Tree Breadth-first Search: Introduction

Let's go over the Tree Breadth-first Search pattern, its real-world applications, and some problems we can solve with it.

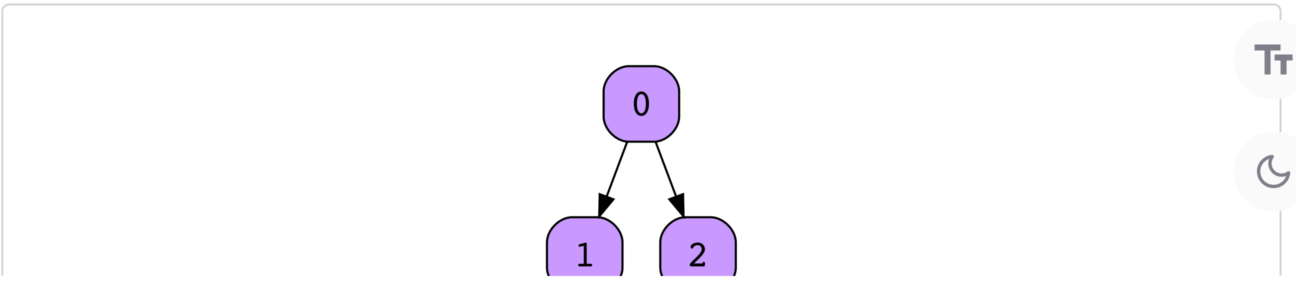
We'll cover the following

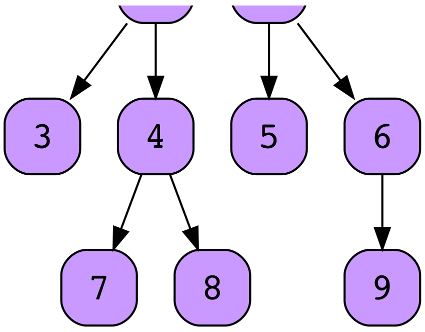
Overview

Tree breadth-first search is an important tree traversal algorithm for finding a node in a tree that satisfies the given constraints. It starts searching at the root node and moves down level by level, exploring adjacent nodes at level $k + 1$.

Essentially, it first visits nodes that are one edge away from the root, followed by nodes that are two edges away, and so on. This helps in efficiently finding neighbor nodes, particularly peer-to-peer networking systems.

Here's how breadth-first search in a tree works:



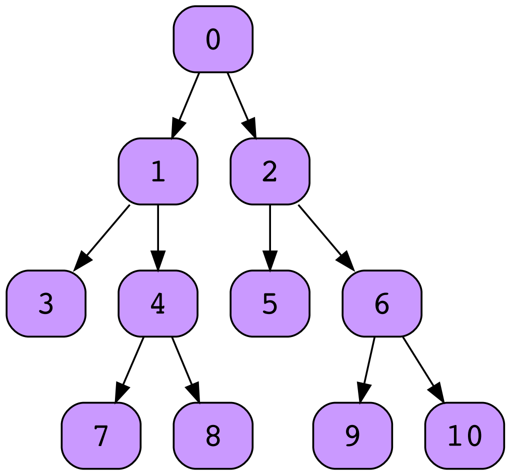


1 of 5

Examples

The following examples illustrate some problems that can be solved with this approach:

Print all nodes of a perfect binary tree in a top-down manner



Order	0	1	2	3	4	5	6	7	8	9	10

1 of 2

Does my problem match this pattern?

