3805ICT Advanced Algorithms – Assignment 1

Chris Vuong – s5193954

**Question 1:** A common geometric problem is, given a set of N lines, how many intersect. The brute force algorithm is O(N2 ). Design an O(N\*logN) algorithm and implement it in C++ for the case where there are only vertical or horizontal lines. Derive and demonstrate the efficiency of your algorithm.

Chart, box and whisker chart

Description automatically generated

The picture above is a graph of segments that are only horizontal and vertical segments. There are 16 intersection points are shown as the red points. This answer is deriving a C++ solution and explaining the algorithms of the solution as below:

The algorithm is applying a sweep line to solve the problem, but before explaining the efficiency of the algorithms, the algorithms need to define some data structure.

Point: storing the x, y coordinates.

Int x, y ;

Line: storing 2 points of a segment.

Point a, b ;

Event: storing a line l and the integer type (0 for the start point of horizontal, 1 for the endpoint, and other for a vertical line)

As the sweep line in this question is assumed as a vertical line, so all the horizontal lines must have 2 events for checking the start point and endpoint. The line of the start point event is set opposite to the endpoint event.

Line l ;

Int type;

According to the graph, there are 11 horizontal lines and 10 vertical lines. Therefore, n is equal to 21.

Pseudo-code:

Let events be the list of the events ( O(n + k) to generate the events n is the number of lines and k is the number of horizontal lines)

Let sweepline is a list of lines. (O(1))

Let count = 0; (O(1))

Looping to each event of the events (O(n))

Case event.type == 0: sweepline.add(event.l); (O(1))

Case event.type == 1: sweepline.delete(event.l); (O(log(n)) this case the sweepline must find where the event line is. In the worst-case scenario, it can be O(n) but it is a special case)

Default:

Loop through the sweepline and check if there is any intersection with the event.l. (O(log(n) this is the loop through the sweepline, and again it may have the special case of O(n)).

If there is the intersection: count ++;

From the pseudo-code above, the formular is O(n+k) + O(2) + O(n \* (log(n) + log(n))) = O(nlog(n)).

In the special cases, if there are only horizontal line in the graph and would not be any intersection the algorithms will be O(2n)+ O(2) + O(n \* (n + n)) = O(n^2).

Here is the output of the C++ generation:

line {(3,9),(10,9)} intersects with line {(4,12),(4,5)}.

line {(3,9),(10,9)} intersects with line {(7,17),(7,5)}.

line {(6,13),(10,13)} intersects with line {(7,17),(7,5)}.

line {(6,16),(13,16)} intersects with line {(7,17),(7,5)}.

line {(6,6),(12,6)} intersects with line {(7,17),(7,5)}.

line {(3,9),(10,9)} intersects with line {(10,12),(10,7)}.

line {(9,11),(16,11)} intersects with line {(10,12),(10,7)}.

line {(6,16),(13,16)} intersects with line {(11,17),(11,12)}.

line {(9,11),(16,11)} intersects with line {(12,13),(13,2)}.

line {(9,11),(16,11)} intersects with line {(15,5),(15,12)}.

line {(14,6),(17,6)} intersects with line {(15,5),(15,12)}.

line {(15,14),(17,14)} intersects with line {(16,15),(16,12)}.

line {(16,7),(20,7)} intersects with line {(18,6),(18,12)}.

line {(17,9),(27,9)} intersects with line {(18,6),(18,12)}.

line {(18,14),(23,14)} intersects with line {(21,18),(21,12)}.

line {(17,9),(27,9)} intersects with line {(22,10),(22,6)}.

Total intersections: 16