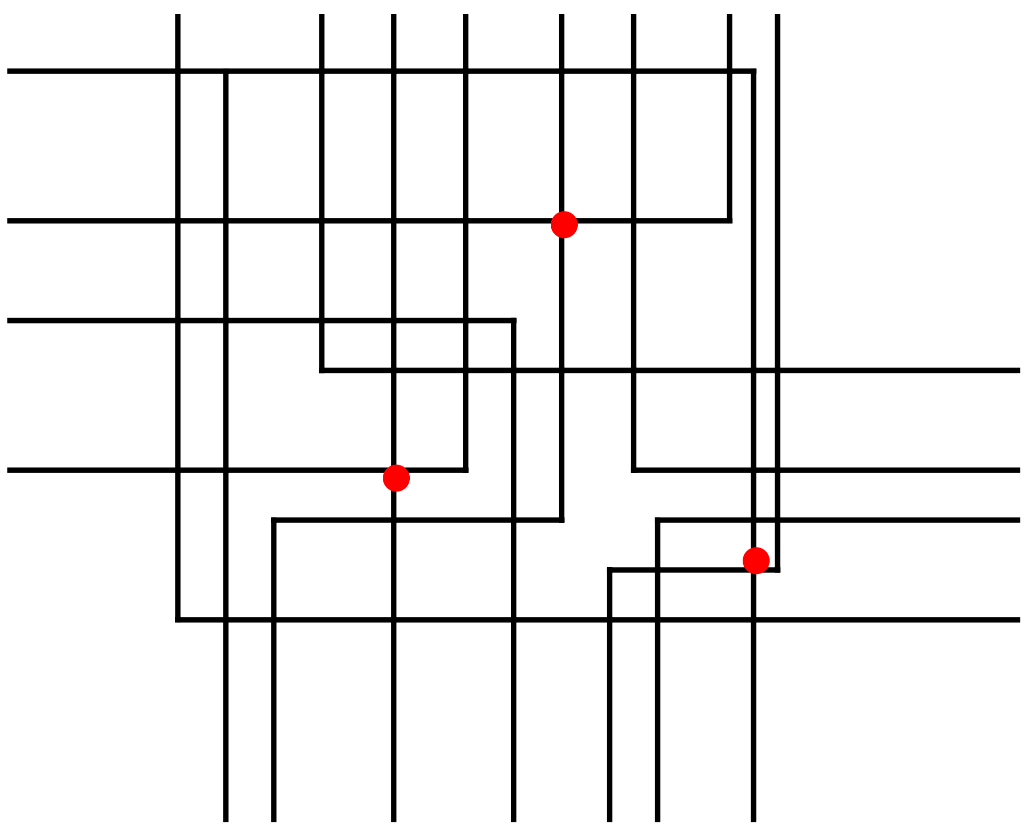
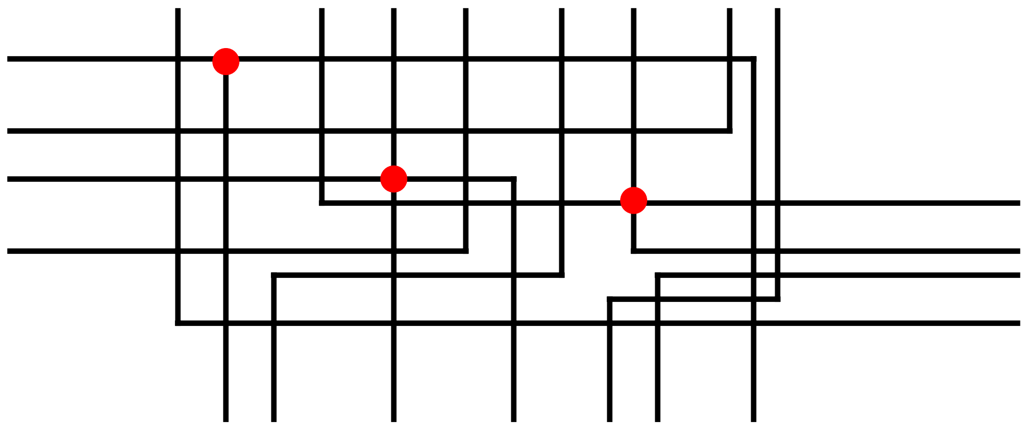
Geometric Intersection

