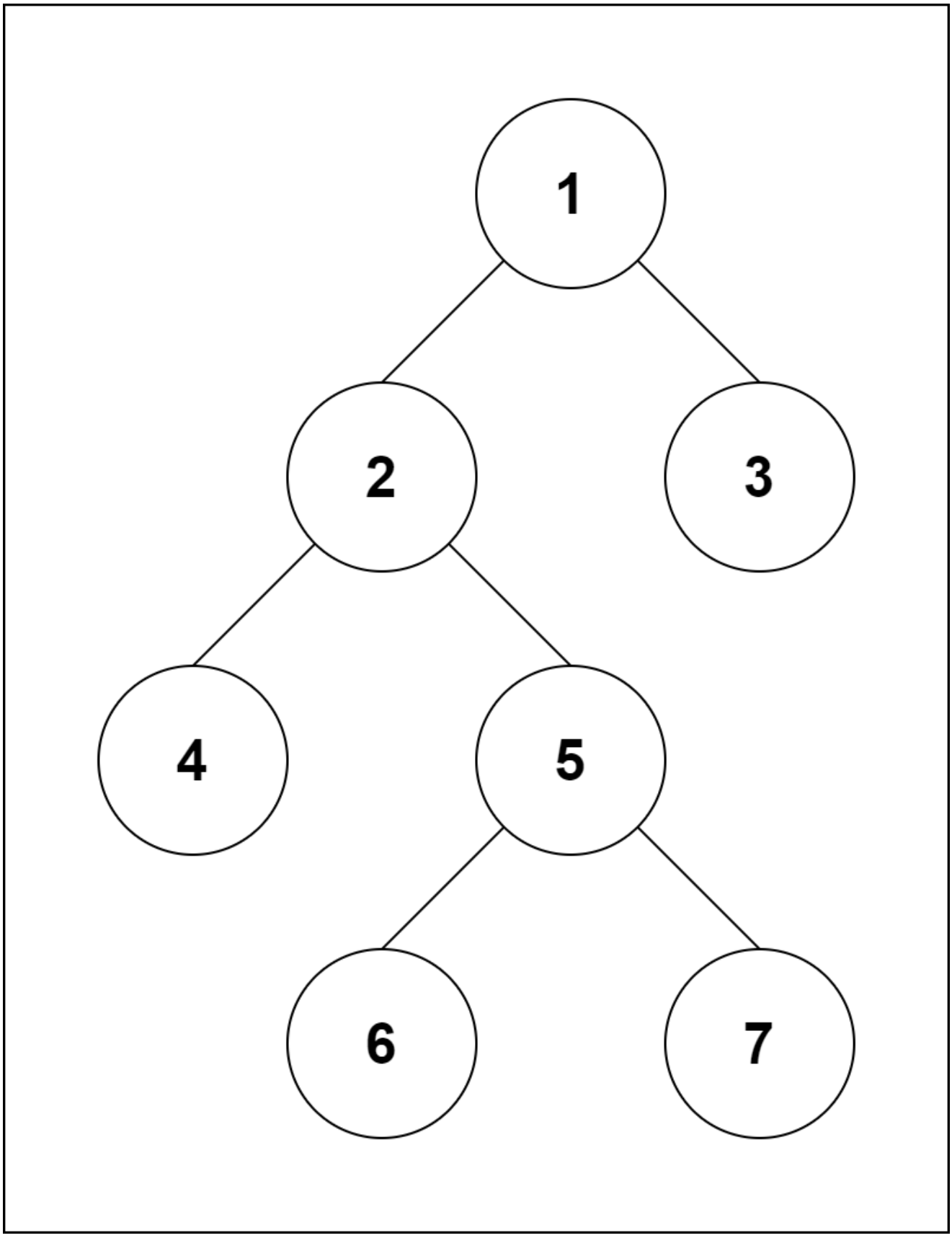
Full Binary Tree

Alright, let's talk about a special type of binary tree called a "full binary tree."

In a full binary tree, every parent node, which we can also call an internal node, has either two children or no children at all. It's like a family tree where every parent has either two kids or none.



