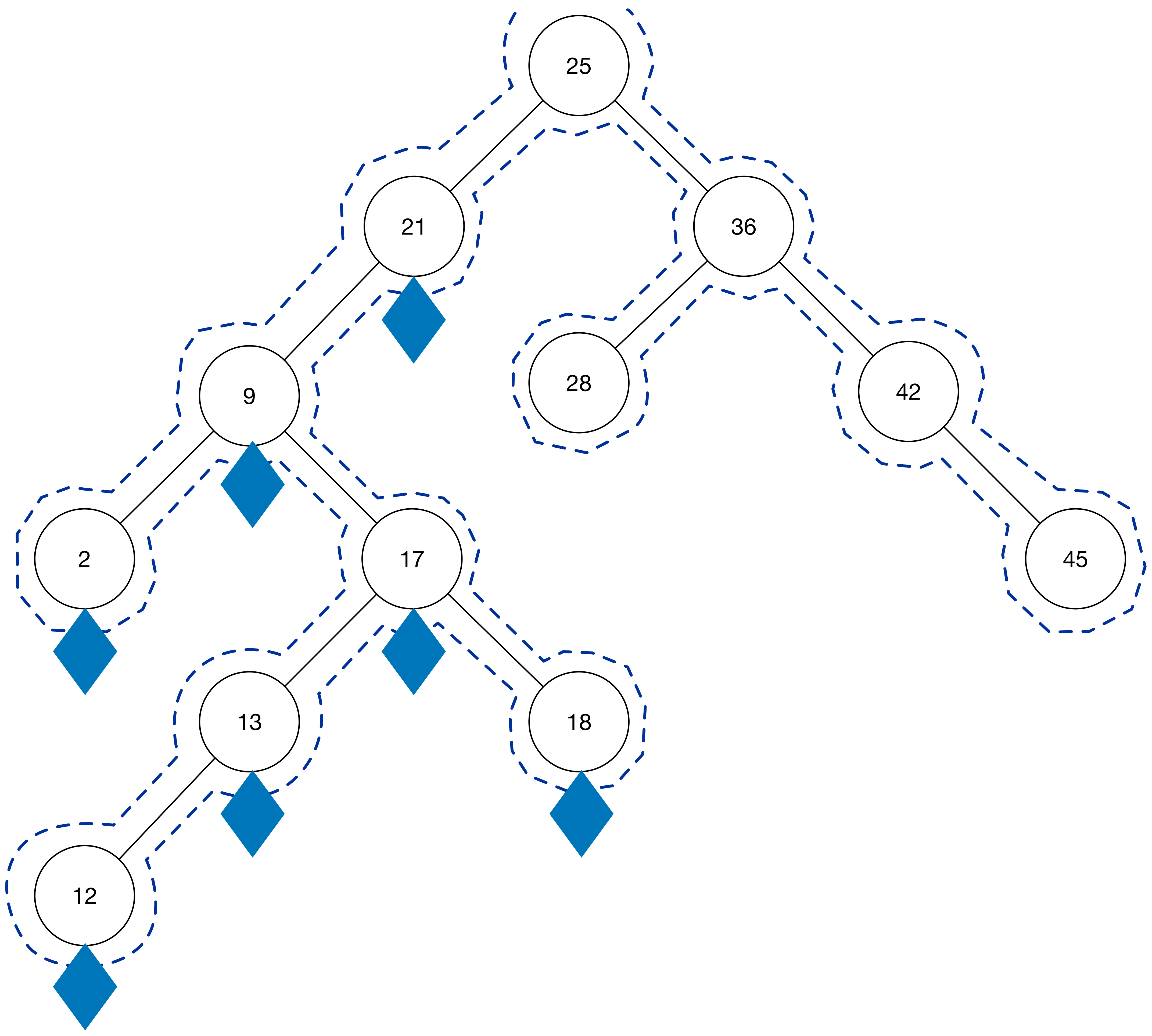
2, 9, 12, 13, 17

Constructing a binary search tree



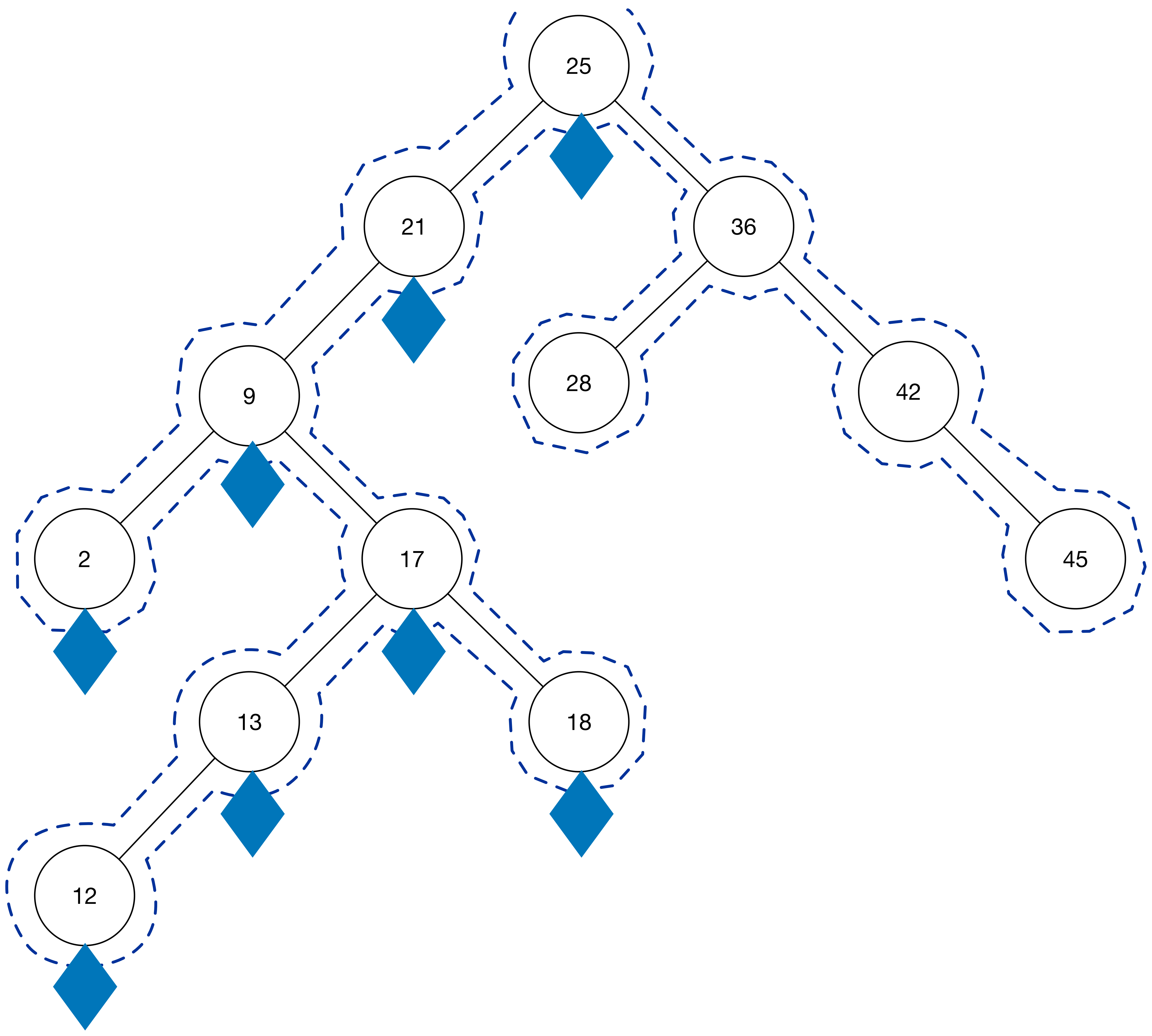
2, 9, 12, 13, 17, 18

Constructing a binary search tree



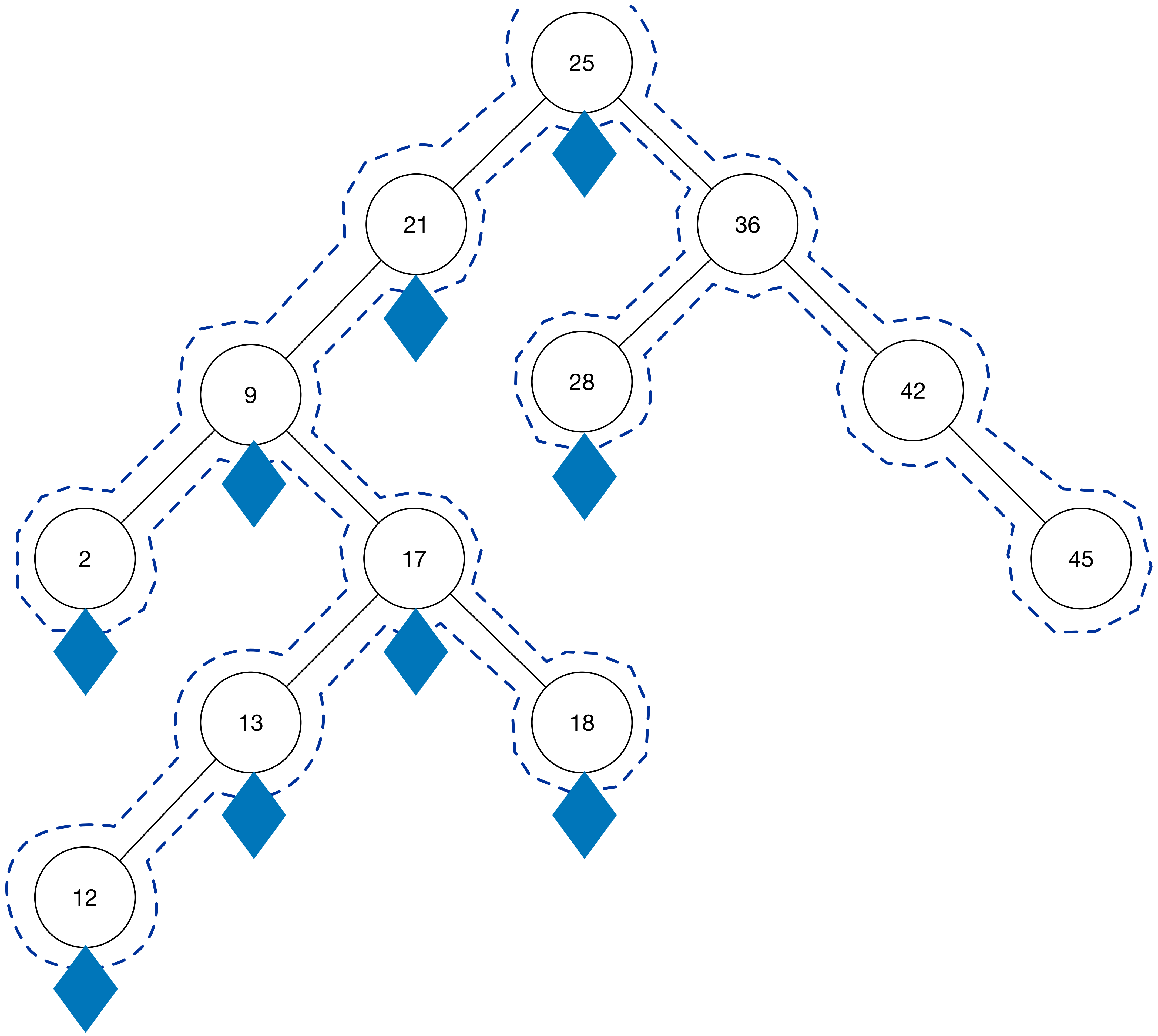
2, 9, 12, 13, 17, 18, 21

Constructing a binary search tree



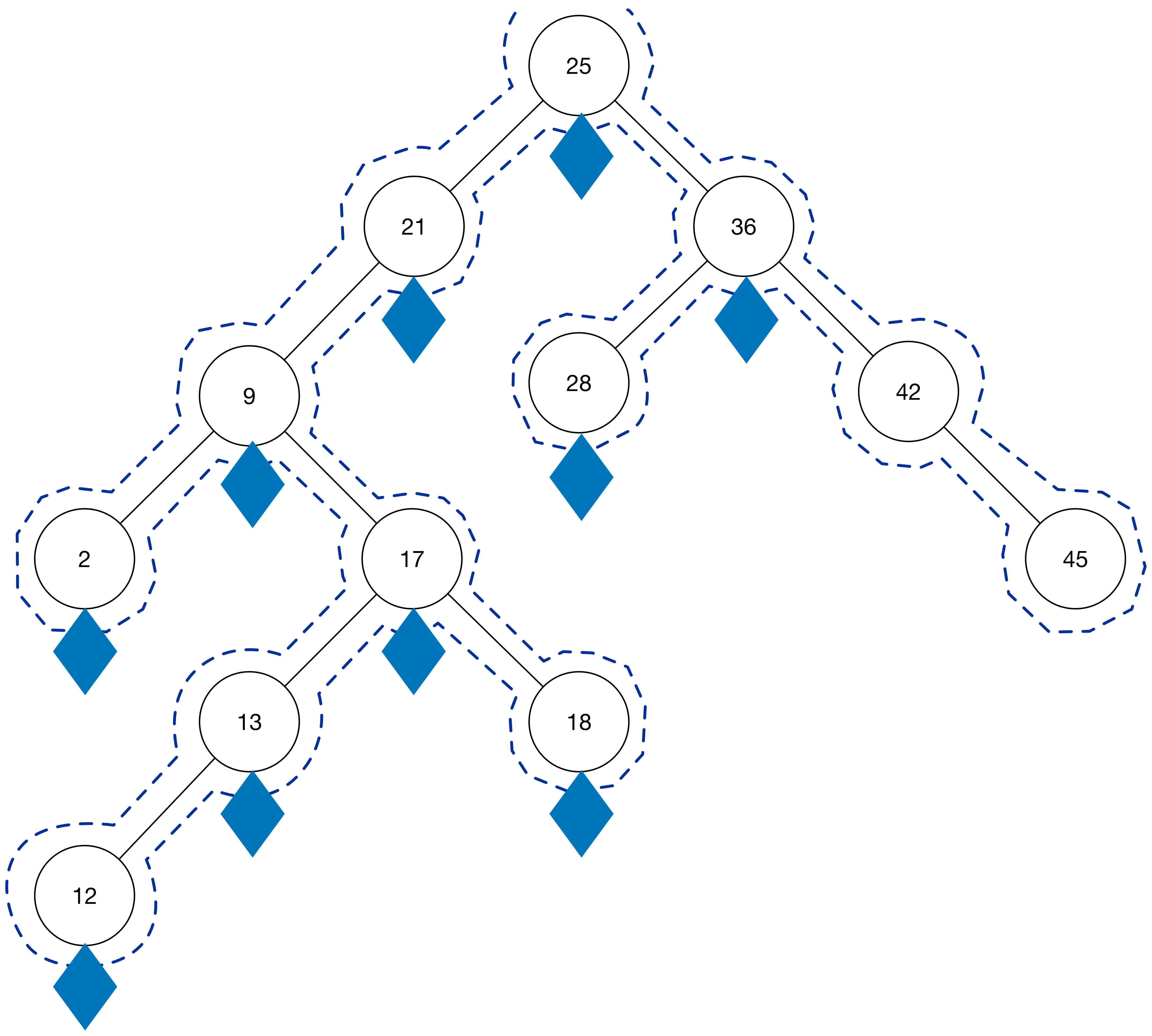
2, 9, 12, 13, 17, 18, 21, 25

Constructing a binary search tree



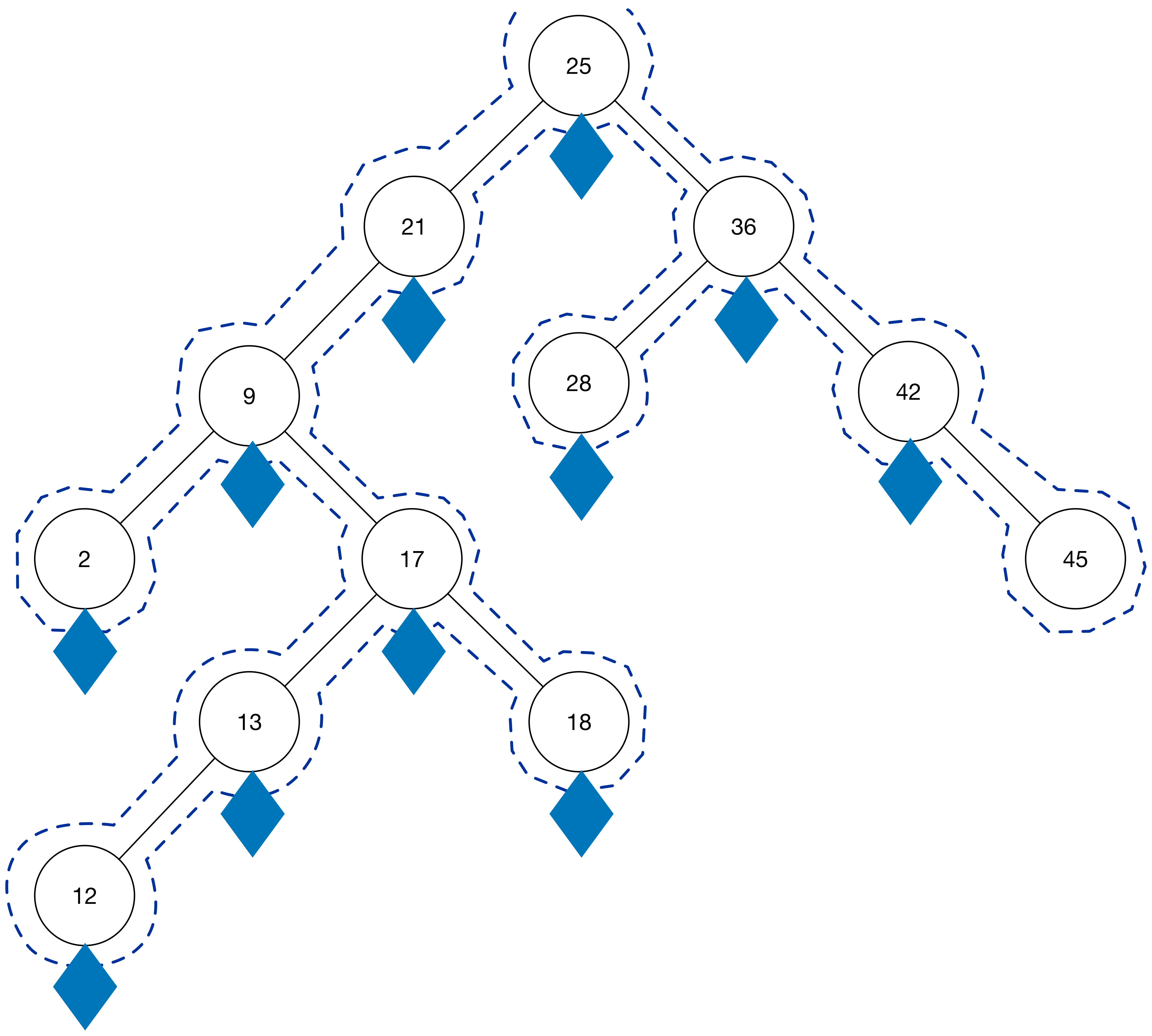
2, 9, 12, 13, 17, 18, 21, 25, 28

Constructing a binary search tree



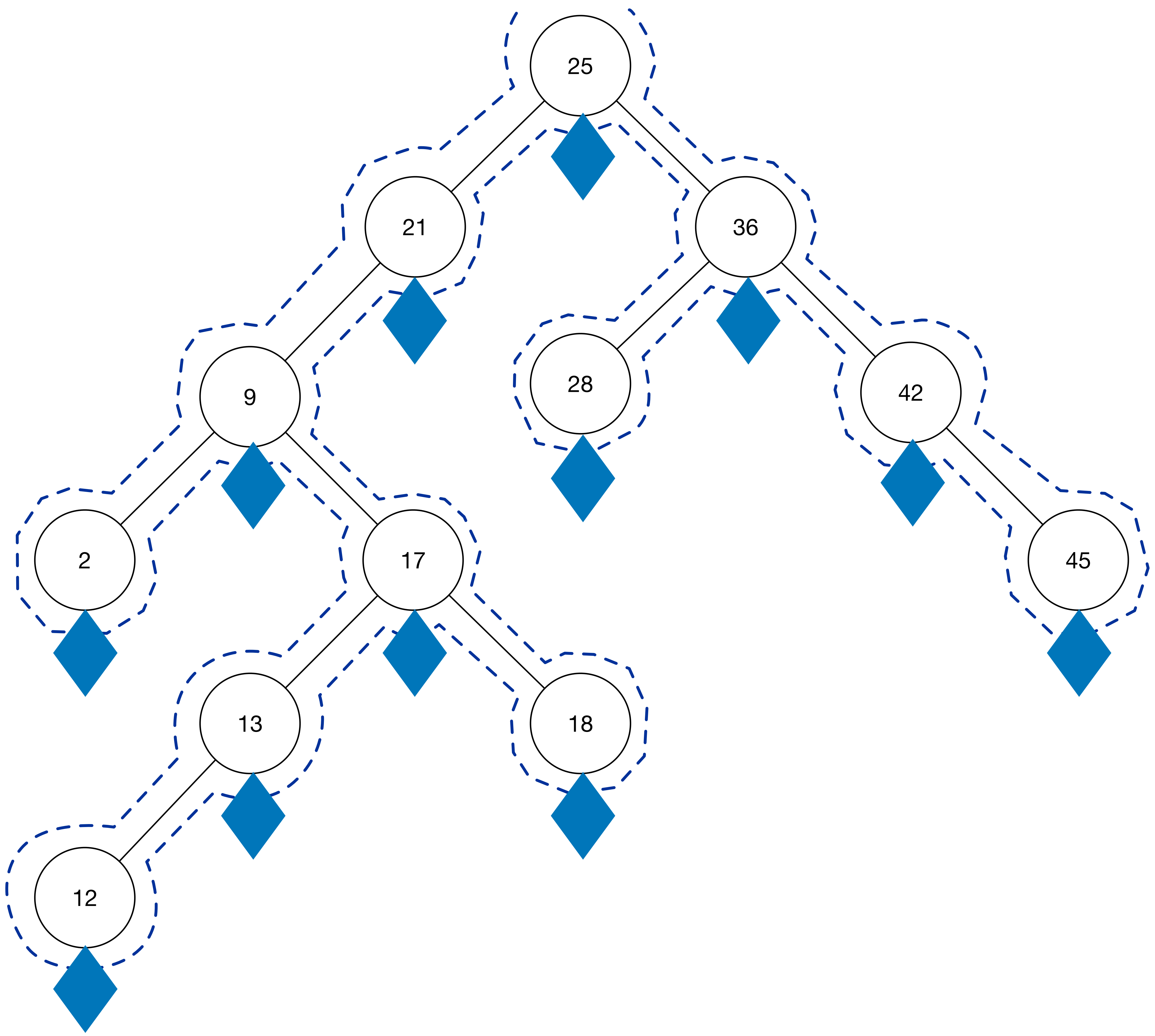
2, 9, 12, 13, 17, 18, 21, 25, 28, 36

Constructing a binary search tree



2, 9, 12, 13, 17, 18, 21, 25, 28, 36, 42

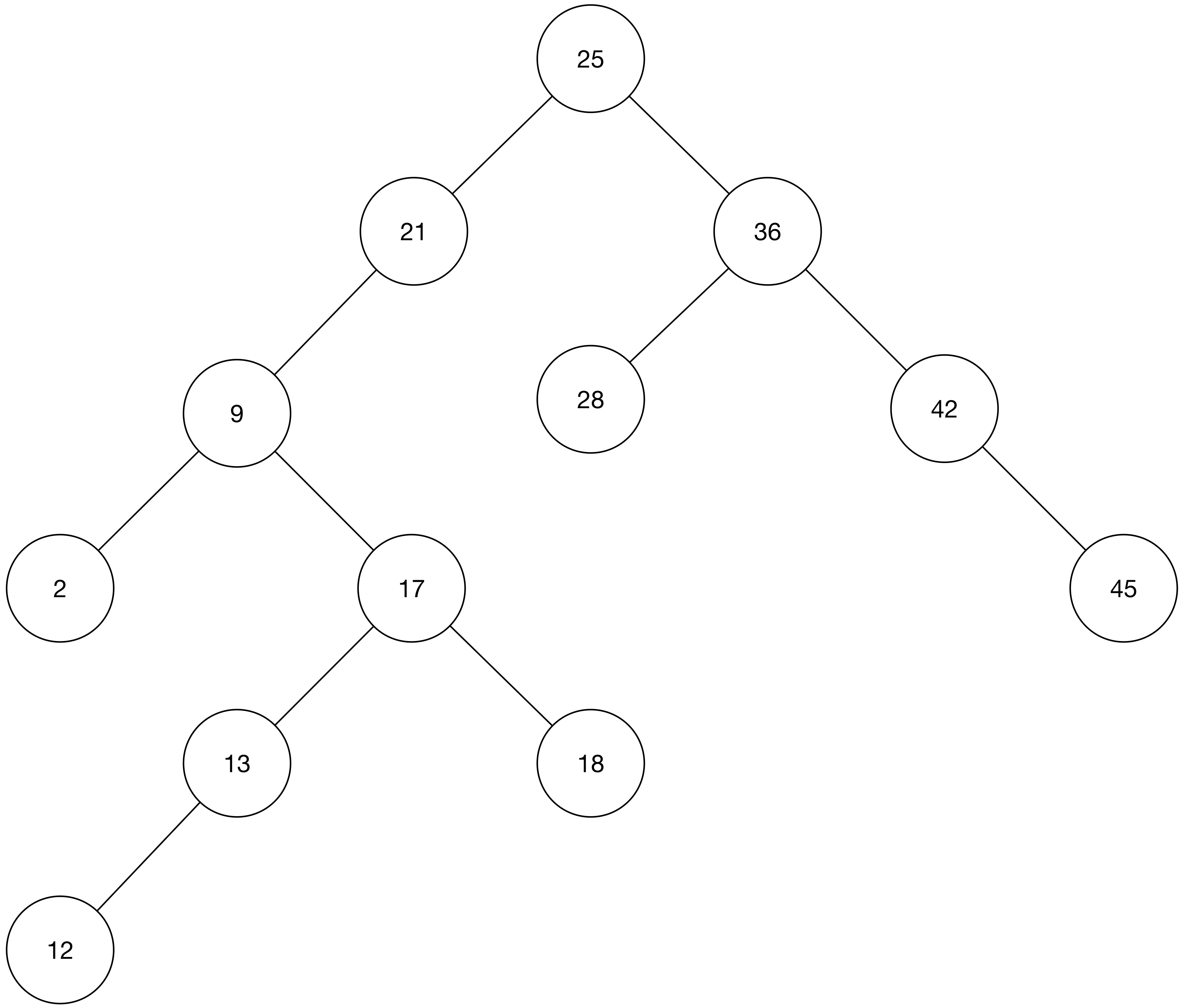