- Determining if there are intersections between graphical objects
- Finding all intersecting pairs
- Brute Force Algorithm
- Plane Sweep Algorithm

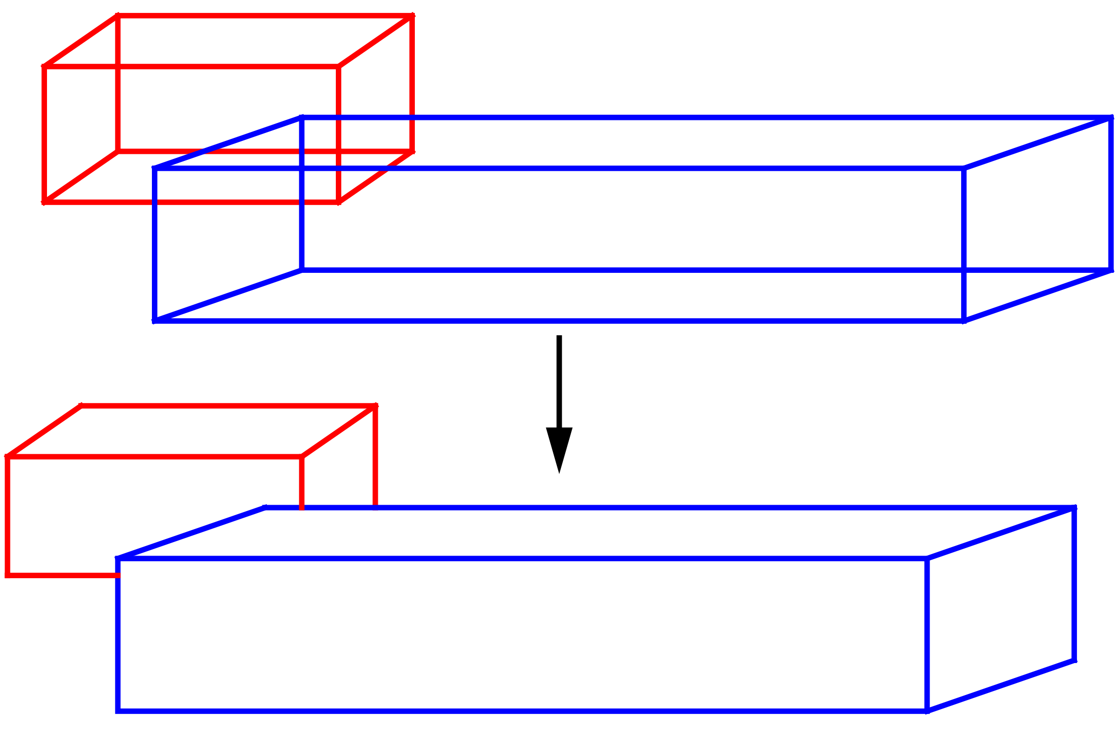


Applications

- Integrated circuit design:



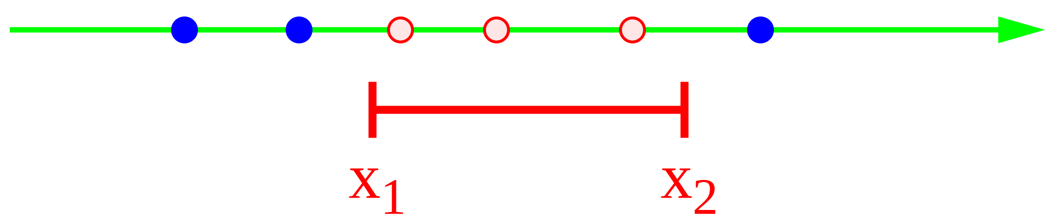
- Computer graphics (hidden line removal):



Range Searching

- Given a set of points on a line, answer queries of the type:

  Report all points x such that $x_1 \le x \le x_2$



- But what if we also want to insert and delete points?

