Tree Traversals

Dr. Mattox Beckman

Illinois Institute of Technology
Department of Computer Science

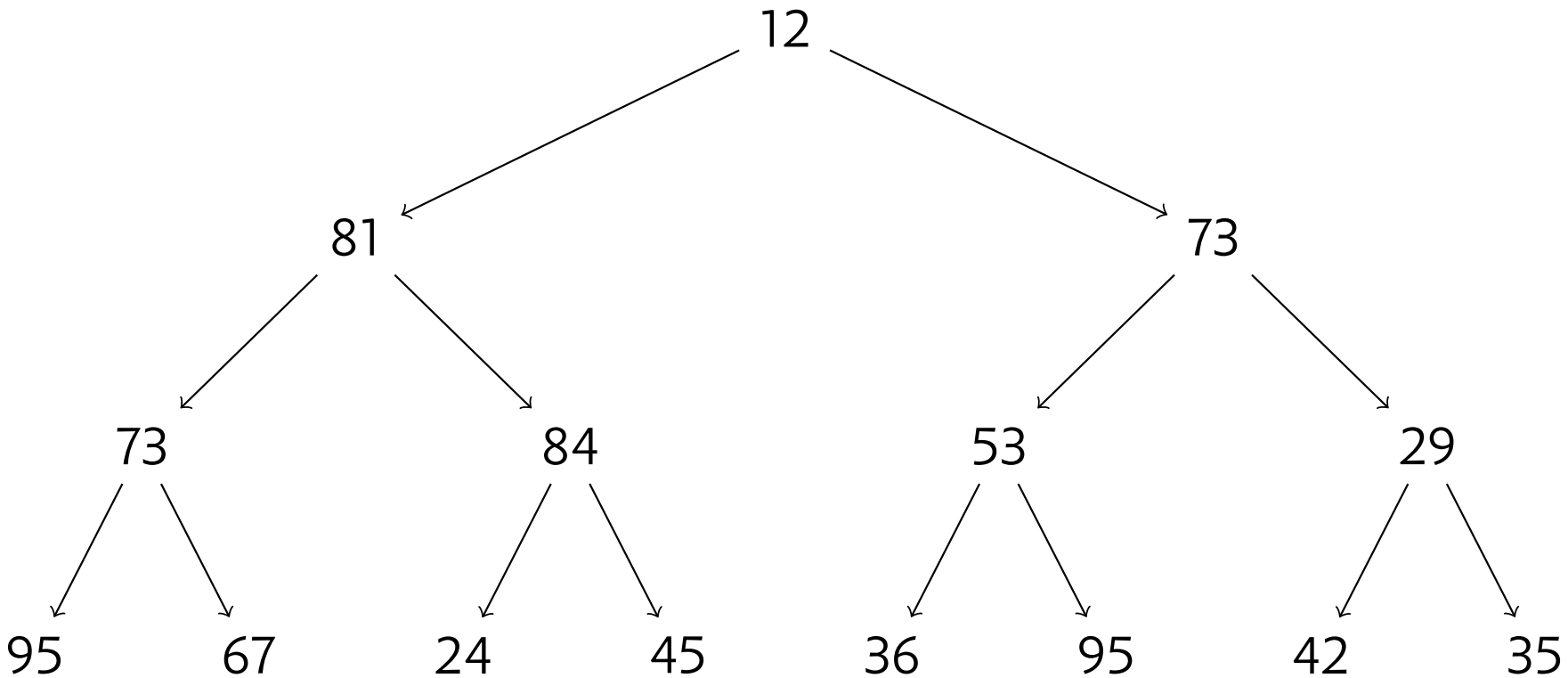
Objectives

You should be able to...

- Know two of the most common tree search patterns:
  - Depth First Search
  - Breadth First Search
- Know five tree traversal algorithms:
  - Preorder
  - Postorder
  - Inorder
  - Next
  - Previous

Looking for the Answer to the Ultimate Question

Suppose we have the following binary tree, and we want to search it for something.

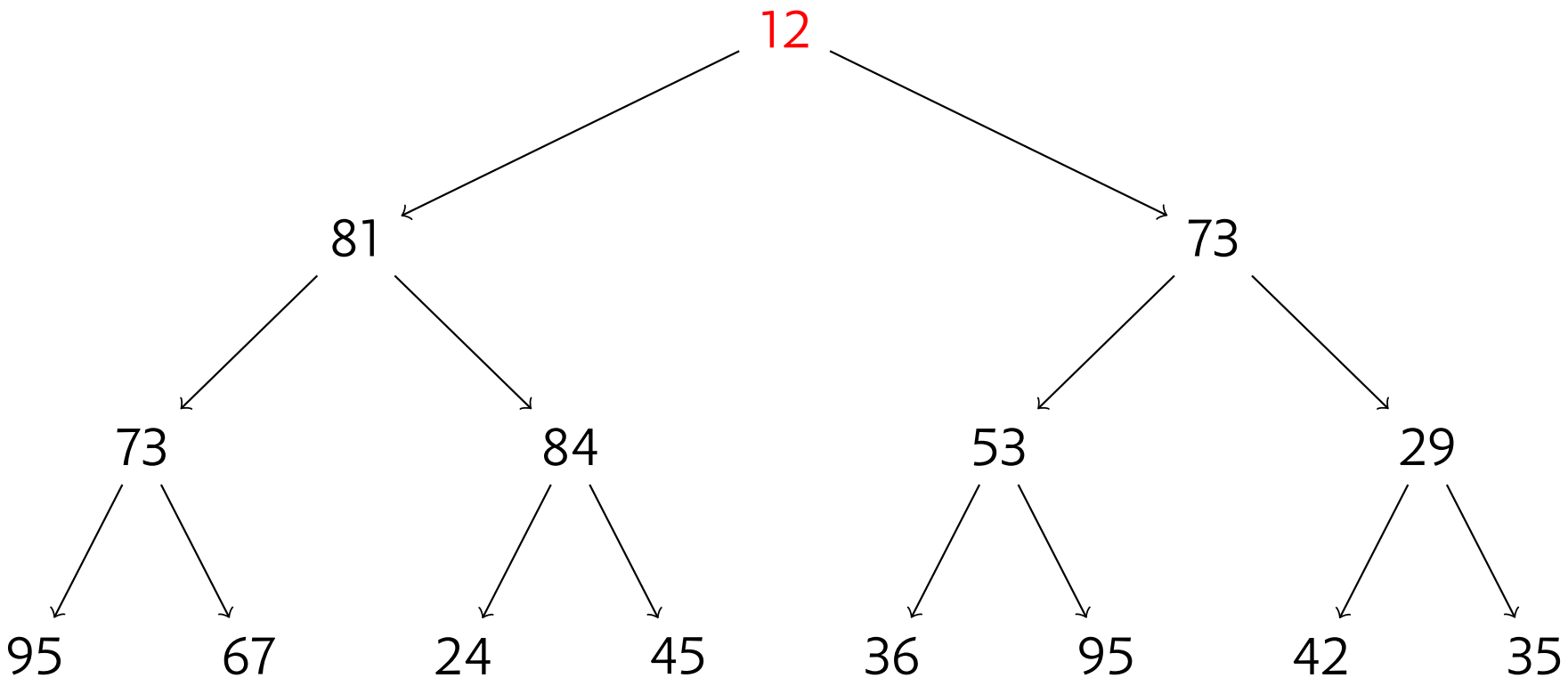


Depth First Search

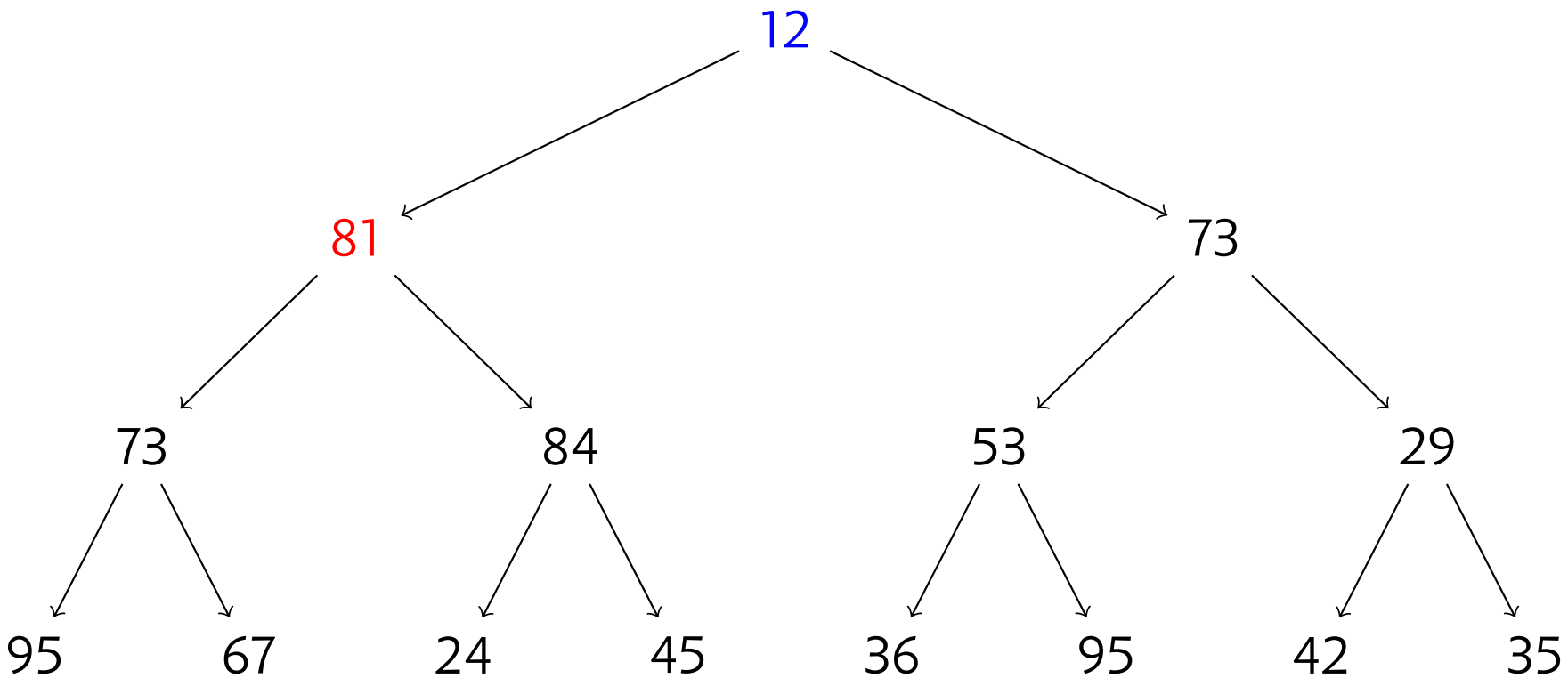
DFS Algorithm

- Check the Current Node
- Recursively Search the Left
- Recursively Search the Right