Constructing a binary search tree



2, 9, 12, 13, 17, 18, 21, 25, 28, 36, 42, 45

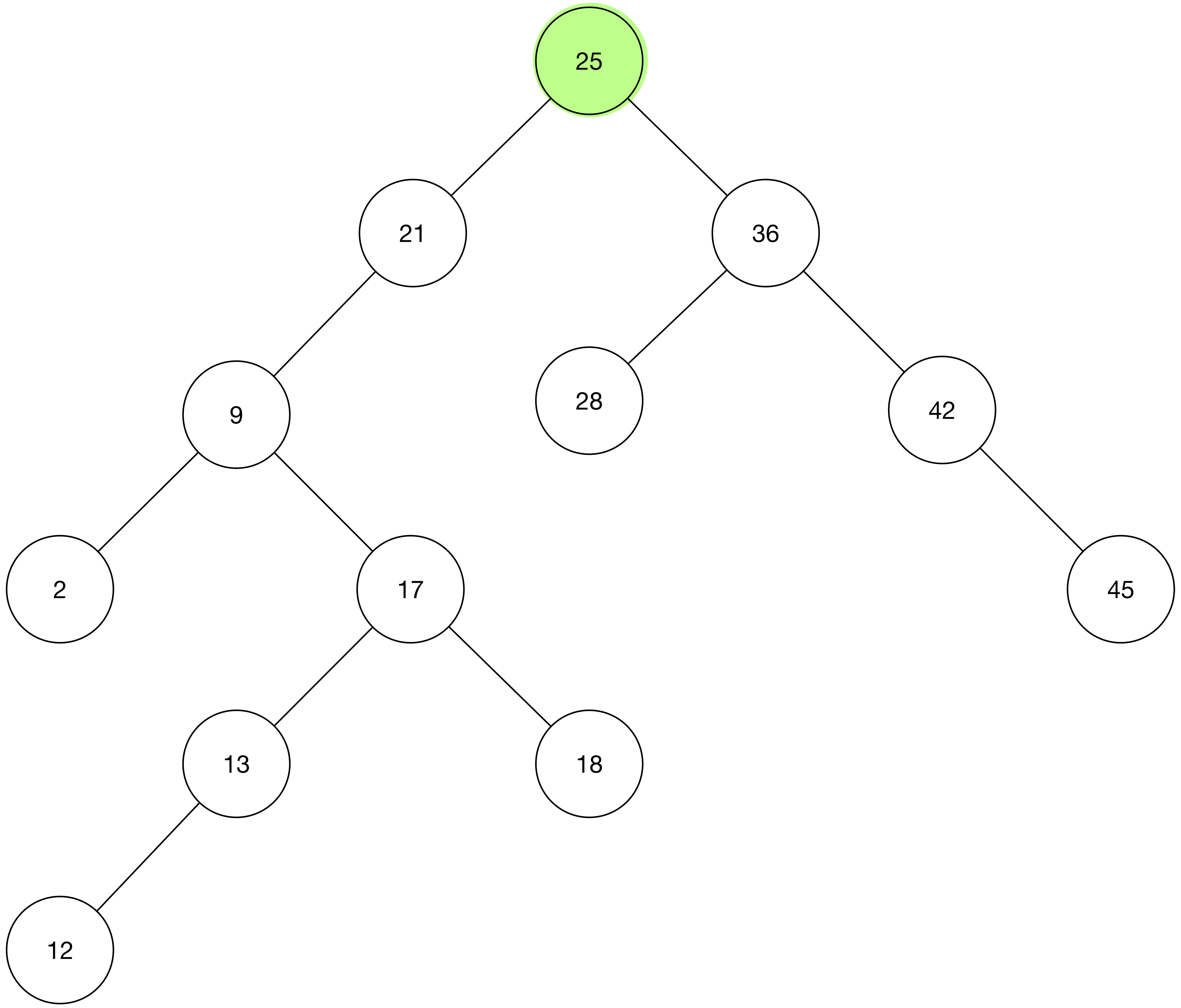
Searching a binary search tree

13



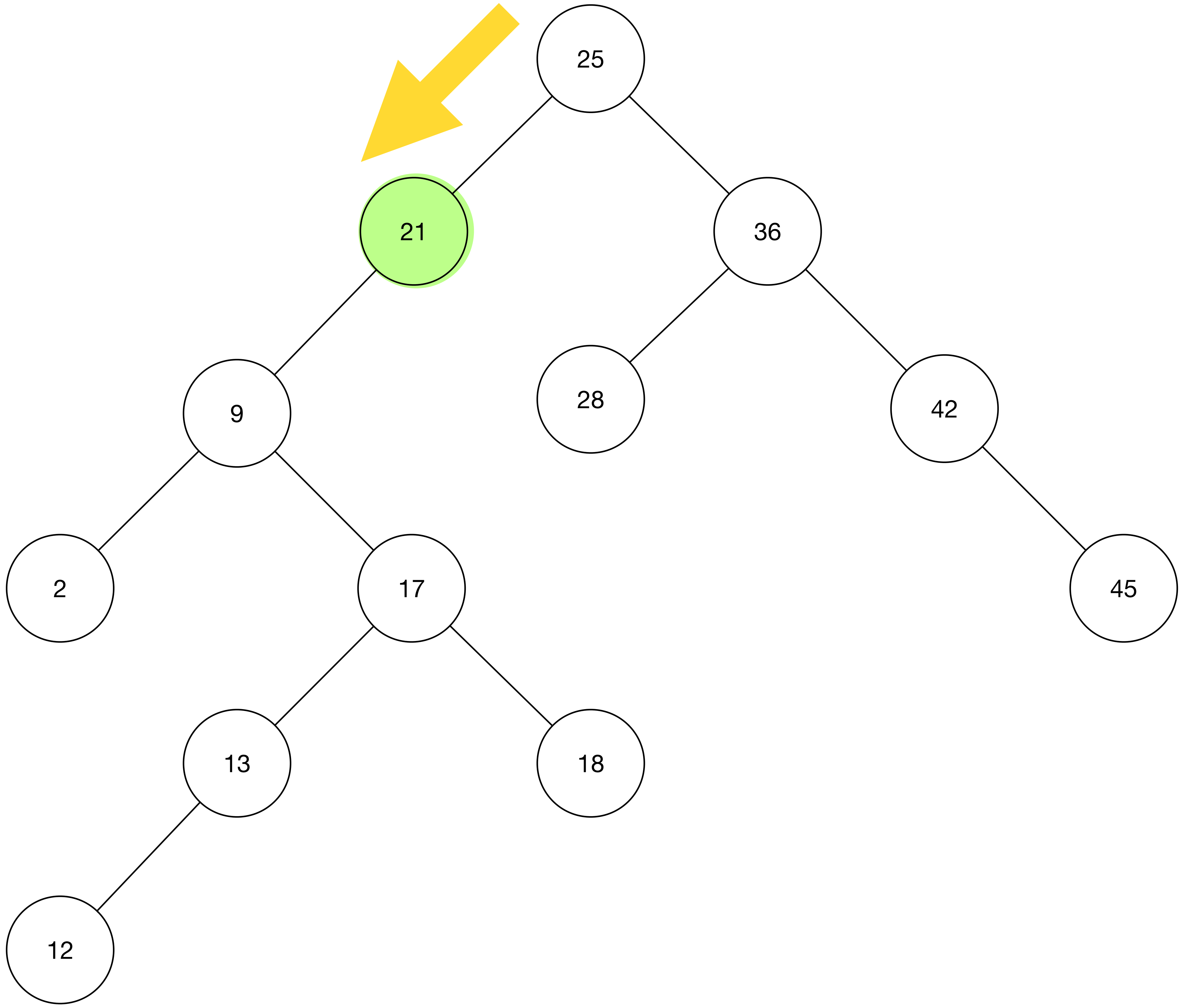
Searching a binary search tree

13
25? 13 < 25. Go left.



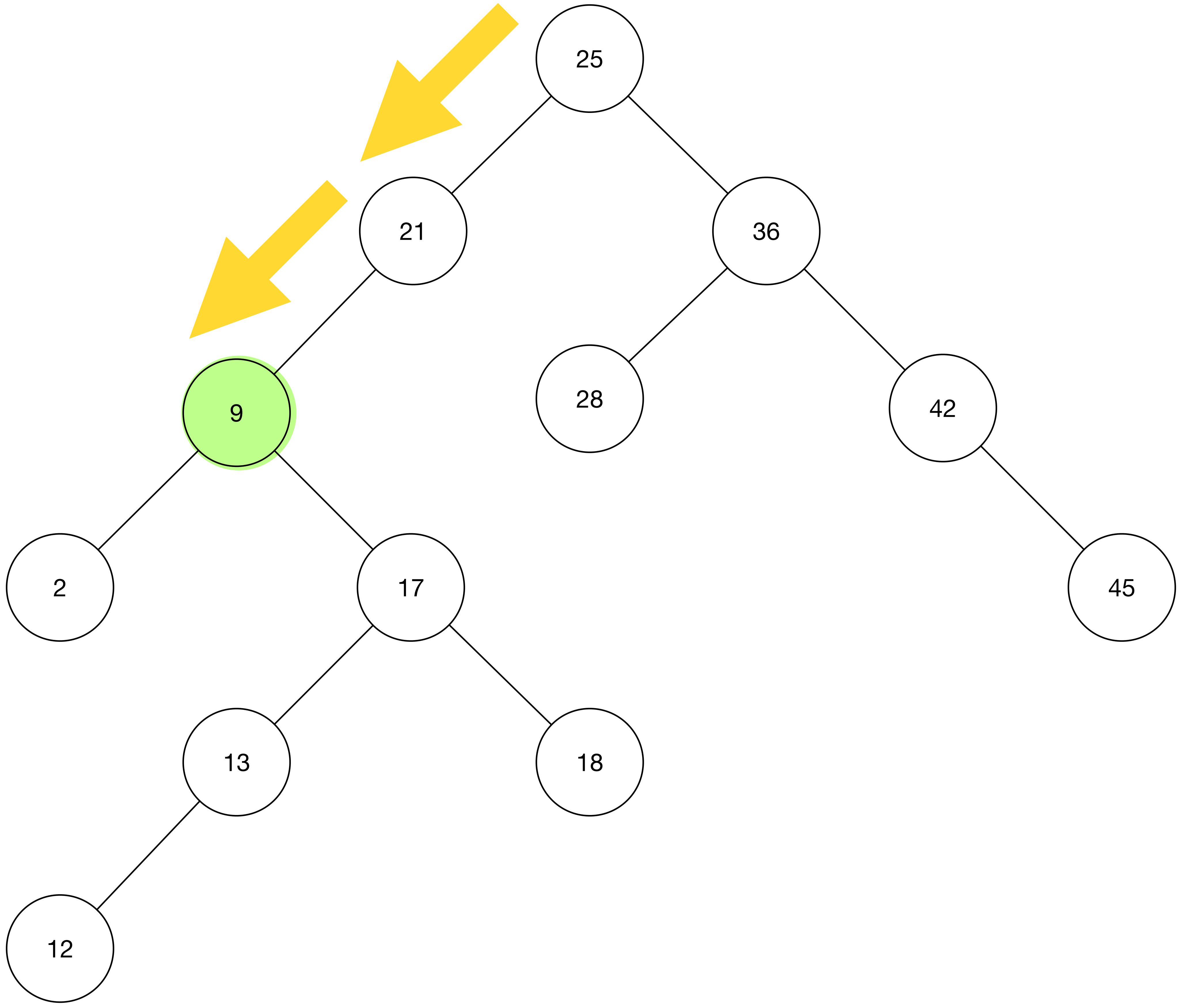
Searching a binary search tree

13
25? 13 < 25. Go left.
21? 13 < 21. Go left.



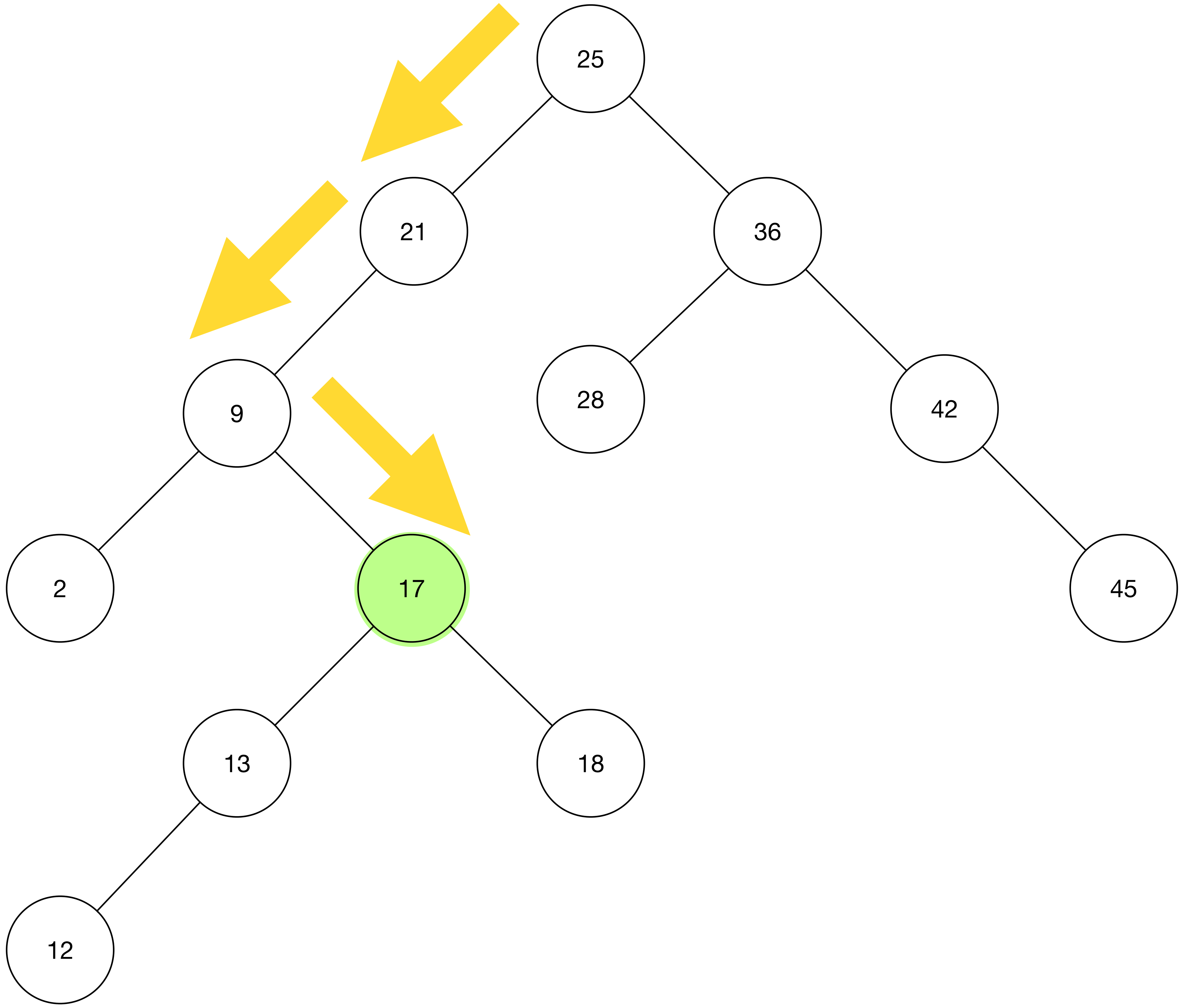
Searching a binary search tree

13
25? 13 < 25. Go left.
21? 13 < 21. Go left.
9? 13 > 9. Go right.



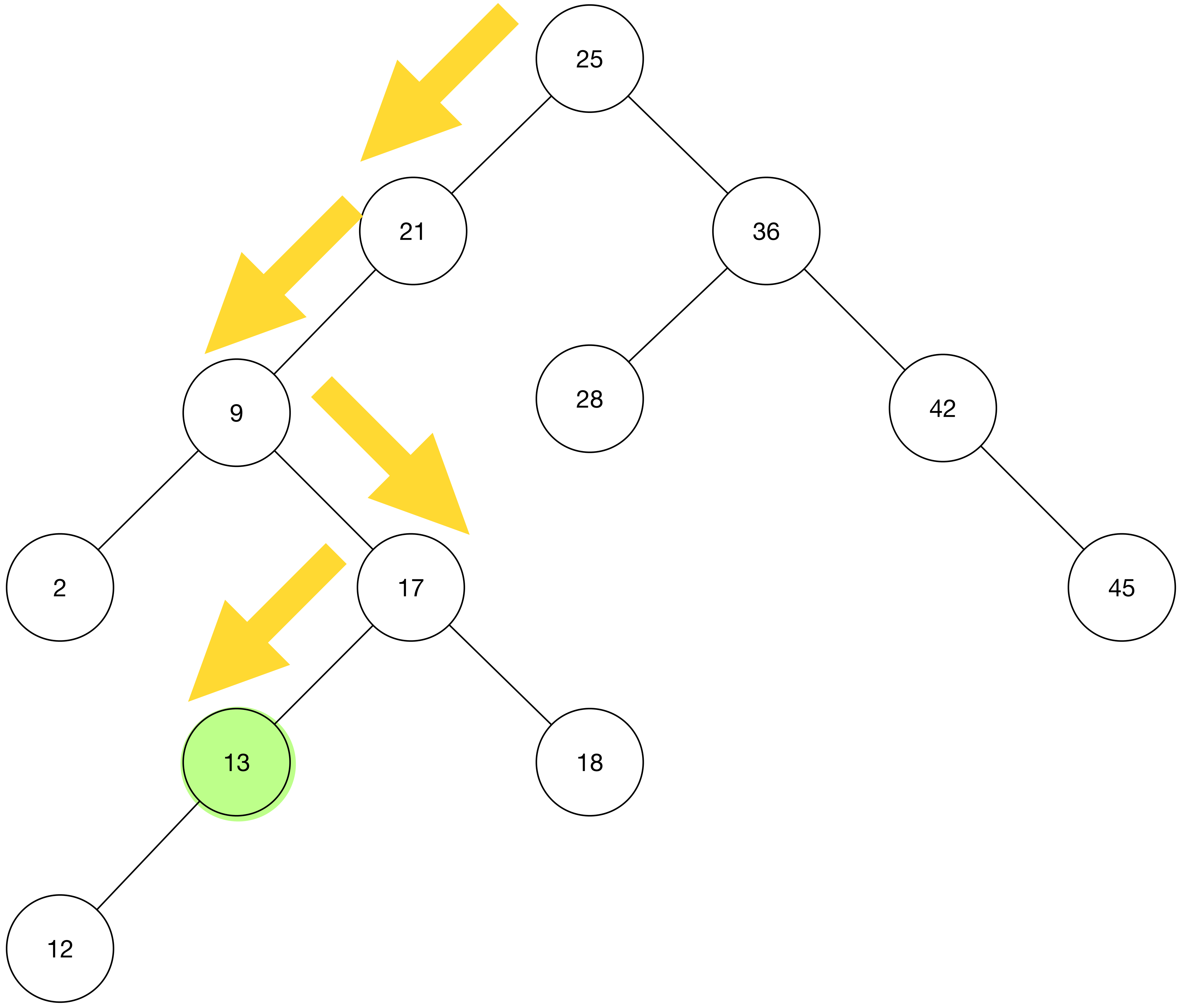
Searching a binary search tree

13
25? 13 < 25. Go left.
21? 13 < 21. Go left.
9? 13 > 9. Go right.
17? 13 < 17. Go left.



