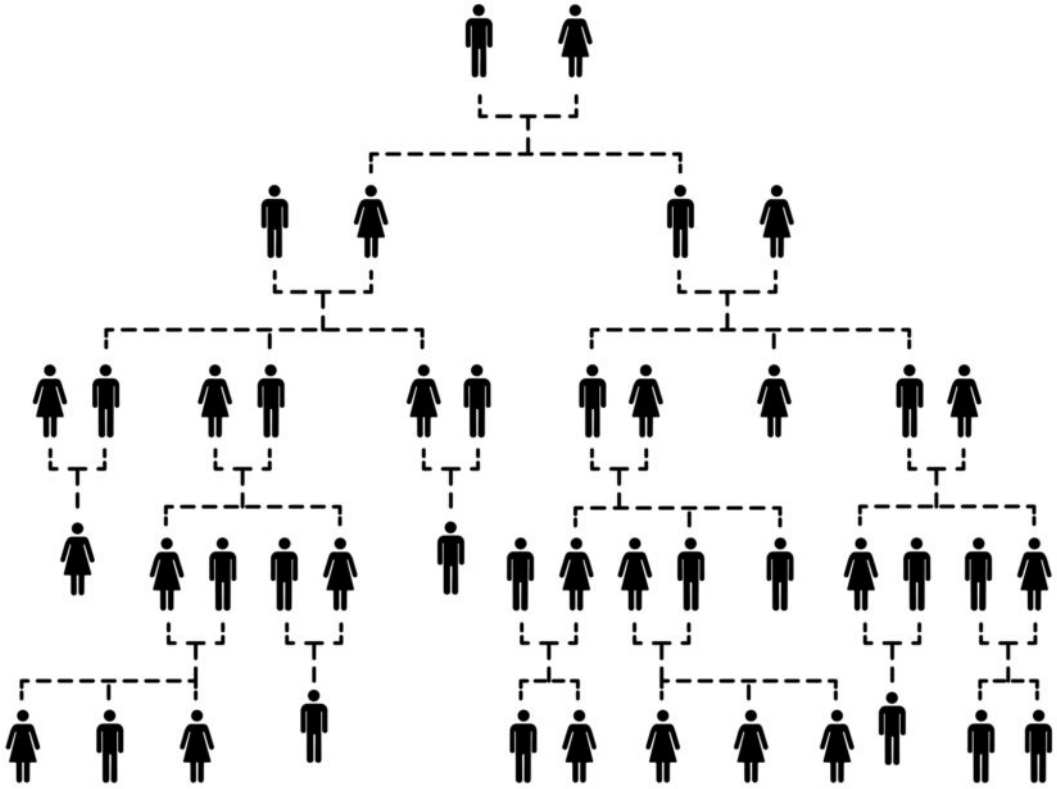
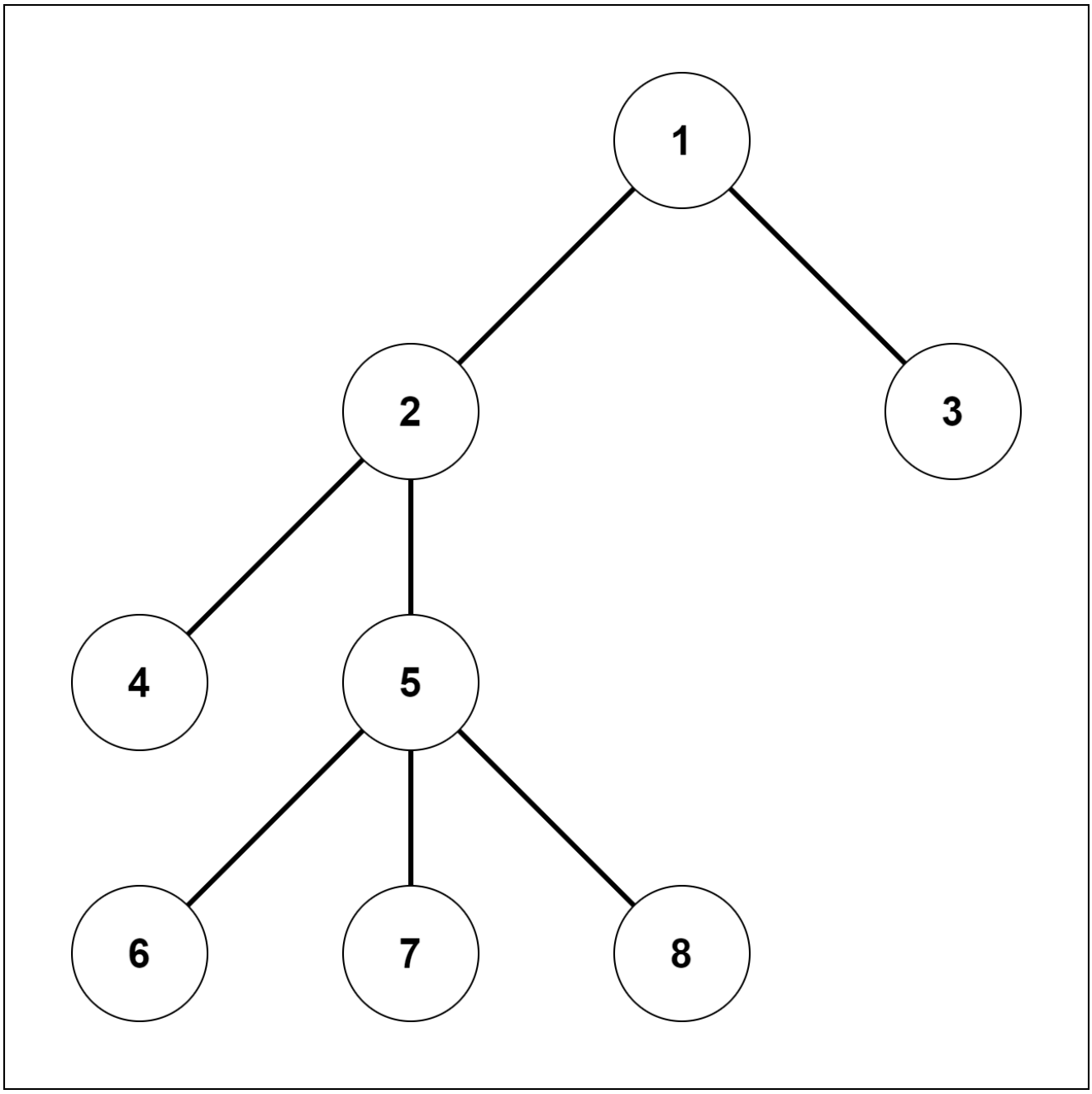
Tree Data Structure



