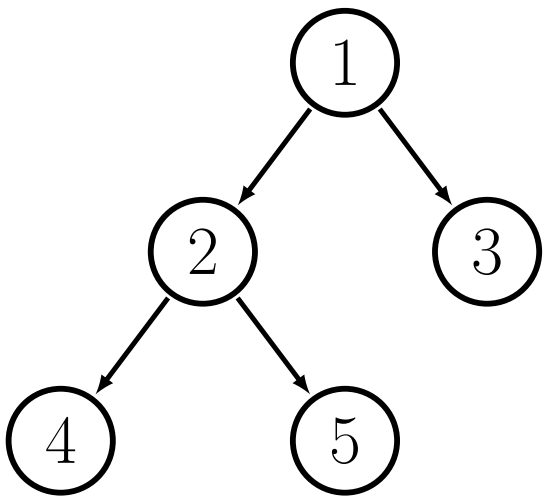
Theory: Tree traversals

15 minutes 8 / 8 problems solved

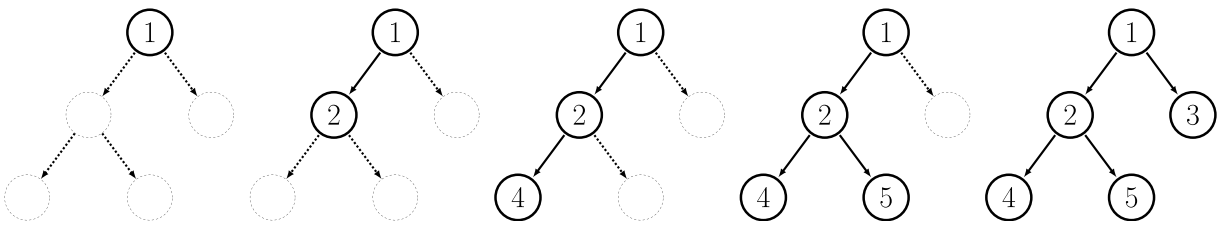
Graphs are usually stored in computer memory either as an adjacency matrix or an adjacency list. Along with the two, there exists another way to represent graphs that fits better for rooted trees. Since each node of a rooted tree keeps the links to its children, it is enough to store only a link to the root in order to have access to all nodes of the tree. To be able to process a rooted tree represented in the described format, we need a way to traverse all its nodes.

There exist four ways to traverse a rooted tree called **pre-order**, **in-order**, **post-order**, and **level-order** traversals. In this topic, we will learn each of them and will see how they work by example. We will use the following rooted binary tree to apply the traversals:



§1. The pre-order traversal

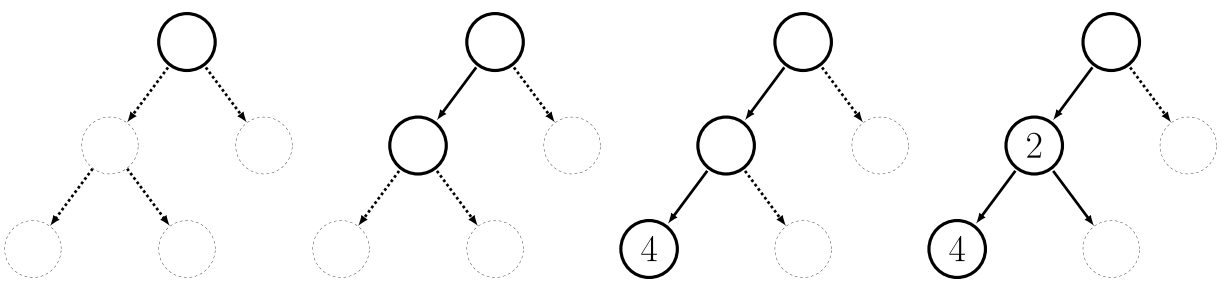
In the **pre-order** traversal, we first visit the current node, then, we recursively visit the left child (if exists), and after that, we visit the right child (if exists). The figures below illustrate how the traversal works for the tree above:



We start the traversal with the root of the tree. First, we mark the root 1 as visited and then move to its left child 2. Next, we mark the node 2 as visited and move to its left child 4. Since the node 4 is a leaf, we stop recursive calls here and return to the previously visited node 2. Next, we visit its right child 5. After we apply the same steps for the remaining nodes, we will have the nodes visited in the following order: $1\ 2\ 4\ 5\ 3$.