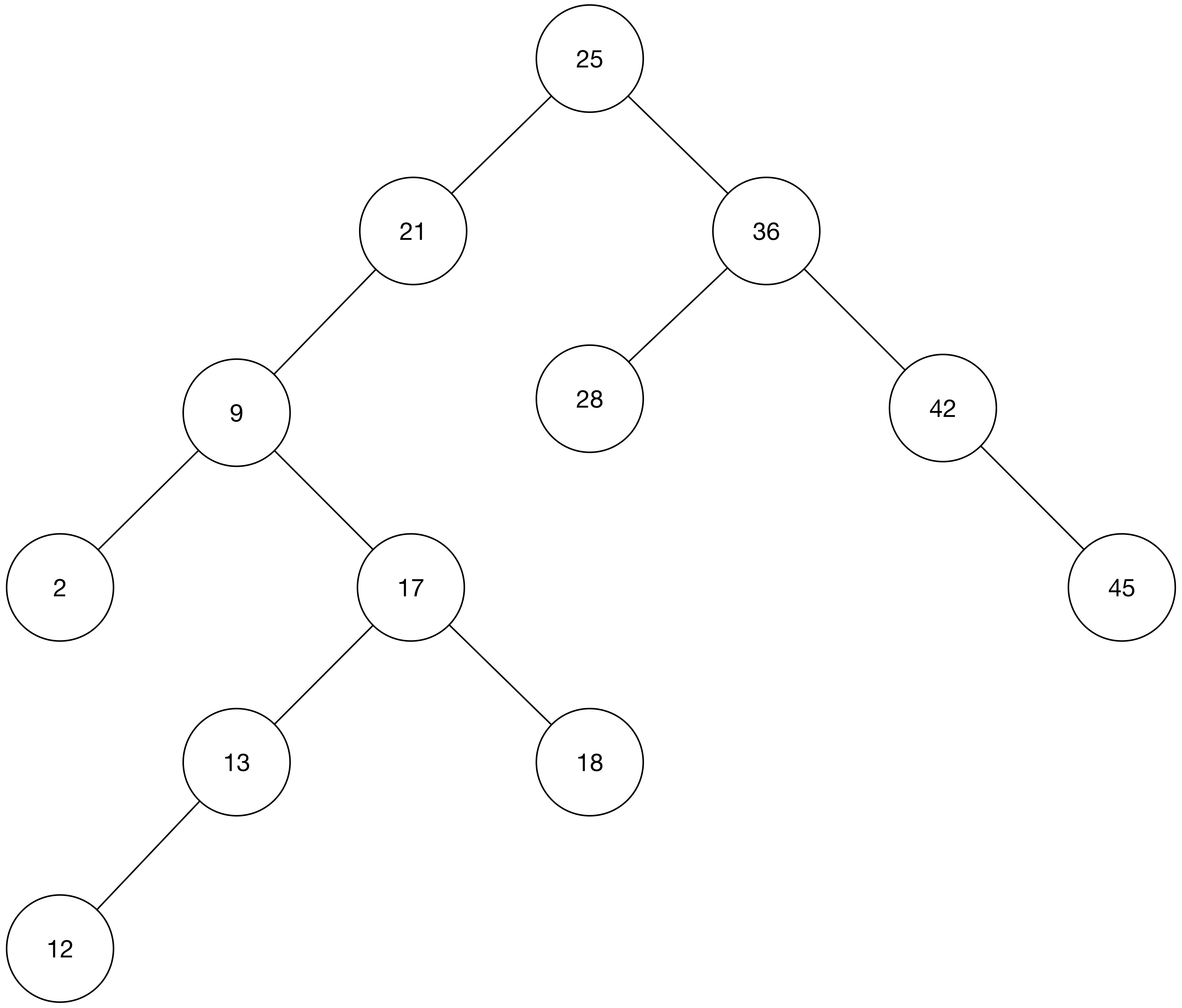
Searching a binary search tree

13
25? 13 < 25. Go left.
21? 13 < 21. Go left.
9? 13 > 9. Go right.
17? 13 < 17. Go left.
13? 13 = 13. FOUND IT!



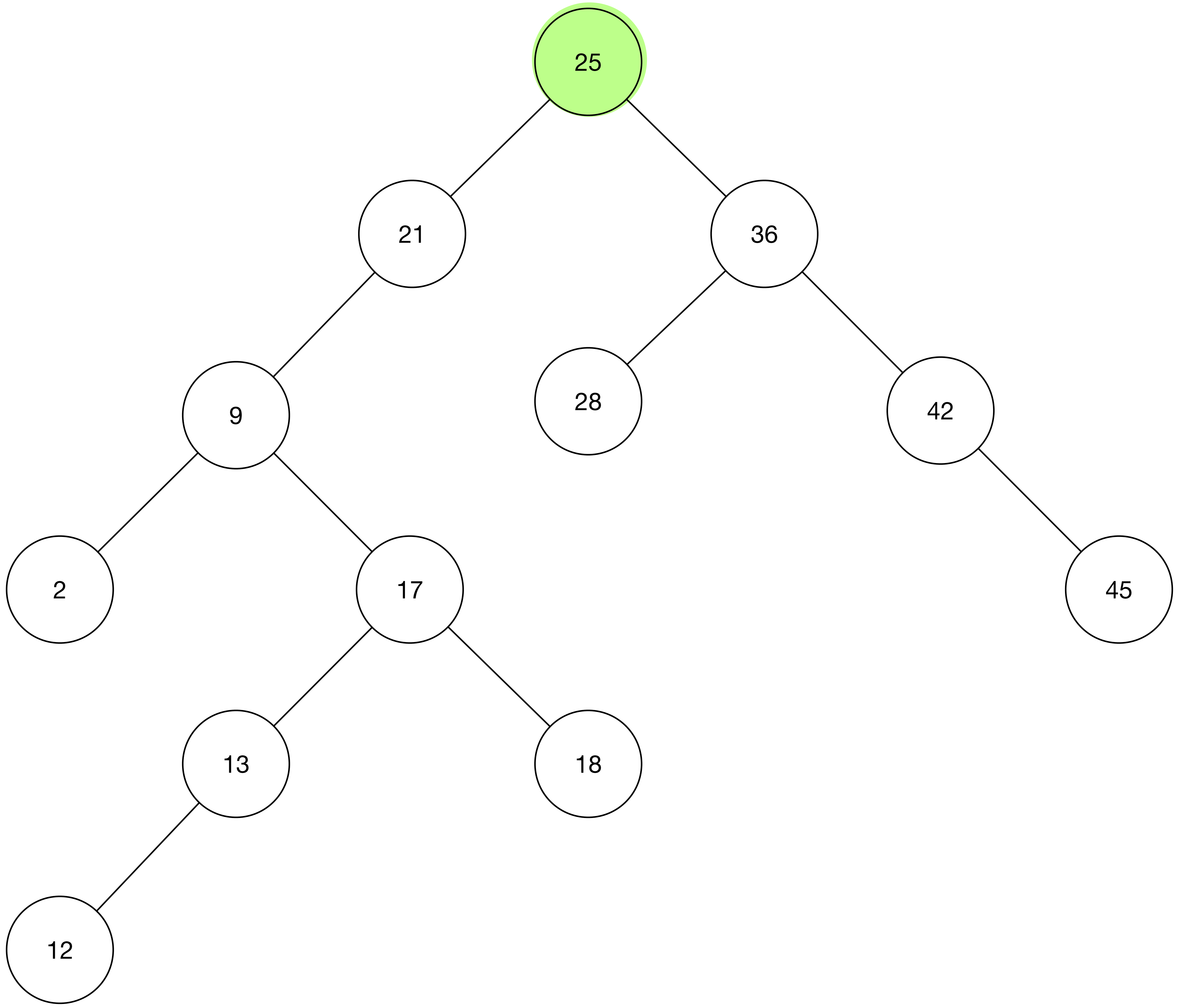
Searching a binary search tree

37 (not in tree)



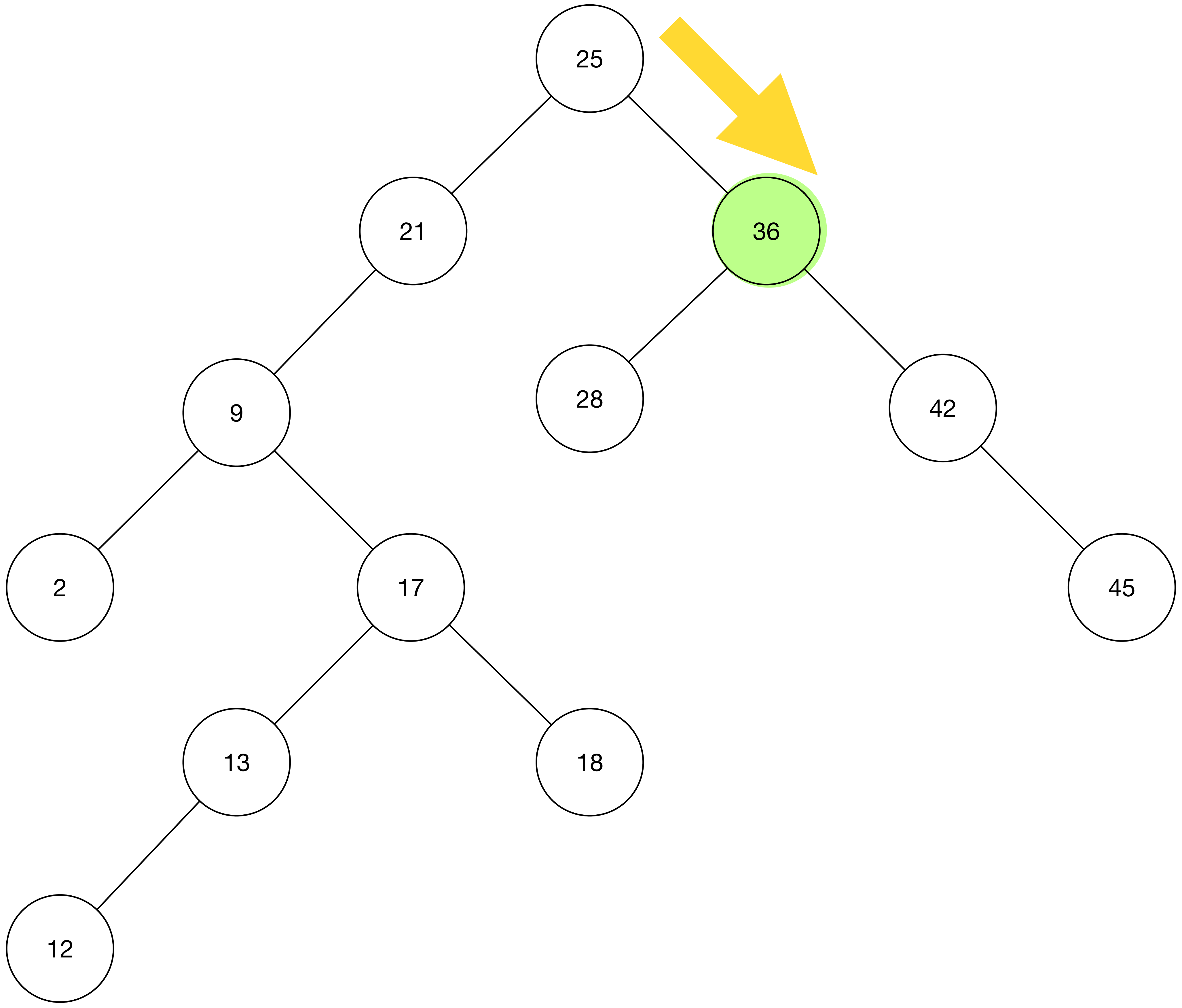
Searching a binary search tree

37 (not in tree)
25? 37 > 25. Go right.



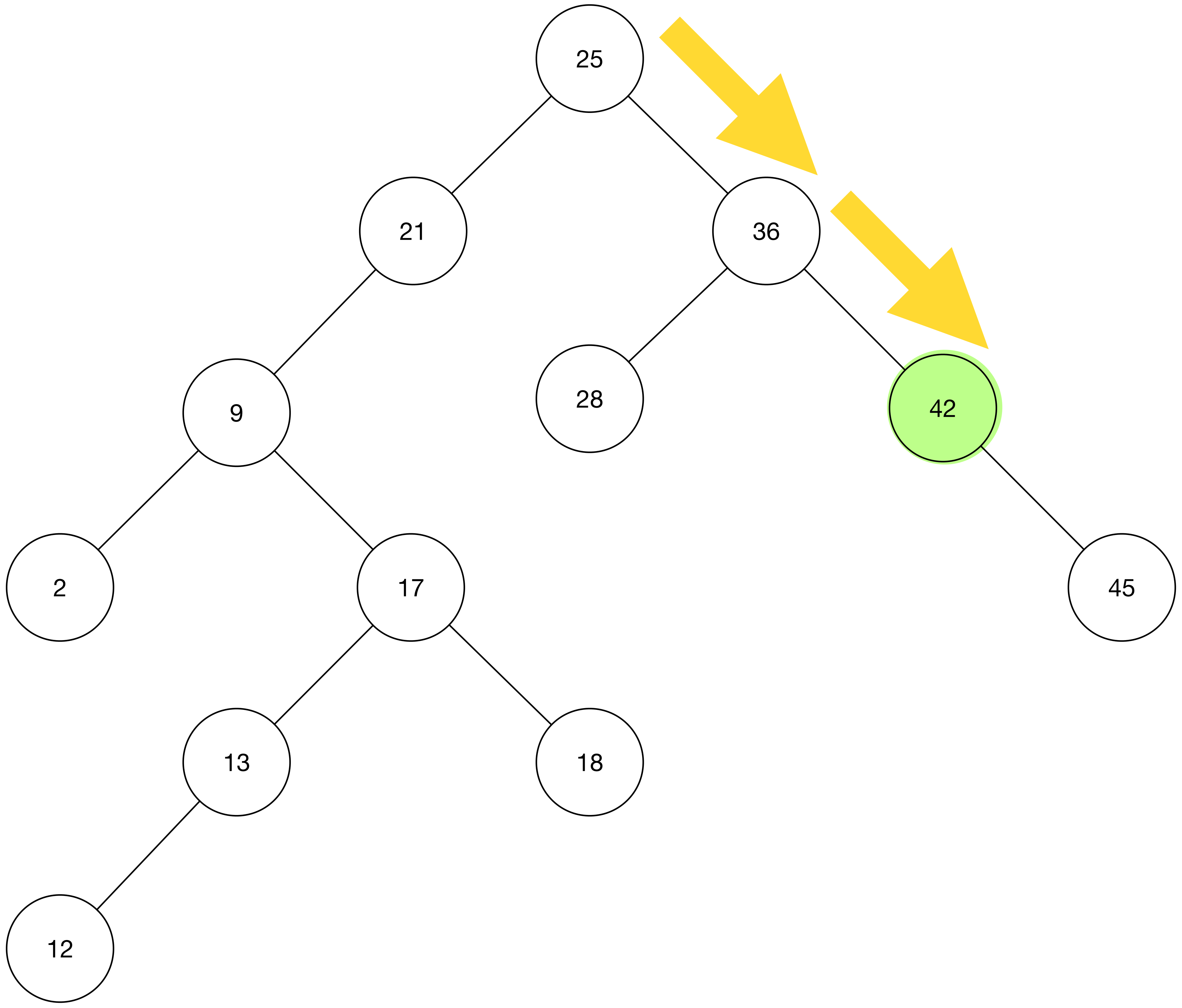
Searching a binary search tree

37 (not in tree)
25? 37 > 25. Go right.
36? 37 > 36. Go right.



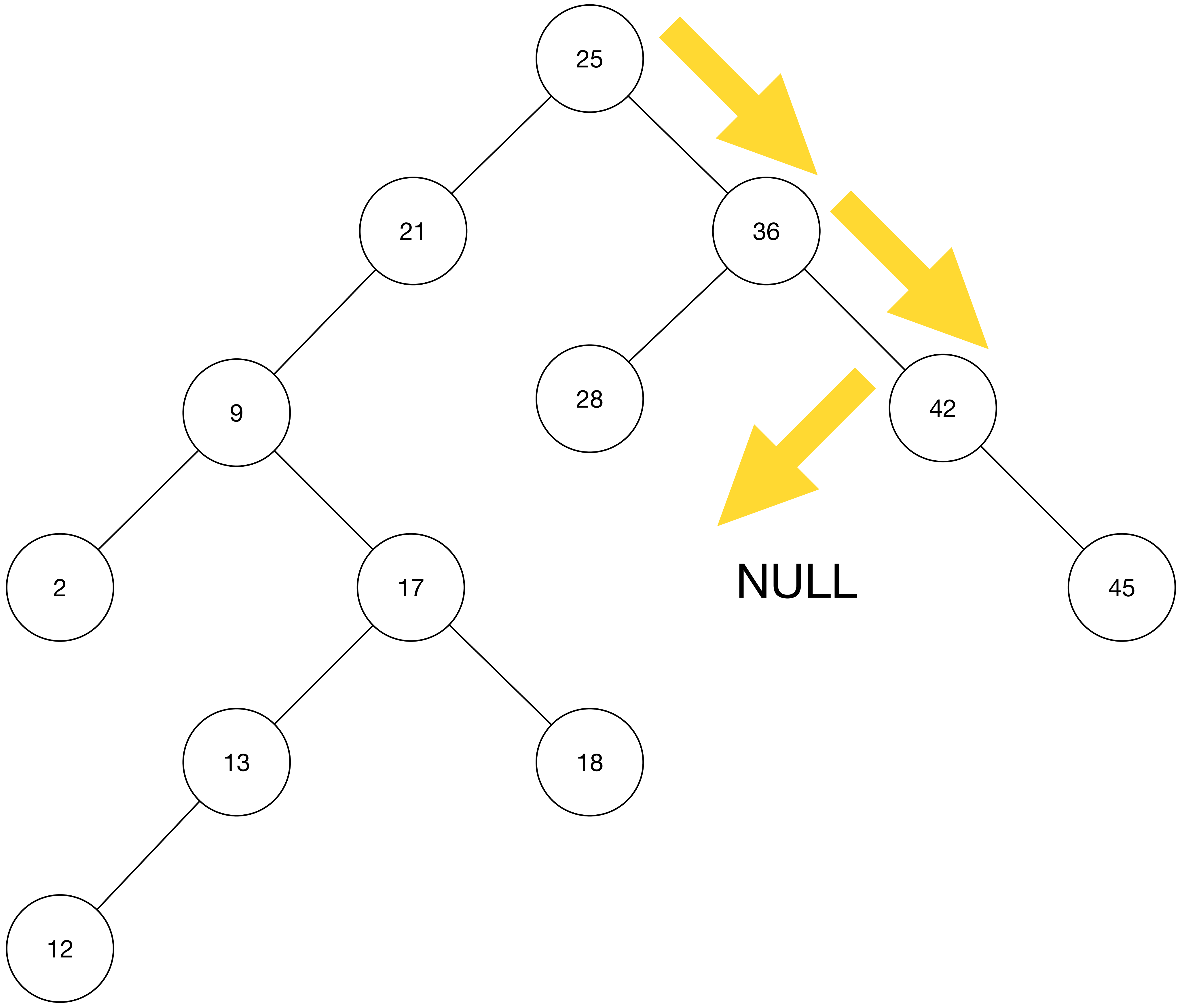
Searching a binary search tree

37 (not in tree)
25? 37 > 25. Go right.
36? 37 > 36. Go right.
42? 37 < 42. Go left.

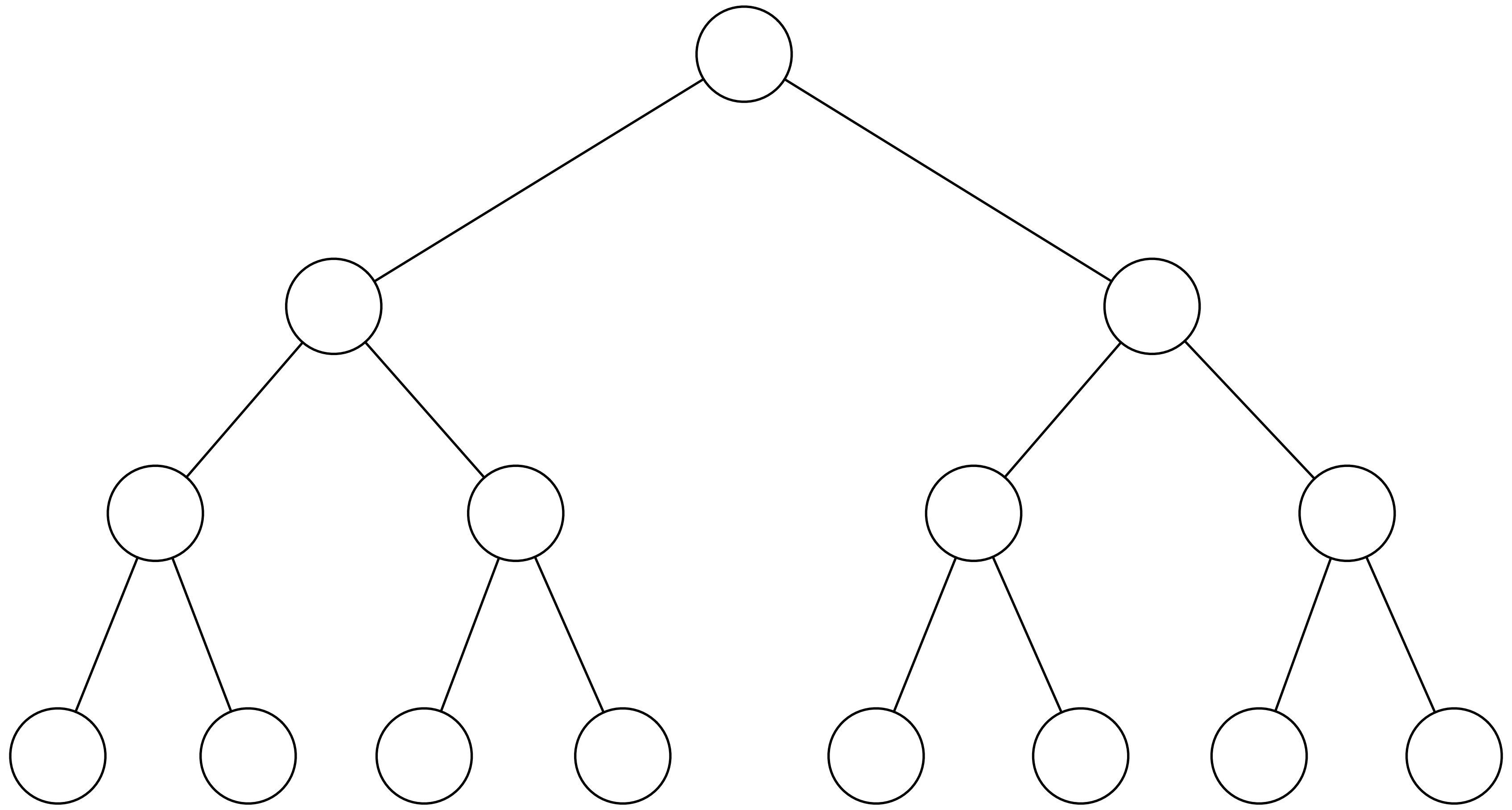


Searching a binary search tree

37 (not in tree)
25? 37 > 25. Go right.
36? 37 > 36. Go right.
42? 37 < 42. Go left.
42 has no left child!
NOT FOUND.