Another name for a full binary tree is a "proper binary tree." So, whether you call it a full binary tree or a proper binary tree, you're talking about a tree where each parent has either two children or no children. It's a neat and organized way to structure data.

Full Binary Tree Theorems

Let's understand some important theorems about a full binary tree.

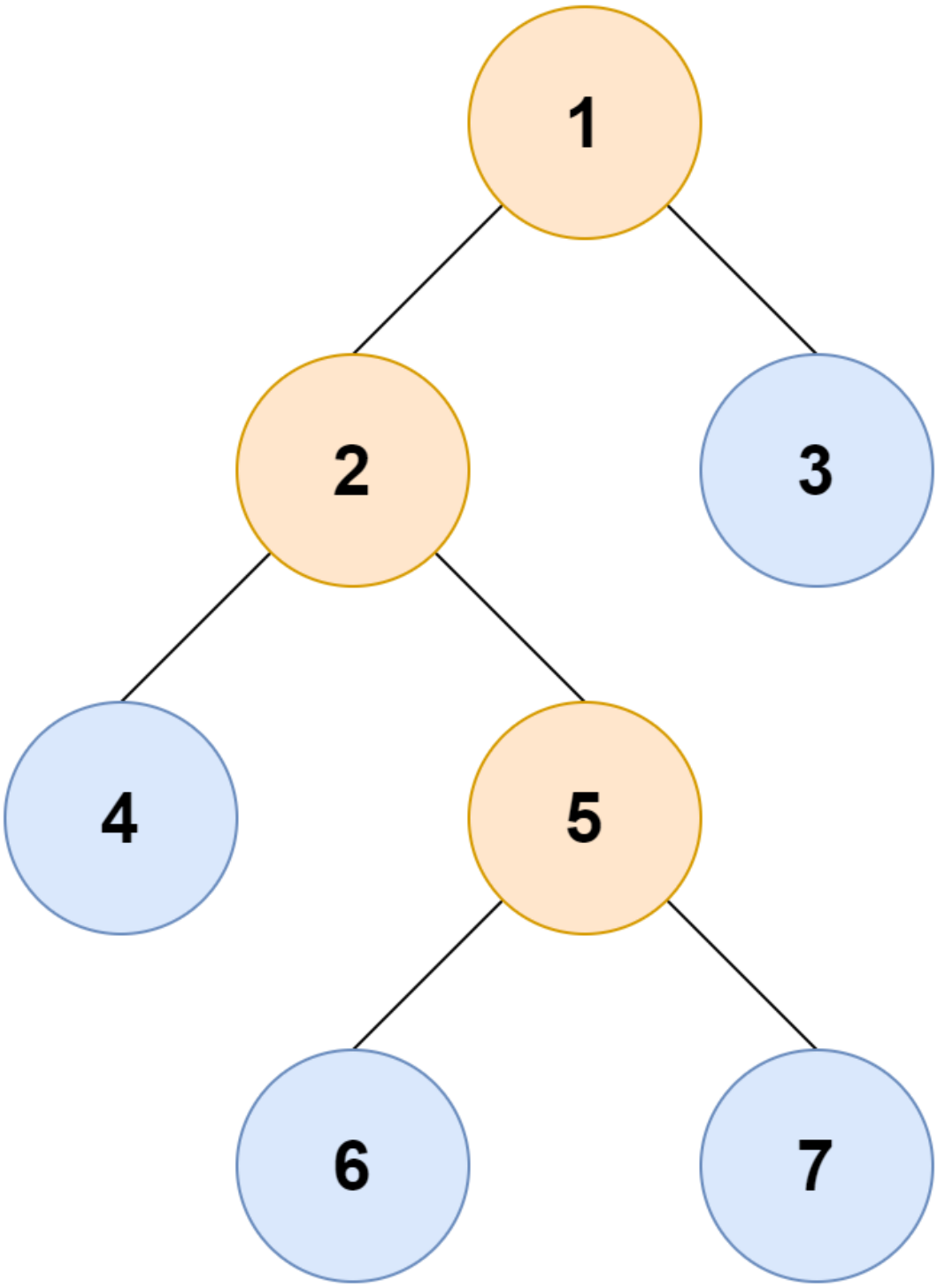
In these theorems:

- "i" represents the number of internal nodes.
- "n" is the total number of nodes.
- "l" is the number of leaves (those are the nodes with no children).
- "λ" is the number of levels in the tree.

