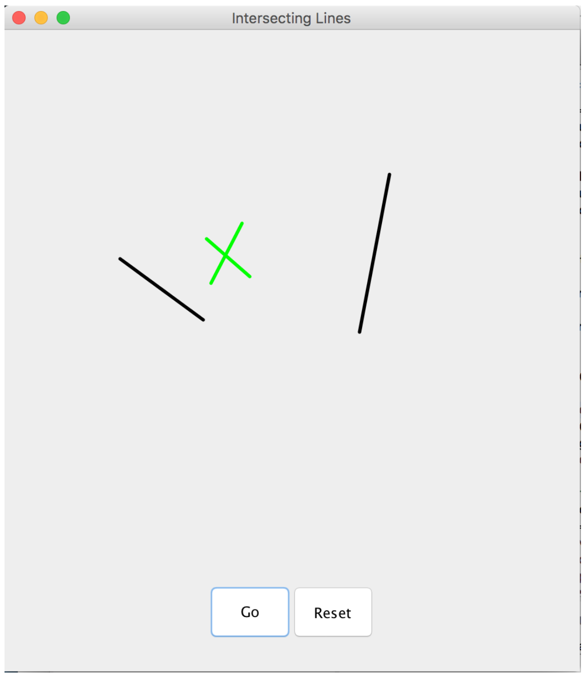
C343 Project - Segment Intersection and Binary Search Trees

1 Project Description

The Line Segment Intersection program detects intersecting lines. In particular, the user draws lines on the screen and once the user presses the 'Go' button, the program should highlight in green the two intersecting lines with the leftmost endpoints. The following shows the result of successfully running the program.



Once you have finished implementing the project and compiling the java files, you will be able start the program by running 'java GUIDriver'.

There are several supporting files, `SegmentIntersection.java`, `SearchTree.java`, `Node.java` which help the driver run your code.

2 Task 1: Unbalanced Search Trees

Create a generic class named `BinarySearchTree` in a new file `BinarySearchTree.java`, which should be parameterized on a `Key` type. Also, `BinarySearchTree` should implement the `SearchTree` interface, which means you will also need to create a class that implements the `Node` interface.

The constructor for `BinarySearchTree` must have a parameter of type `BiPredicate` (binary predicate) that you will use to compare keys. You will need to implement all of the methods required by the `SearchTree` interface:

- `Node<Key> insert(Key key)` - inserts a key in the tree by creating a node for that key and it returns the new node.
- `Node<Key> search(Key key)` - returns the node with a matching key if found, else `null`.
- `void delete(Key key))` - deletes the node with the matching key, if present.

In the node class that you create, you will need to implement all the methods required by the `Node` interface:

- `Node<Key> after()` - returns the node after this one with respect to an in-order traversal.
- `Node<Key> before()` - returns the node before this one with respect to an in-order traversal.
- `Key getKey()` - returns the key associated with the node.

3 Task 2: Balanced Search Trees

Create an `AVLTree` class in a new file named `AVLTree.java`. Implement the search tree interface (same as for `BinarySearchTree`), but this time keep your tree balanced using the AVL approach.

4 Deliverables

Your github `segment_intersection` folder should contain all the files from the zip plus the java files that you've created. Also include a file with a graph with two lines representing the execution times for running segment intersection in batch mode, comparing your BST and AVL implementations. Finally, include a file named `README.md` that explains your code.