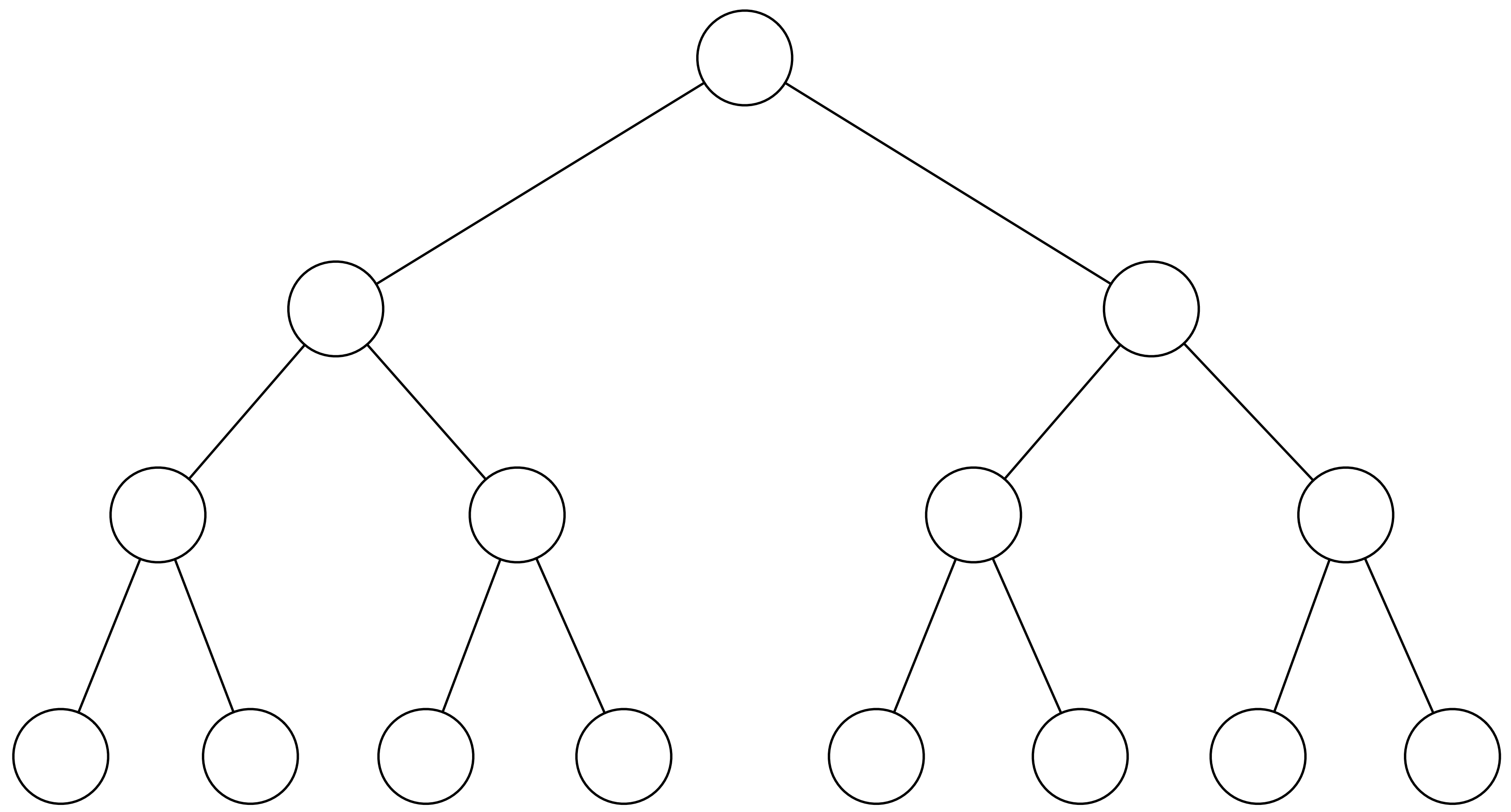


Complexity of search

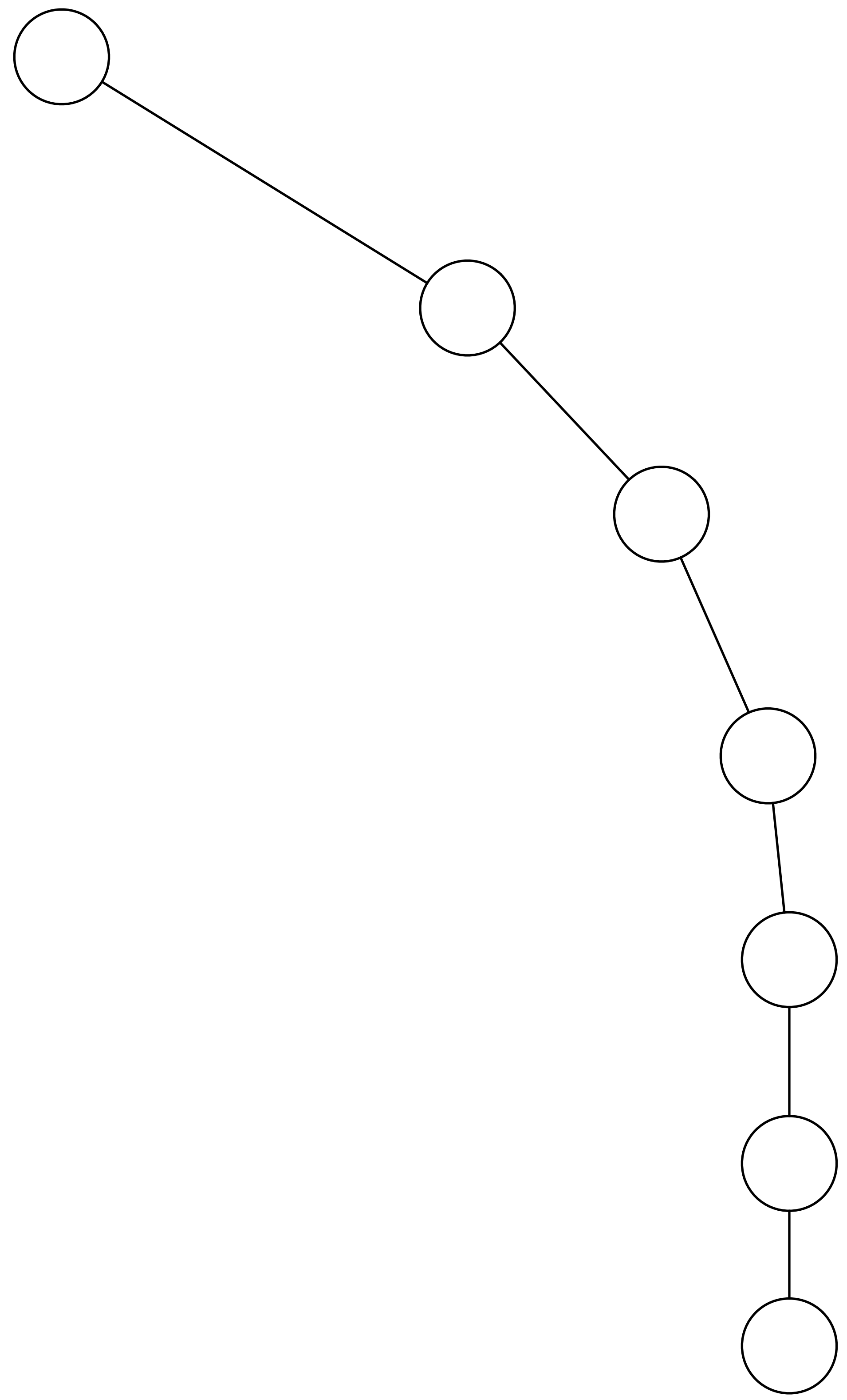


Complete or perfect tree?
$O(\log N)$

Complexity of search



Complete or perfect tree?
$O(\log N)$



Pathological tree?
$O(h) = O(N - 1) = O(N)$

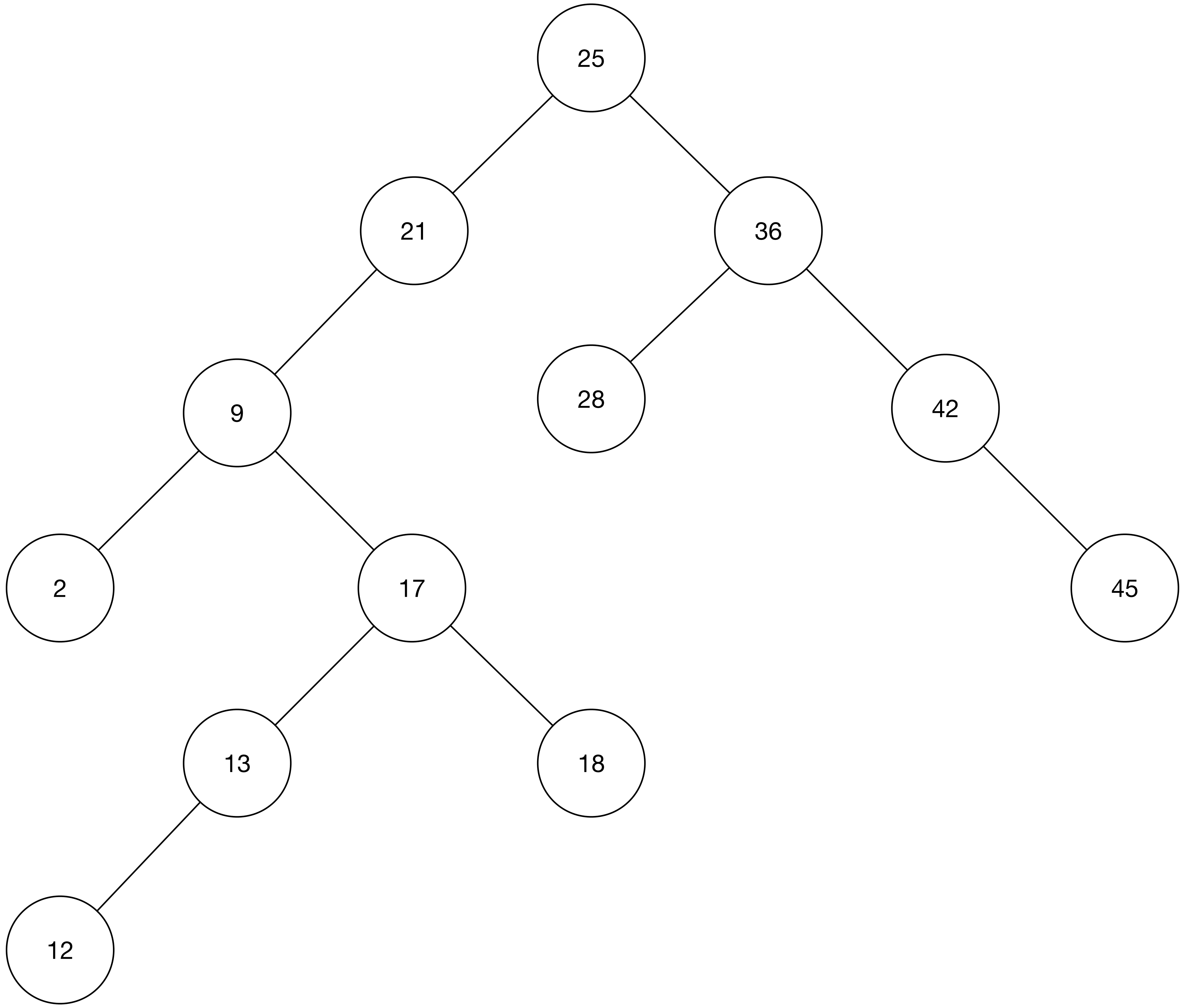
Deleting nodes in a BST

Can get a wee bit tricky / four cases

- Target node is a leaf. Delete the node.
- Target node has one child. Delete the node and replace it with its child.
- Target node has two children:
  - Target node's left child has no right child. Delete the node and replace it with its left child.
  - Target node's left child has a right child. From the target node's left child's right child, continue to probe down through the tree, following right children until you can proceed no further. Replace the target node with the node found by probing. If the node found by probing has a left child, replace that node with its left child.

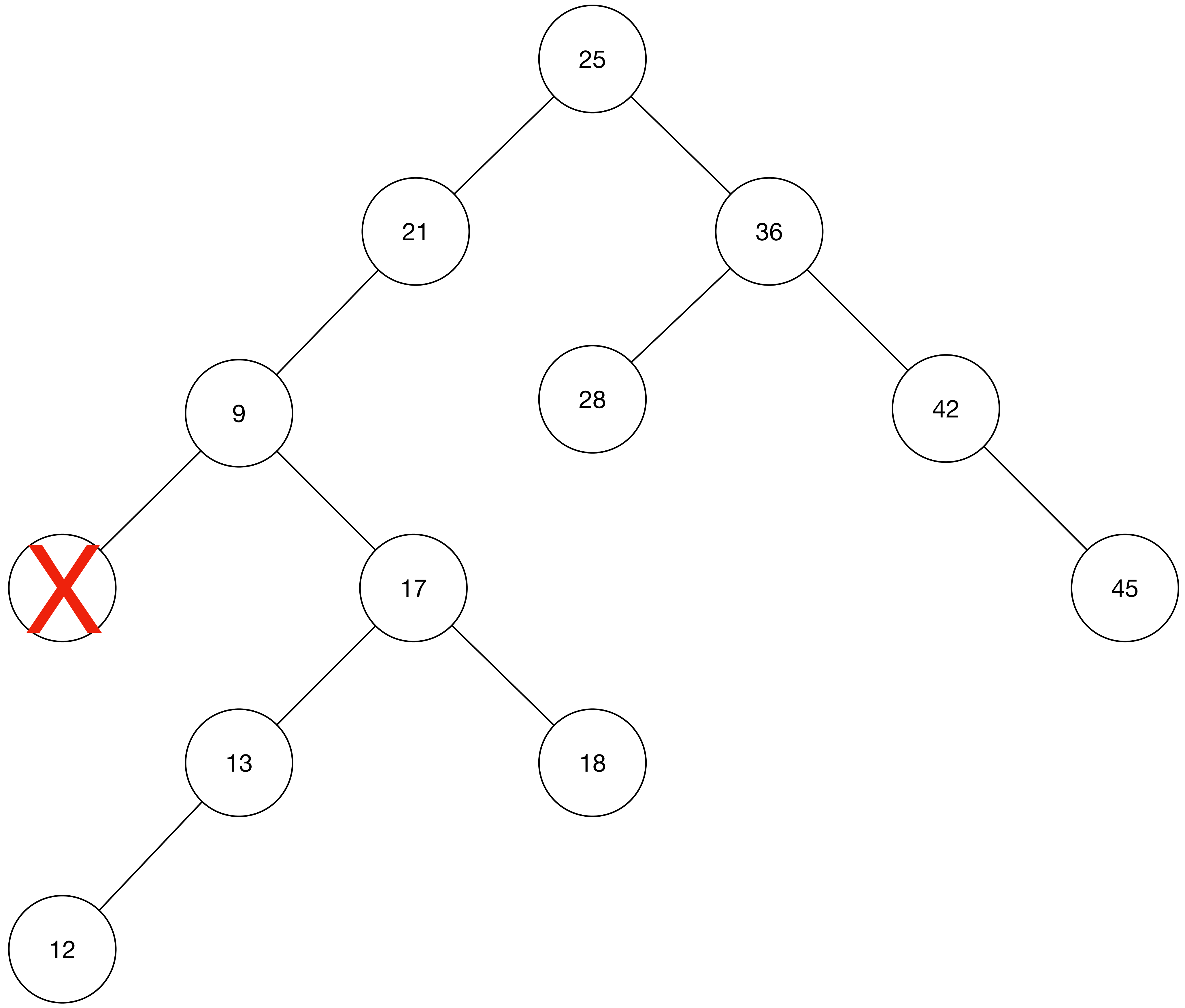
Deleting nodes in a BST

Case 1: Target node is a leaf.



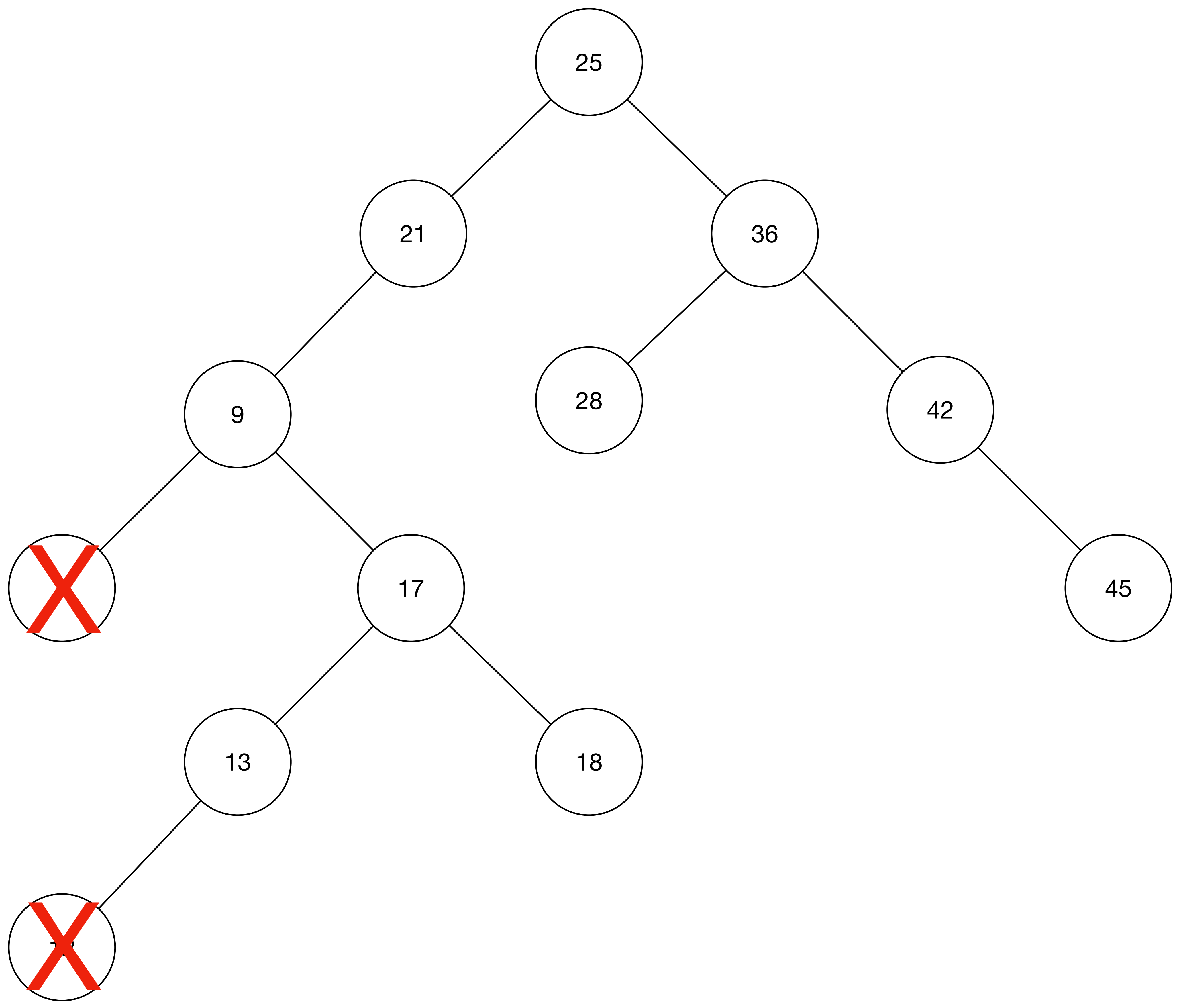
Deleting nodes in a BST

Case 1: Target node is a leaf.



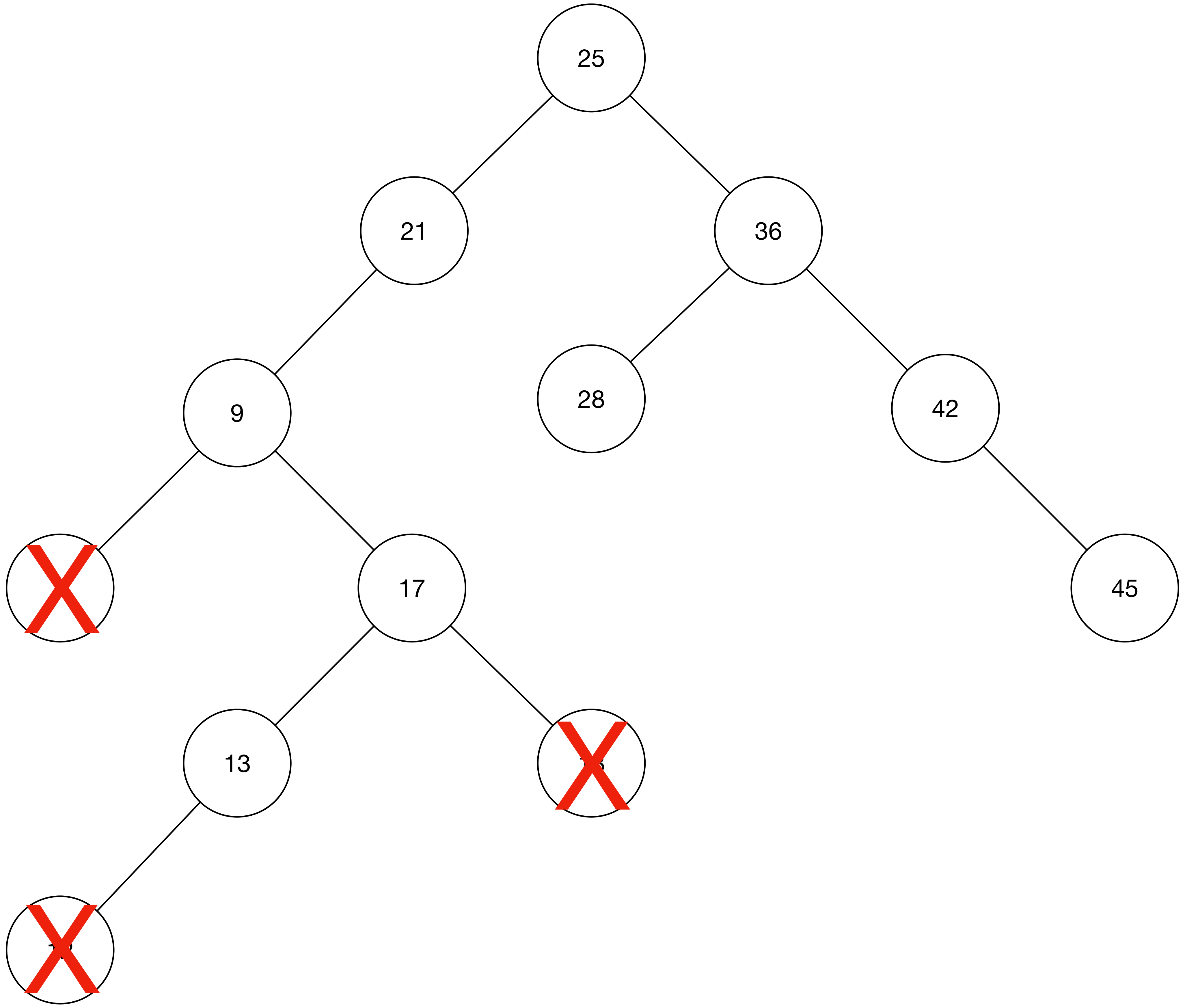
Deleting nodes in a BST

Case 1: Target node is a leaf.



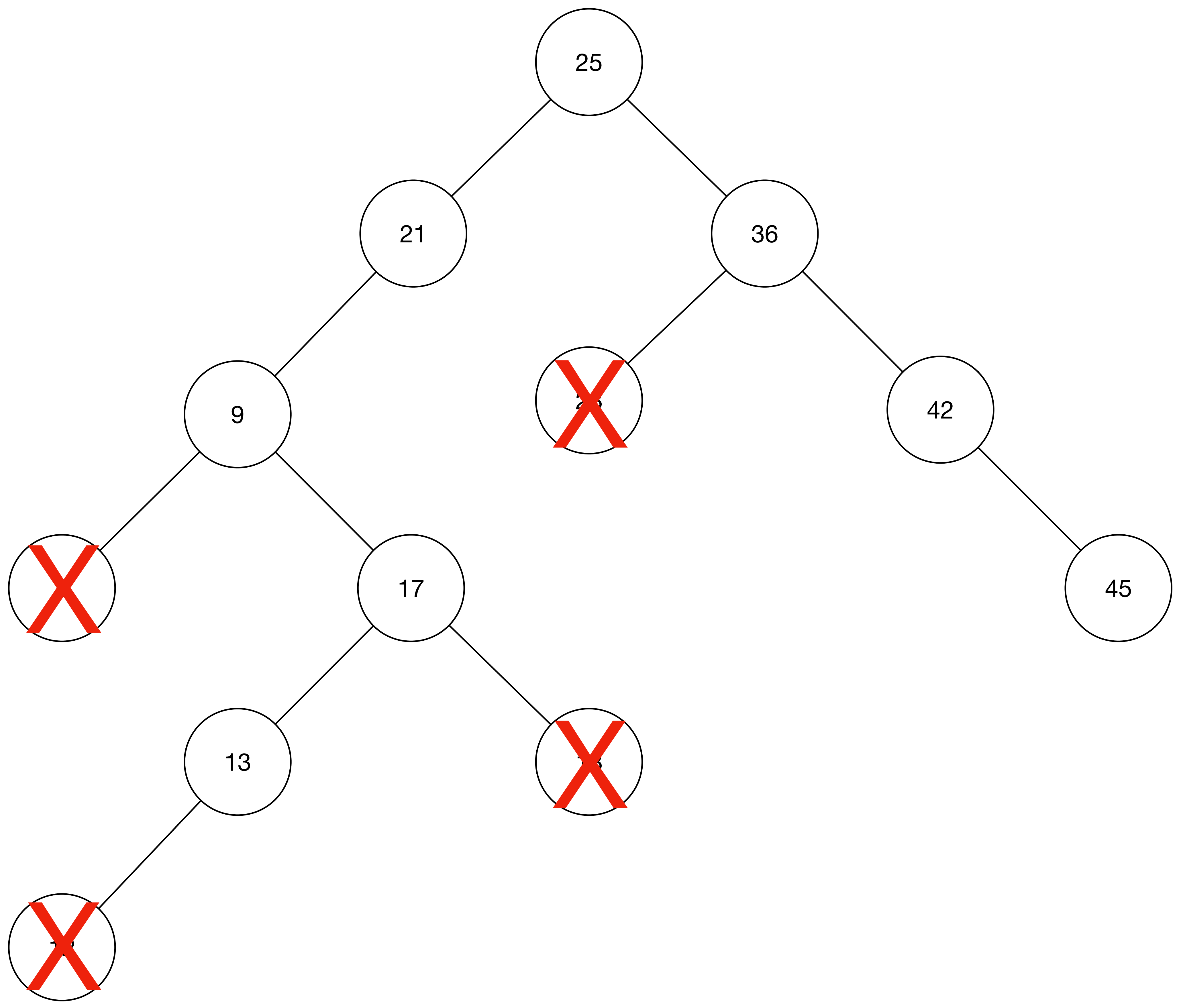
Deleting nodes in a BST

Case 1: Target node is a leaf.