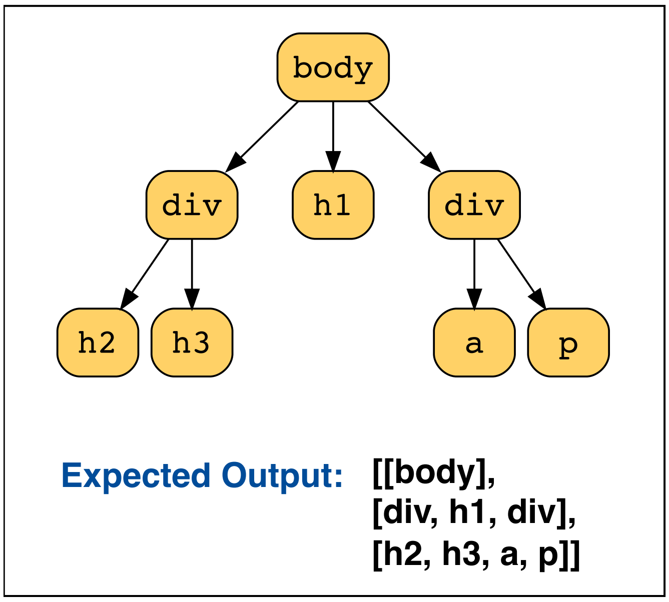
- Yes, if either of these conditions is fulfilled:

    - We have reason to believe that the solution is near the root of the tree.
    - The solution dictates traversing the tree one level at a time, for example, to find the level order traversal of the nodes of a tree, or a variant of this ordering.
- No, if either of these conditions is fulfilled:
    - The tree being searched is very wide.
    - We have reason to believe that the solution is near the leaves of the tree.

Real-world problems

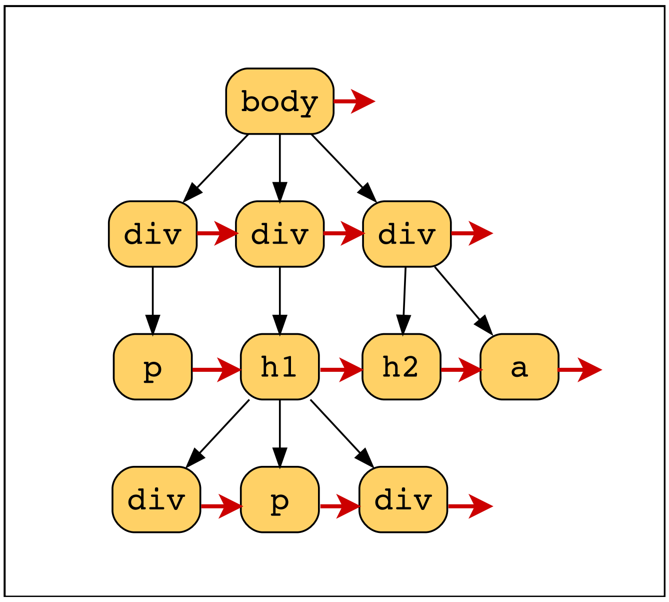
Many problems in the real world use the tree breadth-first search pattern. Let's look at some examples.

- **Traversing DOM Tree I:** We need to figure out a way to traverse the DOM structure that we obtain from a single web page. The HTML can be represented in a tree structure where the children of the HTML tag become the children of a node in the tree. Each tree level can have any number of nodes depending upon the number of nested HTML tags. We need to traverse the nodes in the tree level by level, which can be done using the tree breadth-first search approach.



DOM tree

- **Traversing DOM Tree II:** The number of web pages can be enormous, and traversing all of them consumes a lot of space. So, we create a shadow tree for the DOM where each node has a pointer to the next node to its right on the same level. This next pointer allows us to avoid the queue data structure that we used previously to traverse the tree, resulting in a space-efficient approach.



Red pointers to be obtained

Strategy time!

> **Note:** Select a problem in the left-hand column by clicking it, and then click one of the two options in the right-hand column.

Match The Answer

Select an option from the left-hand side

Find the diameter of a binary tree

Tree Breadth-first Search

Traverse a tree one level at a time and print out the node values in the same order

Serialize the contents of a binary tree for communication, such that the tree can be re-constructed by the receiver

Connect all the siblings of each node in a binary tree

Some other pattern

Reset   Show Solution   Submit

← Back
Maximum Depth of Bi...

Next →
Level Order Traversal ...

✓ Mark as Completed