Binary tree operations practice

Rules:

1. Plagiarism is forbidden.
2. Write your program with C++.

Problem Definition:

- ✓ In this problem, you will first obtain the tree from an array representation. After performing M swap operations constrained by the Lowest Common Ancestor (LCA) of two nodes, you will need to print out its inorder traversal and the maximum depth of the tree.
- ✓ For the swap operation, if one of the nodes is the LCA of the two nodes that will be swapped, swap the index of the nodes only. Otherwise, swap the nodes and their entire subtree.
- ✓ The lowest common ancestor of node A and node B is the deepest node (the node that has the largest depth in a set of nodes) that has both node A and node B as its descendants.

I/O Format:

Example 1:

Input:

```
1
2 3
-1
1
3 1
```

Assume the max depth of the input tree is D, you will get $2^D - 1$ numbers which are the index of the nodes if the number is larger than 0. If the number is 0, it means that the space is empty (no nodes). At the end, a "-1" will mark the end of the tree. In this example, D equals 2, so we can expect there will be $2^2 - 1 = 3$ numbers before "-1".

After "-1", you will get a number M that indicates the number of swap operations. Below M, there will be M pairs of indexes that are the indexes to swap. In this example, it requires you to do one swap operation swapping node 1 and node 3.

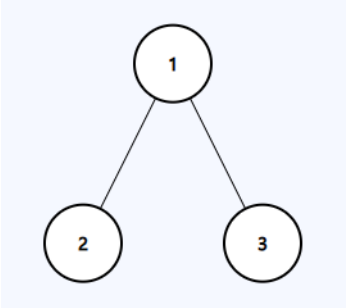
Output:

```
2 3 1
2
```

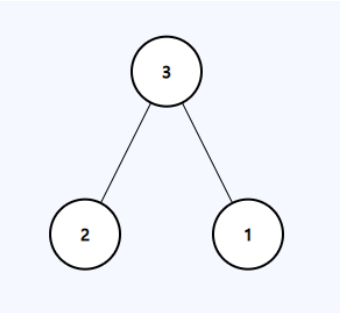
On the first line, print out the inorder traversal of the tree after swap operations.
On the second line, print out the maximum depth of the tree after swap operations.

Explanation:

✧ The corresponding input tree:



✧ The corresponding output tree:



The LCA of **node 1** and **node 3** is node 1. Therefore, we swap only the index of the nodes. The inorder traversal is [2,3,1], and the max depth is 2.

Example 2:

Input:

```
5
2 3
1 4 0 0
-1
2
3 2
4 5
```

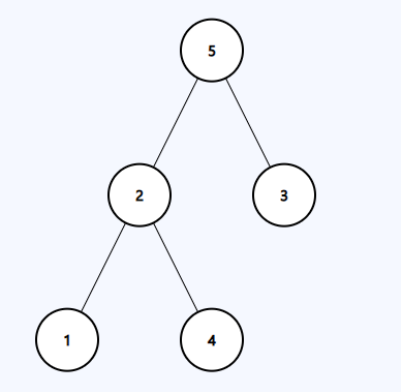
In this example, D equals 3, so we can expect there will be $2^3 - 1 = 7$ numbers before “-1”, and M is 2 which means there will be two swap operations.

Output:

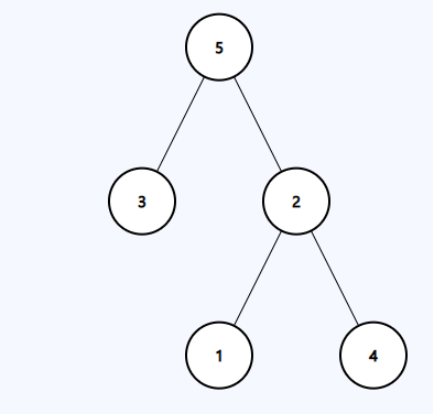
3 4 1 2 5 3

Explanation:

✧ The corresponding input tree:

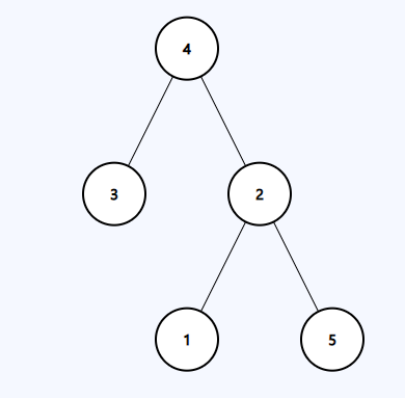


✧ The corresponding tree after swap (3,2):



The LCA of **node 3** and **node 2** is neither node 3 nor node 2. Therefore, we swap the nodes and their entire subtree.

✧ The corresponding tree after swap (4,5), also the output tree:



Once again, the LCA of **node 4** and **node 5** is node 4. Therefore, we swap only the index of the nodes. The inorder traversal is [3,4,1,2,5], and the max depth is 3.

Constraints:

- ✓ $1 \leq$ number of nodes ≤ 20000
- ✓ $1 \leq$ node index ≤ 20000
- ✓ $1 \leq$ D (maximum depth of the input tree) ≤ 20
- ✓ $0 \leq$ M (number of swap operations) ≤ 100000
- ✓ Each node index is unique
- ✓ Memory and time limits specified on the online judge

Hints:

- ✓ Think about level order traversal to build the tree from array representation.
- ✓ Try to build the tree and get the input simultaneously.
- ✓ Add parent pointer in the node for convenience.
- ✓ Record if the node is the root of its parent's left or right subtree for convenience.