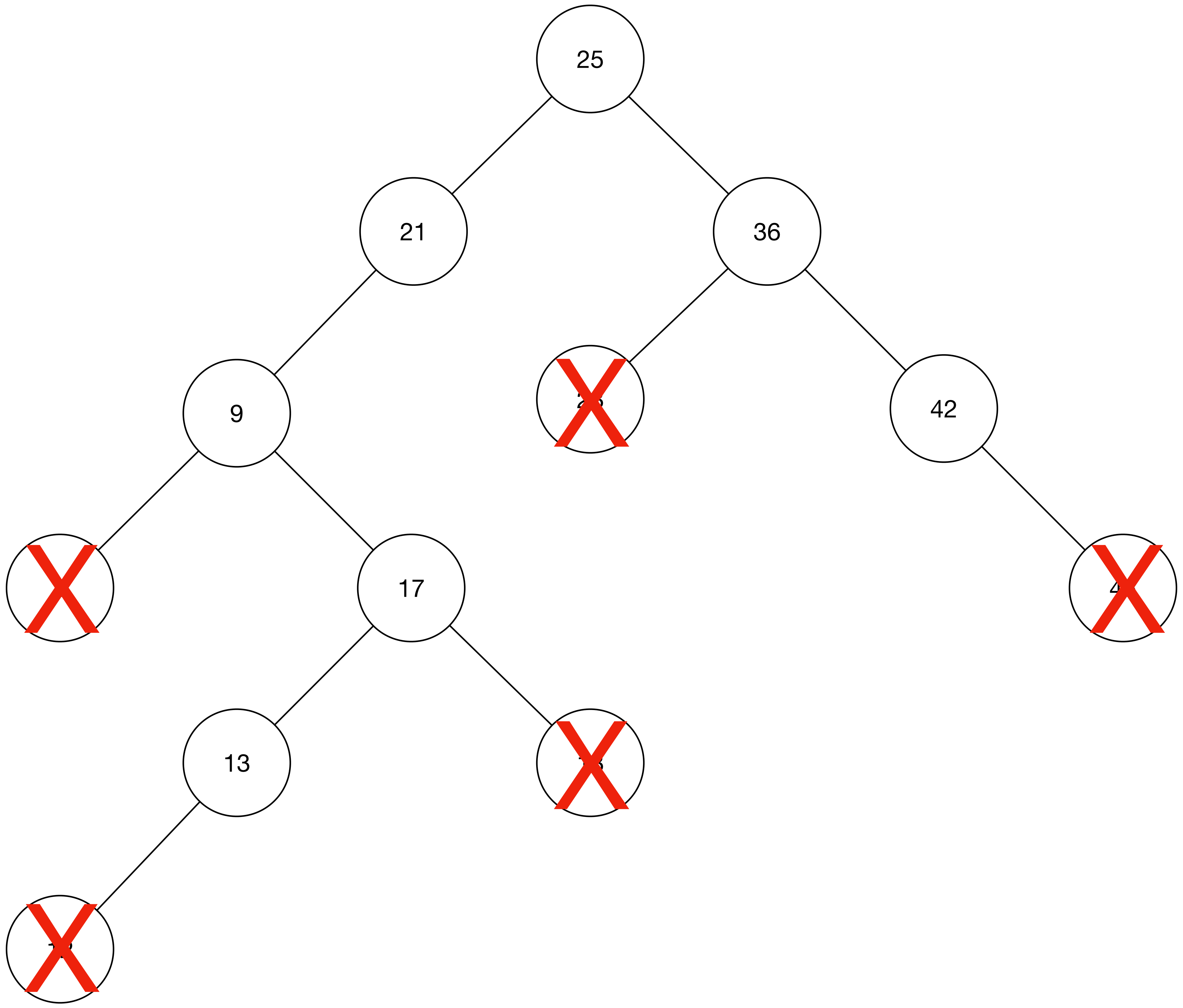
Deleting nodes in a BST

Case 1: Target node is a leaf.



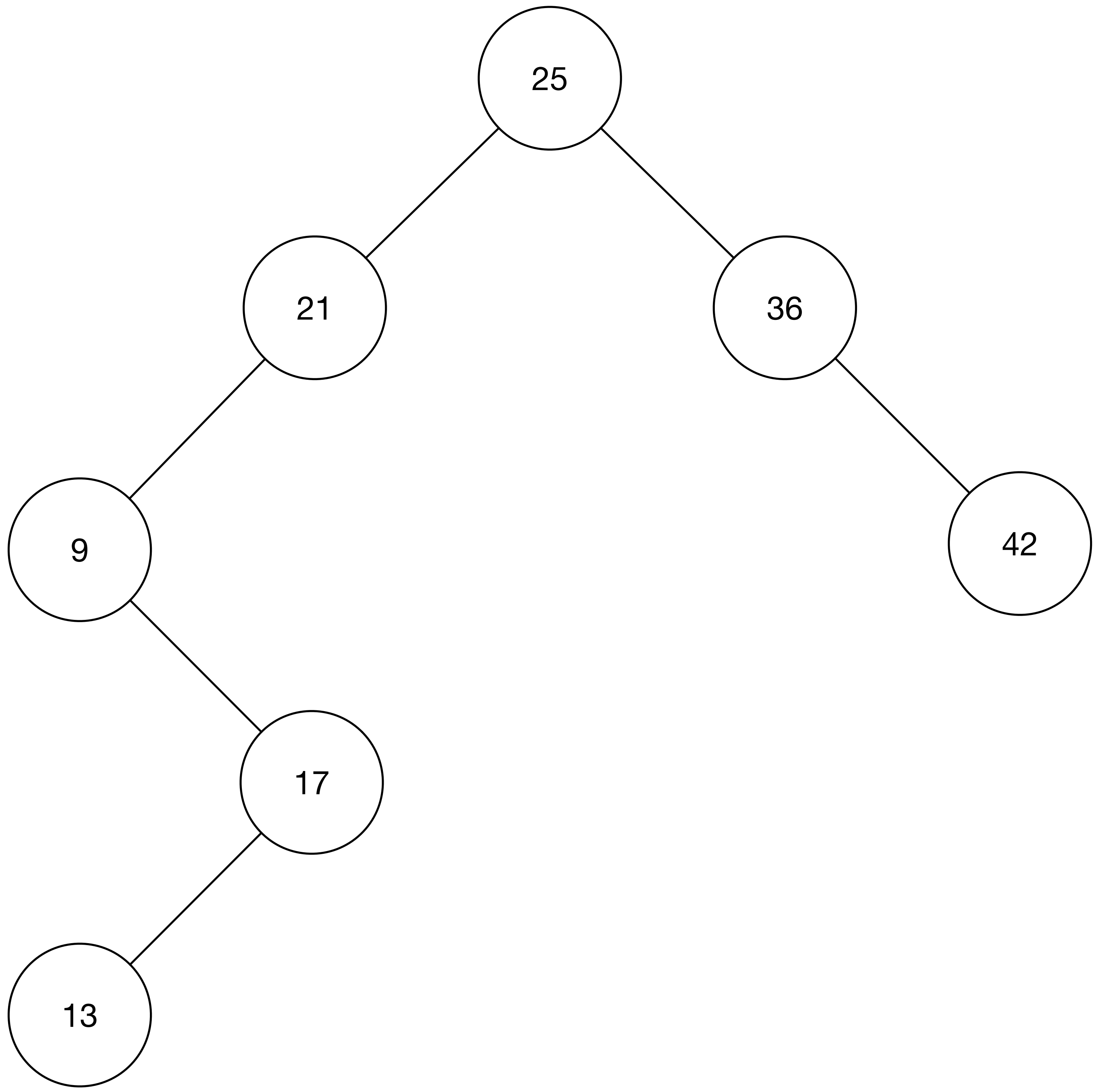
Deleting nodes in a BST

Case 1: Target node is a leaf.



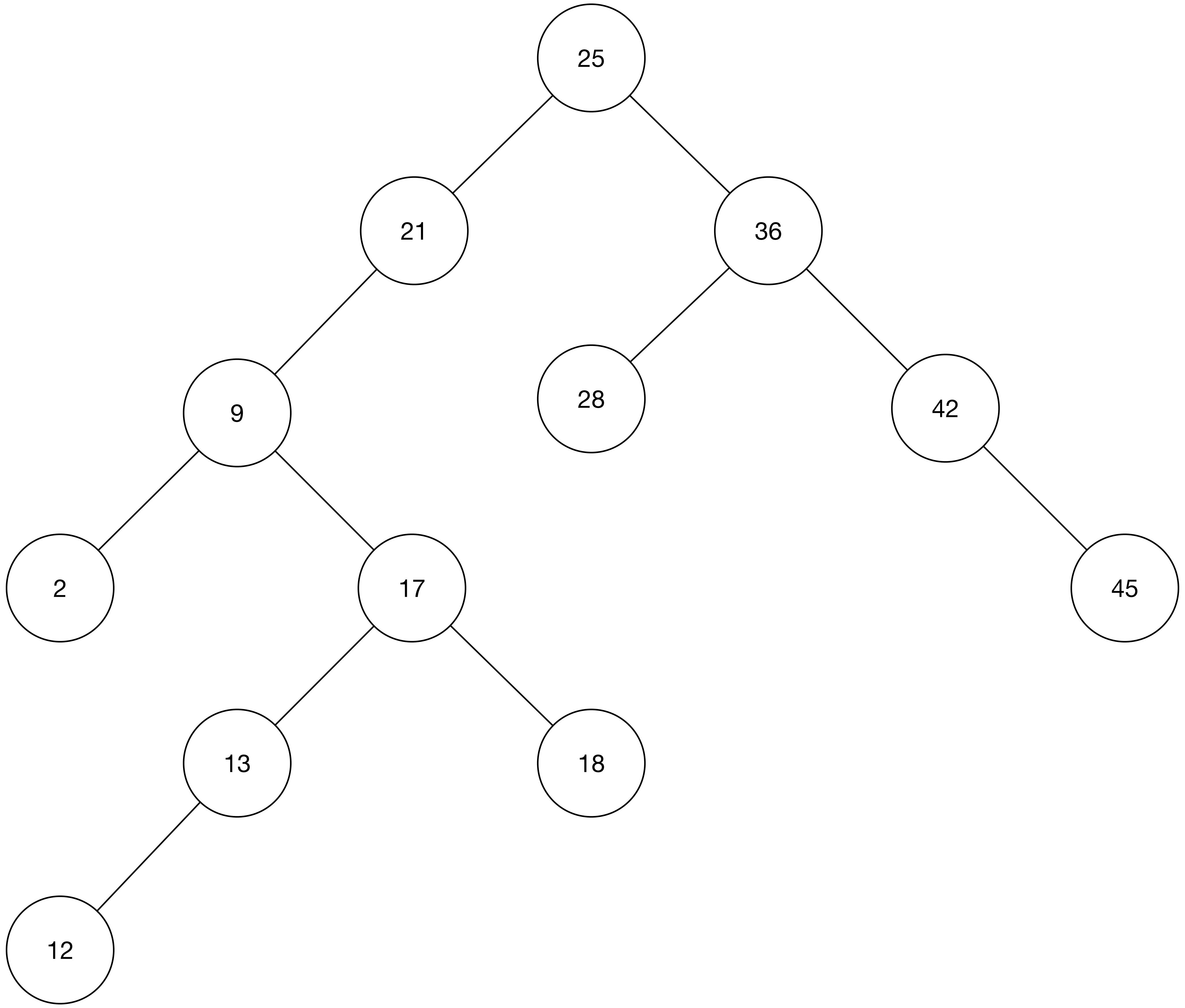
Deleting nodes in a BST

Case 1: Target node is a leaf.



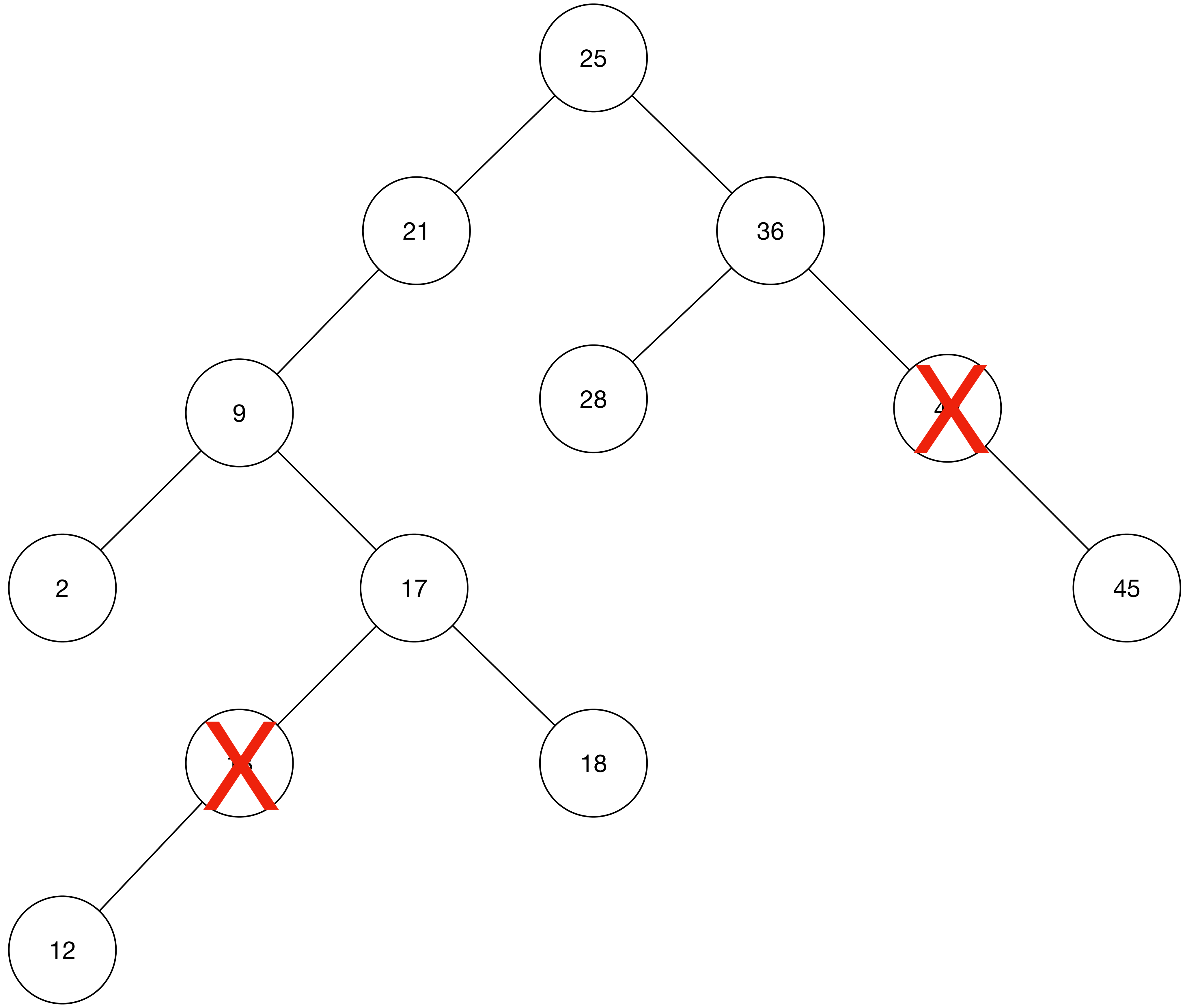
Deleting nodes in a BST

Case 2: Target node has one child.



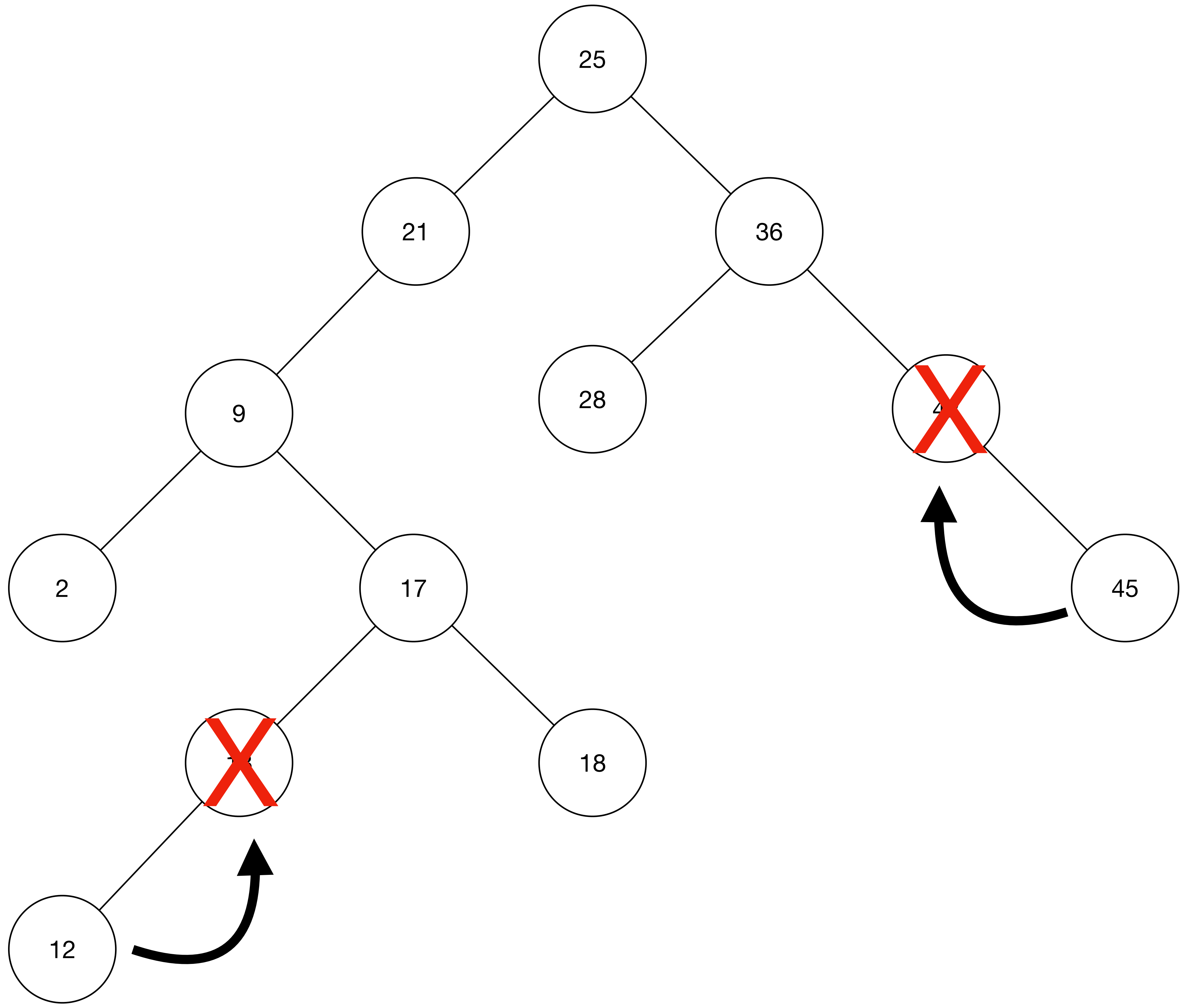
Deleting nodes in a BST

Case 2: Target node has one child.



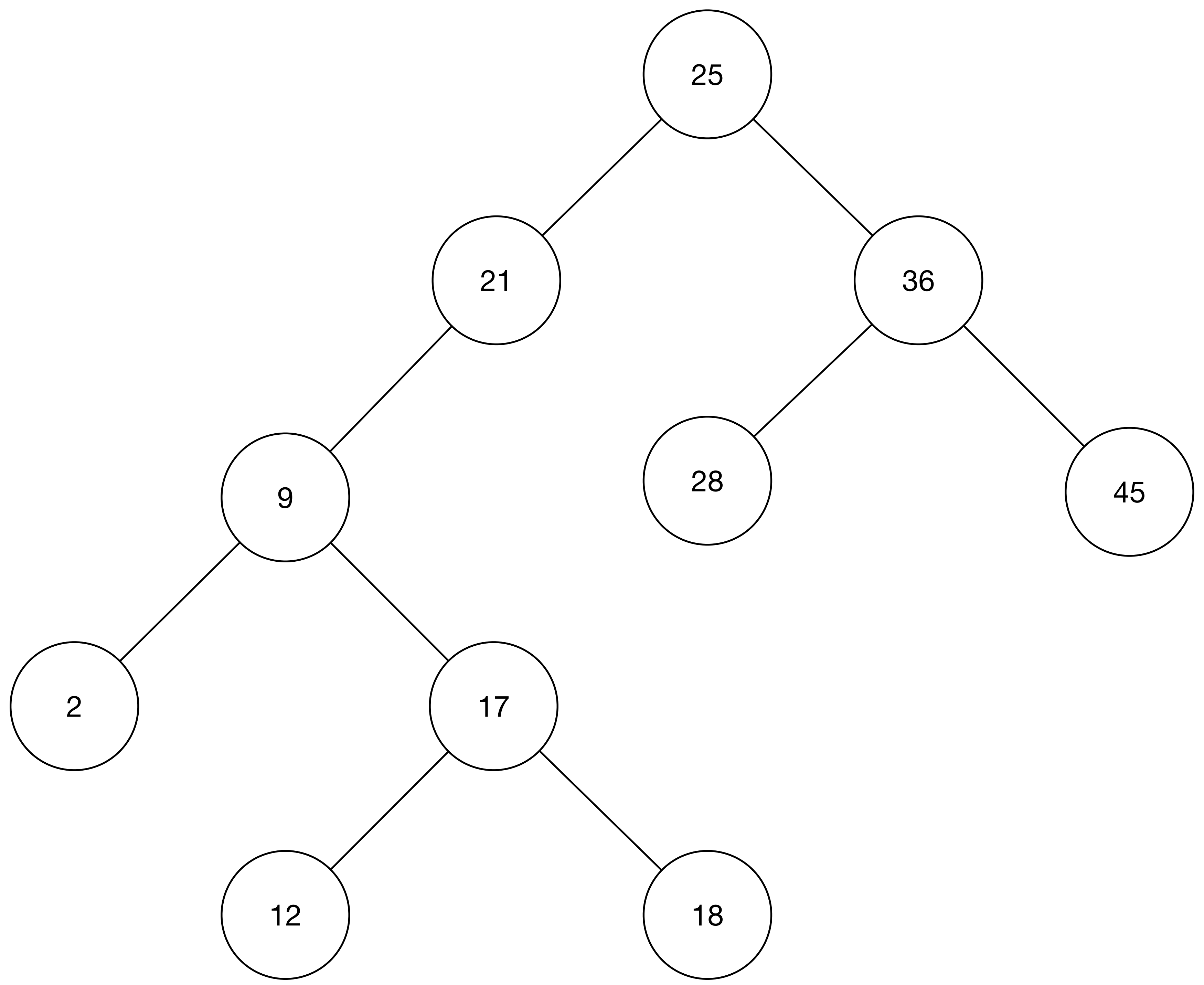
Deleting nodes in a BST

Case 2: Target node has one child.