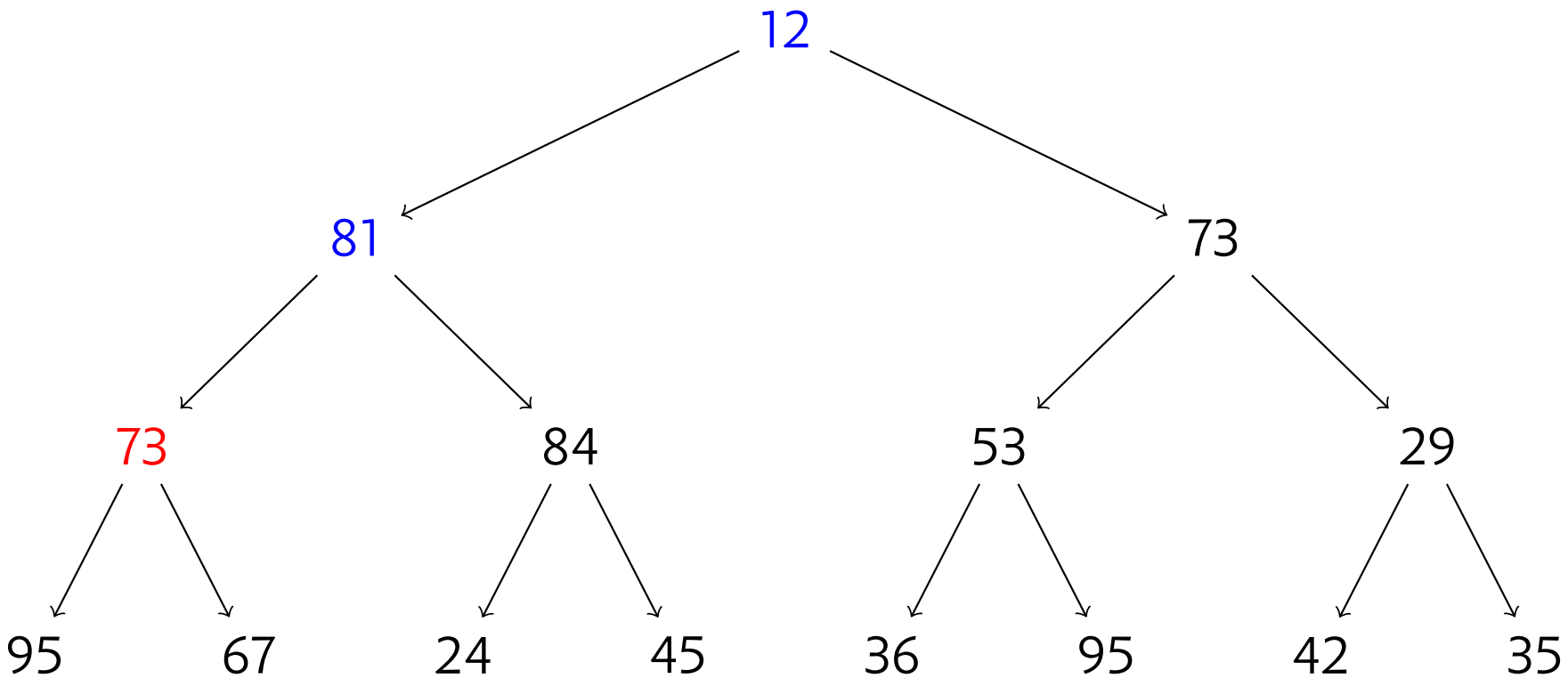
Searching via DFS



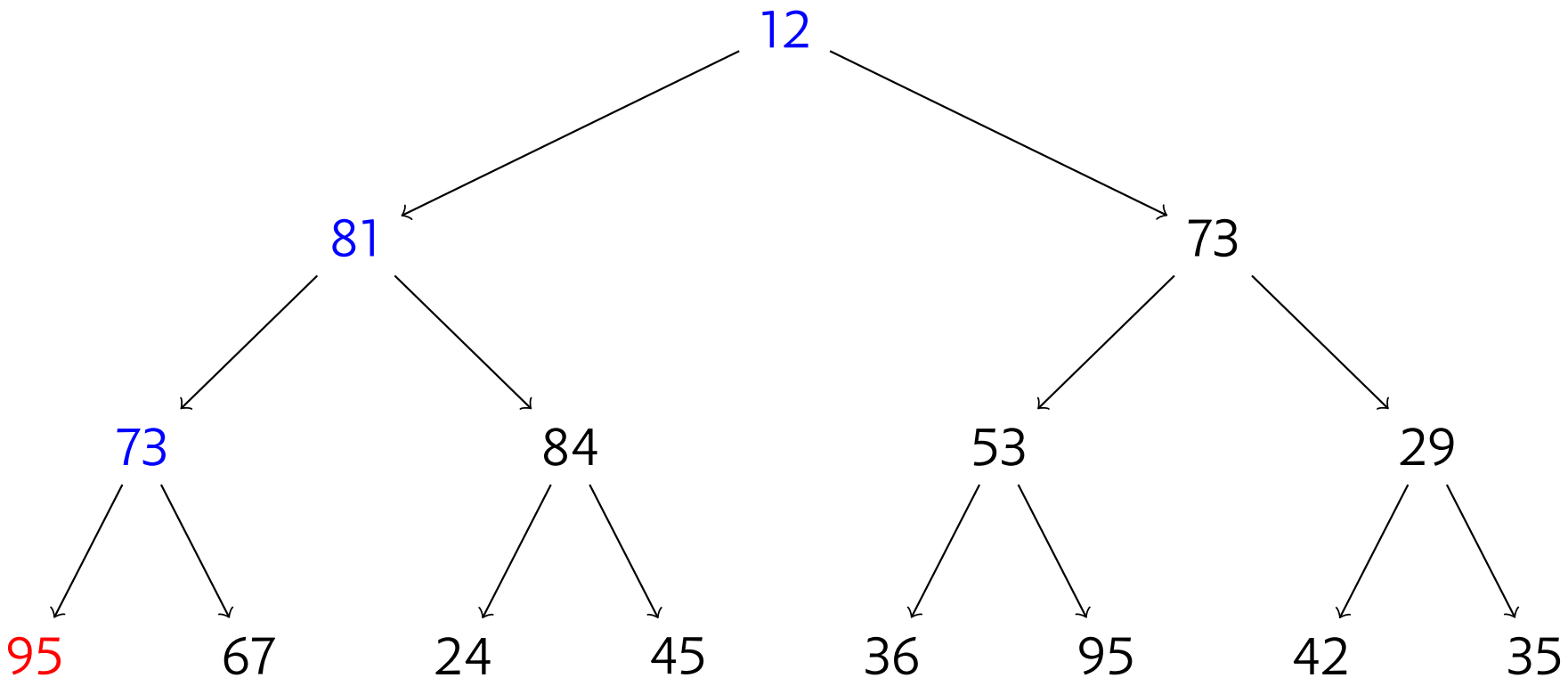
Searching via DFS



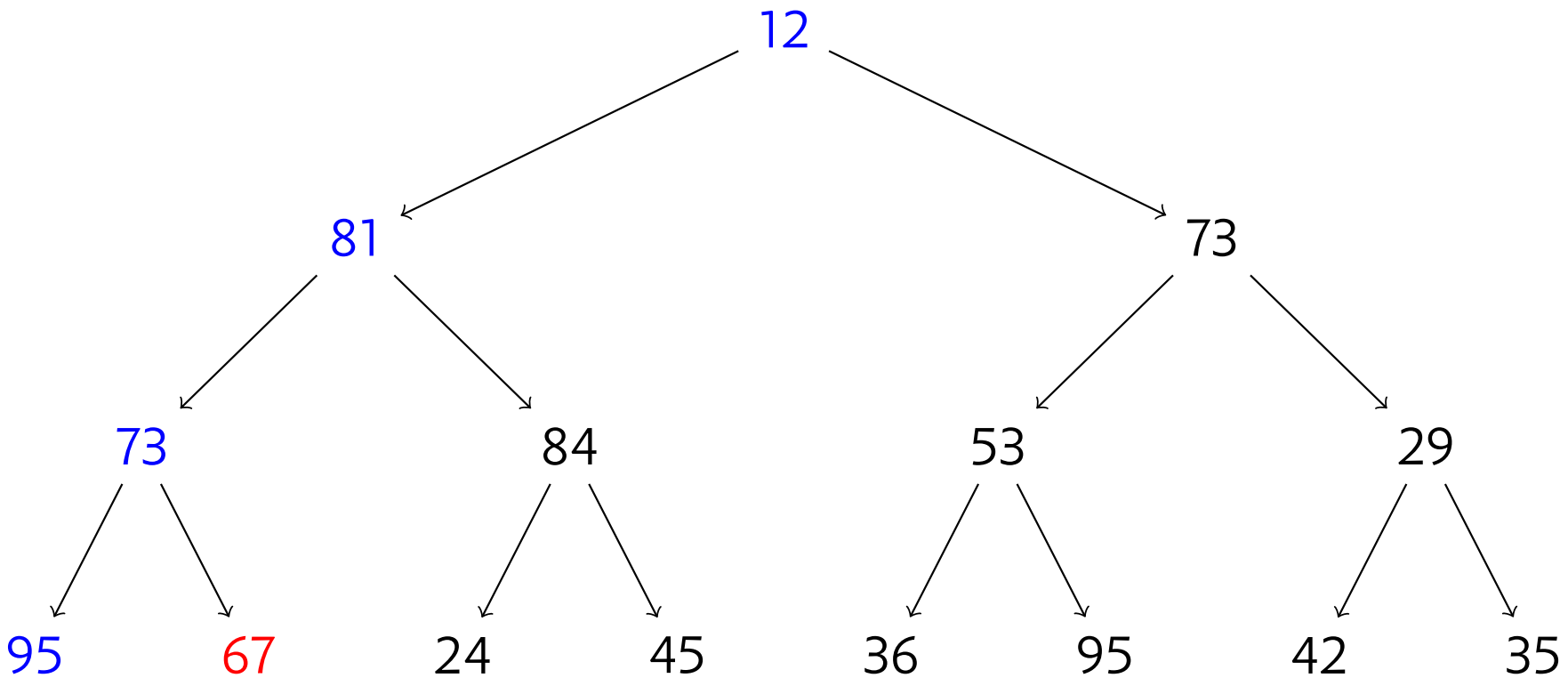
Searching via DFS



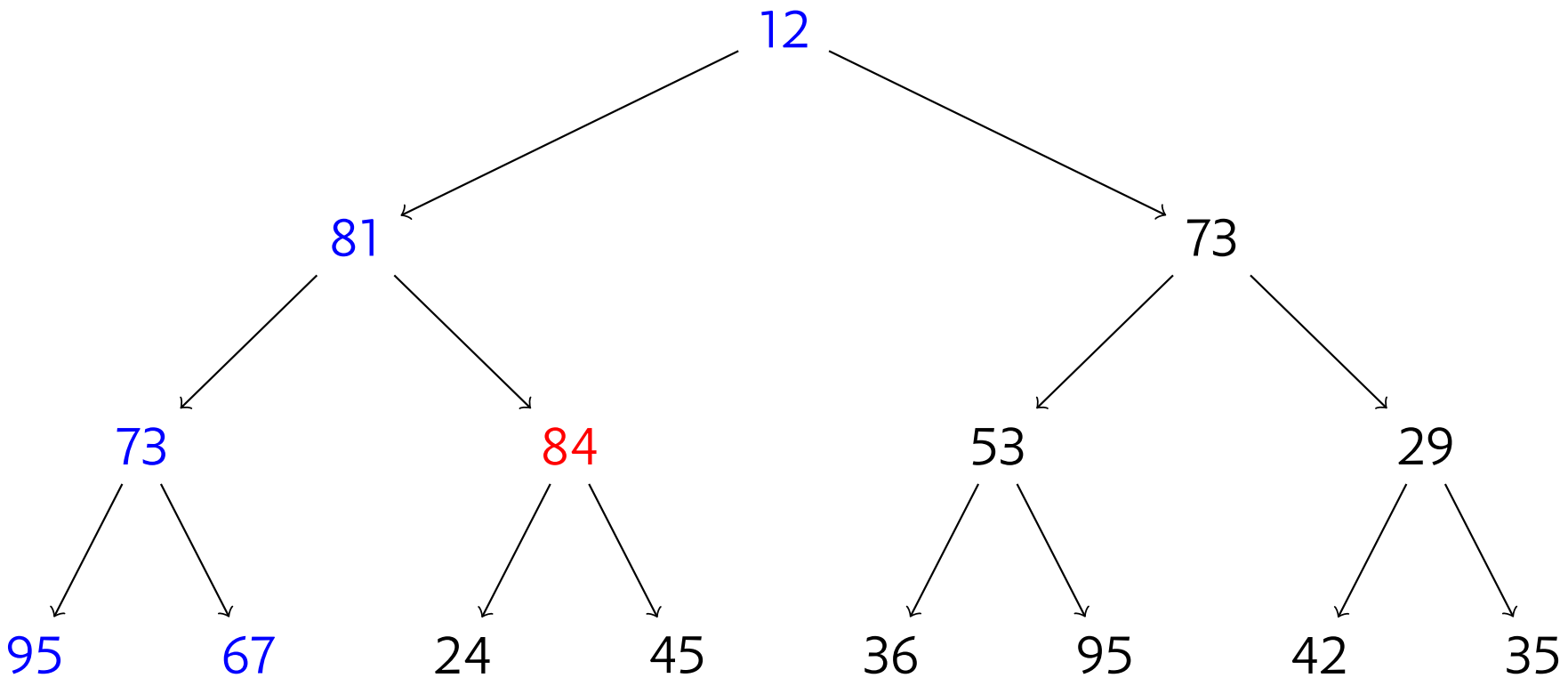
Searching via DFS



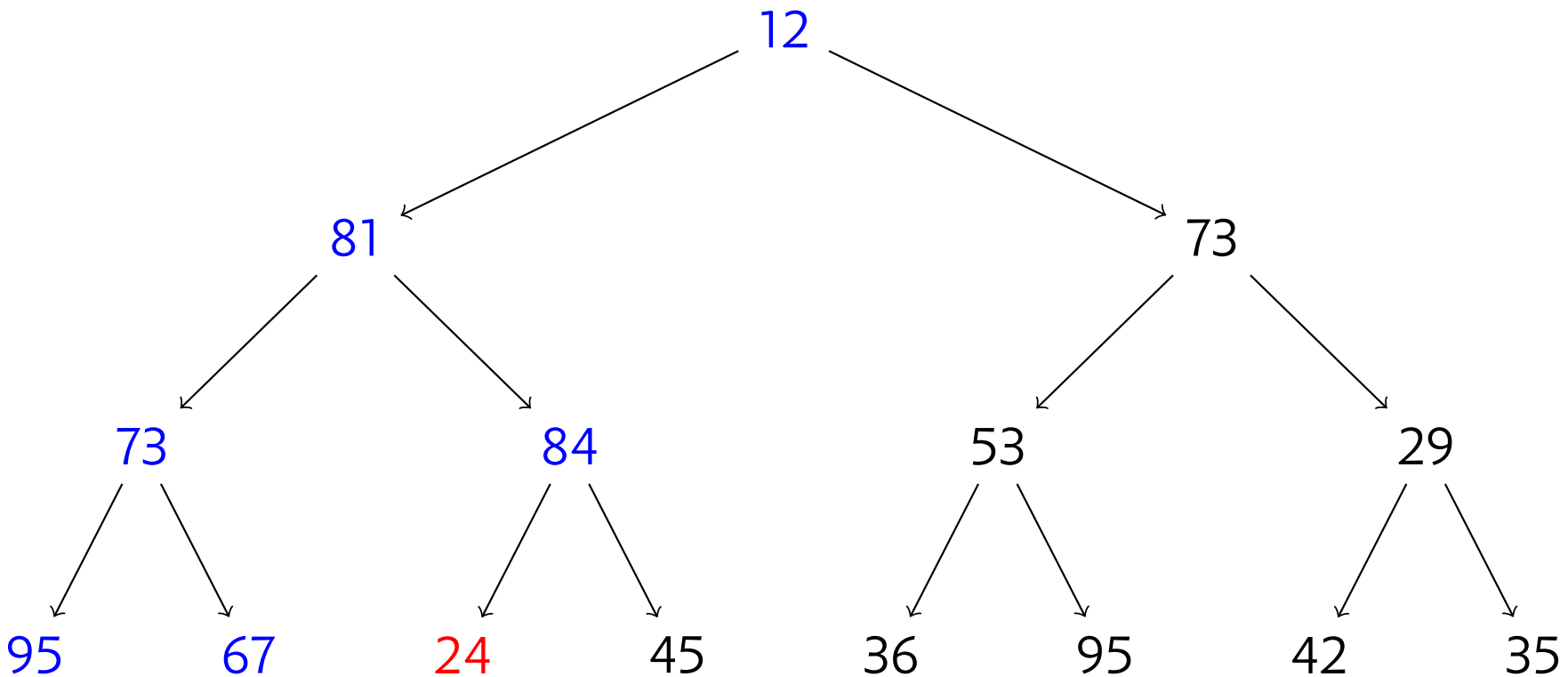
Searching via DFS



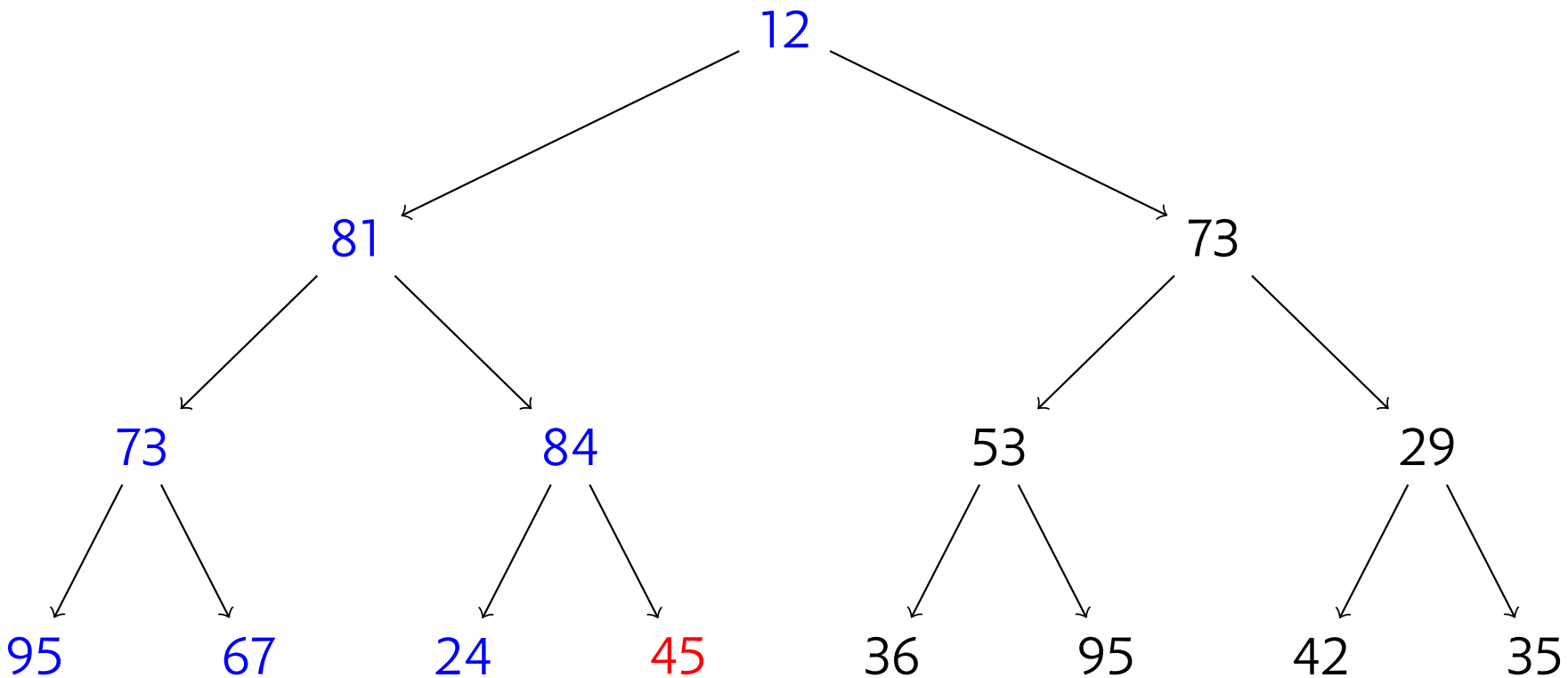
