COMS3005A
Additional Notes
Intersections of Horizontal and Vertical Line Segments

Ian Sanders

First semester, 2023 Academic Year

1 Terminology

- A *point* p is represented as a pair of coordinates (x, y)
- A *line segment* is represented by a pair of points p and q which are its endpoints and is denoted $\overline{pq}$

2 Overview

Let us now consider a geometric problem where we are asked to find all of the intersections between a number of horizontal and vertical line segments in the plane. This presentation is adapted from Manber [1989]. The problem statement is given below:
The Problem: Given a set of n horizontal and m vertical lines segments in the plane, find all of the intersections between them.

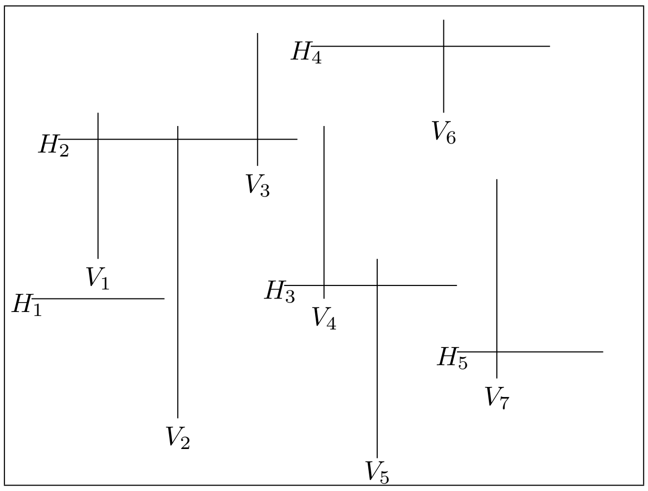

Figure 1: Intersections of Horizontal and Vertical Line Segments

Figure 1 shows an example of the problem. For this example there are seven intersections (H_2 and V_1, H_2 and V_2, H_2 and V_3, H_3 and V_4, H_3 and V_5, H_4 and V_6 and H_5 and V_7).

2.1 A brute force solution

The brute force solution would be to compare each horizontal line segment to each vertical line segment to determine whether they intersect. An algorithm to do this is given in Figure 2.

In this case testing for intersection simply involves checking if the x-coordinate of the vertical line falls in the range of the x coordinates of the horizontal line and that y-coordinate of the horizontal line

```
Algorithm $simpleHVIntersection$
Input:  $h_1, h_2, \cdots, h_n, n \geq 1$ (a set of $n$ horizontal line segments in the plane) and
$v_1, v_2, \cdots, v_m, m \geq 1$ (a set of $m$ vertical line segments in the plane)
Output:    The line segments which intersect.
   For $i$ from 1 to $n$
      For $j$ from 1 to $m$
         If $h_i$ intersects with $v_j$
            Then Output $h_i$ and $v_j$
```

Figure 2: Intersections of Horizontal and Vertical Line Segments – Brute force approach

falls in the range of the y coordinates of the vertical line. If a line segment is represented by a start point (s) and an end point (e), then h is represented by $(h_{sx}, h_{sy}), (h_{ex}, h_{ey})$ where $h_{sy} = h_{ey}$ and v is represented by $(v_{sx}, v_{sy}), (v_{ex}, v_{ey})$ where $v_{sx} = v_{ex}$. Line segment h then intersects with line segment v if $h_{sx} \leq v_{sx}$ and $v_{sx} \leq h_{ex}$ and $v_{sy} \leq h_{sy}$ and $h_{sy} \leq v_{ey}$.

Testing for intersection thus involves testing that these four conditions hold. If we use the testing for intersection, which takes constant time, as the basic operation of this algorithm then there are $n \cdot m$ intersection tests made and the algorithm is O(mn).

The solution method here is clearly correct. The algorithm tests each horizontal line against each vertical line and so if there is an intersection between any two lines this intersection will be found. When the algorithm terminates all intersections will have been found.

Note that the problem of determining whether *any* horizontal line segment in the set intersects with a vertical line segment is a slightly different problem and would be solved differently.