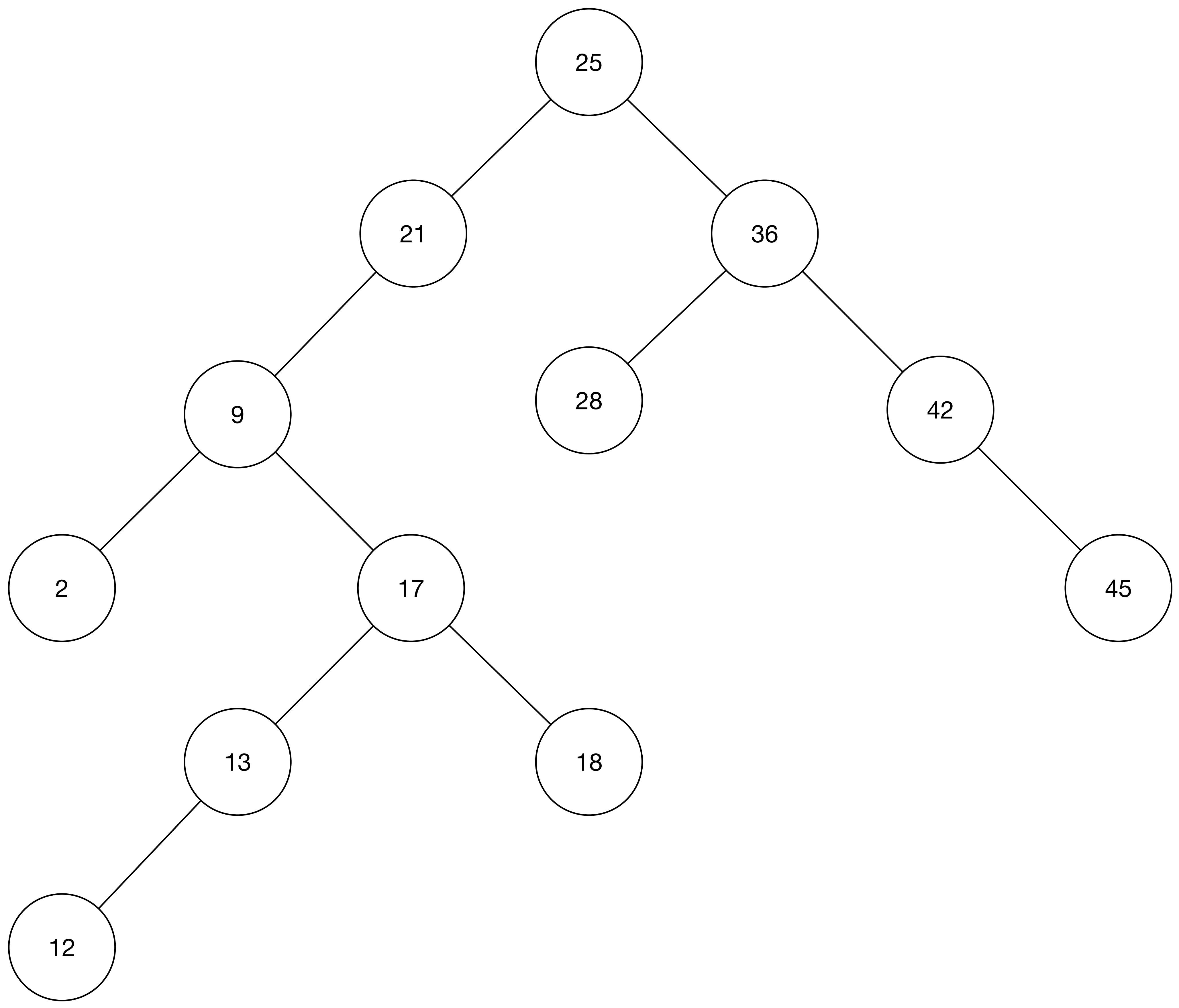
Deleting nodes in a BST

Case 2: Target node has one child.



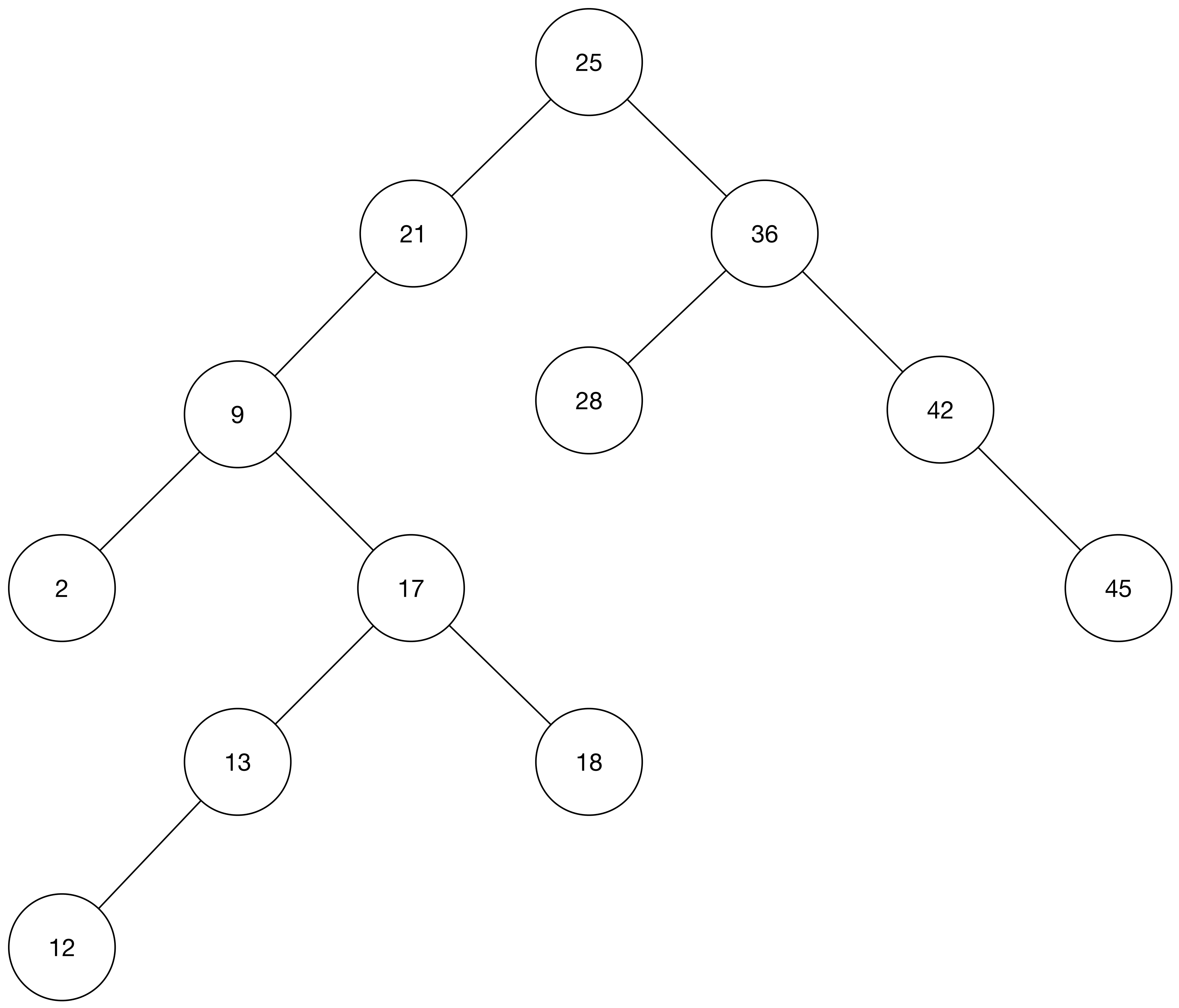
Deleting nodes in a BST

Cases 3 & 4: Target node has two children



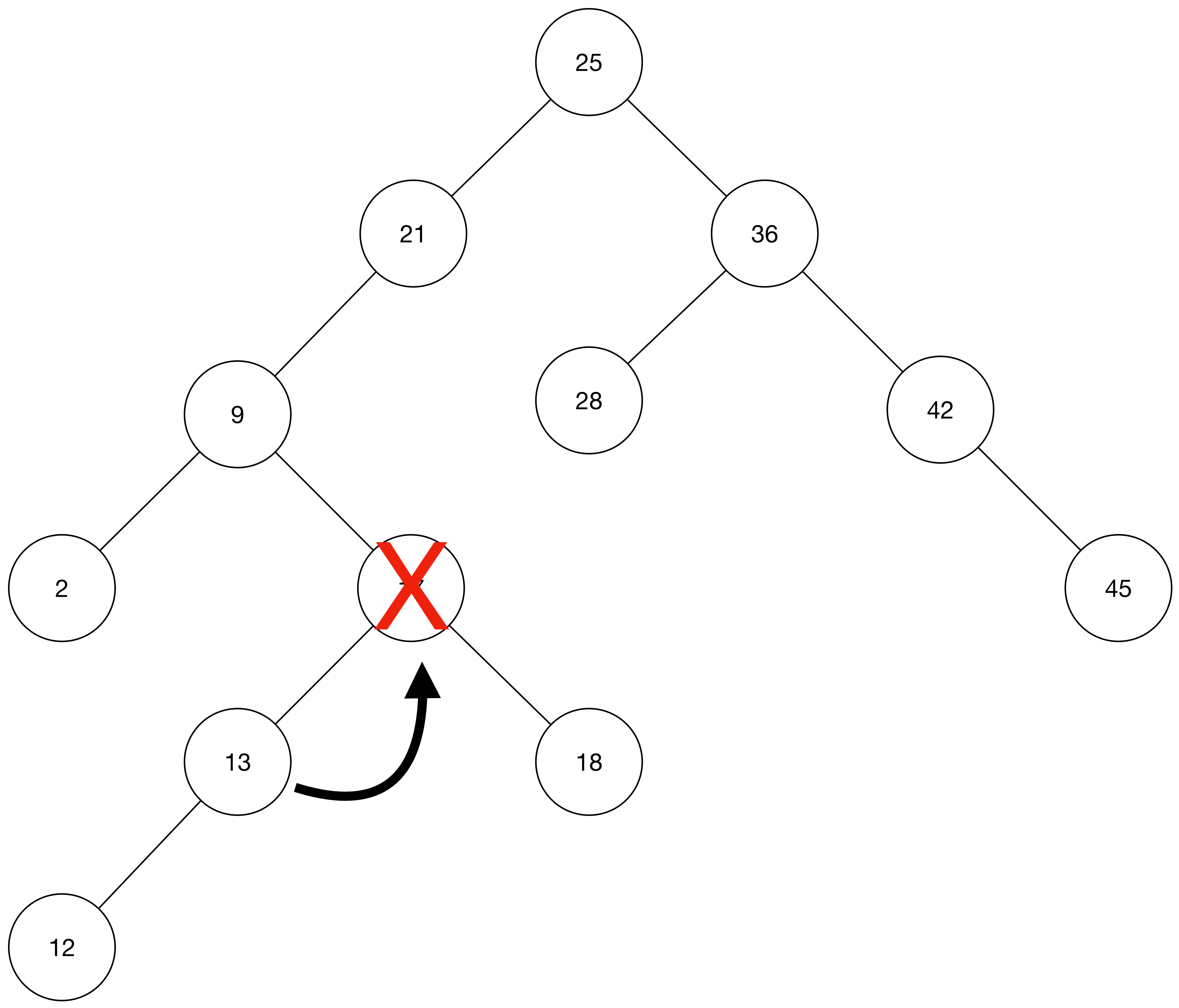
Deleting nodes in a BST

Case 3: Target node has two children, but left child has no right child.



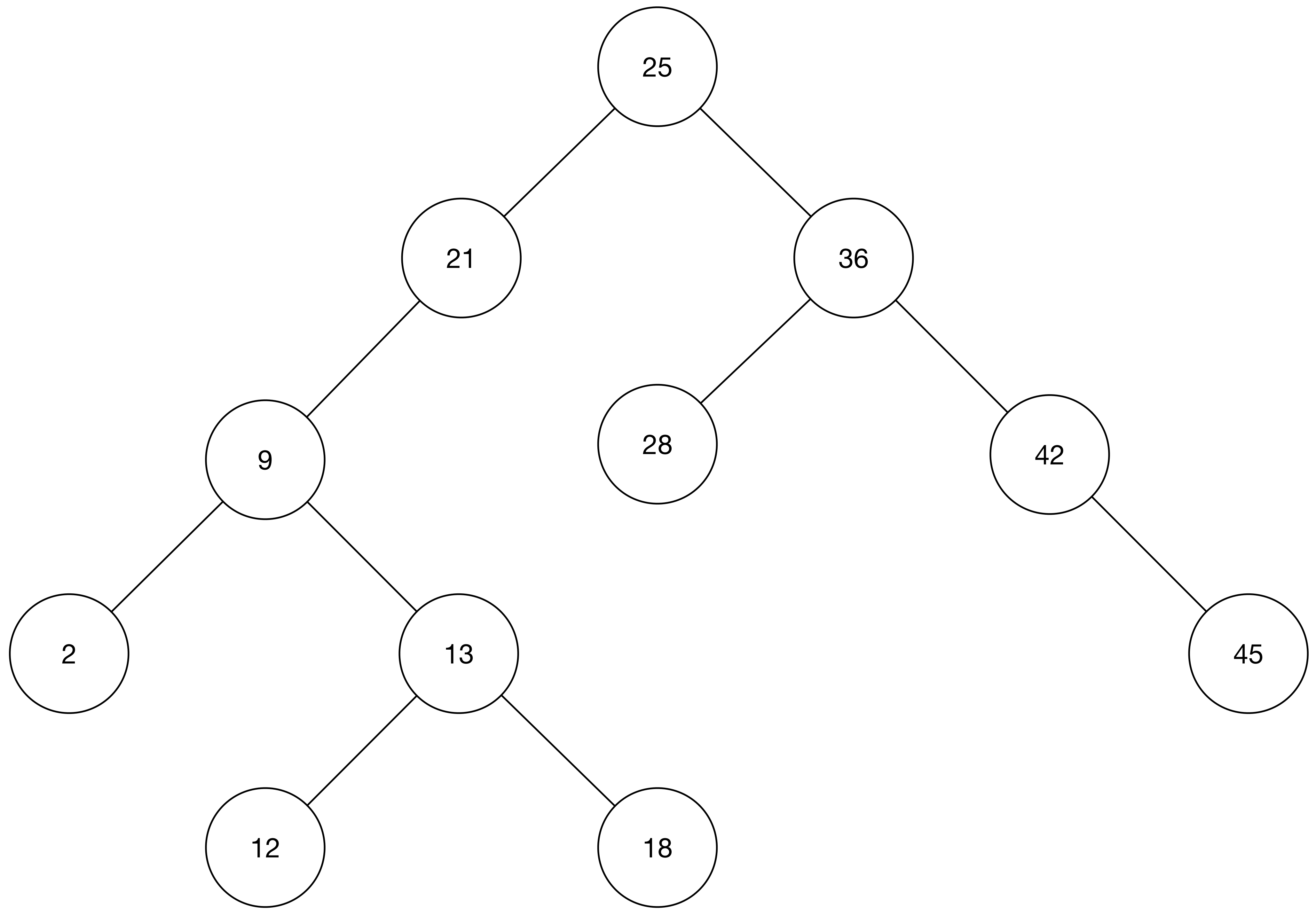
Deleting nodes in a BST

Case 3: Target node has two children, but left child has no right child.



Deleting nodes in a BST

Case 3: Target node has two children, but left child has no right child.

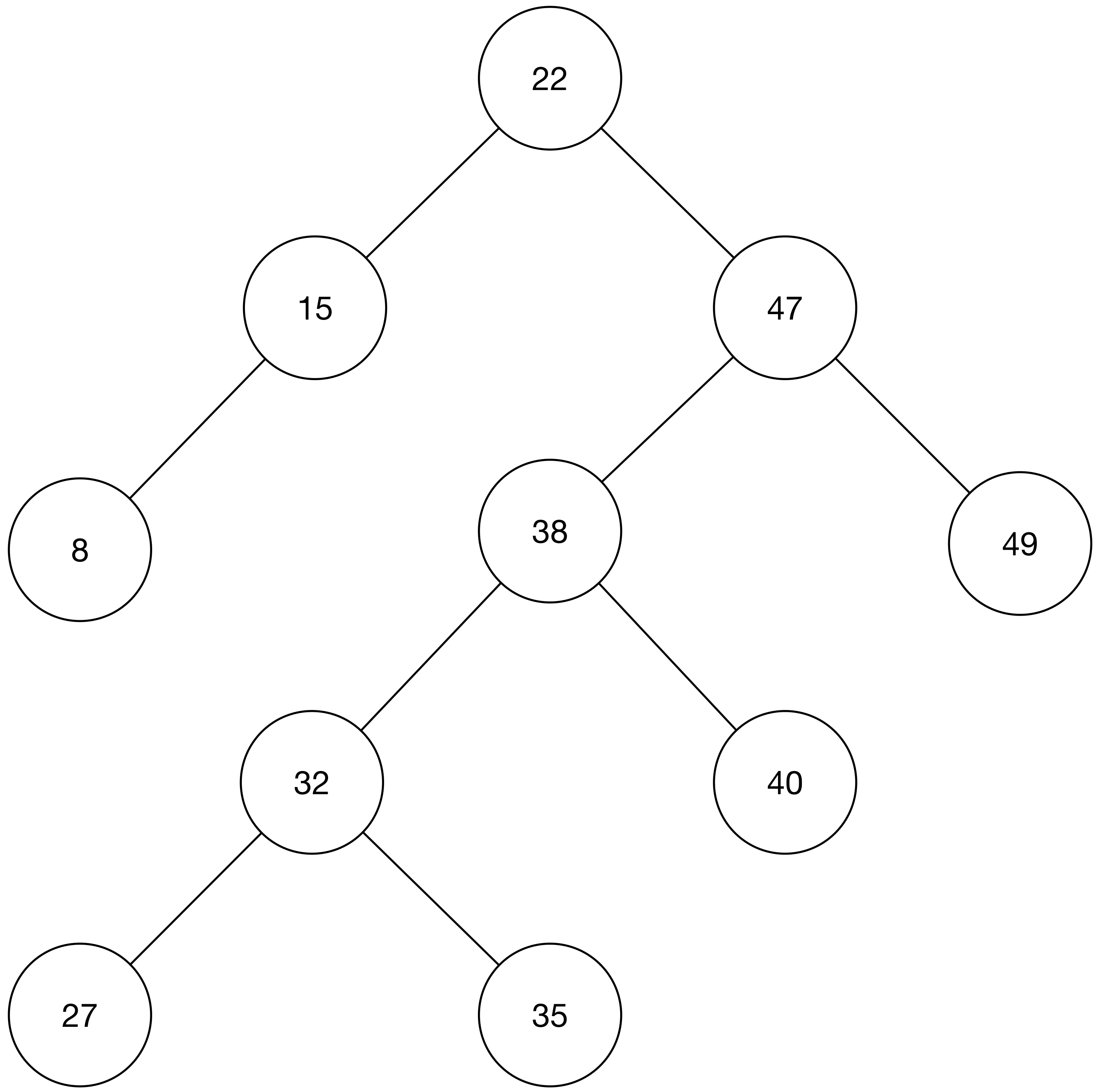


Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.

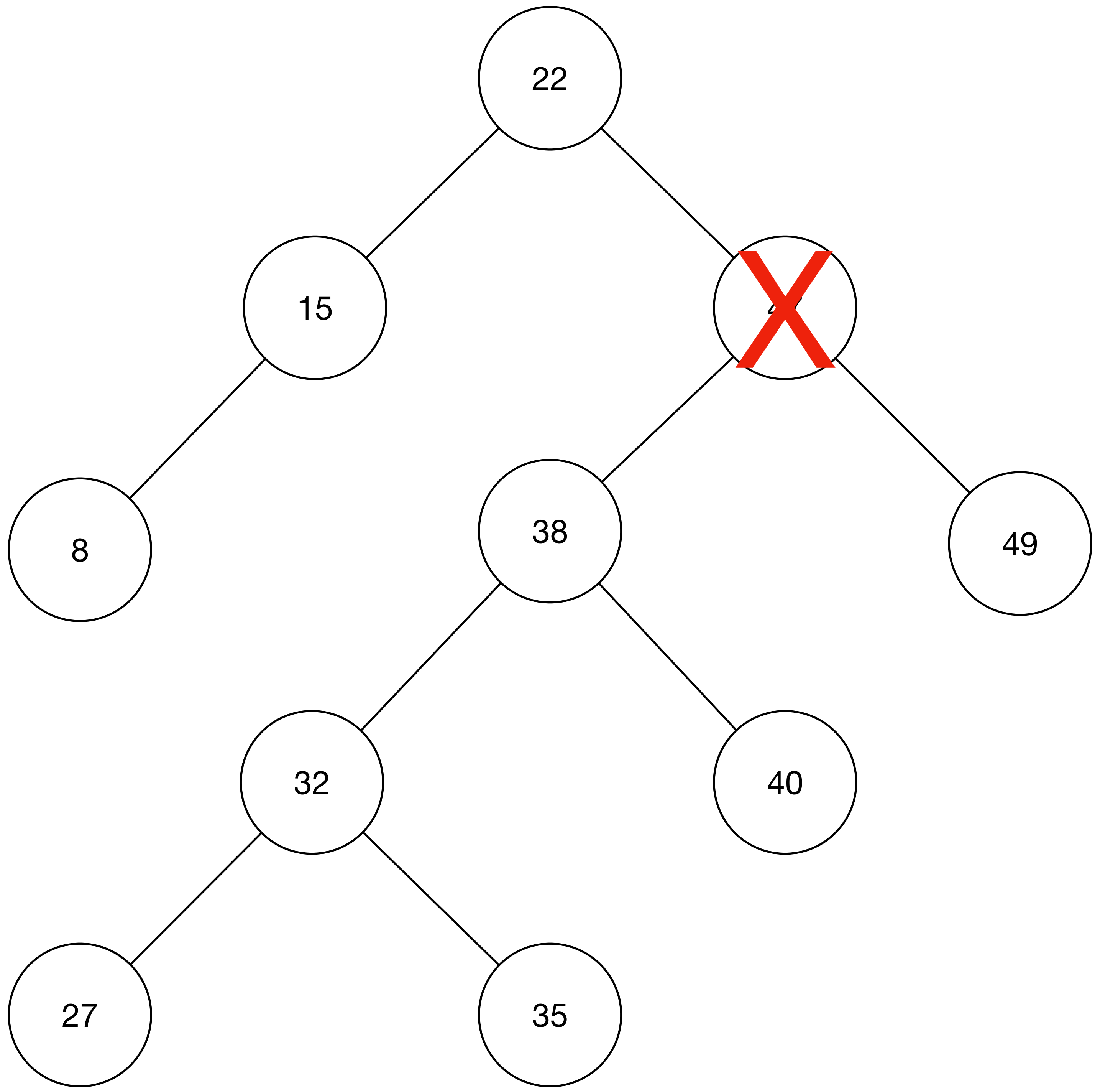
Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.