Searching via DFS



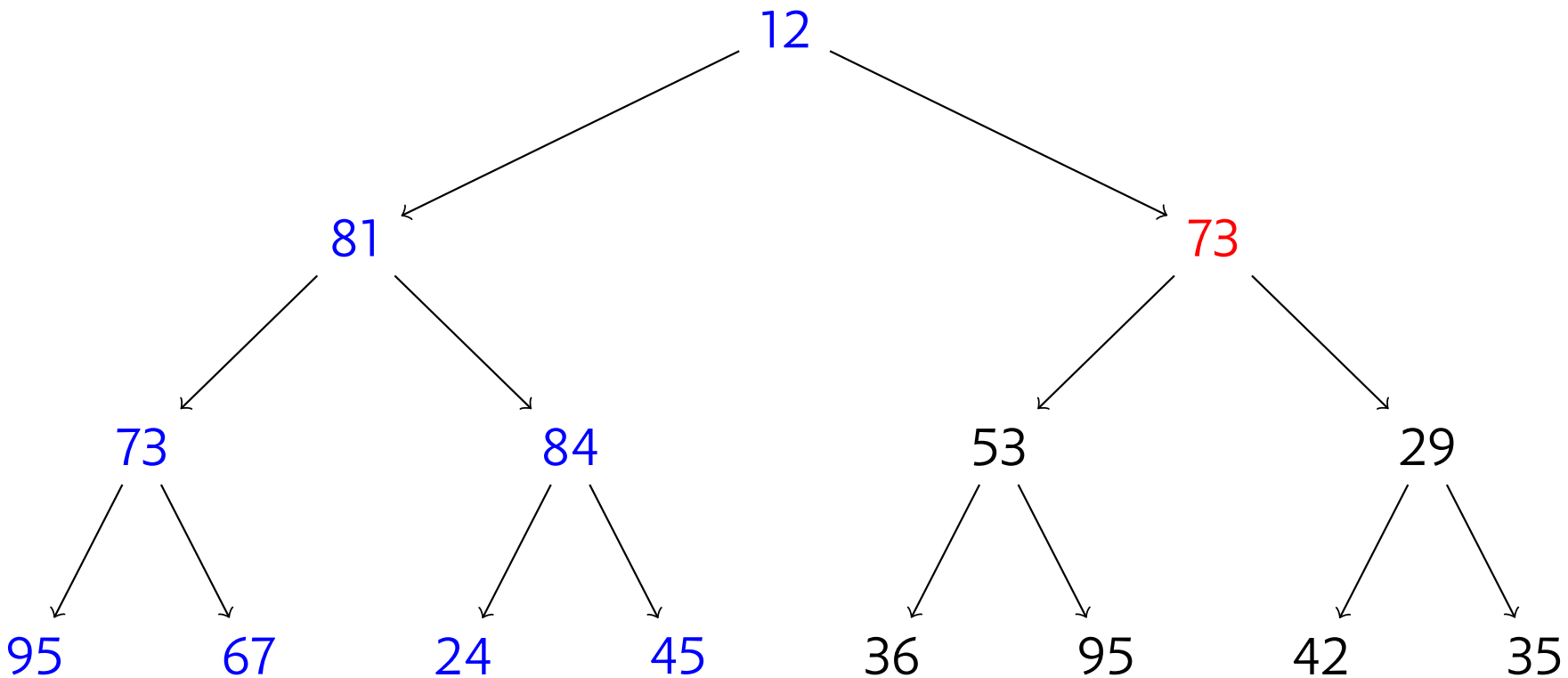
Searching via DFS



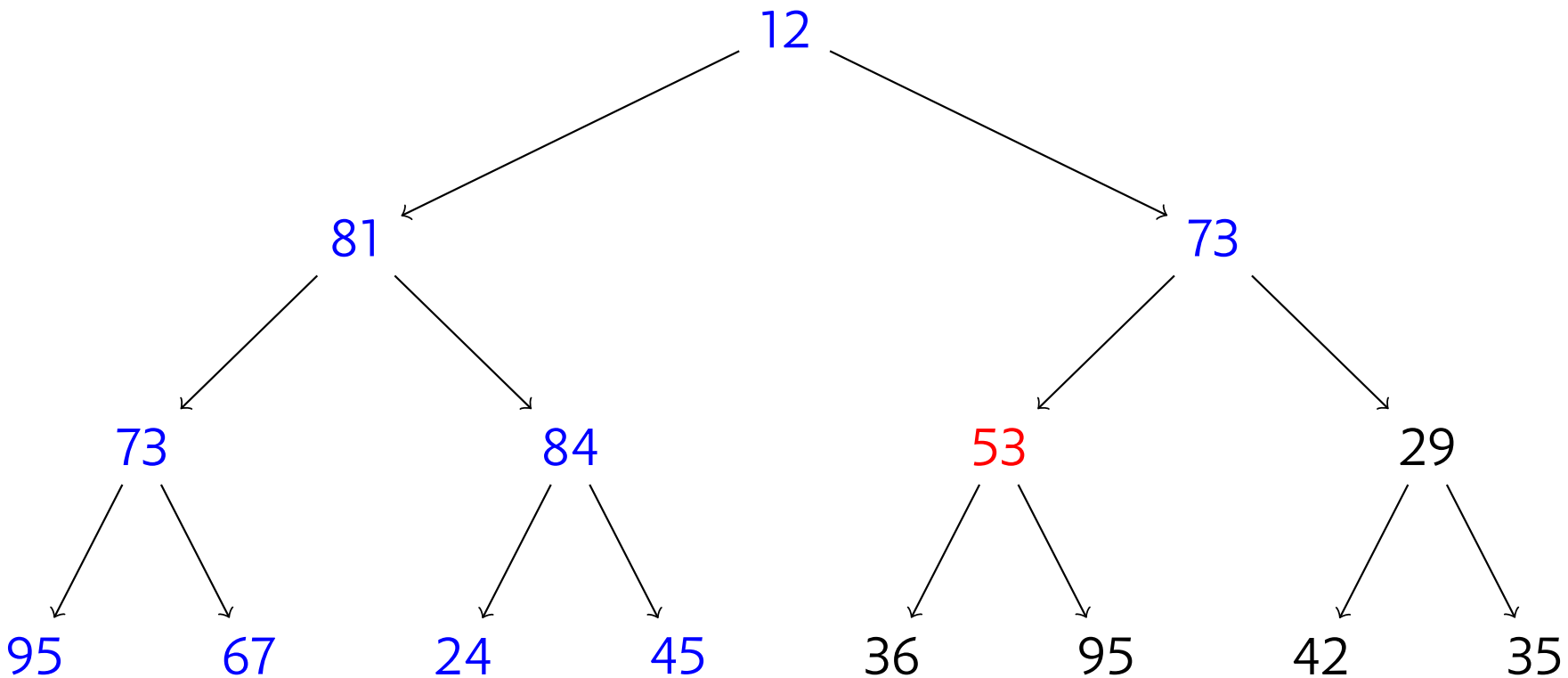
Searching via DFS



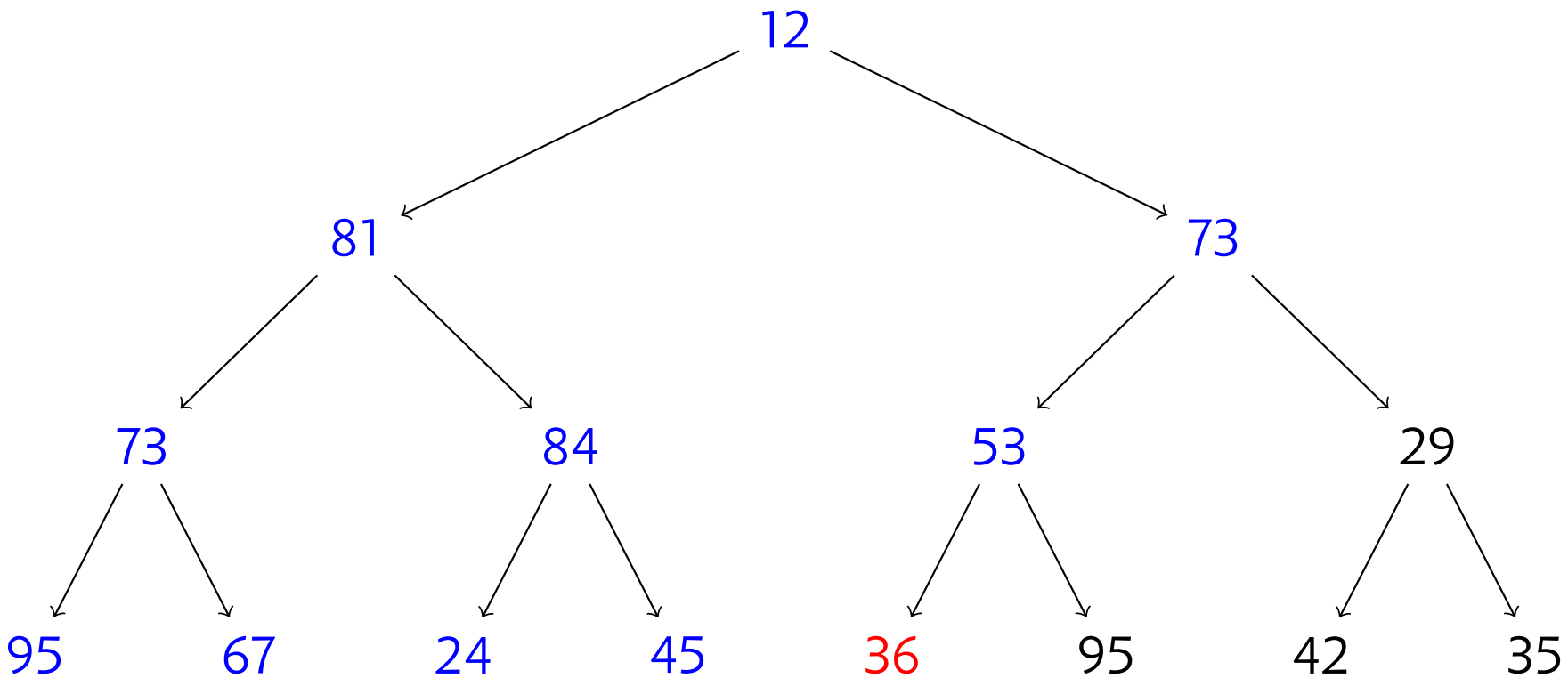
Searching via DFS



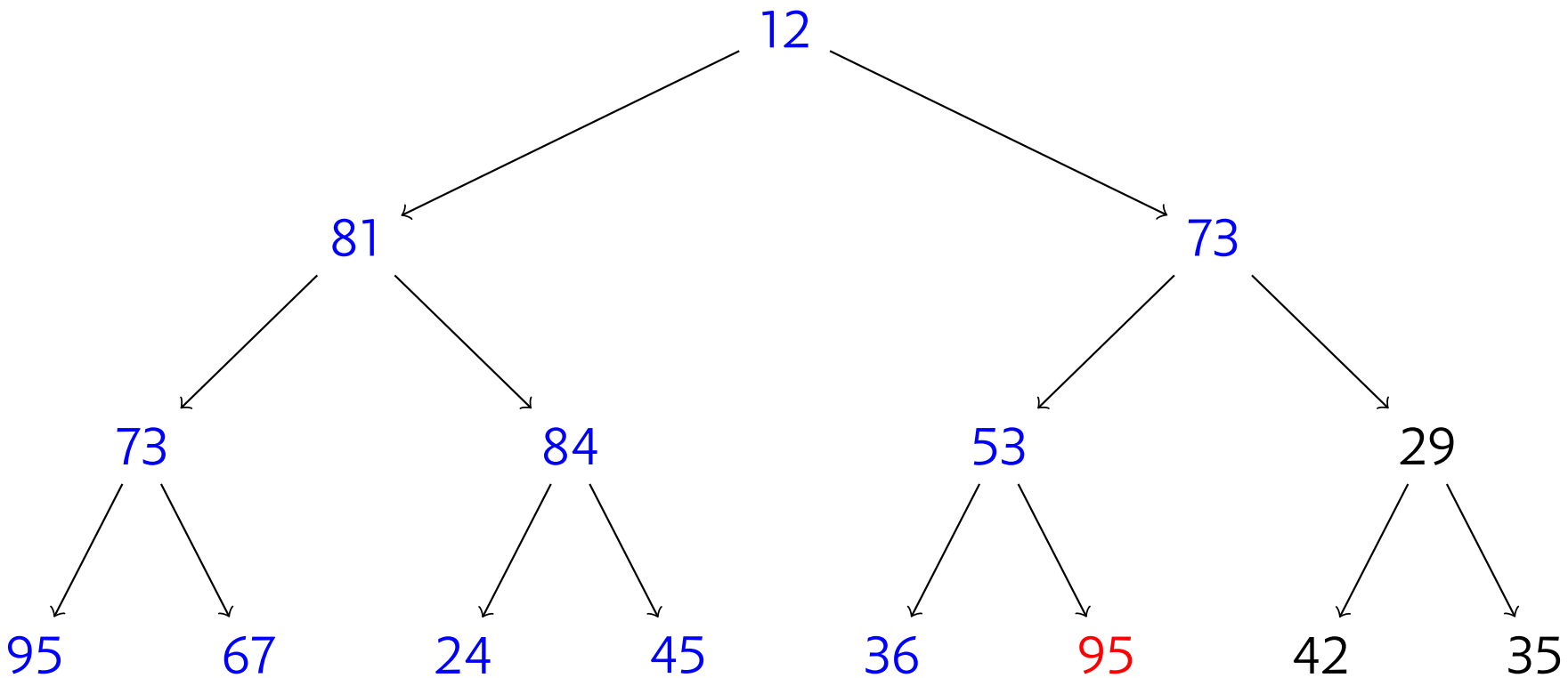
Searching via DFS



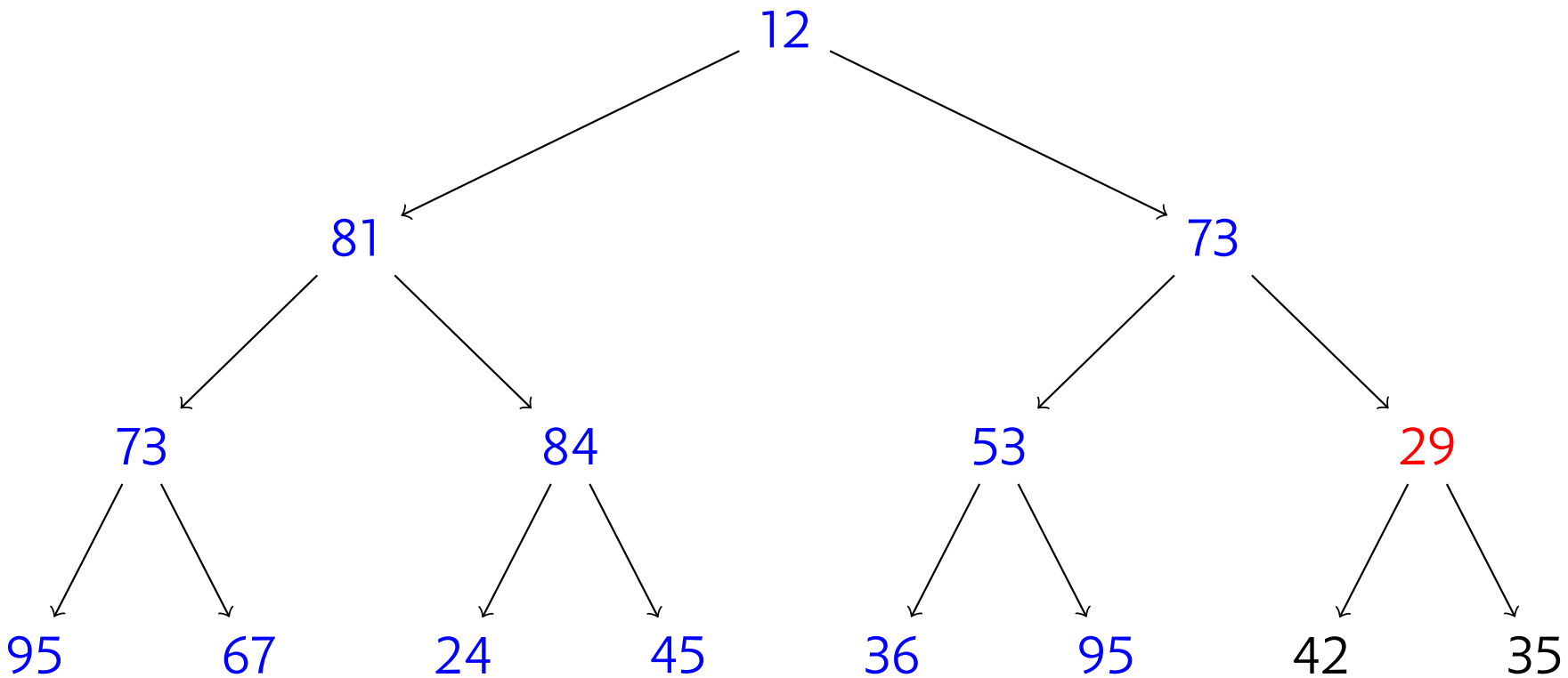
Searching via DFS



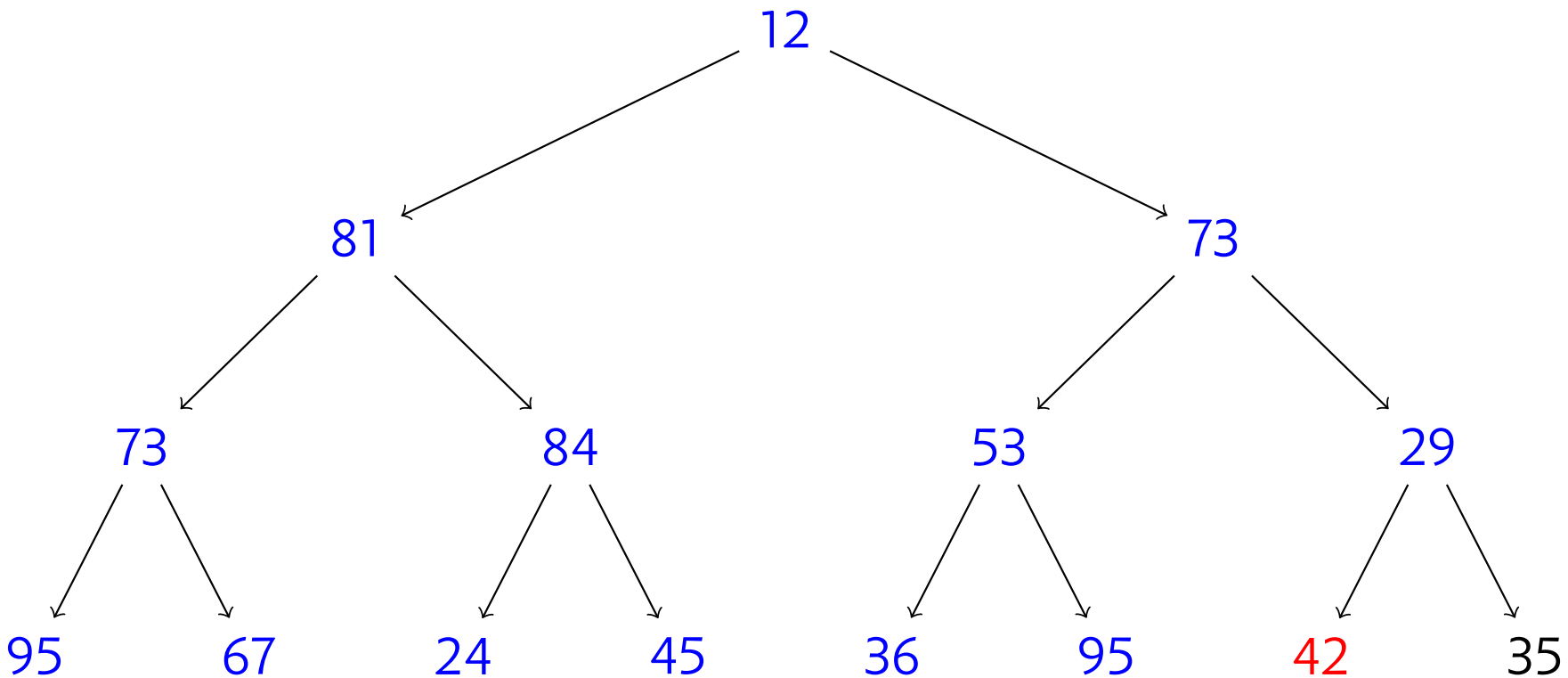