3 An improved algorithm using line sweeping

In this section we consider a solution which makes use of the fact that if two line segments are not in the same region of the plane then they cannot possibly intersect. In this algorithm we check for line segments which overlap in x coordinates and then check if they intersect – if their x coordinates do not overlap then the lines cannot possibly intersect. The checking for which vertical line segments fall within the range of x coordinates of the horizontal lines is accomplished by means of a “line sweep”. Here an infinitely long, imaginary vertical line “sweeps” across the set of line segments from left to right (usually). The sweeping allows us to deal with the objects as we encounter them and allows us to take advantage of any spatial relationships between them. As our line sweeps from the left to the right it encounters three different types of “events” – the start of a horizontal line segment, the end of a horizontal line segment and a vertical line segment.

In general the first thing to do in tackling any problem using a line sweep approach is to identify the *event points* and to draw up an “event point schedule” – an ordered list of the events as they are encountered by the sweep. At each one of these events the relationship between objects in the plane could change. We then need to figure out what we need to know about the relationships between the objects in order to make working with the problem easier and what we actually have to do at each event point.

In our intersecting horizontal and vertical line segment problem, the event points are the start of a horizontal line segment, the end of a horizontal line segment and a vertical line segment. We draw up an event point schedule by sorting the events (the endpoints of horizontal line segments and position of vertical line segments) by x-coordinate. We would then work through the sorted list and at each event would follow the rules given below.

- If the event is a left endpoint of a horizontal line segment then add the line to the list of lines to be considered (the candidates)
- If the event is a right endpoint of a horizontal line segment then remove the line segment from the candidates
- If the event is a vertical line segment then test each candidate for intersection with the vertical line segment

An algorithm to solve this problem using this approach is given in Figure 3.

```
Algorithm $sweepHVIntersection$
Input:  $h_1, h_2, \cdots, h_n, n \geq 1$ (a set of $n$ horizontal line segments in the plane) and
$v_1, v_2, \cdots, v_m, m \geq 1$ (a set of $m$ vertical line segments in the plane)
Output:   The line segments which intersect.
   Create set $eventPoint$ which is made up of the endpoints of horizontal line
   segments and the position of vertical line segments
   Sort $eventPoint$ based on $x$ coordinate
   {break ties by putting starts of horizontal line segments before vertical
    line segments before ends of horizontal line segments and break
    further ties based on lower $y$ value}
   Set $candidates$ to be empty
   For each $event$ in $eventPoint$
      If $event$ is the start of a horizontal line segment $h$
         Then
            Add the line segment $h$ to $candidates$
      If $event$ is the end of a horizontal line segment $h$
         Then
            Remove the line segment $h$ from $candidates$
      If $event$ is a vertical line segment $v$
         Then
            For each line segment $l$ in $candidates$
               If $l$ intersects with $v$
                  Then
                     Output $l$ and $v$
```

Figure 3: Intersections of Horizontal and Vertical Line Segments – the line sweep approach

It is relatively simple to argue that this solution method is correct. Each horizontal line segment will be added to the set of candidates when its start point is encountered and will remain in $candidates$ until its end point is encountered. While it is in $candidates$ it will be checked for intersection against any vertical line segment whose x coordinate falls in the range between the horizontal line segment's start and end points. The order of breaking ties will ensure that, even if the position of a vertical line segment coincides with the start or end point of the horizontal line segment, all the line segments are tested for intersection.