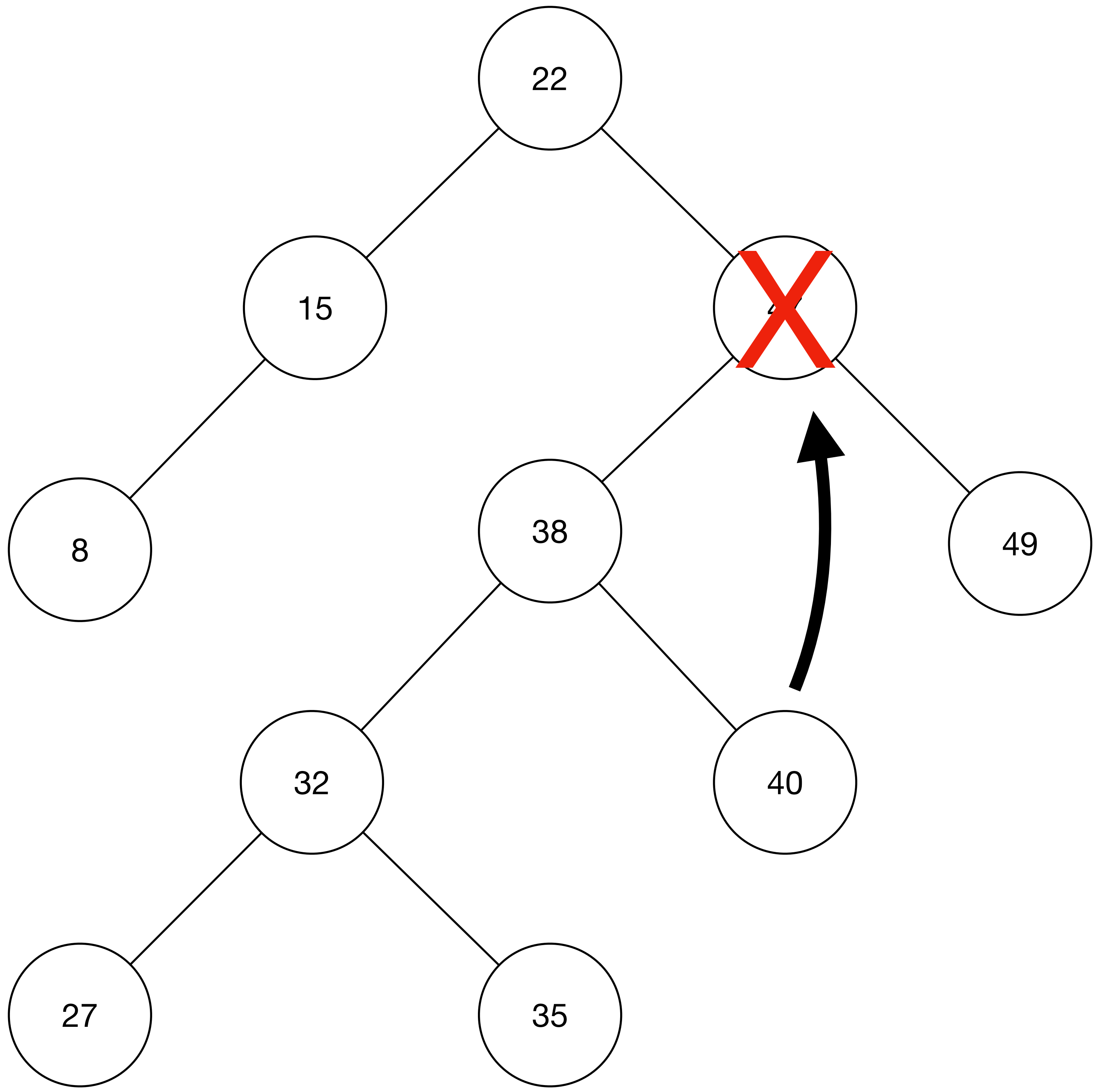
Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.



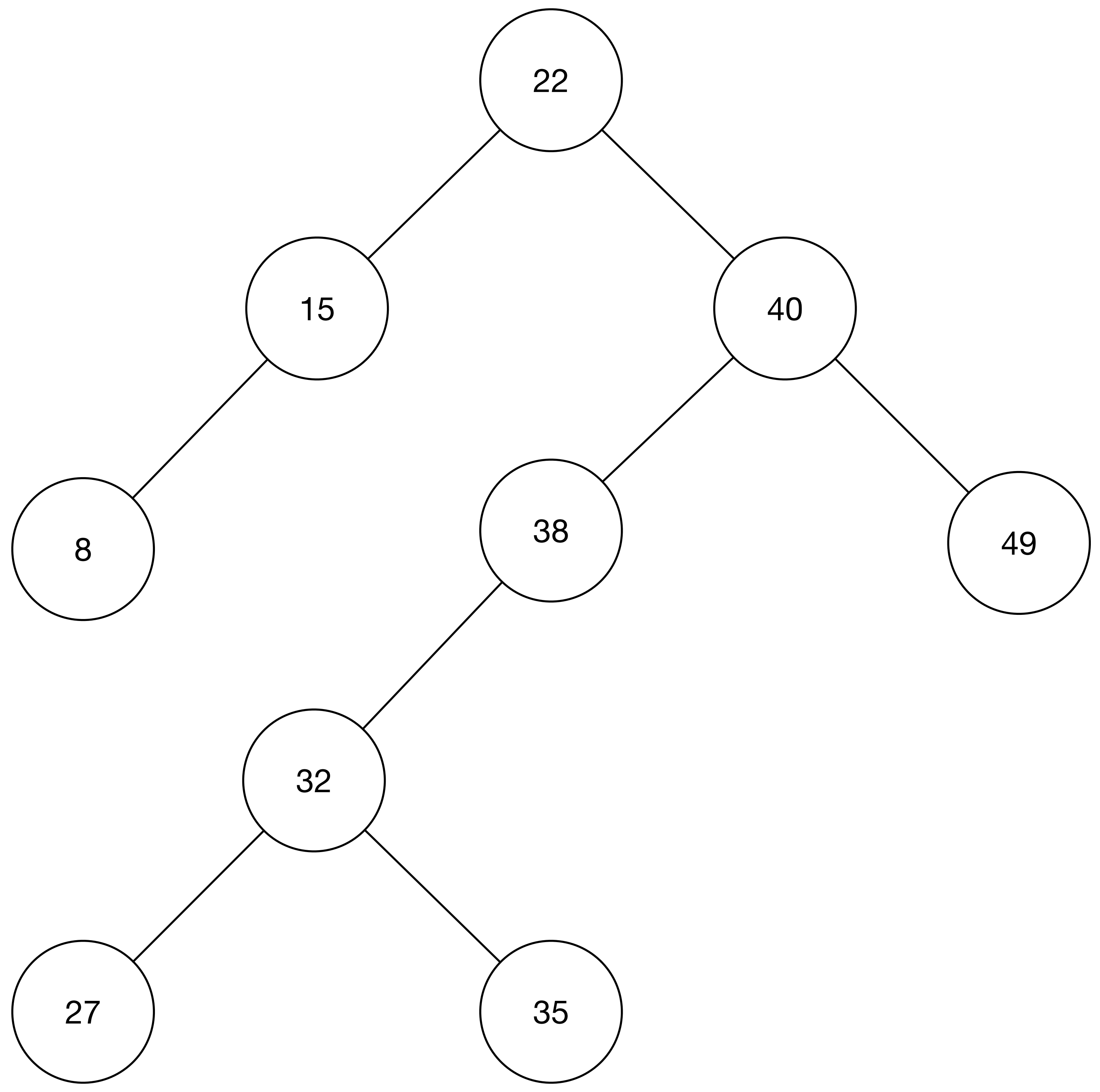
Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.



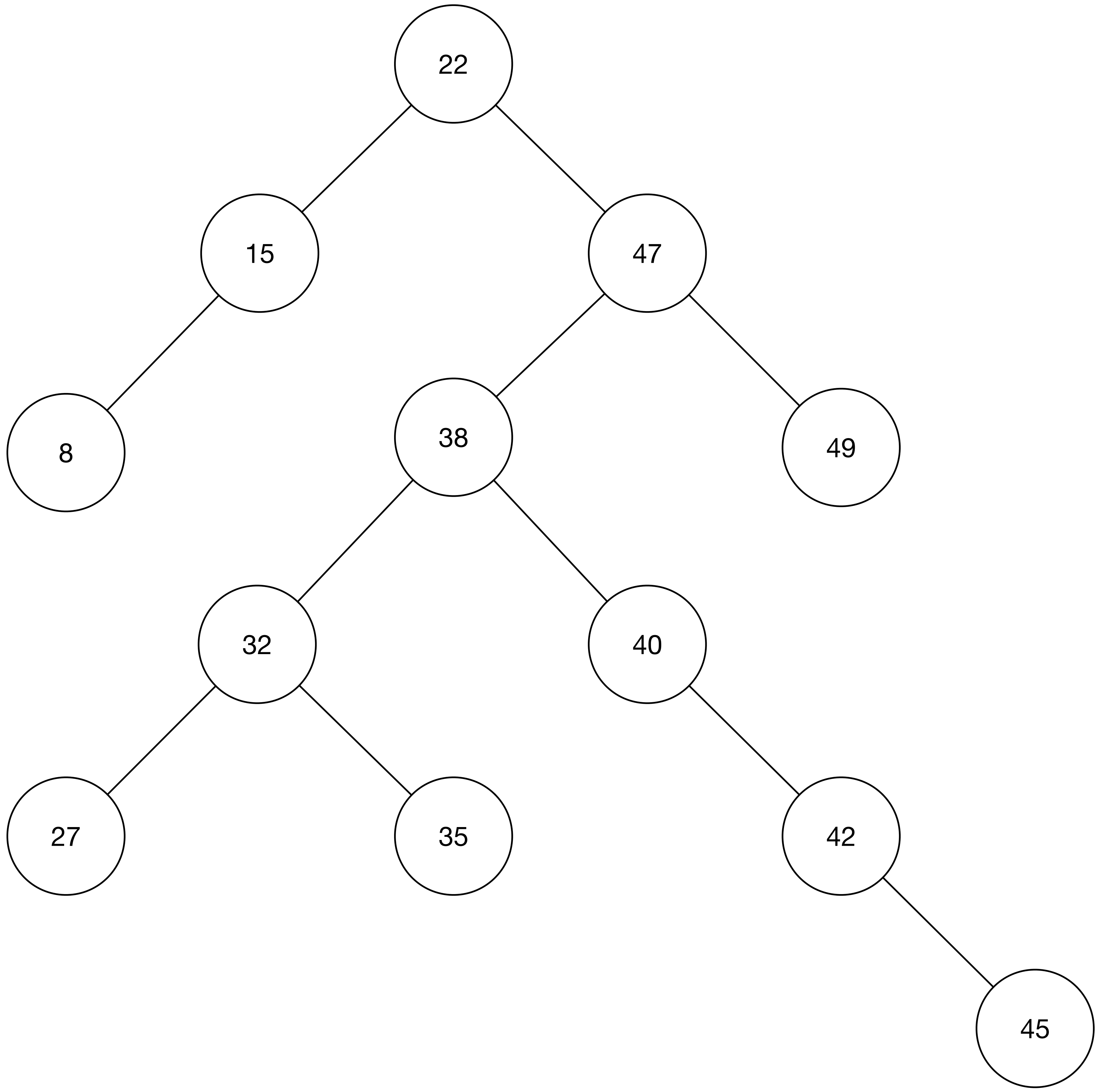
Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.



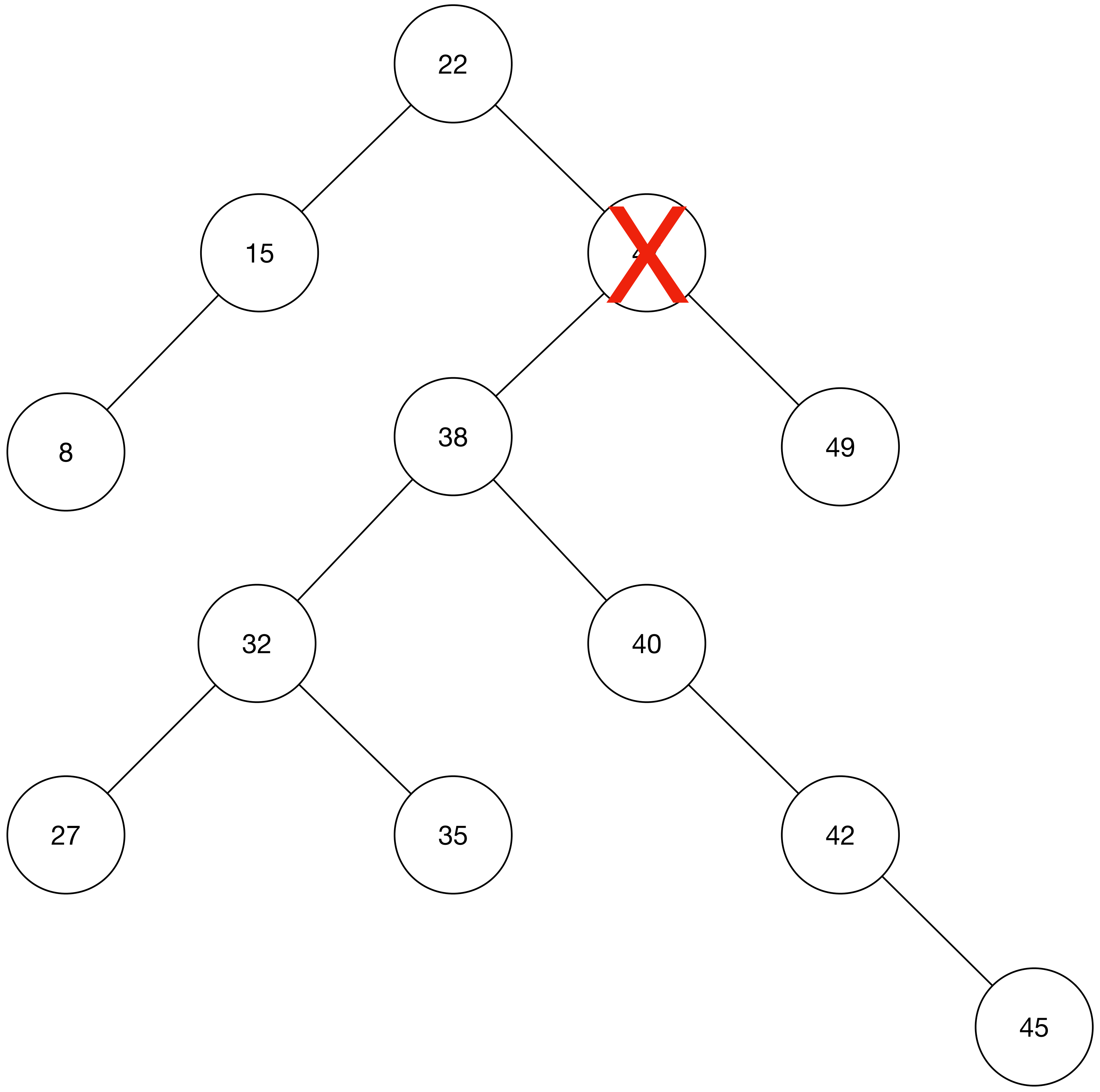
Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.



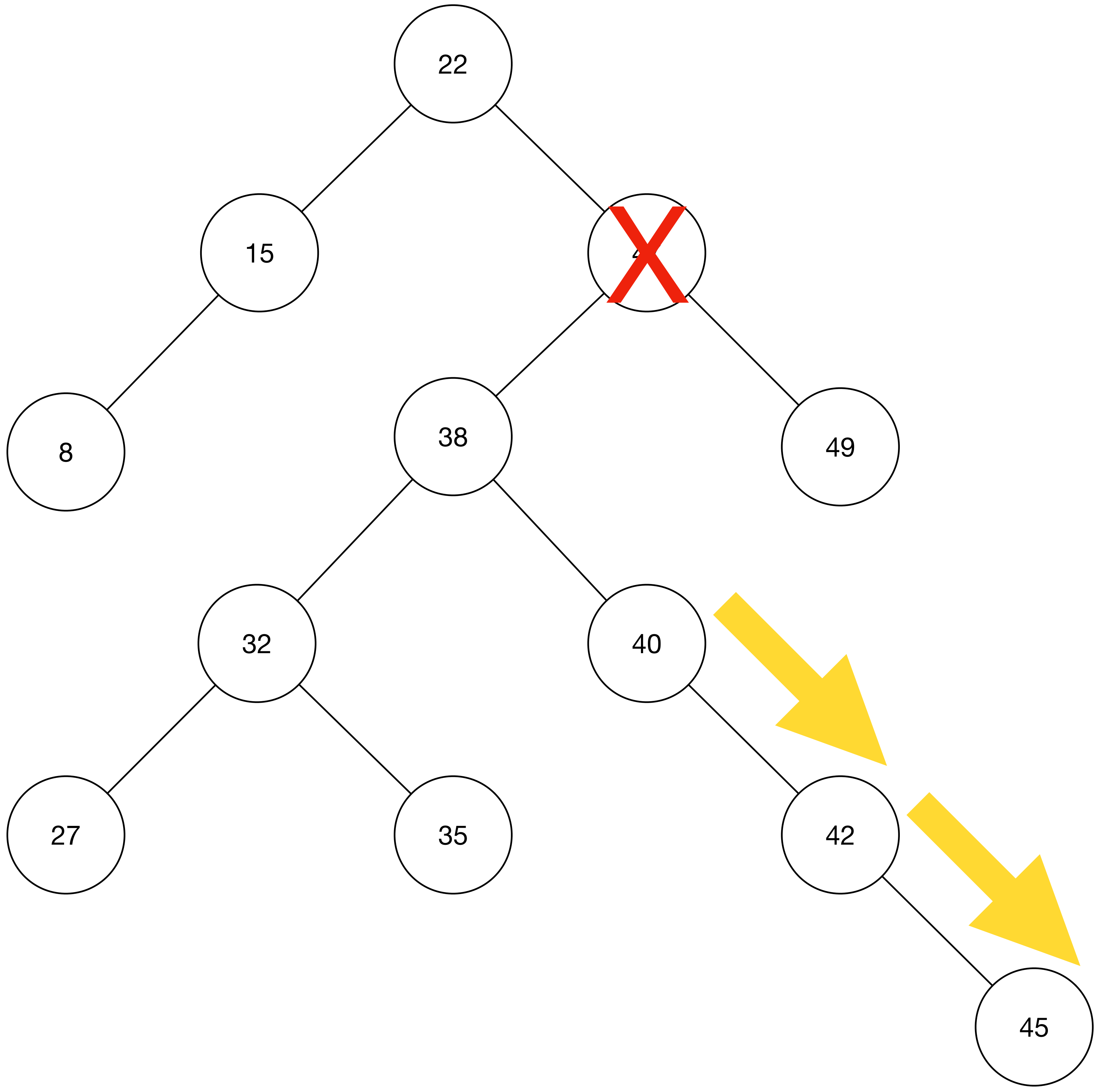
Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.



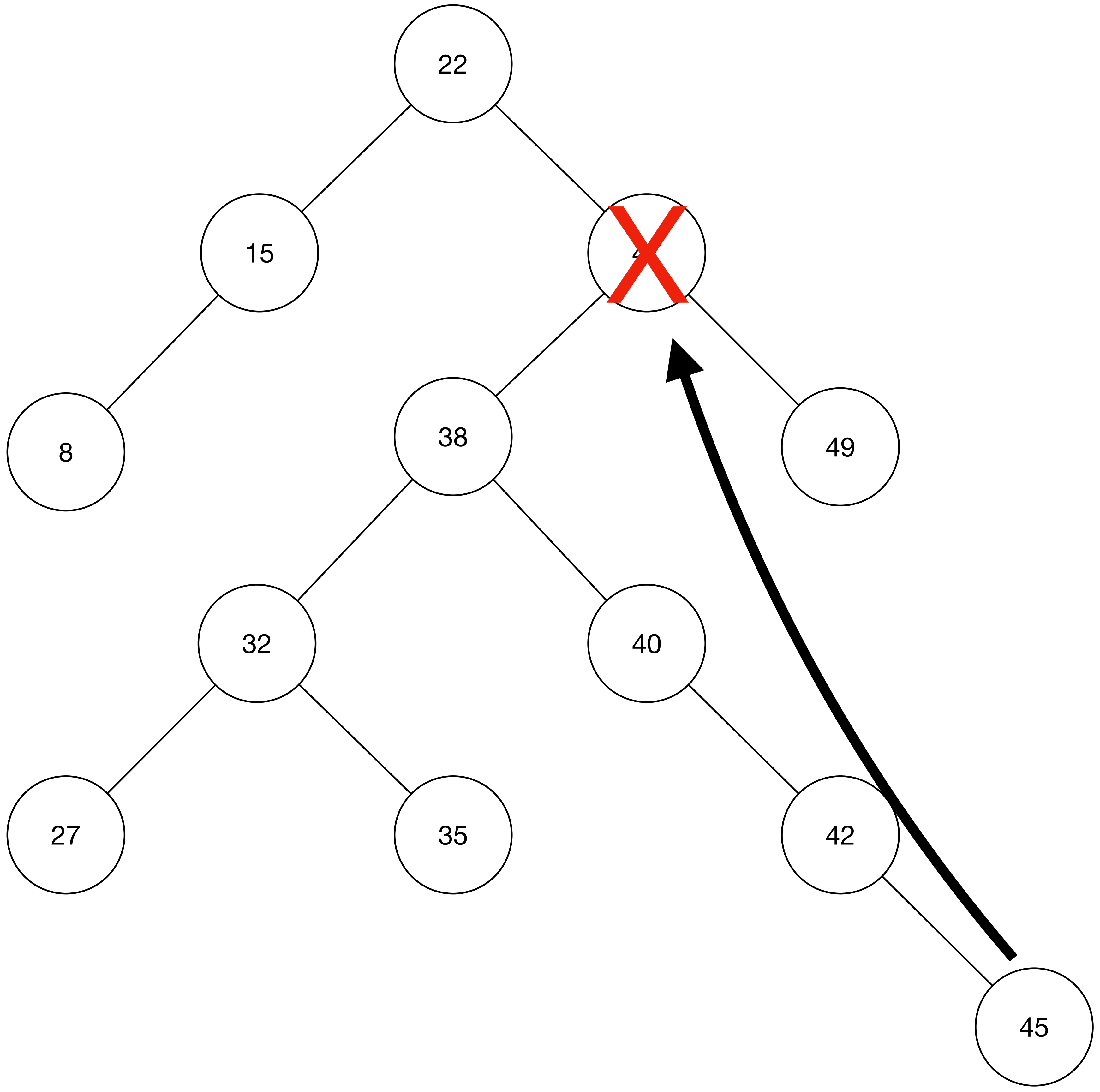
Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.



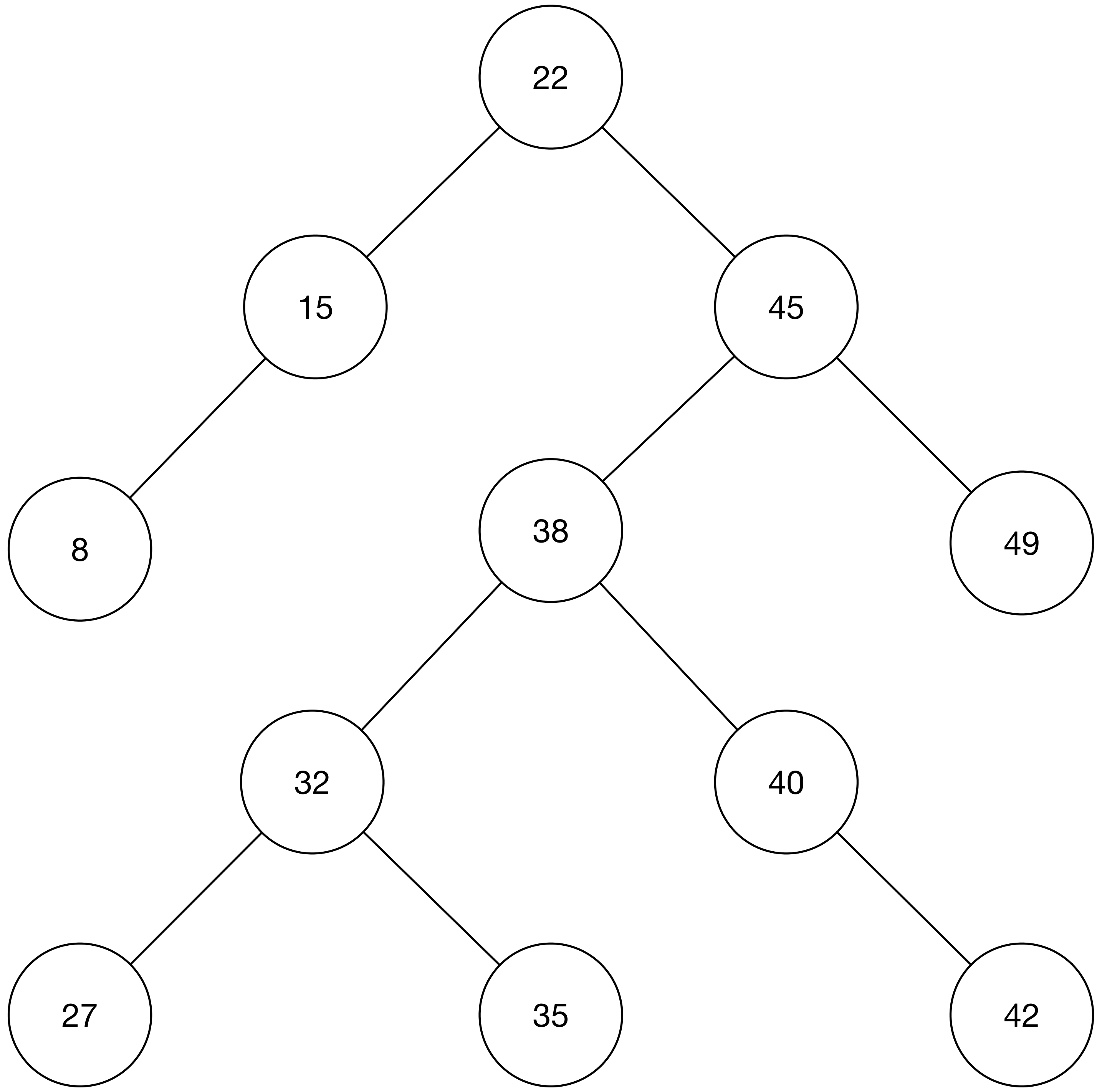
Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.



