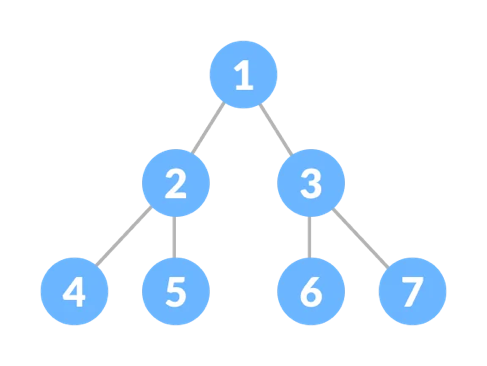
# **Kinds of Binary Trees**

## **Perfect Binary Tree**

* A **perfect binary tree** is a type of binary tree in which

1. **All nodes have exactly two children**
2. **All the leaf nodes are at the same level**



### **Number of Nodes**

* A perfect binary tree of height *h* has the following properties:
  + 2*h* total nodes at **current level** *h* (number of nodes only on this row)
  + 2*h+1* − 1 total nodes **up to level *h*** (number of nodes in this and all previous rows)
* If we traverse down the tree from the root to the leaves, we see that at every level, the number of nodes double.
  + Level 0 has 1 node.
  + Level 1 has 2 nodes.
  + Level 2 has 4 nodes.
  + and so on…
* From this, we can see a pattern.
* The number of nodes on a level is based on a power of 2.
* For every level we go down, we double the number of nodes.
* The top level contains the root, which is the only node which contains no parent.
  + Total Nodes At Current Level: We account for this in the calculation of total nodes at current height h (2h) if we call the top level of tree height 0. The number of nodes at height 0 is 20 = 1.
  + Total Nodes Up To Current Level: We account for this in the calculation of total nodes at height h (2h+1 − 1) by adding that extra – 1 after the 2h+1. The – 1 account for the top level, which only has 1 node

### **Analysis**

|  |  |  |
| --- | --- | --- |
| Height | Number of Nodes at This Level | Number of Nodes at This Level and All Previous |
| 0 | 20 = 1 | 20+1 – 1 = 1 |
| 1 | 21 = 2 | 21+1 – 1 = 3 |
| 2 | 22 = 4 | 22+1 – 1 = 7 |
| 3 | 23 = 8 | 23+1 – 1 = 15 |

Chart, line chart

Description automatically generatedChart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

0

1

2

3

Chart, scatter chart

Description automatically generated

|  |  |  |
| --- | --- | --- |
| h | 2h | 2h+1 – 1 |

* Looking at the diagram, we can derive that
  + The number of nodes on level 𝑖 of a binary tree is 2*h*.
  + The total number of nodes in a perfect binary tree of height h is Text

    Description automatically generated
* So if we have 𝑛 nodes in a perfect tree, we can use the formula 𝑛 = 2h+1 − 1 to conclude that h = 𝑂(log 𝑛).

We use the formula

2h = n 🡪 log2n = h

To derive O(log N) from the formula 𝑛 = 2h+1 – 1, we first do a little math by adding 1 to both sides

𝑛 = 2h+1 – 1

n + 1 = 2h+1

Then using the logarithmic formula

n + 1 = 2h+1 🡪 log2n = h + 1

* This implies that the height of a complete binary tree is ⌊log *N*⌋, which is clearly *O*(log *N*).

https://courses.cs.vt.edu/~cs3114/Fall10/Notes/T03a.BinaryTreeTheorems.pdf

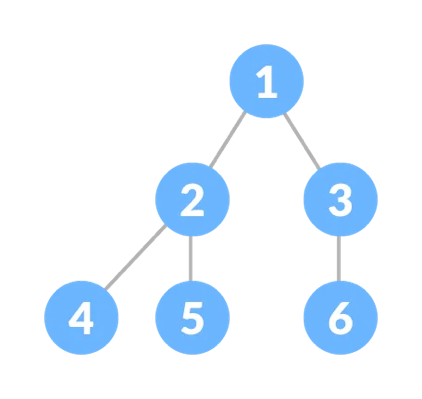
A perfect tree with 𝑛 nodes has height 𝑂(log 𝑛).  
A similar argument can show the same statement for complete trees.

**Examples**

* Example: number of nodes of height 3
  + 23 = 8 nodes
* Example: total number of nodes of height 4
  + 24+1 – 1 = 8 nodes

## **Complete Binary Tree**

* A complete binary tree has every level, except possibly the deepest, completely filled.
* If the last level is not filled, then the nodes in the last level are as far left as possible.



**Complete Binary Tree Properties**

* A perfect binary tree of height *h* has the following property:
  + There are **between** 2*h* and 2*h*+1 − 1 nodes at height *h*
* This implies that the height of a complete binary tree is ⌊log *N*⌋, which is clearly *O*(log *N*).
* This property is especially true when using the total height of the tree *h* to try and find the number of leaves a binary tree has.

**Example**

* We can say for certain that this complete binary tree with height 3 has between 23 and 23+1 - 1 nodes.
* This means that the tree can have between 8 and 15 nodes, which holds to be true.
* H is the first possible node that holds the completeness property at level 3, and 15 is the rightmost node we could put in the tree at height 3.

A picture containing looking, linedrawing

Description automatically generated

8th node

15th node at rightmost part of tree.