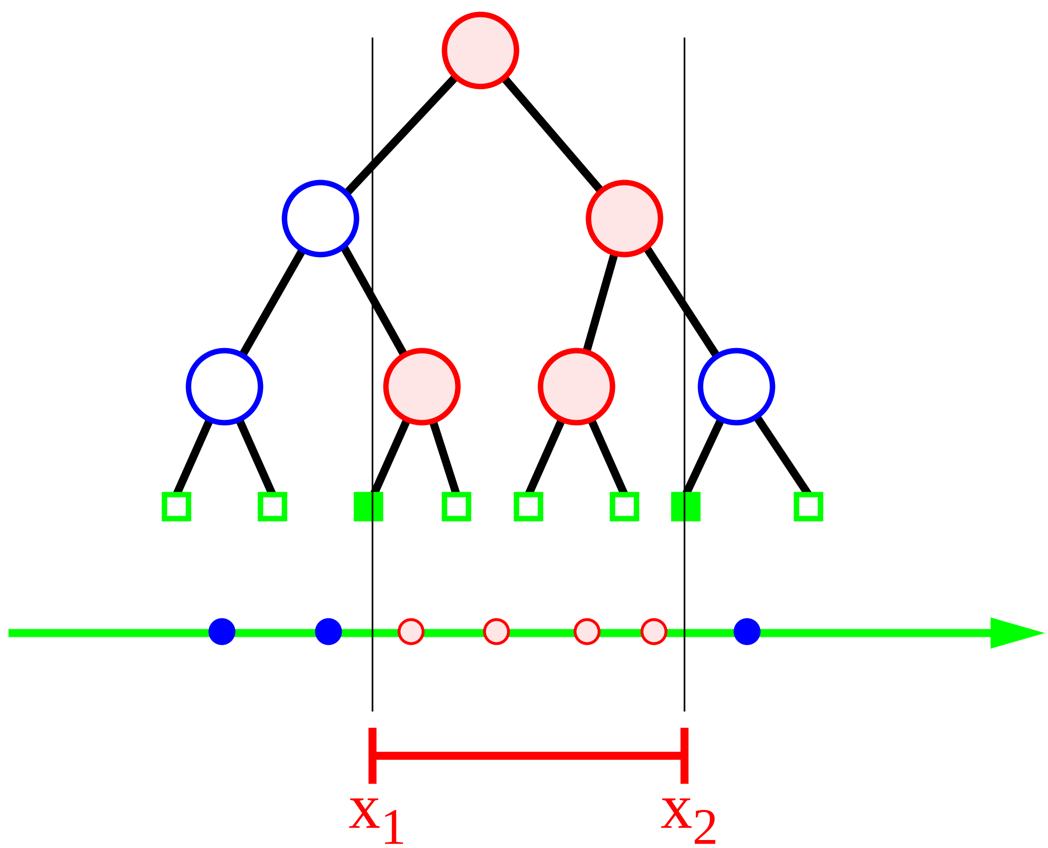
- We'll need a dynamic structure. One which supports these three operations.

  - insert (x)
  - remove (x)
  - range_search (x1, x2)

