Stabbing Problem
Outline

• Stabbing problem
• Method 1
• Method 2
• Related problems
Stabbing Problem

- Given a set of $n$ line segments $S$
- Input: query point $q$
- Output: the intervals that contain $q$
Stabbing Problem

• Brute force algorithm:
  - For query point q and every interval $s_i$ in S, check if q overlaps with $s_i$
  - Time complexity: $O(n)$
Stabbing Problem

• Brute force algorithm:
  - For query point q and every interval \( s_i \) in S, check if q overlaps with \( s_i \)
  - Time complexity: \( O(n) \)

• Can we do faster?
Outline

• Stabbing problem
• Method 1 - segment tree
Segment Tree

• Preprocess:
  - Step 1: Sort by all the start points and end points
  - Step 2: By the 2n points, build a balanced binary search tree $T$
    • Height of $T = O(\log n)$
  - Step 3: Insert the line segments into $T$
    • Insert a line segment needs $O(\log n)$ time
Example
Segment Tree

- Property: any segment is stored at most twice at each level of T
- Space complexity: $O(n \log n)$
- Preprocessing time: $O(n \log n)$

- Note: every node represents a segment
Segment Tree Query
Segment Tree Query
Segment Tree

• Query time:
  - $O(\log n + k_1 + k_2 + k_3 + \ldots + k_{\log n})$
  - $= O(\log n + k)$
  - $k_L$: number of nodes reported on level $L$

• Output-sensitive
  - algorithms whose running time depends not only on the size of the input but also on the size of the output
Outline

• Stabbing problem
• Method 1 - segment tree
• Method 2 - interval tree
Interval Tree

• Preprocess:
  - Build a balanced binary search tree $T$ for the $n$ line segments by the start points
    • Each node $v$ of $T$ has information of the line segment and Max
    • Max: position of the righmost end points in subtree of root $v$
Interval Tree

• Preprocess time:
  - Build BBST: $O(n \log n)$
    • Insert a line segment into $T$: $O(\log n)$
  - Maintain Max: $O(1)$
Example

Max = 14

Max = 10

Max = 5

Max = 10

Max = 11

Max = 14
Interval Tree

- Space: $O(n)$
  - Each node represents a line segment

- Query time: $O(k \log n)$
Interval Tree

• Query:
  - Step 1: check if query point q intersects with the line segment in node x
    • Yes -> report
  - Step 2: check if q > x.max
    • Yes -> complete
  - Step 3: check if q > x.startpoint
    • Yes -> recursively run on x.leftchild and x.rightchild
    • No -> recursively run on x.leftchild
Example

Max = 14
Max = 10
Max = 5
Max = 10
Max = 11
Max = 14
Max = 14
Max = 14
Max = 10
Max = 5
Max = 10
### Interval Tree

<table>
<thead>
<tr>
<th></th>
<th>$q &lt; x\text{.max}$</th>
<th>$q &gt; x\text{.max}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q &lt; x\text{.start}$</td>
<td><img src="image" alt="Diagram - $q &lt; x\text{.start}$" /></td>
<td><img src="image" alt="Diagram - $q &gt; x\text{.start}$" /></td>
</tr>
<tr>
<td>$q &gt; x\text{.start}$</td>
<td><img src="image" alt="Diagram - $q &gt; x\text{.start}$" /></td>
<td><img src="image" alt="Diagram - $q &gt; x\text{.start}$" /></td>
</tr>
</tbody>
</table>
Outline

• Stabbing problem
• Method 1 - segment tree
• Method 2 - interval tree
  (a completely different version)
Outline

- Stabbing problem
- Method 1 - segment tree
- Method 2 - interval tree
- Related problems
Related Problem

• Higher-dimension Stabbing Problem
  - Solved by multi-level of segment trees
  - Space improvement if we use interval tree at deepest level

• Given a set of points, query rectangle
  - called Range Query Problem