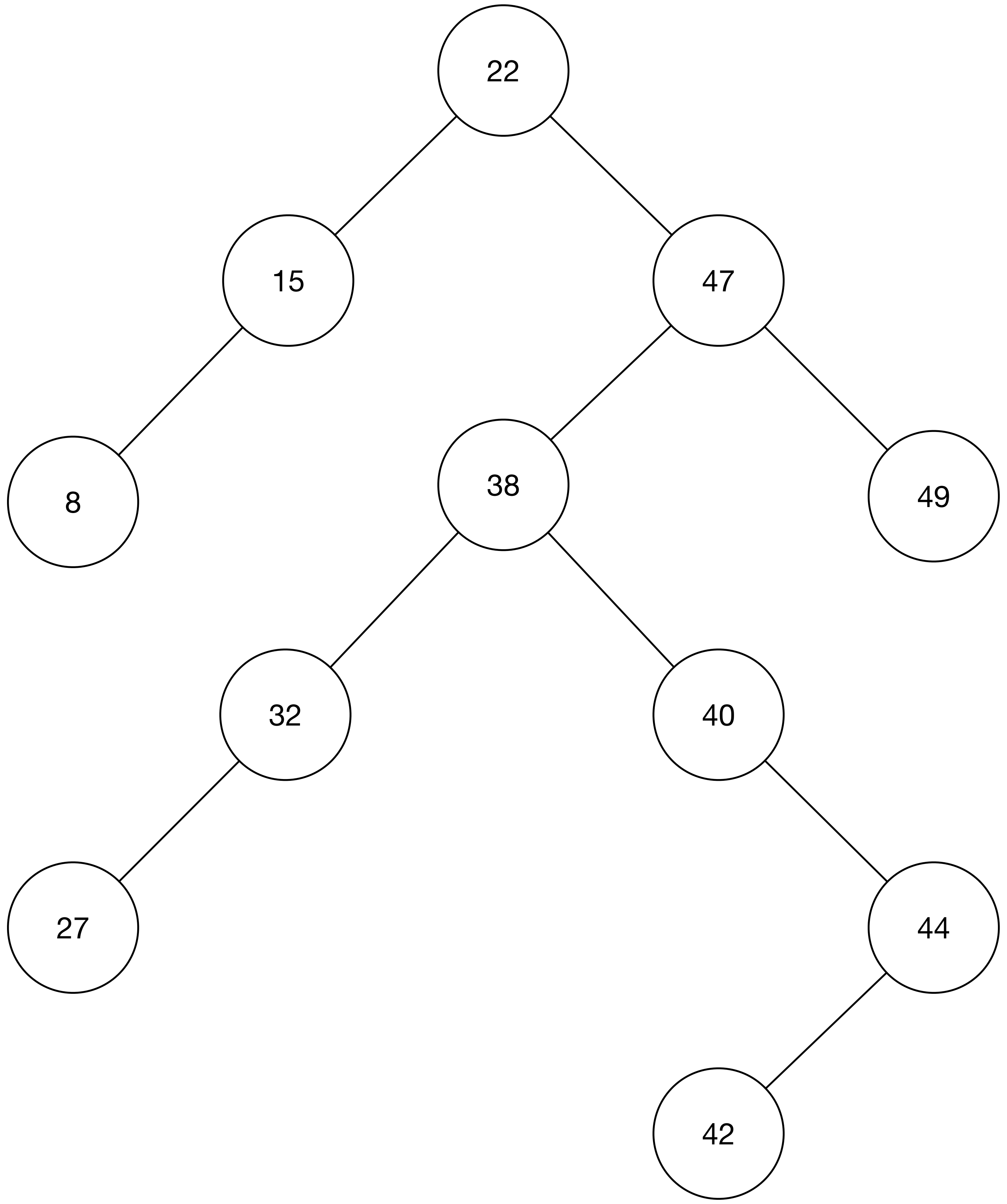
Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.



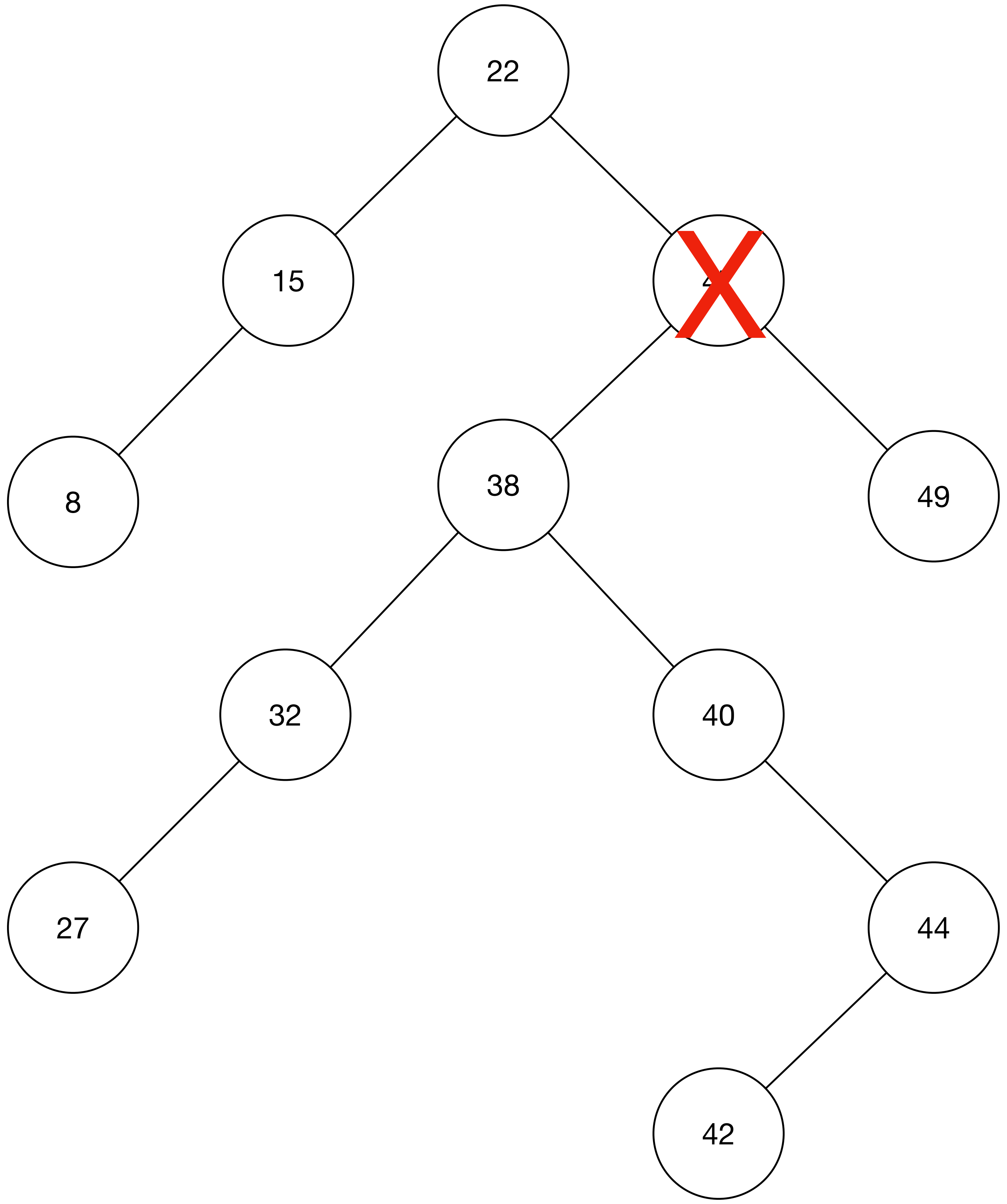
Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.



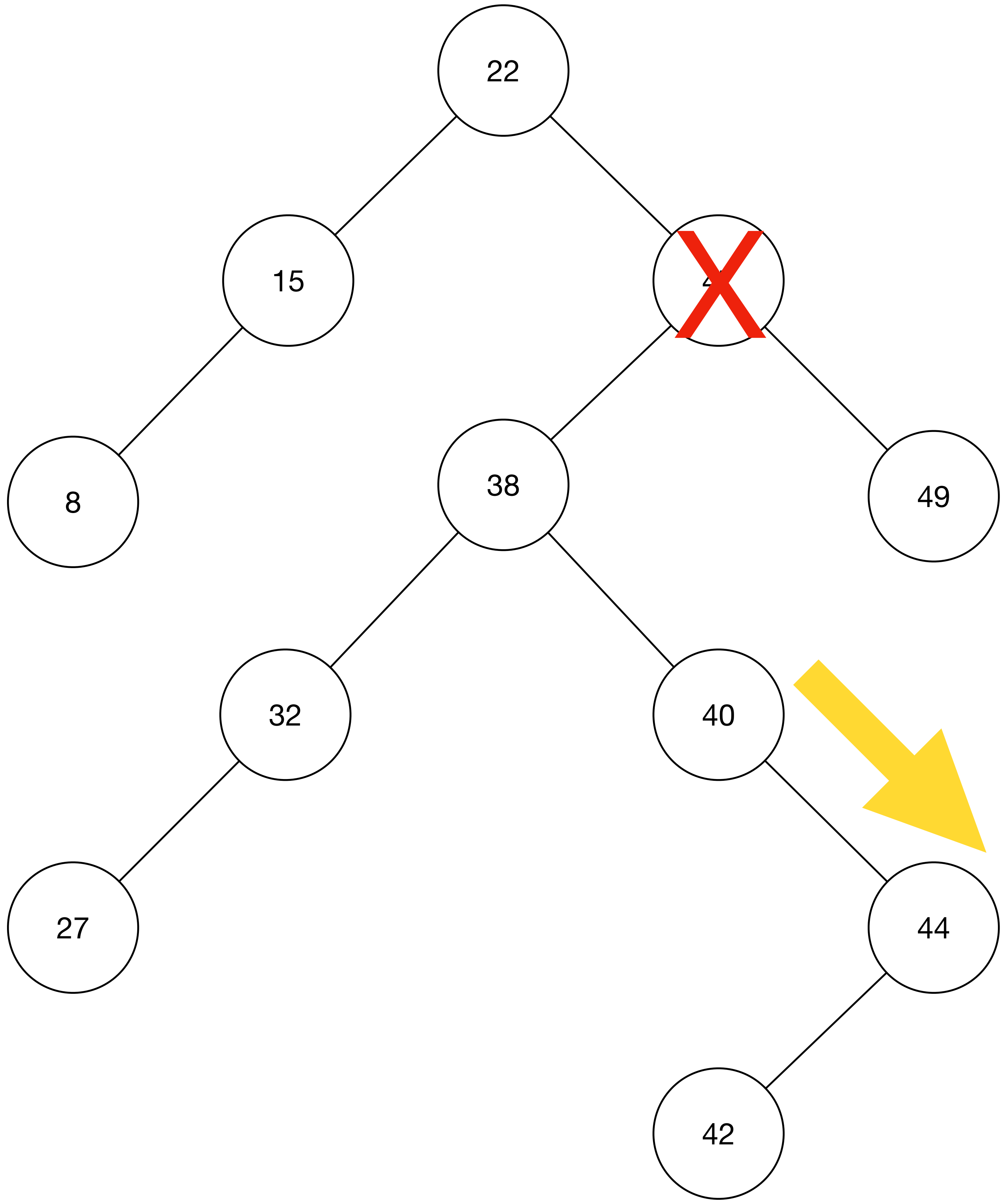
Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.



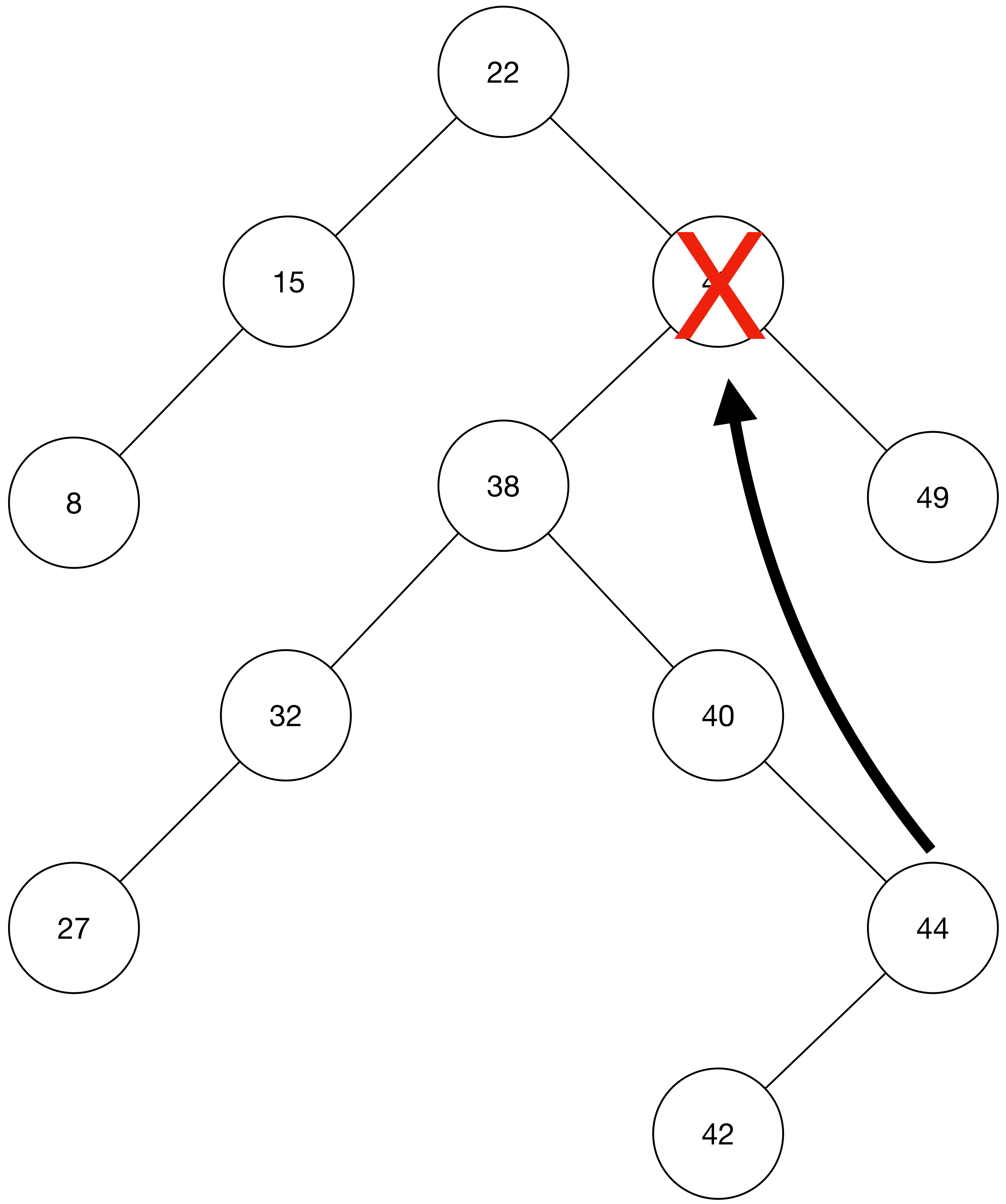
Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.



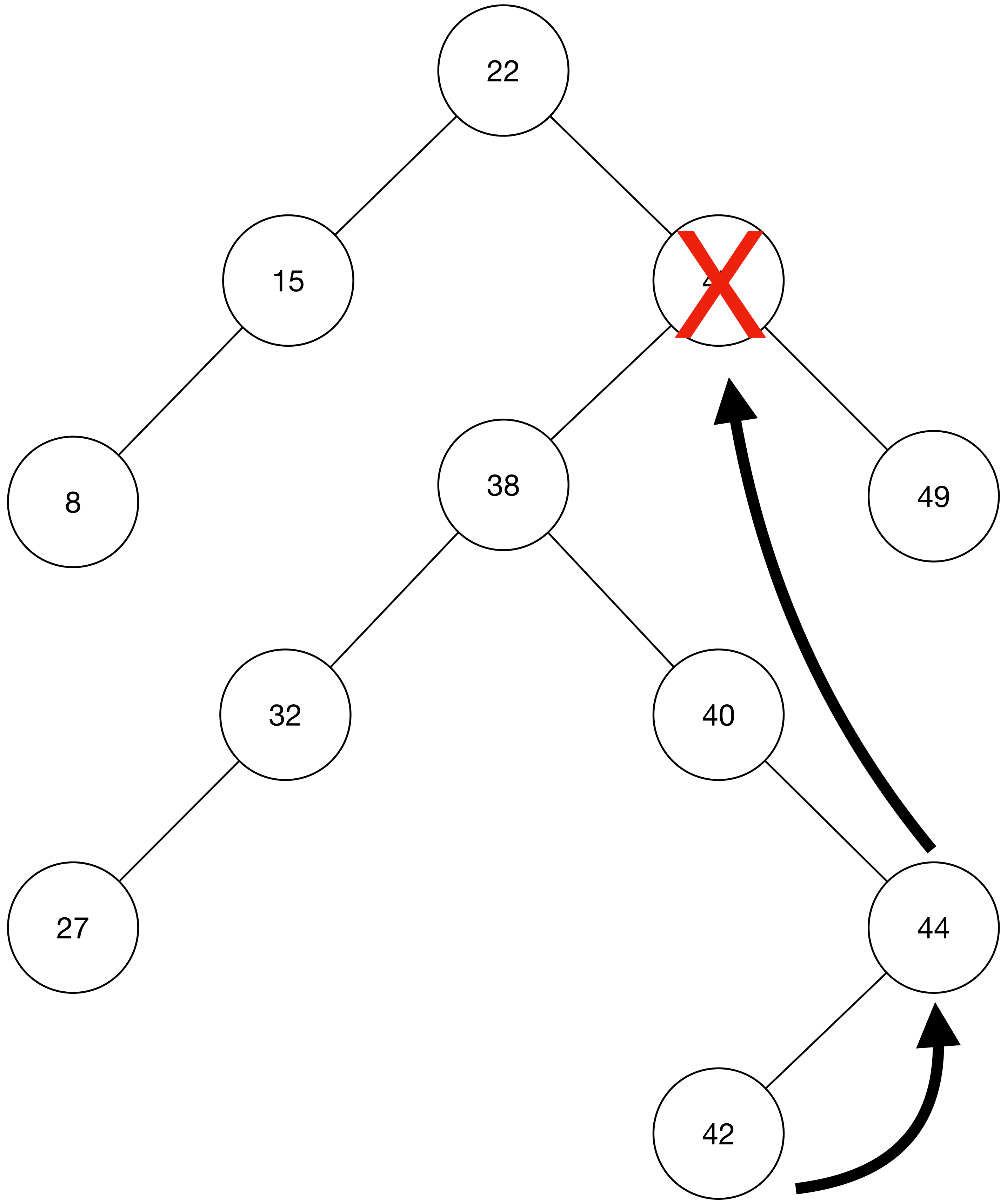
Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.



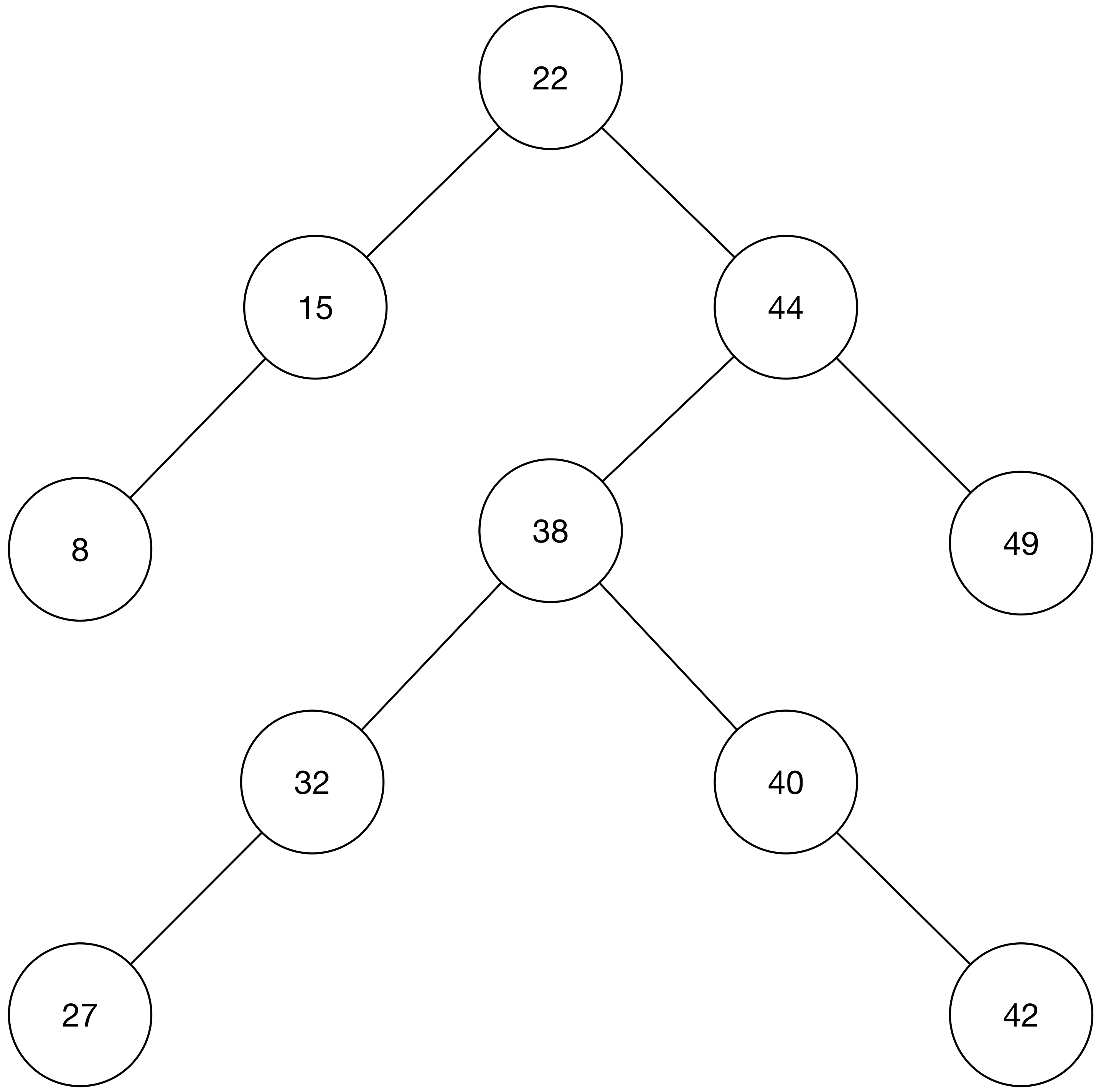
Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.



Deleting nodes in a BST

Case 4: Target node has two children, but left child has a right child.



Complexity of BST operations

	Insert node	Search	Delete node
Average case	$O(\log N)$	$O(\log N)$	$O(\log N)$
Worst case	$O(N)$	$O(N)$	$O(N)$