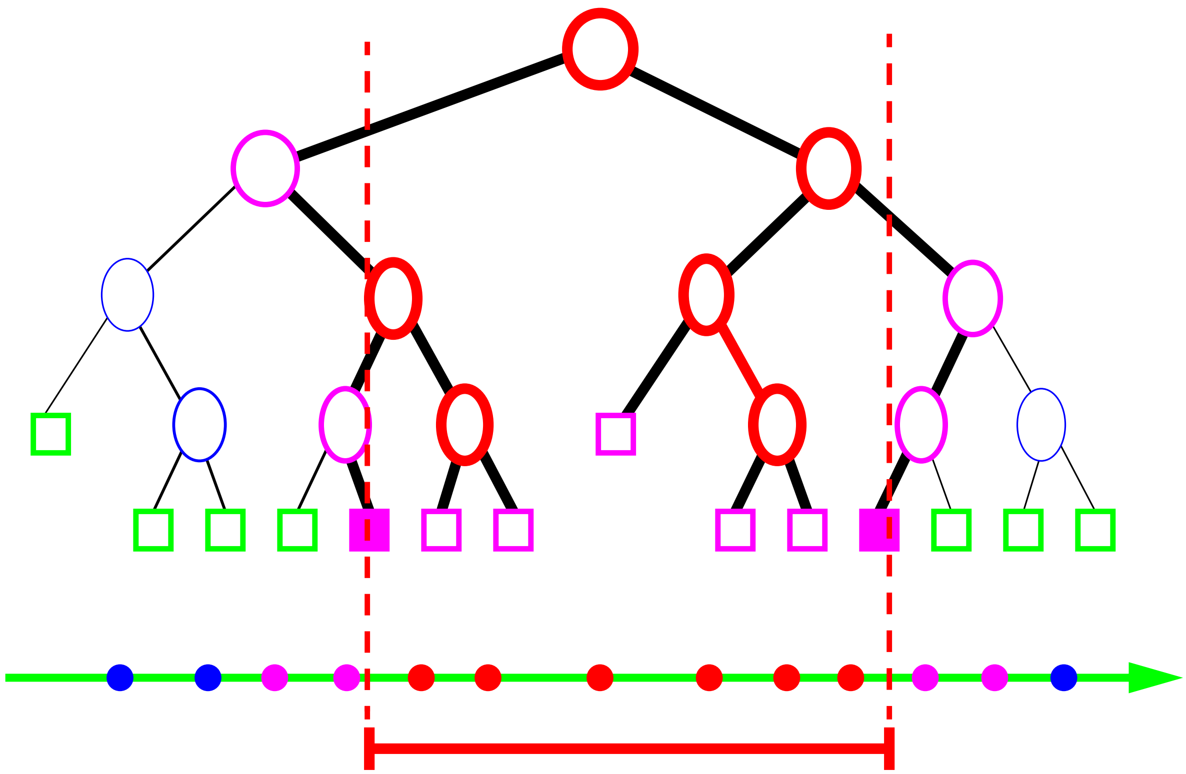
- That's right. It's Red-Black Tree time.

On-Line Range Searching

- Store points in a red-black tree
- Query by searching for x_1 and x_2 (take both directions)

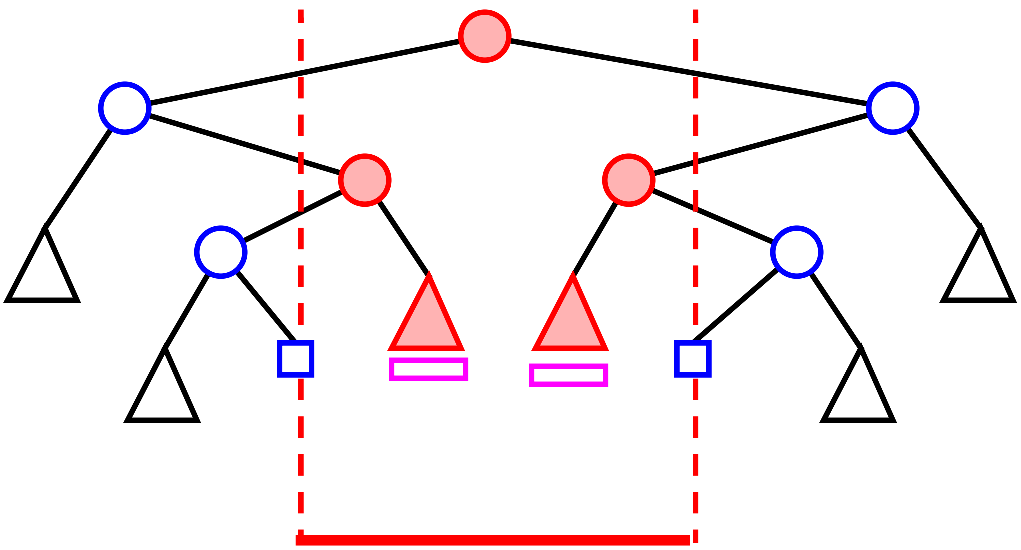


Example



The range search reports the nodes that are highlighted in red. The search also visits the purple nodes that are not reported.