Searching via DFS



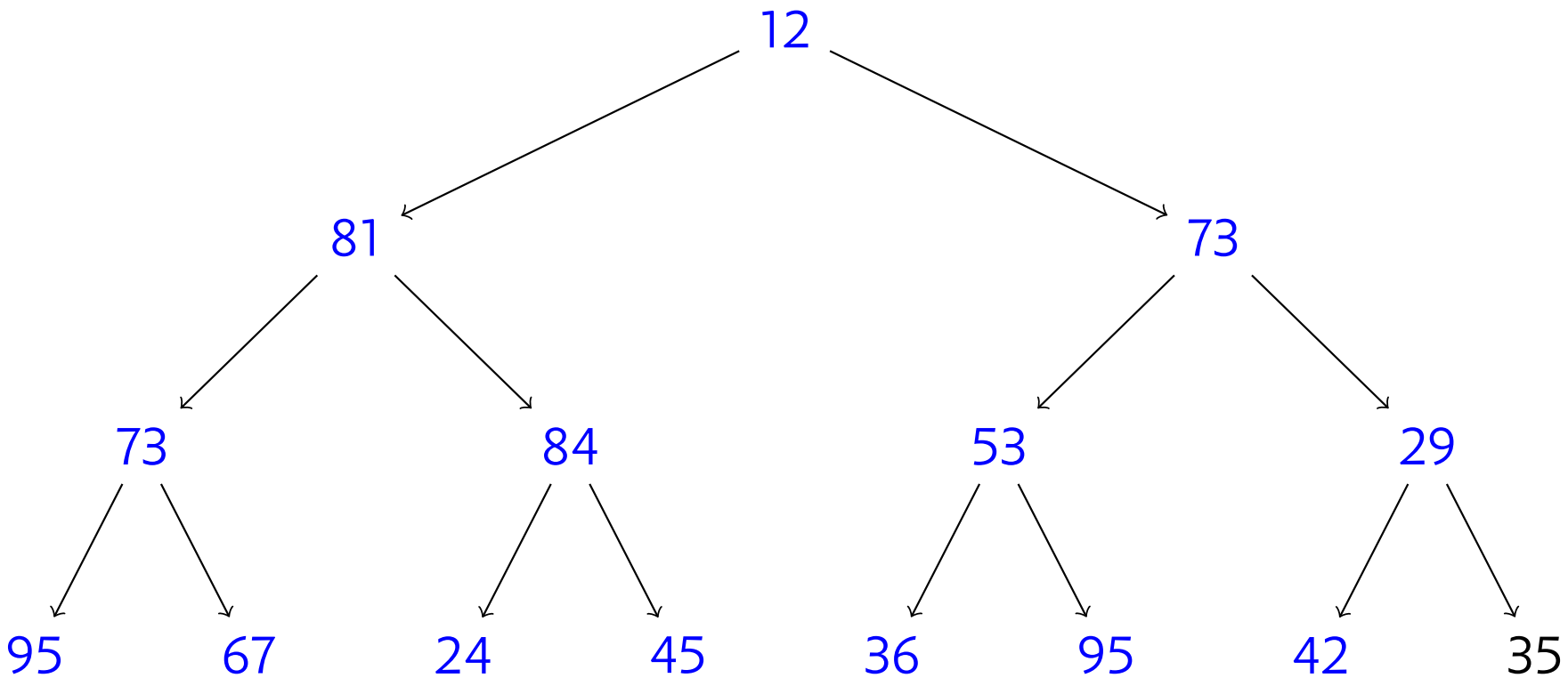
Searching via DFS



Searching via DFS



Searching via DFS



Things to know.

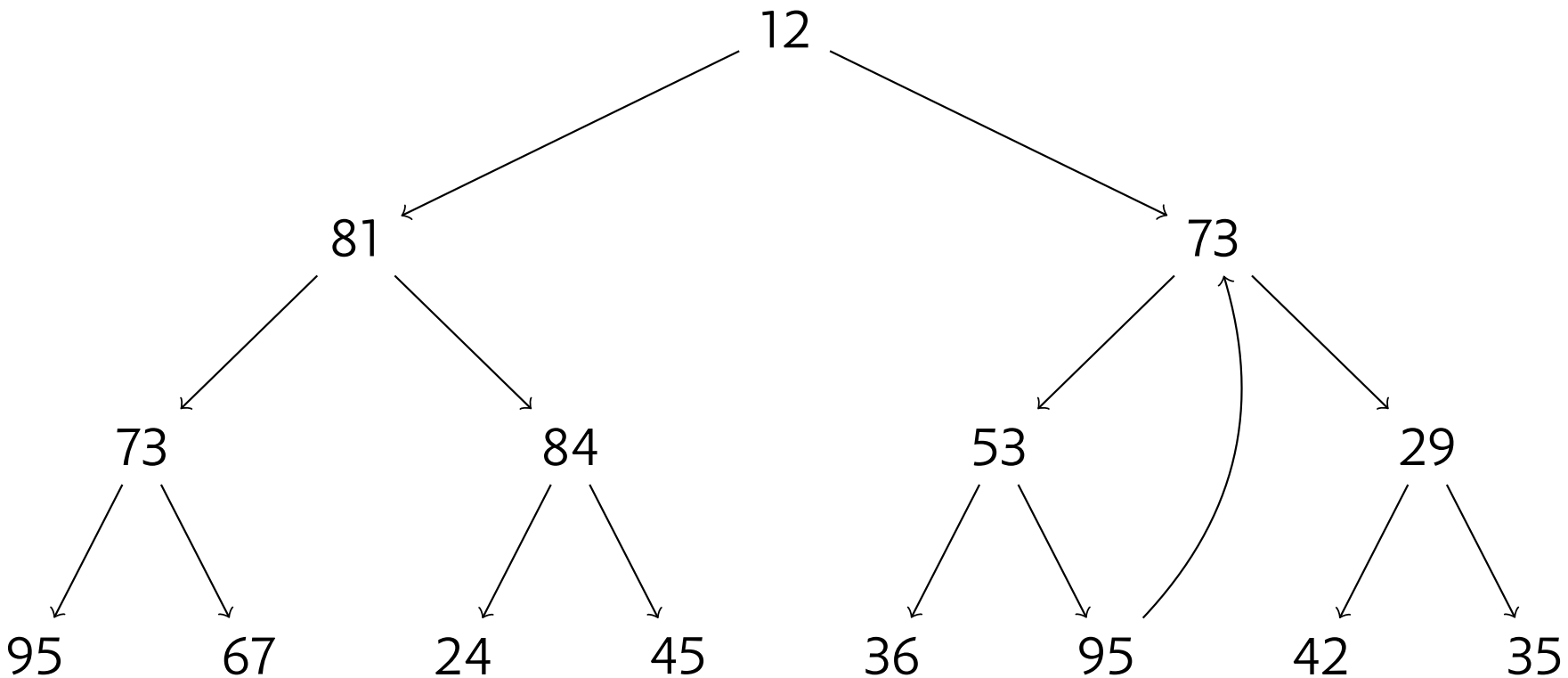
Pros

- Very easy to write this.
- Uses very little memory. (How much?)

Cons

- Does not handle back-edges well.
- Does not handle infinite trees at all.