Imagine a tree. Not the kind with leaves and branches, but a special kind of structure we use in computer science. This structure is like a family tree, where you have a bunch of individuals (we call them "nodes") and lines connecting them (we call these "edges").



Now, in this tree structure, each individual or node is connected to others below it. It's like a parent connecting to their children and those children can have their own children, and so on. This is why we call it hierarchical – it's like a family tree where there's a clear order and hierarchy.

So, in short, a tree in computer science is a way to organize data in a hierarchical, structured manner, just like a family tree helps organize family relationships. It's a useful tool for solving all sorts of problems in computing!

Why Tree Data Structure?

So, we've got different ways of organizing data in computer science. You might have heard of things like arrays, linked lists, and so on. Now, these are what we call "linear" data structures because they store data one after the other, like a line of books on a shelf.

Here's the thing: when you want to do something with these linear structures, like find a specific piece of data, it can take a while if there's a lot of data. Think of it like finding a specific book in a long row of books. You have to go through each one until you find the right one. That can be slow, especially when there's a ton of data.

But, in our modern world of computing, we often need things to be fast and efficient. That's where trees come in.

A tree is a different kind of structure. It's not linear; it's more like a family tree with branches going in different directions. In a tree, data is organized in a way that makes it quicker and easier to find what you're looking for.

So, when we're dealing with lots of data and we want things to happen fast, we use tree structures because they help us do just that.

