Names:
Ryan Munz (rtm8889)
Jayanth Prathipati (jayanth)
Omid Anvar (omid8)

Project 1 Phase 2:

Files Included With Project:

+ bruteForce.java
+ bruteForceTest.java
+ Intersection.java
+ Line.java
+ lineSweep.java
+ lineSweepTest.java
+ LineTest.java
+ mainClass.java
+ mergeSort.java
+ Point.java
+ README.txt
+ Graph.jpg
+ DataAlgoGraph.xlsx

How to run our code:

We did this in Eclipse. You need to import the project into eclipse and run the class mainClass. You can change the number of lines to generate or grid size by changing the constants NUMLINES or GRIDSIZE respectively. You should also be able to compile and run the code from command line, but we didn't.

Notes:

+ We ran the mainClass for different number of lines and recorded the run times. We then use these times to create a graph in excel. The graph is shown in Graph.jpg. The excel file used to make the graph is also included.

+ We used the java TreeSet for the binary tree and ArrayList for all the arrays.

Description:

The project implements the brute force and line sweep algorithms for finding

Merge Sort:
This class takes a list of points that represent vertical and horizontal and performs a merge sort on the points to sort them by their Y value in descending order. Then the points with the same Y values are then sorted by

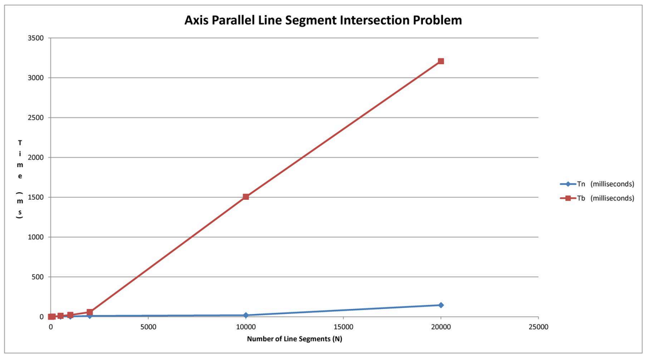
precedence where Vertical start points have the highest, horizontal start points are next, finally the vertical or horizontal end points. The newly sorted list is then returned.

Line Sweep:
Performs the line seep algorithm as designed in phase 1 of this project. The Algorithm finds all intersections between vertical and horizontal lines. The algorithm uses the java implementation of TreeSet for the balance binary tree since in the documentation, they say it guarantees O(logn) for insertion, deletion, and find. We also used the ArrayList from java for all arrays.
This is because the ArrayLists automatically grow and have many methods for ease of use. We implemented everything else.

Brute Force:
Implements the traditional algorithm by comparing every vertical line to every horizontal line and checked for intersections.

Results:

After creating our results, we were able to generate out the following graph below which shows that our results for our brute force are quadratic, as the algorithm is O(n^2) where n is the number of lines in the graph. In addition, our result shows that our line sweep approach is significantly faster as it is O(nlog(n) + k) as a result, you can see that our line sweep takes a fraction of the time of brute force.


Axis Parallel Line Segment Intersection Problem
Time (ms)
Number of Line Segments (N)
Tn (milliseconds)
Tb (milliseconds)