A Back Edge

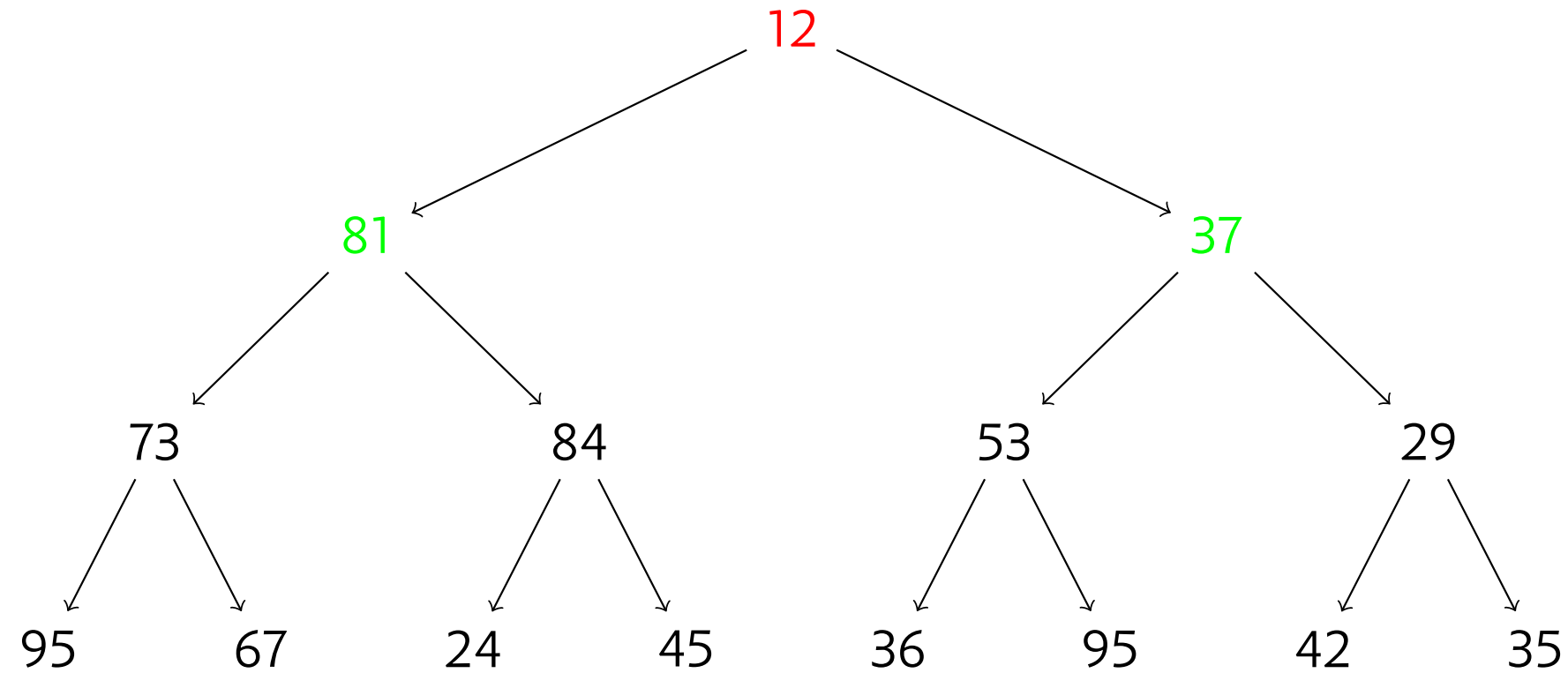


Breadth First Search

BFS Algorithm

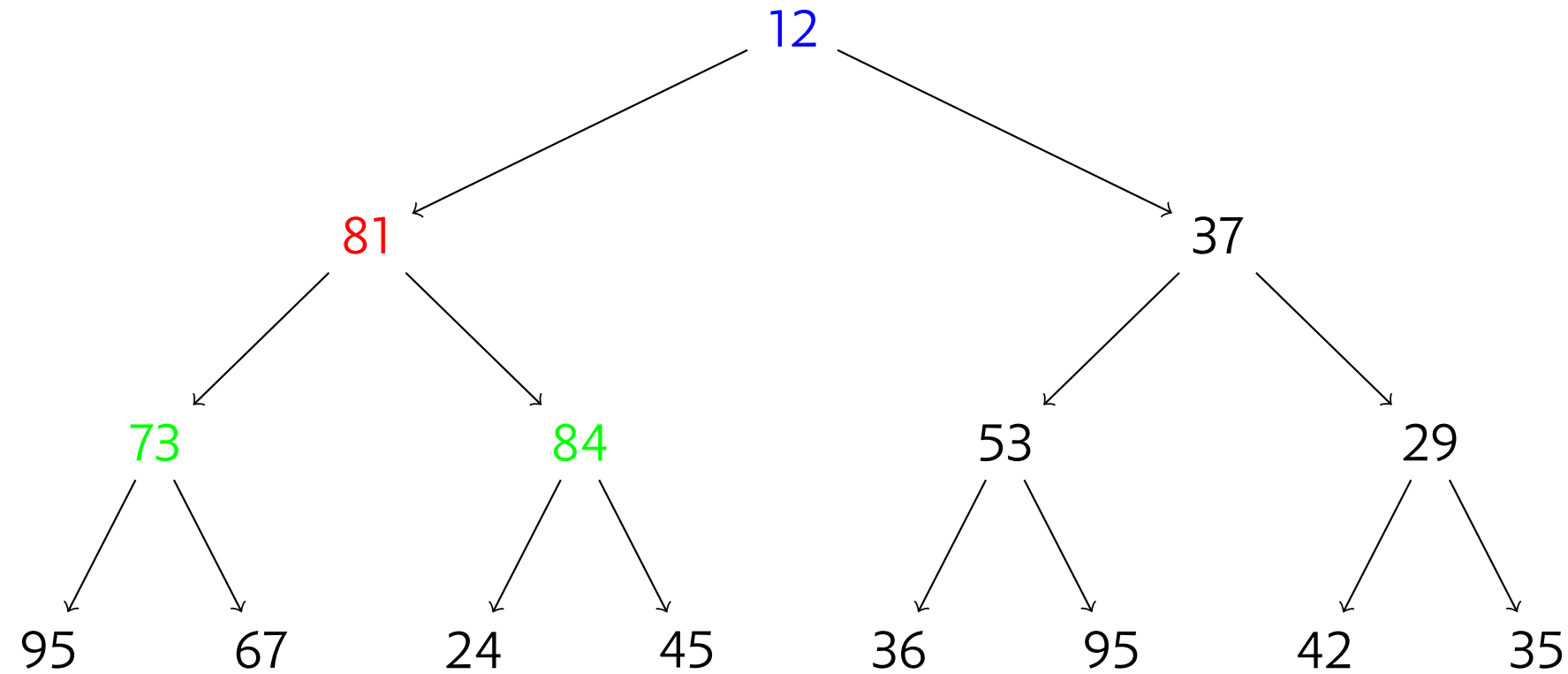
- Enqueue the Root
- Then...
  - Dequeue a Node
  - Check Node
  - Enqueue Children