Here are the theorems:

1. The number of leaves in a full binary tree is equal to "i + 1". So, if you count the leaves, you'll have one more than the internal nodes.
2. The total number of nodes in a full binary tree is "2i + 1". This includes both internal nodes and leaves.
3. The number of internal nodes in a full binary tree is "(n – 1) / 2". You can find this by subtracting 1 from the total number of nodes and then dividing by 2.
4. The number of leaves in a full binary tree is "(n + 1) / 2". This is also found by manipulating the total number of nodes.
5. The total number of nodes in a full binary tree is "2l – 1", where "l" is the number of leaves.
6. The number of internal nodes in a full binary tree is "l – 1", again based on the number of leaves.
7. The number of leaves in a full binary tree is at most "$2^{\lambda - 1}$", where "λ" is the number of levels in the tree.

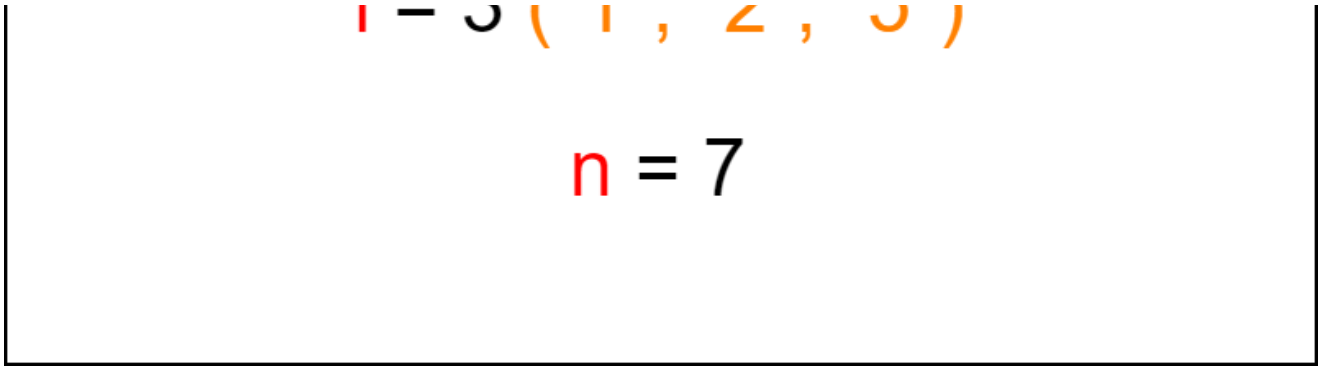
These theorems help us understand the relationships between the various components of a full binary tree, such as internal nodes, leaves, and levels.



$\lambda = 4$

$l = 4$ ("3", "4", "6", "7")

$i = 3$ ("1", "2", "5")



C++ Example

The following code is for checking if a tree is a full binary tree.

```
// Checking if a binary tree is a full binary tree in C++

#include <iostream>
using namespace std;

struct Node {
  int key;
  struct Node *left, *right;
};

// New node creation
struct Node *newNode(char k) {
  struct Node *node = (struct Node *)malloc(sizeof(struct Node));
  node->key = k;
  node->right = node->left = NULL;
  return node;
}

bool isFullBinaryTree(struct Node *root) {

  // Checking for emptiness
  if (root == NULL)
    return true;

  // Checking for the presence of children
  if (root->left == NULL && root->right == NULL)
    return true;

  if ((root->left) && (root->right))
    return (isFullBinaryTree(root->left) && isFullBinaryTree(root->right));

  return false;
}
```

```
int main() {
  struct Node *root = NULL;
  root = newNode(1);
  root->left = newNode(2);
  root->right = newNode(3);
  root->left->left = newNode(4);
  root->left->right = newNode(5);
  root->left->right->left = newNode(6);
  root->left->right->right = newNode(7);

  if (isFullBinaryTree(root))
    cout << "The tree is a full binary tree\n";
  else
    cout << "The tree is not a full binary tree\n";
}
```