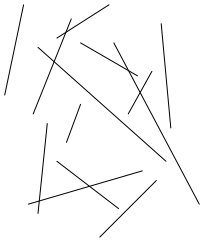
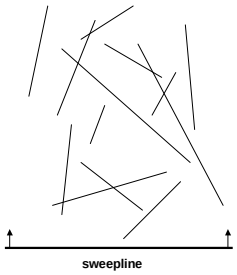
PQ (and Balanced Tree) Application: Line Segment Intersection

- Given a set of line segments in the plane, report all intersections
- Obvious method:
  - Try each pair of segments
  - Running time = $O(n^2)$
- If there are a lot of intersections then this is the best we can do, but what if there are few intersections?



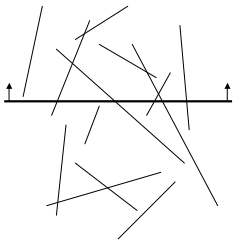
Idea: Use a Sweepline



- Move a sweepline across the line segments
  - At each moment of time, the sweepline intersects a number of segments
  - Just *before* an intersection, the two intersecting segments are adjacent on the sweepline

Sweepline Intuition

- The idea is to efficiently find intersections by checking only those segments that are adjacent along the sweepline
- Adjacencies change only when there is an *event*
- Only the segments adjacent to an event change their adjacencies



Sweepline Events

- We use an event-driven simulation to move the sweepline
- Events are
  - Segment start
    - When we first encounter a segment
  - Segment end
    - When we reach the segment's other endpoint
  - Intersect segments
    - When we reach a segment intersection
- Some of the events, we know ahead of time
  - Segment start
  - Segment end
- If we knew all events ahead of time we could sort them and do without a Priority Queue, but we don't...
- For segment intersection, we have to determine these events as we move the sweepline, so we need a PQ

A Sweepline Program

```
Load PQ with segment-start and segment-end events;
loop
    Get next event from PQ;
    if (intersection) then Report it;
    Update sweepline;
    Check for new adjacencies;
    If two segments are newly adjacent and they intersect then
        Insert their intersection-event into the PQ;
end loop;
```


Adjacency using a Balanced Tree

Need the following operations:

- Insert segment
- Delete segment
- Tell me the two segments adjacent to this one (along the sweepline)
- Swap two segments along the sweepline (needed when an intersection occurs)

We can use a balanced tree!

- Runtime analysis
  - The loop is executed O(n+k) times where k is the number of intersections
  - Each PQ operations within the loop takes time O(log n) (using a heap)
  - The operations needed to maintain adjacency information each take time O(log n) (using a balanced tree)
- Total time: O((n+k)log n)