Searching via BFS



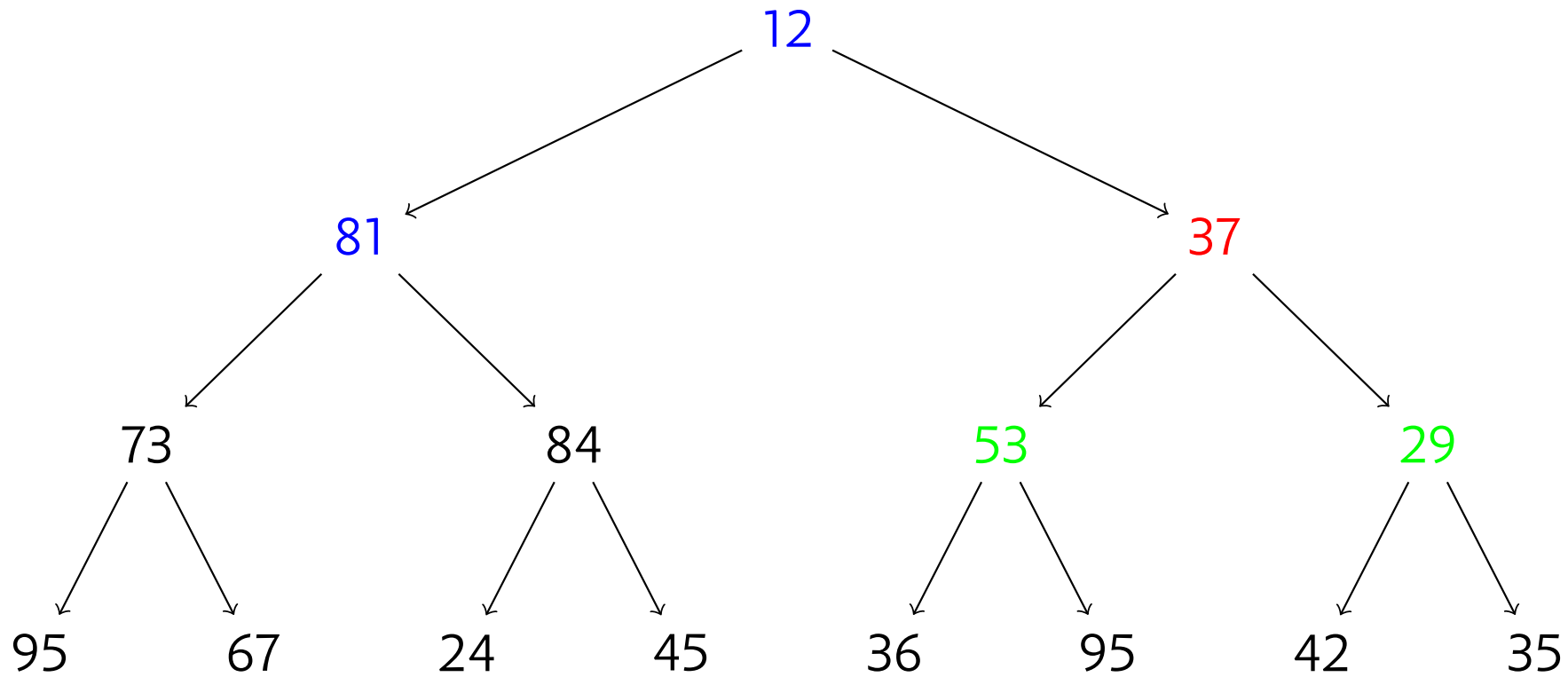
Queue: 81 37

Searching via BFS



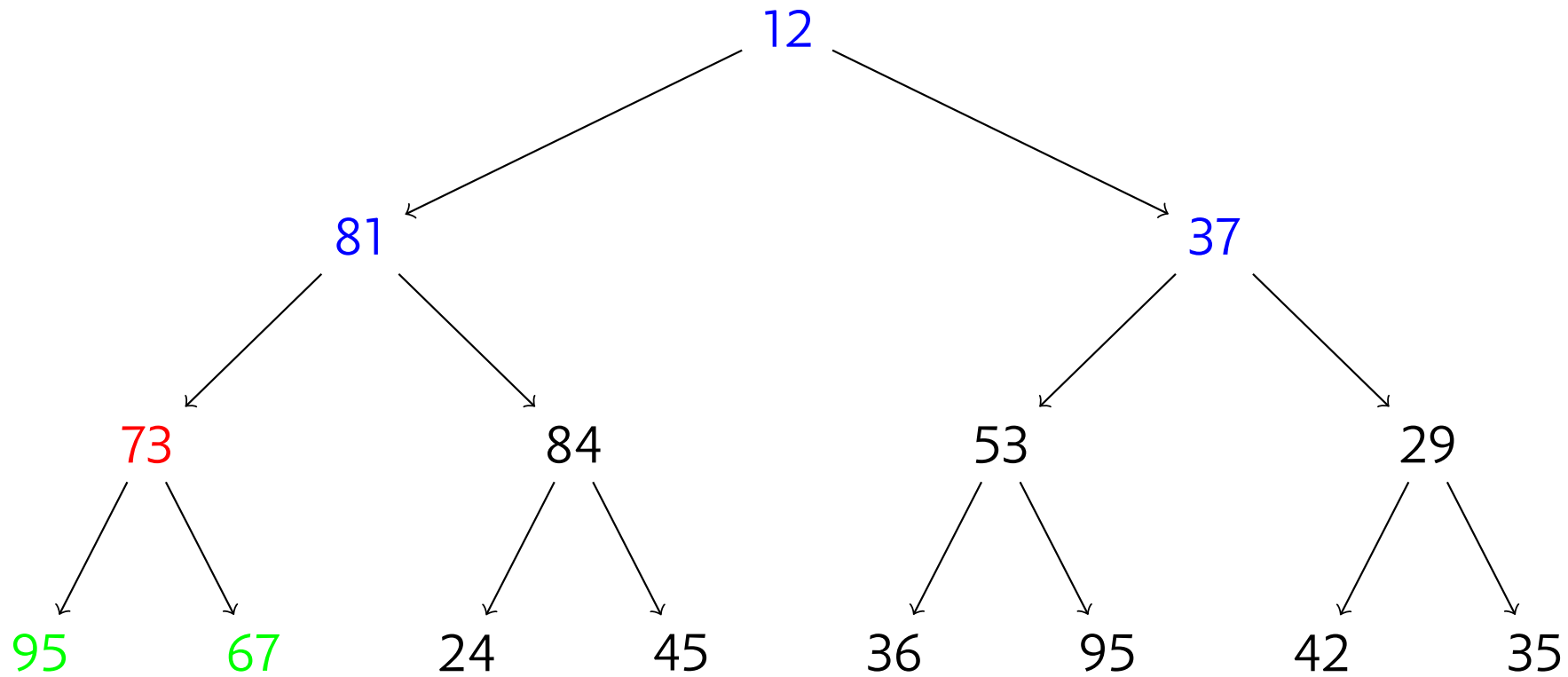
Queue: 37 73 84

Searching via BFS



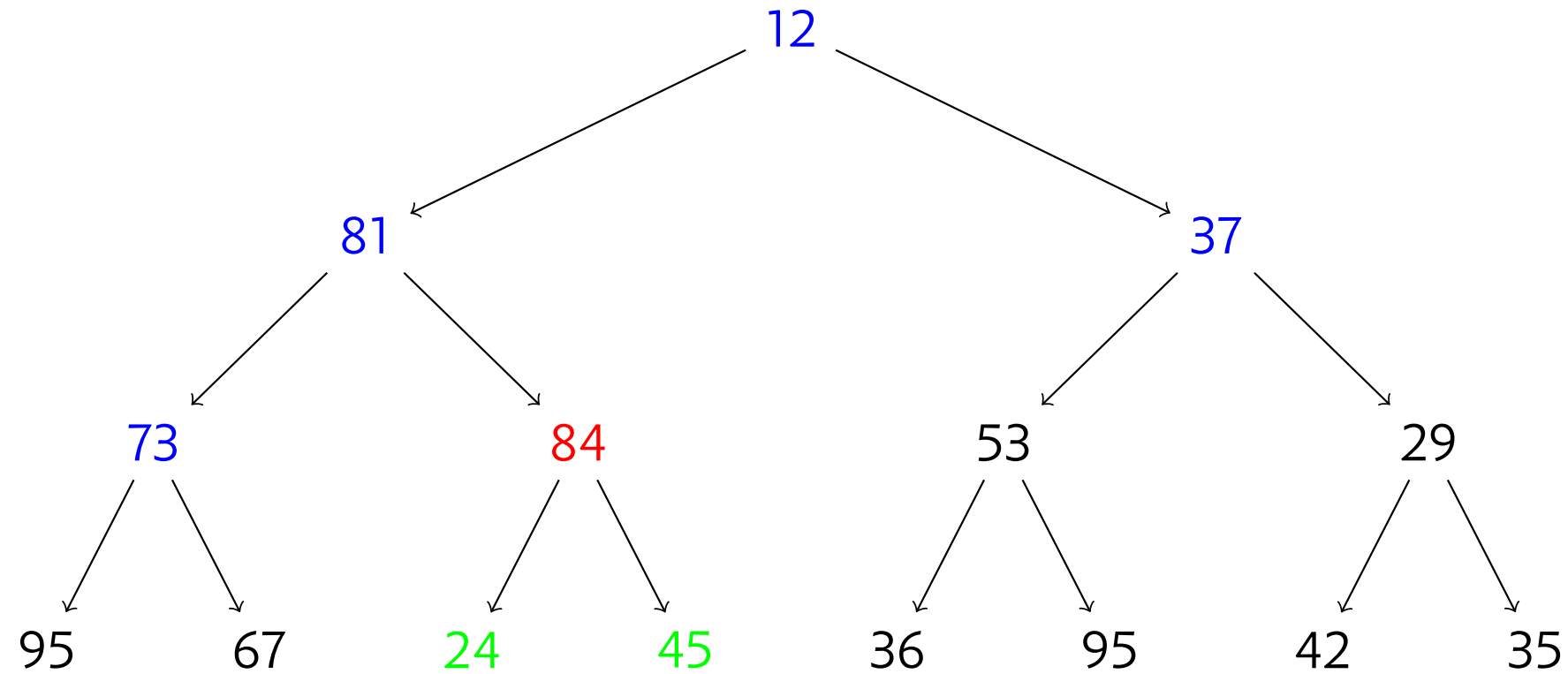
Queue: 73 84 53 29

Searching via BFS



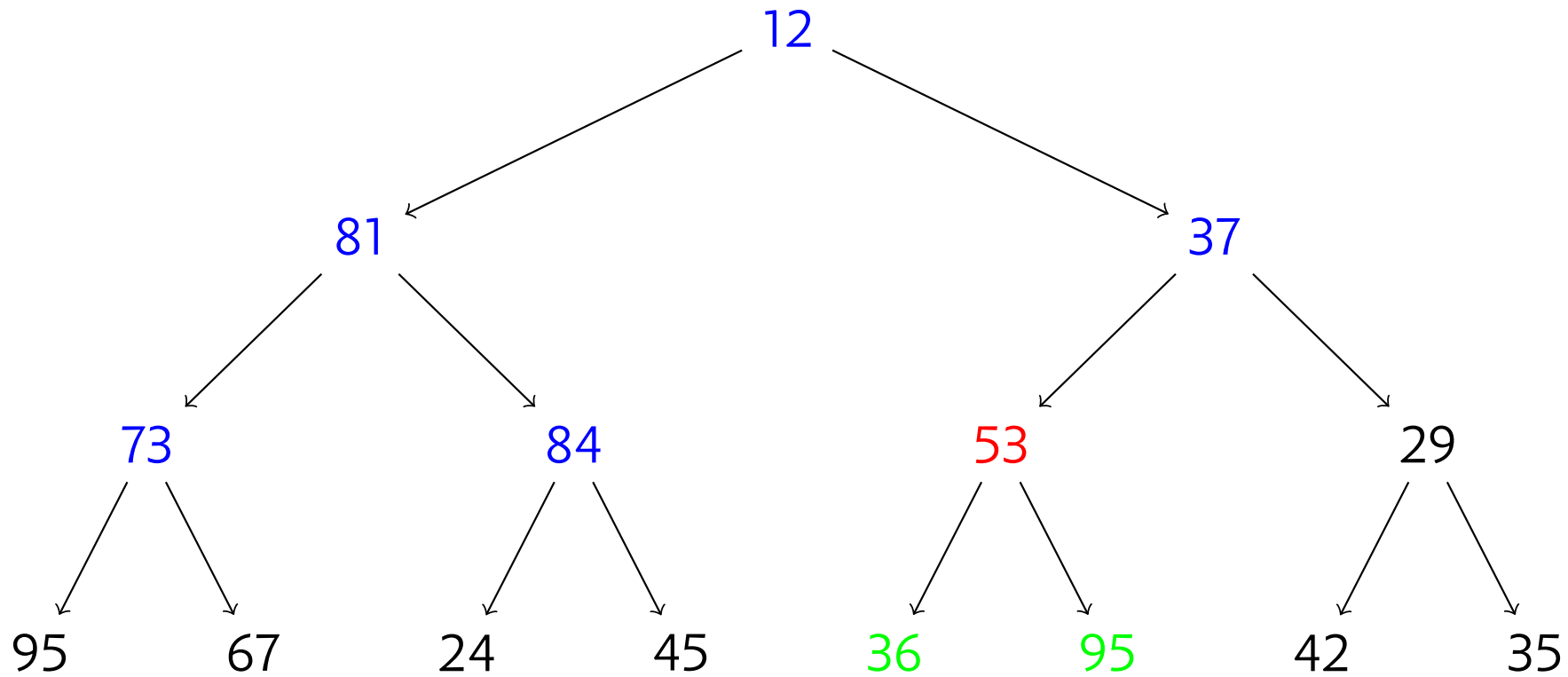
Queue: 84 53 29 95 67

Searching via BFS



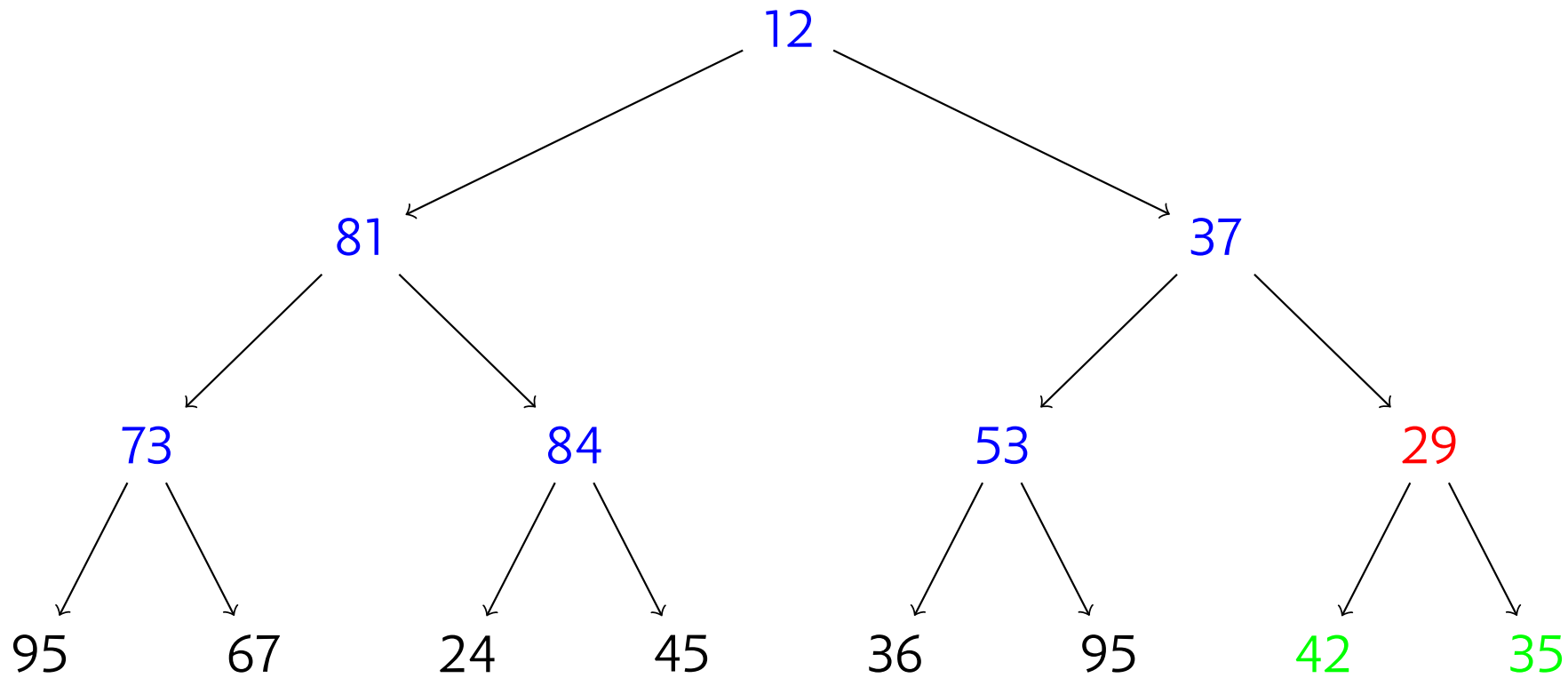
Queue: 53 29 95 67 24 45

Searching via BFS



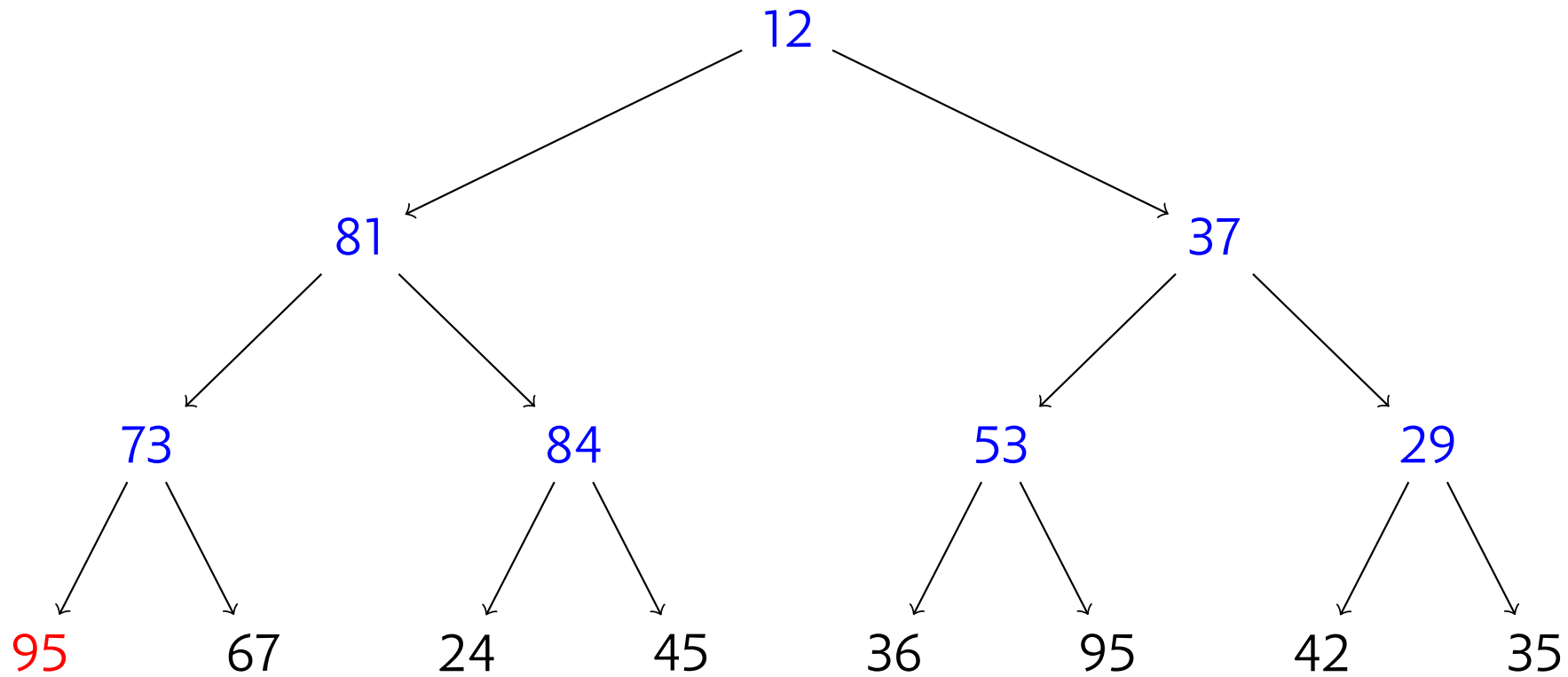
Queue: 29 95 67 24 45 36 95

Searching via BFS



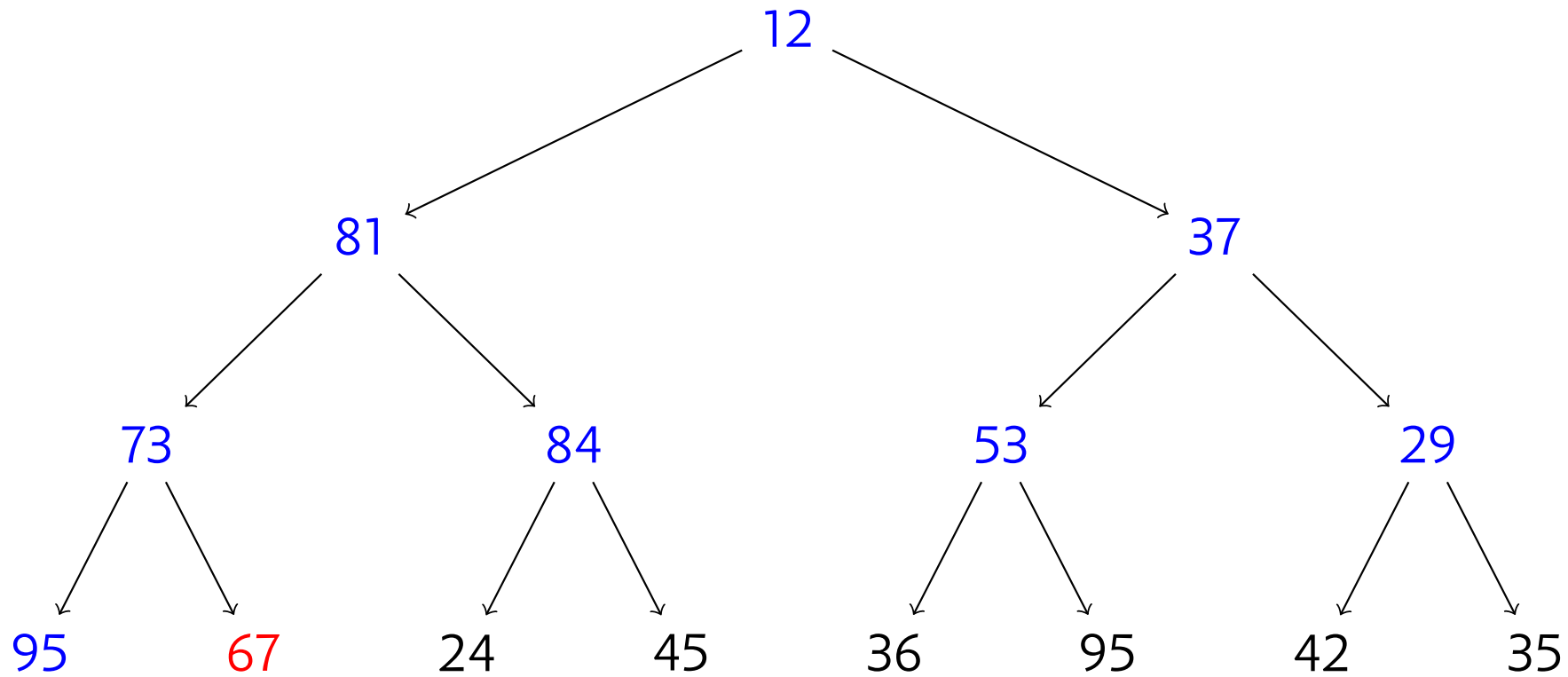
Queue: 95 67 24 45 36 95 42 35

Searching via BFS



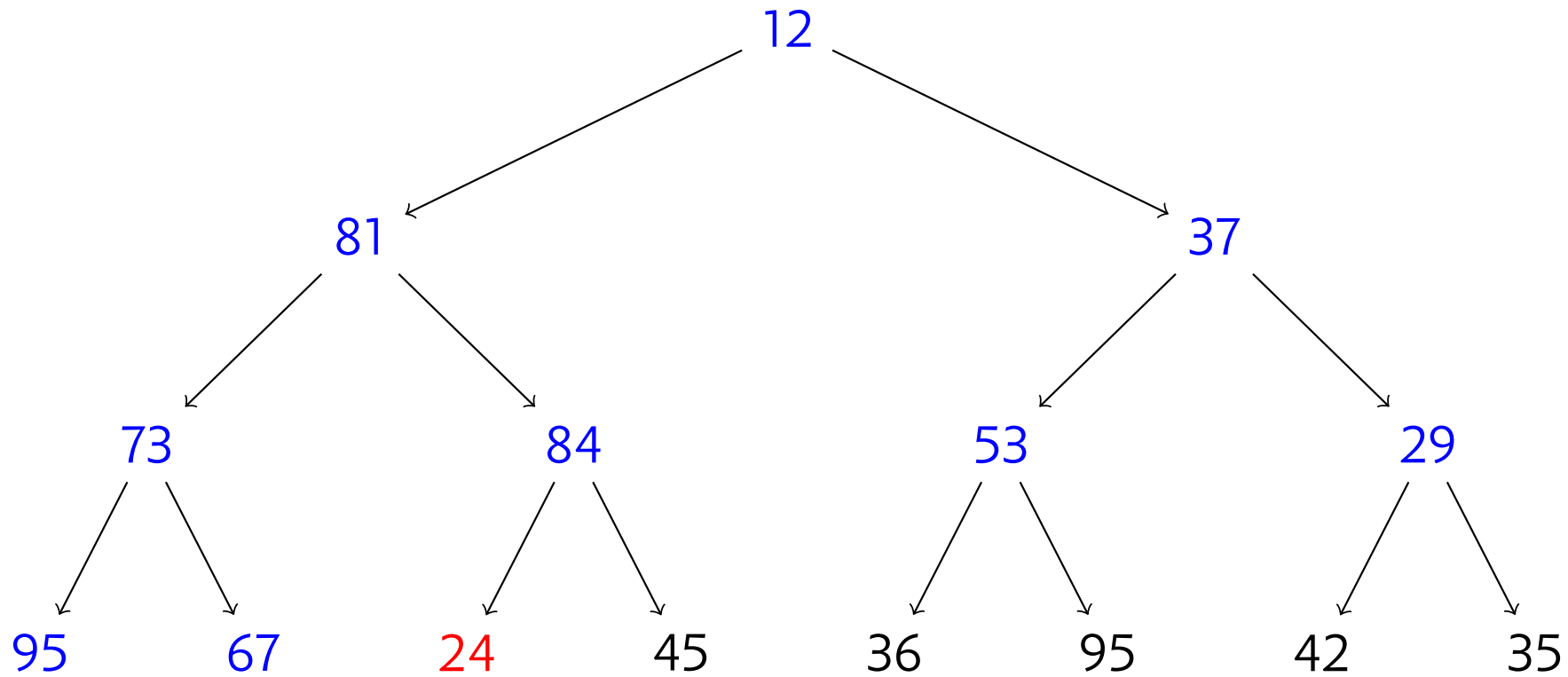
Queue: 67 24 45 36 95 42 35

Searching via BFS



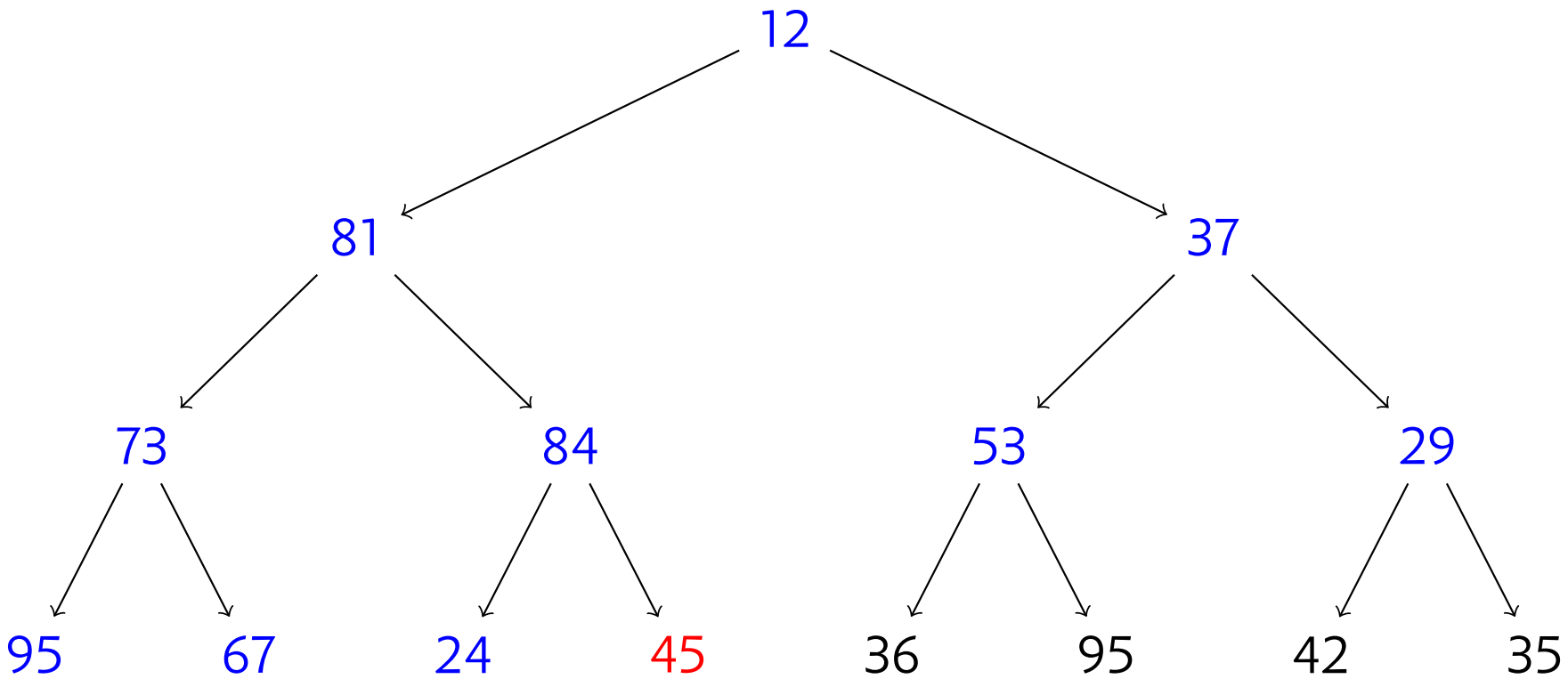
Queue: 24 45 36 95 42 35

Searching via BFS