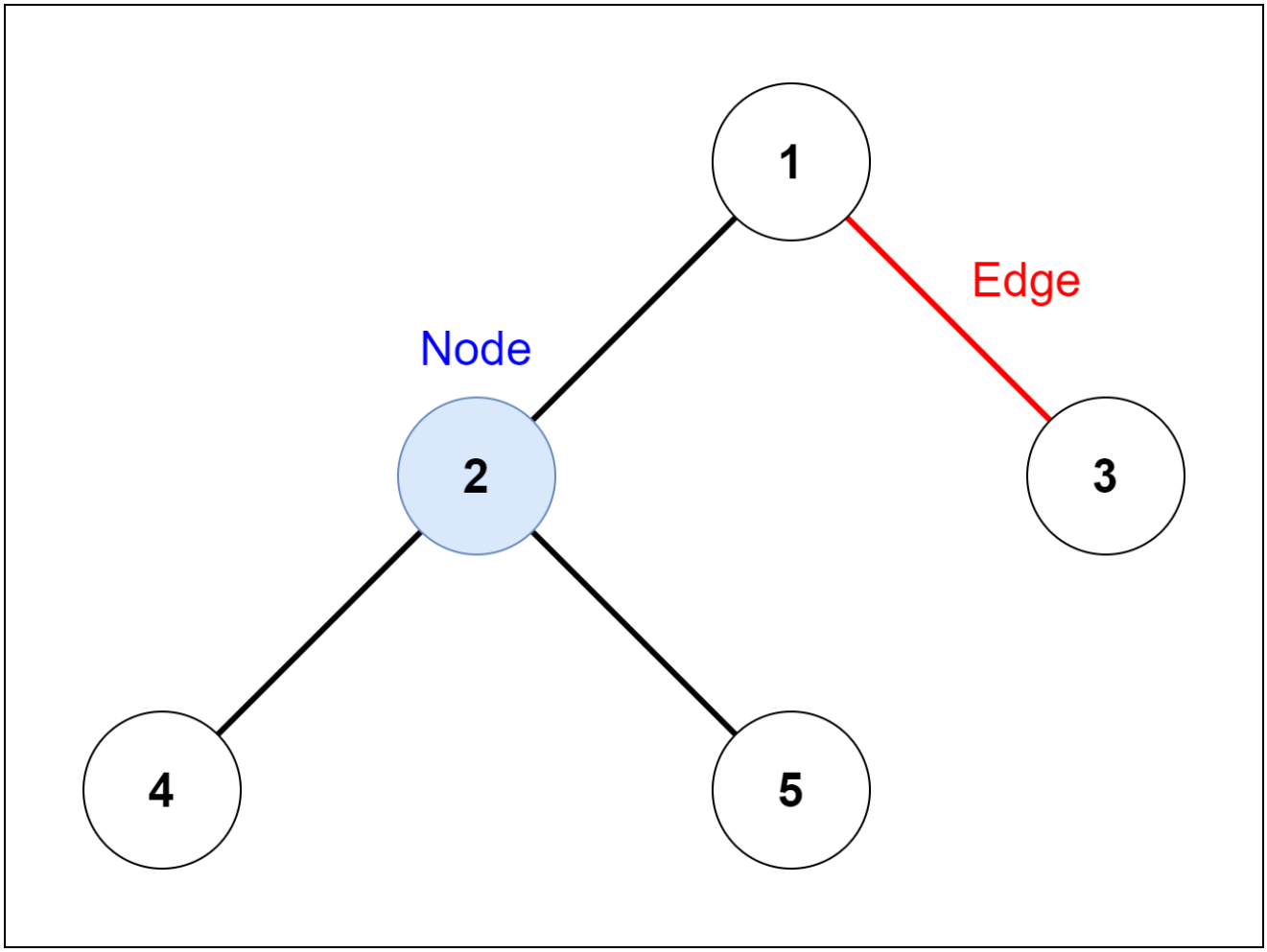
Tree Terminologies

Node

Think of a node as a building block of a tree. Each one holds some information and can point to other nodes. Now, some nodes are special. The ones at the very ends, they're called "leaf nodes" or "external nodes." They don't point to any other nodes. But if a node has at least one child, we call it an "internal node."

Edge

An edge is like a bridge that connects two nodes. It's the line that says, "Hey, you can get from this node to that node."



Root

The root is the big boss node, right at the top. Imagine it as the head of the tree.

Height of a Node

This is like measuring how far you'd have to climb down from a particular node to reach the very bottom of the tree. Just count the steps, or in our case, the edges.