A problem with this algorithm is that it would still be O(mn) in the worst case even if there were no intersections between horizontal and vertical line segments. (Note that if every vertical line intersects with every horizontal line then there will be $n \cdot m$ intersections and so the algorithm would be O(mn) in this case but this algorithm would be O(mn) even if there were no intersections at all!) This could happen if all of the horizontal line segments are in $candidates$ when each of the vertical line segments is checked against all of the lines in $candidates$. A configuration of line segments as shown in Figure 4 would cause

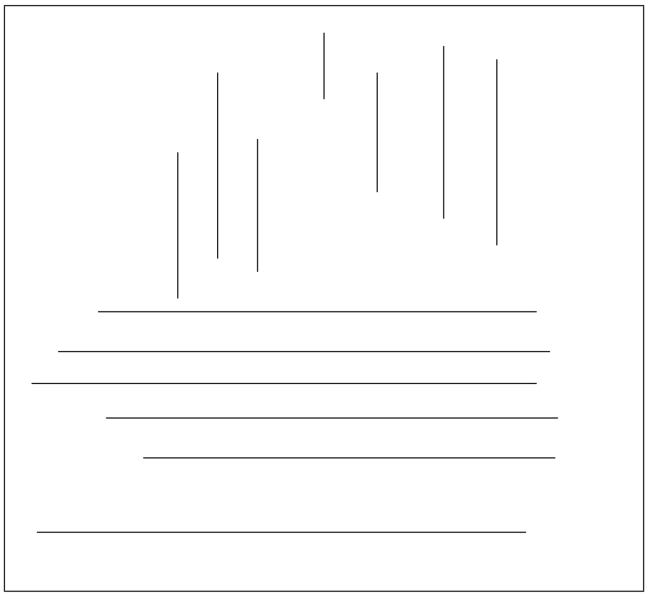

Figure 4: A configuration of horizontal and vertical line segments that results in $\mathrm{O}(mn)$ performance for the line sweep algorithm in Figure 3

this to happen. In this case there are actually no intersections but each horizontal line segment is added to $candidates$ as it is encountered and all of the horizontal line segments are in $candidates$ when each of the vertical line segments is encountered and tested against the whole of the $candidates$ set.

To improve the algorithm we need to minimise the number of comparisons between the y-coordinates of a vertical line and those of the candidates if at all possible. In the example above we would like to make at most a constant number of tests to know that we do not need to test every line segment in the candidate set for intersection with the current vertical line segment.

To accomplish this we need to maintain a sorted list of the y-coordinates of the candidate line segments $y_1 \leq y_2 \leq \ldots \leq y_k$ and then only look at those candidates where the current vertical line segment *could* possibly intersect. We would do this by doing binary searches on the sorted list to find where in the sorted list y_b and y_t of the vertical line would appear and then considering only those candidates that appear in the sorted list between where y_b and y_t would appear. We could also do one binary search to find where y_b should appear in the sorted list and then traverse the list considering the candidates in order until we reach y_t. Our modified algorithm is shown in Figure 5.

Clearly the solution method is still correct – it is not changed by this modification.

Let us now consider the complexity of the algorithm. Sorting the event points at the beginning is $\mathrm{O}((m+n)\lg(m+n))$. To find the position of y_b in the sorted candidate list takes $\mathrm{O}(\lg n)$ in the worst case. Then the intersection tests will be done for however many lines the vertical line segment intersects with – if any candidate line segment has a y value that falls between y_b and y_t then it will be tested for intersection and must intersect. Thus we have a cost of $O(\lg n + r_i)$ for each vertical line segment i (r_i is number of intersections for line i). So the cost for all the vertical line segments is $O(m \lg n + R)$ ($R = \sum_{i=1}^{m} r_i$)