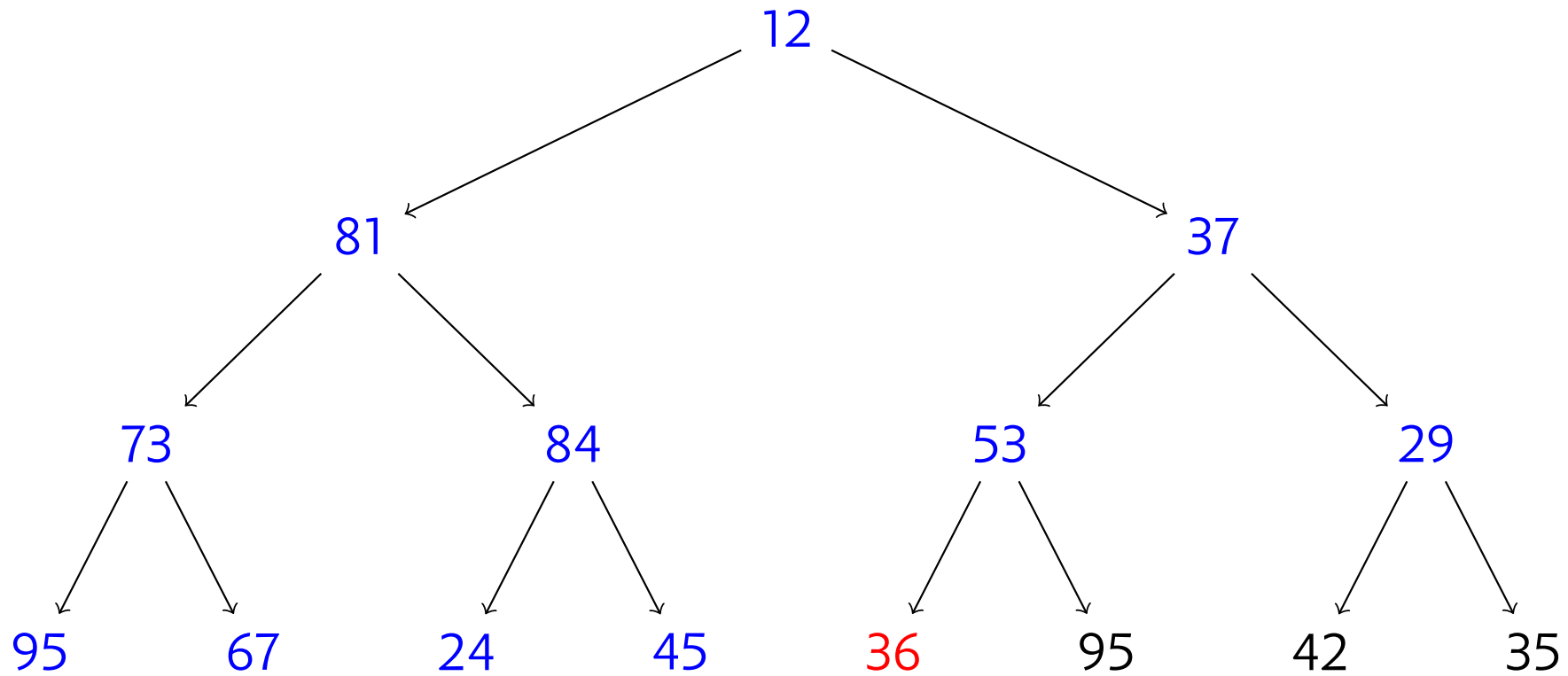
Queue: 45 36 95 42 35

Searching via BFS



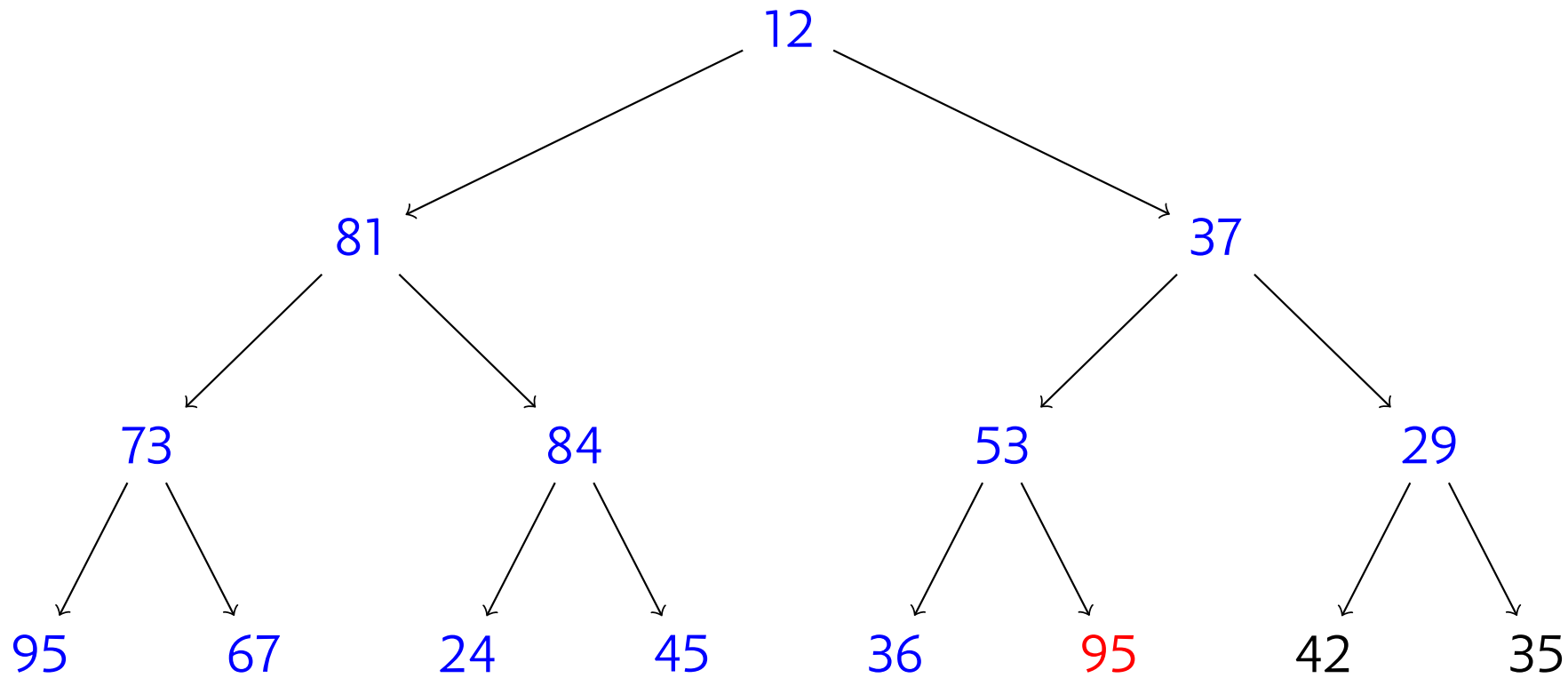
Queue: 36 95 42 35

Searching via BFS



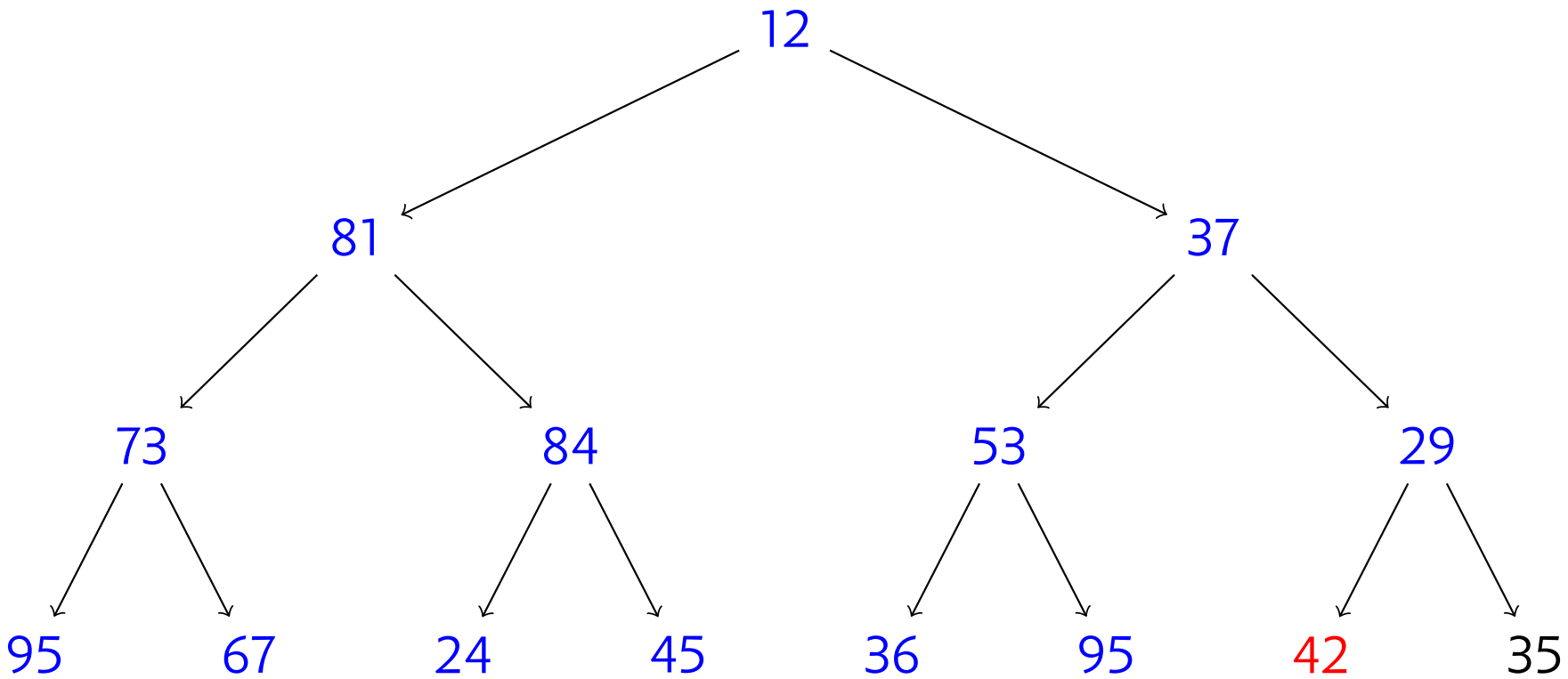
Queue: 95 42 35

Searching via BFS



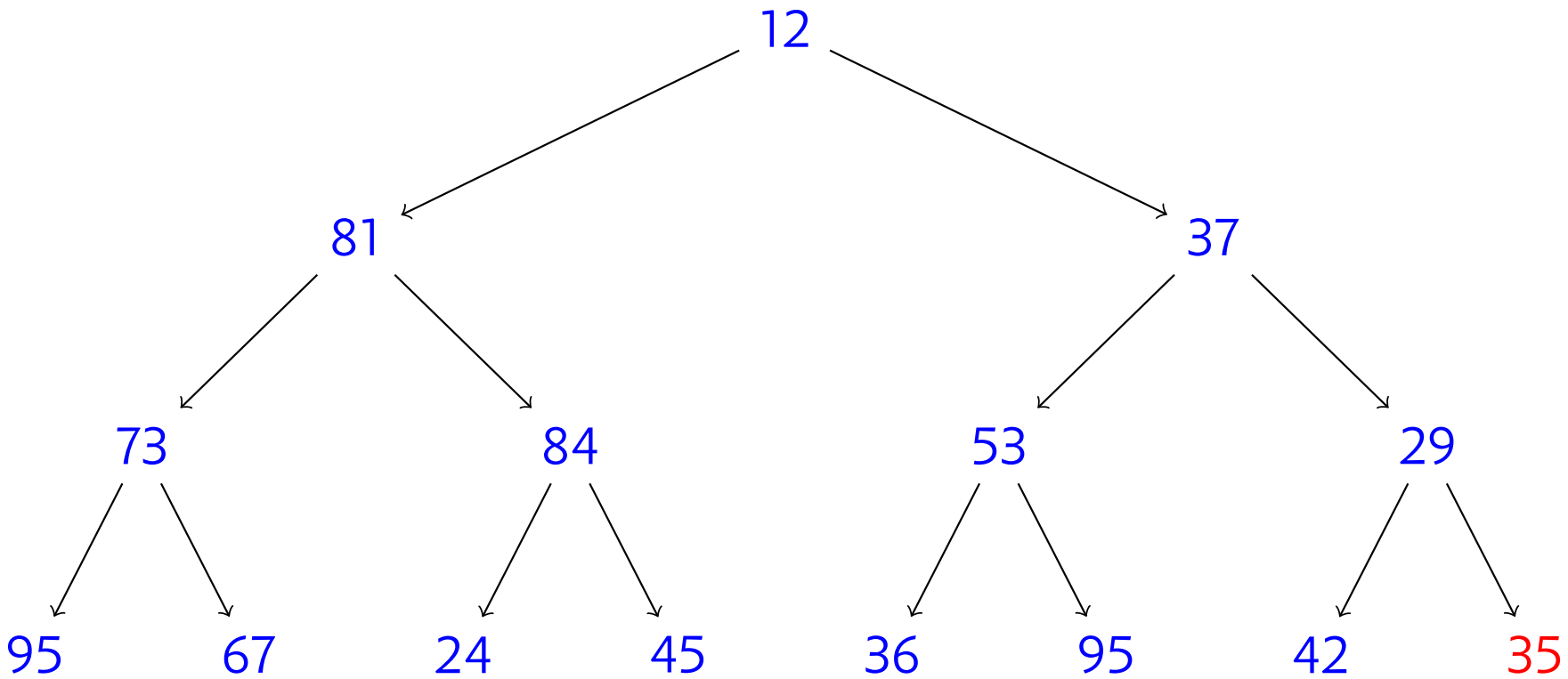
Queue: 42 35

Searching via BFS



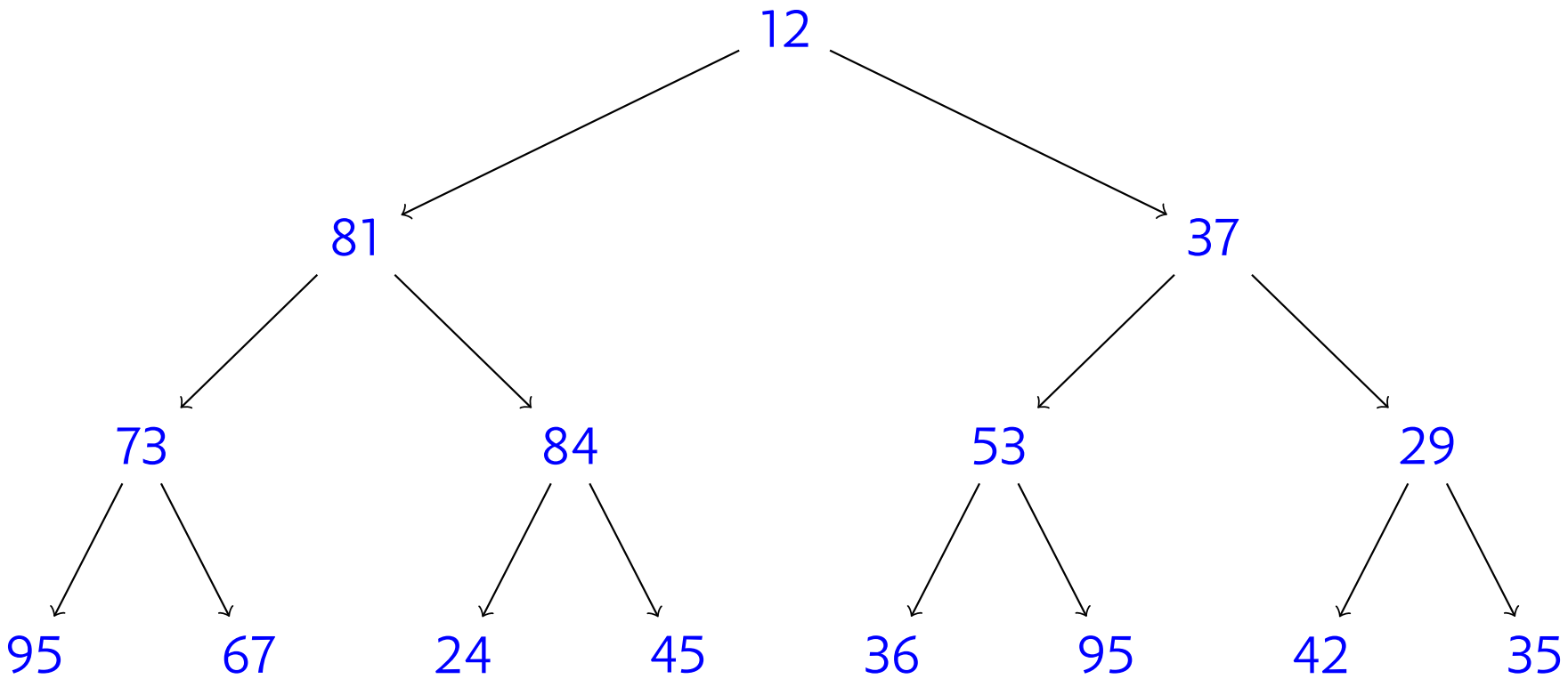
Queue: 35

Searching via BFS



Queue:

Searching via BFS



Queue:

Things to know.

Pros

- Finds node closest to root.
- Handles infinite trees and back edges.

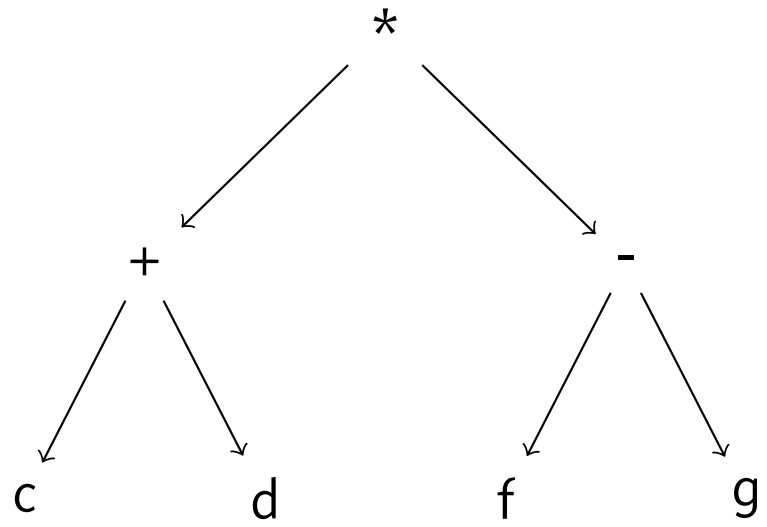
Cons

- Can use a lot of memory. (How much?)
- Usually takes a bit longer to write.

- BFS is also called “level order traversal”.

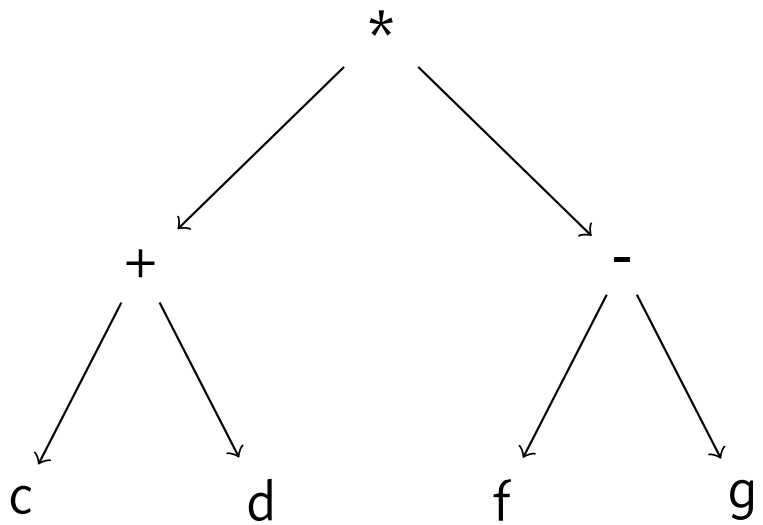
Three Kinds

- There are three kinds of traversals you should know.