Depth of a Node

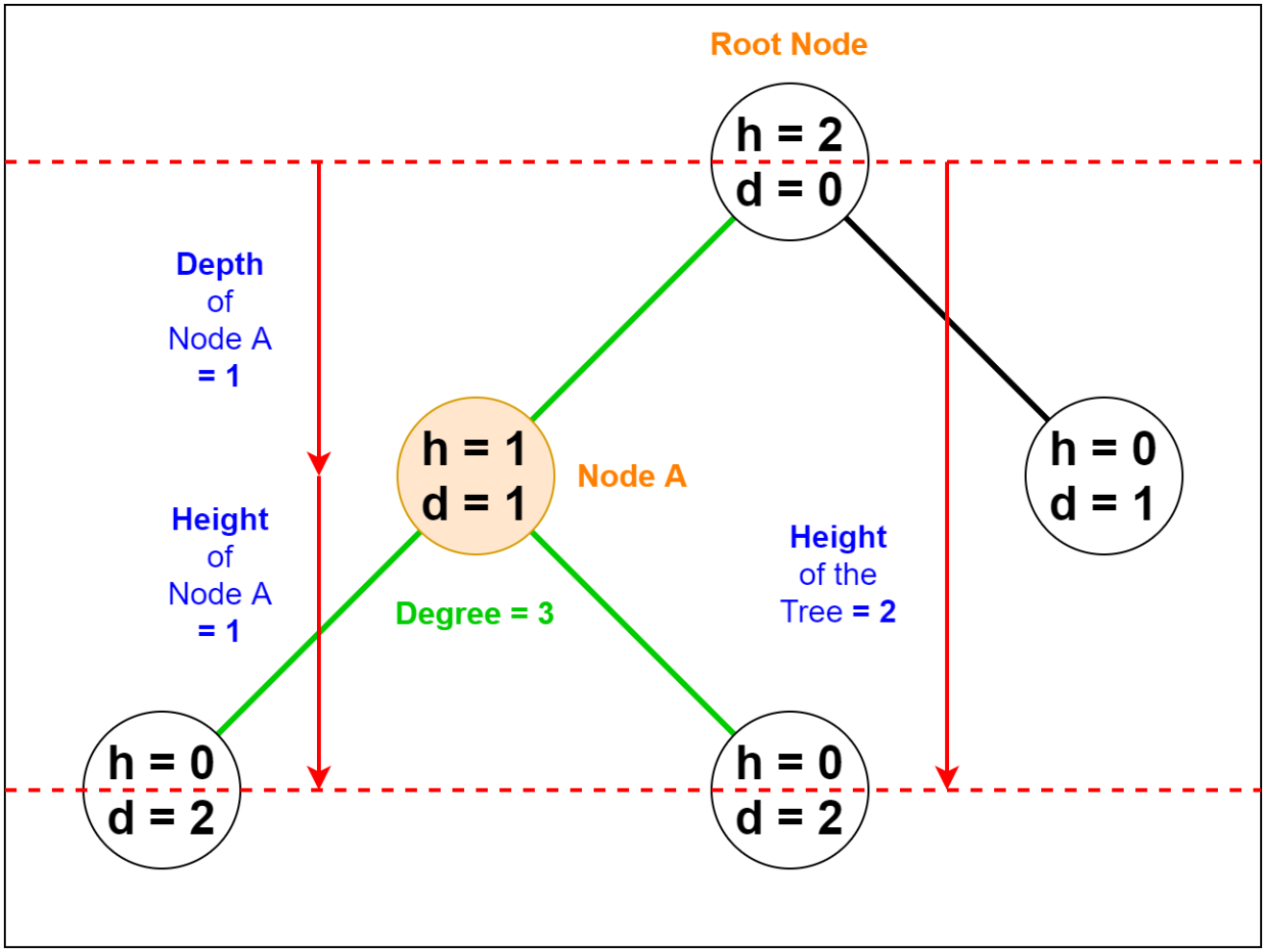
Now, this is kinda the opposite. It's like counting how many steps you need to climb up from the root to reach a specific node.

Height of a Tree

This is just like the height of the root, or you can think of it as how deep the deepest point in the tree goes.

