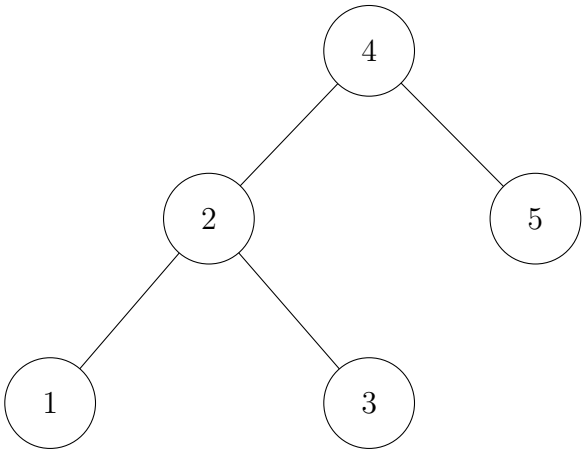
1 (2). Construct an algorithm that convert a binary search tree into a linked list such that the linked list is ordered descending.

2 (3). A concatenate operation takes two sets S_1 and S_2, where every key in S_1 is smaller than any key in S_2, and merges them together. Give an algorithm to concatenate two binary search trees into one binary search tree. The worst-case running time should be $O(h)$, where h is the minimal height of the two trees; heights of the trees are known.

3 (4). Describe how to modify any balanced tree data structure such that search, insert, delete, minimum, and maximum still take $O(\log n)$ time each, but successor and predecessor now take $O(1)$ time each. Which operations have to be modified to support this?

4 (5). The keys of the binary search tree had been written to the array according to the preorder traversal (NLR). So, for a tree from the picture, the array is $[4, 2, 1, 3, 5]$. The array for a tree is stored in RAM. Construct an $O(n)$ algorithm that gets on the input keys u and v and outputs whether v is a descendent of u in the original tree.



5 (1+2+5). An array of size n contains elements from the range from 1 to $n-1$ (may be not all of them). Construct an algorithm that finds a duplicate satisfying the conditions (subproblems are separate)

1. Algorithm runs in $O(n)$.

2. Algorithm runs in $O(n)$ and uses $O(1)$ RAM; the array is available in read-only mode. It is known that the array contains all numbers from 1 to $n-1$.

3*. Algorithm runs in $O(n)$ and uses $O(1)$ RAM; the array is available in read-only mode. You can solve only this subproblem.