Review tree traversal

Ancestry relationship in tree

Given a binary tree $T$, we say that a node $n$ is an ancestor of $m$ if $n$ is on the path connecting the root of $T$ to $m$. Write a program to determine and print out the relationship of two given nodes $n$ and $m$. Possible output would be:
- $n$ is an ancestor of $m$
- $m$ is an ancestor of $n$
- None of the above

```java
class Node<T>
{
    Node<T> left, right;
    T data;
}

void ancestor(Node<T> root, Node<T> n, Node<T> m)
{
}
```

Analysis

- What is the time complexity?
- Can you improve the performance?
void preprocessing_recursive(Node<T> root) {
}

class Node<T> {
    Node<T> left, right;
    T data;
}

void ancestor(Node<T> root, Node<T> n, Node<T> m) {
}

void preprocessing_iterative(Node<T> root) {
}

---

**Analysis**

- What is the time and space complexity?
Find in BST

Find $k$-th smallest node

Given a binary search tree, implement a recursive method that can find $k$-th smallest node in a given tree or report null if no such node.

Analysis

- What is the time complexity?
- Can you implement an iterative version?

```java
void kthMin(Node<T> root, int k) {
}

void kthMin_iterative(Node<T> root, int k) {
}
```
Operate BST

**Merge two trees**

Given two binary search trees, implement a method that can merge these two trees into one. We can assume that all nodes in both trees hold unique values.

```java
void merge(Node<T> a, Node<T> b) {
}
```

**Analysis**

- What is the time complexity?
- Can you do better?
Build a BST from input

- Build and draw a BST by inserting these numbers in order: 5, 13, 18, 21, 23, 31, 5, 7, 89, 130
- Build and draw a BST by inserting these numbers in order: 31, 18, 130, 89, 21, 5, 57, 13, 23
- What do you observe?
- Can you come up with an algorithm that will give you the best BST?
Heap tree

Binary min heap

A binary min heap is a binary tree in which a node must hold a value smaller than the value stored in its children nodes.

Your task

Using the node structure below to implement two methods: insert a value to a heap and removing the smallest value from a heap.

class Node<T>
{
    Node<T> left, right;
    T data;
}