Degree of a Node

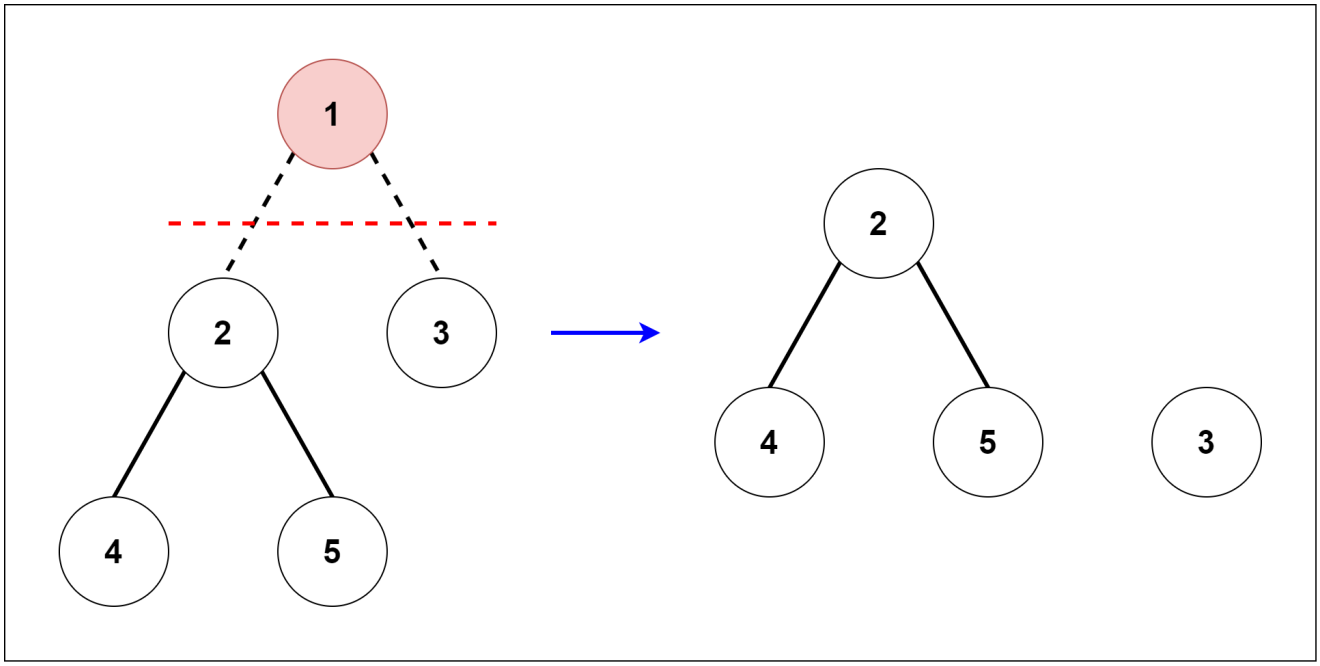
Think of this like counting how many roads lead out of a city. For a node, it's the total number of branches it has.



Forest

Imagine you have a bunch of trees, but they're not all connected to each other. We call this a forest. It's like having several little woods instead of one big forest. And you can make a

forest by chopping down the big tree at the top, which is like cutting the root.



Tree Applications

Let's talk about where trees come in handy:

1. **Binary Search Trees (BSTs)**: These are like the Sherlock Holmes of data structures. They help us quickly figure out if something's in a set or not. Imagine you have a massive phonebook, and you want to see if your friend's number is in there. A BST helps you do this super fast.
2. **Heap**: This is a type of tree, but not like the ones you see in a park. It's more like a mountain. We use it for something called heap sort, which is a way to sort a jumbled-up list of things.
3. **Tries**: These are like special trees, kind of like family trees but for words. Modern routers use them to figure out where to send your internet data. It's like a super-smart postman finding the right mailbox for your letters.
4. **B-Trees and T-Trees**: These are like the super organizers of the database world. You know how your parents keep all the important stuff in those labeled folders? That's what these trees do for data in big, fancy databases.
5. **Syntax Trees**: Imagine you're writing a super cool program, but you miss a comma or something. Compilers, the programs that turn your code into something the computer understands, use something called a syntax tree to catch those mistakes. It's like having a grammar teacher check your work.