§2. The In-order traversal

In the **in-order** traversal, we first visit the left child of the current node (if exists), next, we visit the current node, after that, we visit the right child of the current node (if exists). Let's see how it works for the above graph:



We start the traversal with the root of the tree. The label of the root is not shown indicating that the node has not been processed yet. Next, we visit the left child of the root. Since the child also has the left child, we apply the same procedure for it. After that, we reach the node 4 and since it is a leaf, we mark it as visited and return to its parent node. Next, we mark the current node 2 as visited and start processing the right child of the tree.

Current topic:

✓ Tree traversals ···

Topic depends on:

✓ Tree ···

Table of contents:

↑ Tree traversals

§1. The pre-order traversal

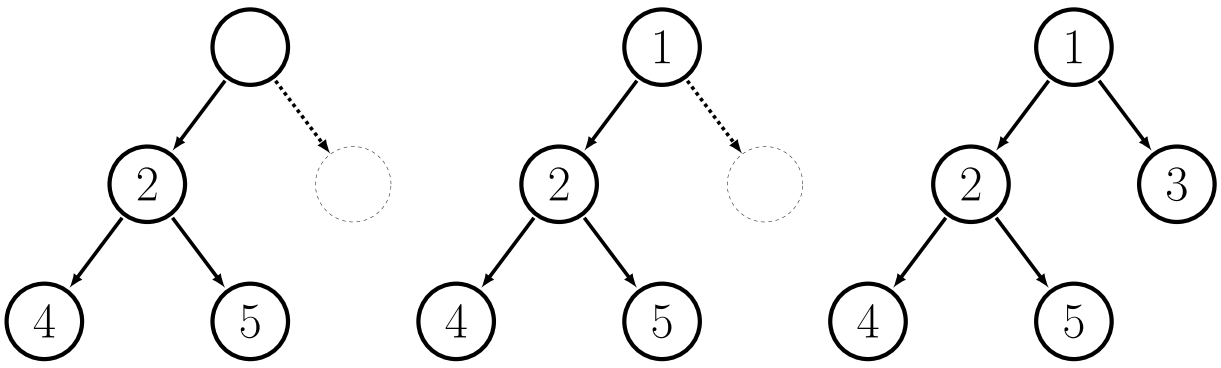
§2. The In-order traversal

§3. Post-order traversal

§4. Level-order traversal

§5. Summary

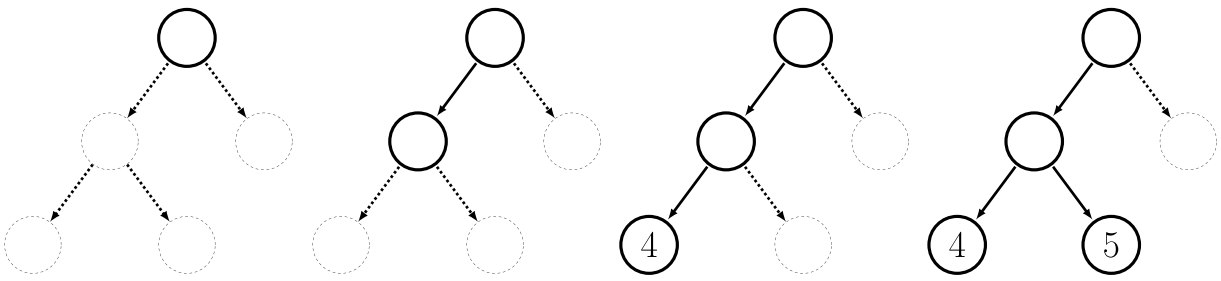
Feedback & Comments



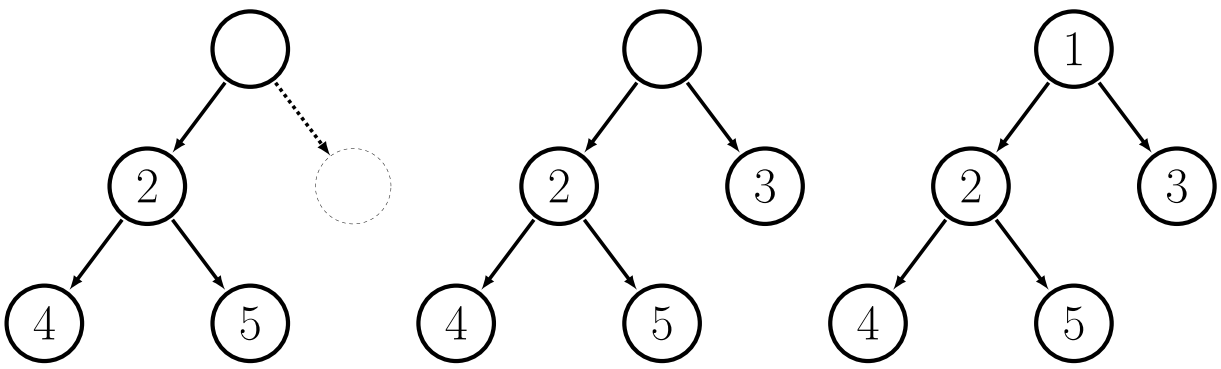
Having applied the same steps for the remaining nodes, we will have the nodes of the tree traversed in the following order: $4\ 2\ 5\ 1\ 3$.

§3. Post-order traversal

In the post-order traversal, we first visit the left child of the current node (if exists), after that, we visit the right child of the current node (if exists), and then, we visit the current node. For the above tree, it works as follows:



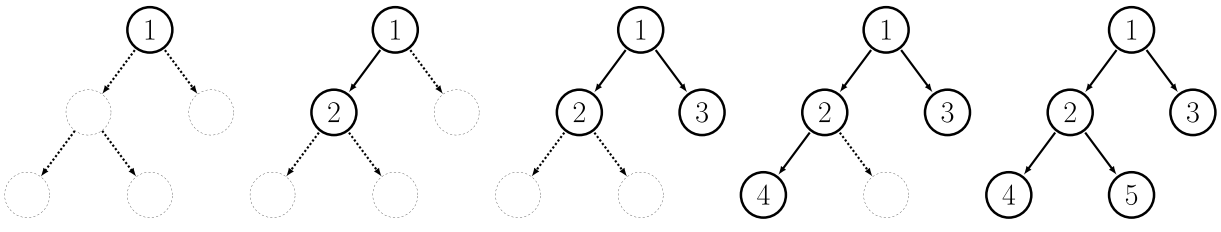
We start the traversal with the root of the tree. The label of the root is not shown indicating that the node has not been processed yet. Next, we visit the left child of the current node. After that, we apply the same for that child and reach the node 4. Since the node 4 is a leaf, we mark it as visited and move to its parent node. After that, we visit the right child of the current node. The right child is a leaf, we mark the node as visited and again return to the parent.