This makes it seem as if we have improved our algorithm but we still have a problem. In our algorithm we might need to insert and delete $\mathrm{O}(n)$ horizontal lines into the set of condidate lines. Insertion into and deletion from an ordered list are operations which have costs that are proportional to the length of the list (i.e. linear) and in the worst case all of our horizontal line segments could be in $candidates$ which would mean that each insertion and deletion is $\mathrm{O}(n)$. This would then make the overall cost of inserting and deleting $\mathrm{O}(n^2)$ which is more expensive than all the other operations. So our algorithm is not any

```
Algorithm $sweepHVIntersection$
Input:  $h_1, h_2, \cdots, h_n, n \geq 1$ (a set of $n$ horizontal line segments in the plane) and
$v_1, v_2, \cdots, v_m, m \geq 1$ (a set of $m$ vertical line segments in the plane)
Output:    The line segments which intersect.
   Create set $eventPoint$ which is made up of the endpoints of horizontal line
    segments and the position of vertical line segments
   Sort $eventPoint$ based on $x$ coordinate
   {break ties by putting starts of horizontal line segments before vertical line
    segments before ends of horizontal line segments and break
    further ties based on lower $y$ value}
   Set $candidates$ to be empty
   For each $event$ in $eventPoint$
      If $event$ is the start of a horizontal line segment $h$
         Then
            Insert $h$ into the correct position in $candidates$ based on $y$ value
      If $event$ is the end of a horizontal line segment $h$
         Then
            Remove the line segment $h$ from $candidates$
      If $event$ is a vertical line segment $v$
         Then
            Do a binary search on $candidates$ to find the position of $y_b$
            While $y$ value of line segment $l$ in $candidates$ is $\leq y_t$
               If $l$ intersects with $v$
                  Then
                     Output $l$ and $v$
```

Figure 5: Intersections of Horizontal and Vertical Line Segments – the modified line sweep approach

better than our straightforward brute force algorithm.

We are, however, making progress towards an efficient algorithm to solve this problem. To do this we need a data structure which allows us to insert and delete candidate line segments efficiently. We know that in the worst case we might have to do O(n) insertions and deletions so we need something which is cheaper than O(n) for each insert and delete. That is, we need a data structure that allows us to do each insert or delete in O$(\lg n)$ time, or O$(n \lg n)$ overall.

One solution here is to use balanced trees. We will be considering red-black trees (as an example of balanced trees) later in the course. We will see later that if we use these then we can get O$(n \lg n)$ performance overall or O$(\lg n)$ for each insert or delete.

4 The final algorithm which uses a data structure to give improved performance

Red-black trees (which we consider later in the course) allow us to maintain an ordered structure and to do insertions and deletions in O$(\lg n)$ time. The final version of our line sweep algorithm to find the intersections between horizontal and vertical lines segments is given in Figure 6.

Clearly the solution method is still correct – it is not changed by this modification.

Each insertion into and deletion from $candidates$ can now be done in O$(\lg n)$ time as can searching for y_b

The final running time of the line sweep algorithm would thus be O$(((m+n)\lg(m+n)+R)$ where R is the total number of intersections between horizontal and vertical line segments.

```
00  Algorithm $sweepHVIntersection$
    Input:  $h_1, h_2, \cdots, h_n, n \geq 1$ (a set of $n$ horizontal line segments in the plane) and
$v_1, v_2, \cdots, v_m, m \geq 1$ (a set of $m$ vertical line segments in the plane)
    Output:   The line segments which intersect.
01    Create set $eventPoint$ which is made up of the endpoints of horizontal line
      segments and the position of vertical line segments
02    Sort $eventPoint$ based on $x$ coordinate
      {break ties by putting starts of horizontal line segments before vertical line
       segments before ends of horizontal line segments and break
       further ties based on lower $y$ value}
03    Set $candidates$ to be empty
04    For each $event$ in $eventPoint$
05       If $event$ is the start of a horizontal line segment say $h$
06          Then
07             Insert the line segment $h$ into the correct position (based on $y$ value)
                in the red-black tree representation of $candidates$
08       If $event$ is the end of a horizontal line segment $h$
09          Then
10             Delete the line segment $h$ from the red-black tree $candidates$
11       If $event$ is a vertical line segment $v$
12          Then
13             Do a tree search on $candidates$ to find the position of $y_b$ in $candidates$
14             While the $y$ value of line segment $l$ in $candidates$ (found by
                traversing the red-black tree) is $\leq y_t$
15                If $l$ intersects with $v$
16                   Then
17                      Output $l$ and $v$
```

Figure 6: Intersections of Horizontal and Vertical Line Segments – the modified line sweep approach

References

Manber, U. (1989). *Introduction to Algorithms: A creative approach.* Addison Wesley Publishing Company, USA.