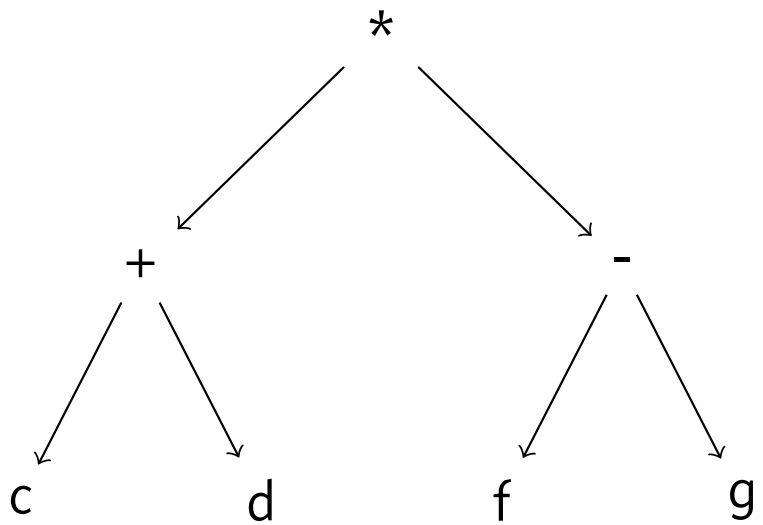
- Preorder: * + c d - f g — used by Scheme and Lisp
- Inorder: c + d * f - g — used by scientific calculators
- Postorder: c d + f g - * — used by Reverse Polish Notation

Preorder



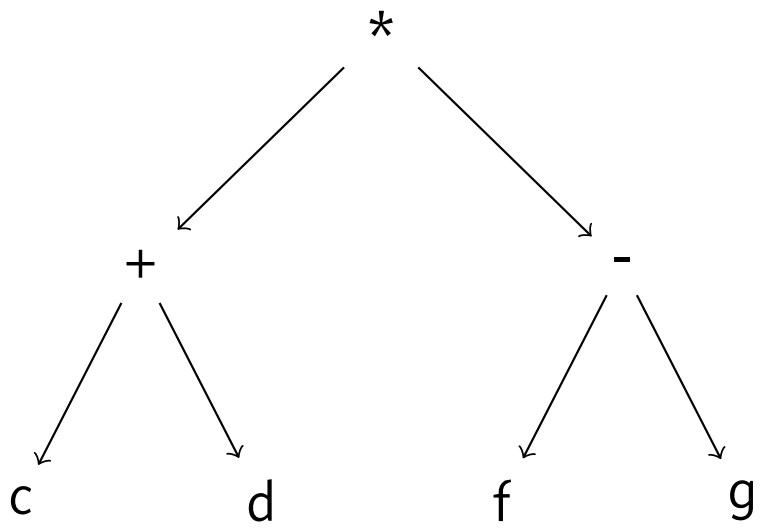
- Preorder: * + c d - f g — used by Scheme and Lisp
- How can you code this traversal scheme?
- Note: if you can distinguish leaves from nodes, you can reconstruct the tree from the traversal!

Postorder



- Postorder: c d + f g - * — used by Reverse Polish Notation
- How can you code this traversal scheme?
- Note: if you can distinguish leaves from nodes, you can reconstruct the tree from the traversal!

Inorder



- Inorder: c + d * f - g — used by scientific calculators

This one is tricky! Consider: can you reconstruct the tree given the string?