**Intro to Algorithms, COMP-160, Homework #9**

Benjamin Tanen, 04/06/2016

1. **Consider a set $S$ of $2n$ line segments in 2D: $n$ in horizontal and $n$ in vertical. Assume that no two segments have endpoints at the same $x$-coordinate or $y$-coordinate. It is easy to determine if two given segments intersect, in constant time, by comparing the coordinates of their 4 endpoints. Provide a sub-quadratic-time algorithm to count how many intersections there are among the segments in $S$.**

   We can begin our algorithm by first sorting all $2n$ of our lines (vertical and horizontal) based on their left-most position. Since we will use an efficient sorting algorithm, this then takes $O(n \log n)$ time to sort. Next, we should create a BST to hold our horizontal lines. This BST tree will be sorted based on the $y$-position of each line.

   Now that we have all of our lines sorted by their left-most position (where each node also holds the right-most position) and a BST to store horizontal lines, we can "sweep" by iterating over our $2n$ lines. For each line $l$ that we will hit, we do one of the following:

   (a) If line $l$ is a horizontal line, add line $l$ to our BST based on its $y$-position. For each line, we will also note when line $l$ ends (when our sweep passes the right-most point $x_{l2}$ of our line $l$), we should delete line $l$ from our BST. Inserting each line takes $O(\log n)$ and deleting each line takes $O(\log n)$.

   (b) If line $l$ is a vertical line, use the range finding algorithm on our BST where our left bound is the bottom $y$-position $y_{l1}$ and our right bound is the upper $y$-position $y_{l2}$. The range finding algorithm takes $O(\log n)$ to compute the lines that overlap our vertical line $l$.

      Since our BST only contains horizontal lines for which we have swept over their left-position and not yet swept over their right-position, we know that all lines in our BST are potential horizontal candidates to overlap with our vertical line $l_v$ (for each horizontal line $l_h$, the left position of $l_h$ is less than the horizontal position of $l_v$ which is less than the right position of $l_h$). Thus we know that any lines in the correct vertical range intersect / overlap with our vertical line $l_v$.

   Thus, to originally sort our $2n$ lines, we use $O(n \log n)$ time. Then for each of our $n$ horizontal lines, we will add the line and then delete it later, taking $O(\log n + \log n) = O(\log n)$. For each of our $n$ vertical lines, we will conduct the range finding algorithm on our BST, which takes $O(\log n)$ time. Thus for each of our $2n$ lines, we will take $O(\log n)$ time, so in total, our entire algorithm takes $O(n \log n) + 2nO(\log n) = O(n \log n)$.