Since both the left and the right child of the current node are visited, we may mark the node as visited. After the same steps are applied for the remaining nodes, we will have all of them traversed in the following order: $4\ 5\ 2\ 3\ 1$.

§4. Level-order traversal

In the level-order traversal, we visit the nodes of a tree *level-by-level*. First, we visit the root, after that, we visit all nodes that are at a distance of one from the root, next, we visit the nodes that are at a distance of two from the root and so on. Let's see how it works by example:



First, we visit the root 1 of the tree. Next, the visit the nodes 2, and then 3, since they are at a distance of one from the root. After that, we visit the node 4 and 5, since they are at a distance of two from the root. Note that the nodes belonging to one level are visited from the left to the right. Finally, we have the nodes traversed in the following order: $1\ 2\ 3\ 4\ 5$.

§5. Summary

In this topic, we learned basic tree traversals algorithms: pre-order, in-order, post-order, and level-order traversals. Each of these traversals might be used to solve different problems. For example, if you want to access all elements of a binary search tree in the sorted order, you may use the in-order traversal. If you want to remove some subtree of a tree, you can run the post-order traversal from the root of the subtree to remove children first, and then the root itself. There are some other problems that may require different approaches. If you encounter with a similar one, just think what type of traversals fits best and apply it.

Report a typo

27 users liked this theory. **1** didn't like it. **What about you?**

Start practicing

Comments (1) Hints (0) Useful links (0) Show discussion