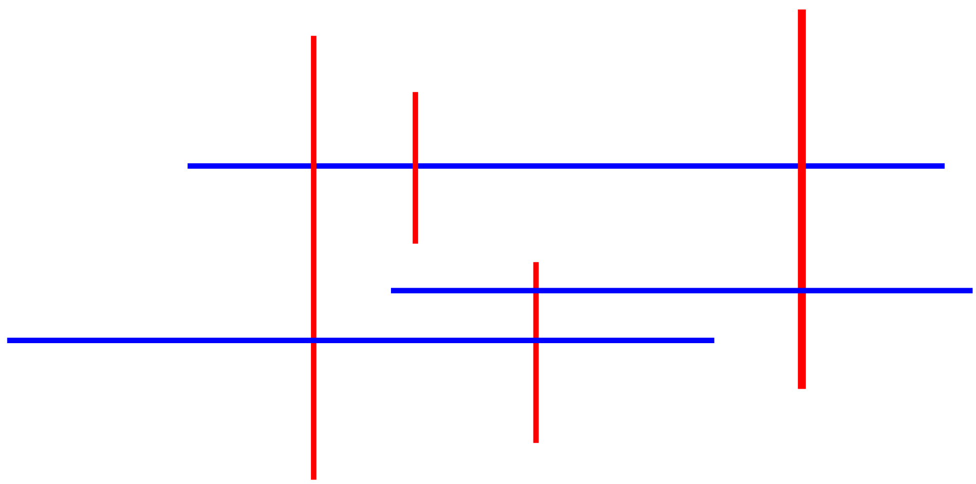
Time Complexity



- The search reports each of the K points that lie within the search range. All of the nodes of the K points reported are visited.
- In addition, ***O***(log N + K) nodes may be visited whose points are not reported.
- Query Time: ***O***(log N + K)

Intersection of Horizontal and Vertical Segments

- Given:



  - H= horizontal segments
  - V= vertical segments
  - S= H ∪ V
  - N= total number of segments
- Report all pairs of **intersecting segments**. (Assuming no coincident horizontal or vertical segments.)

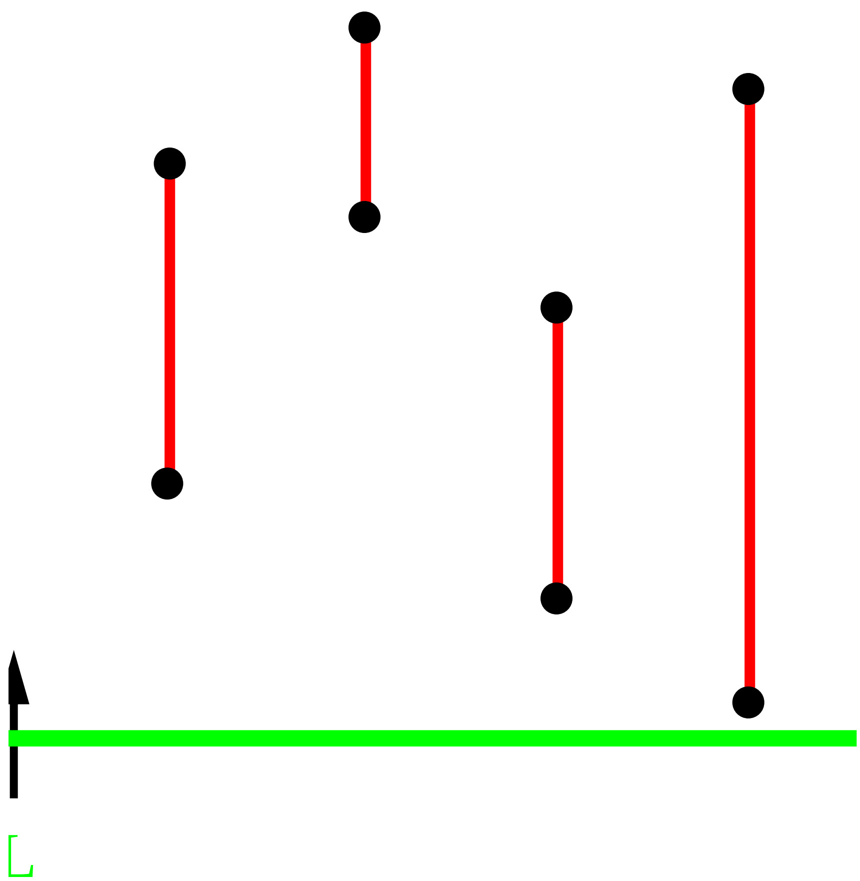
The Brute Force Algorithm

```
for each h in H
  for each v in V
      if h intersects v
          report (h,v)
```

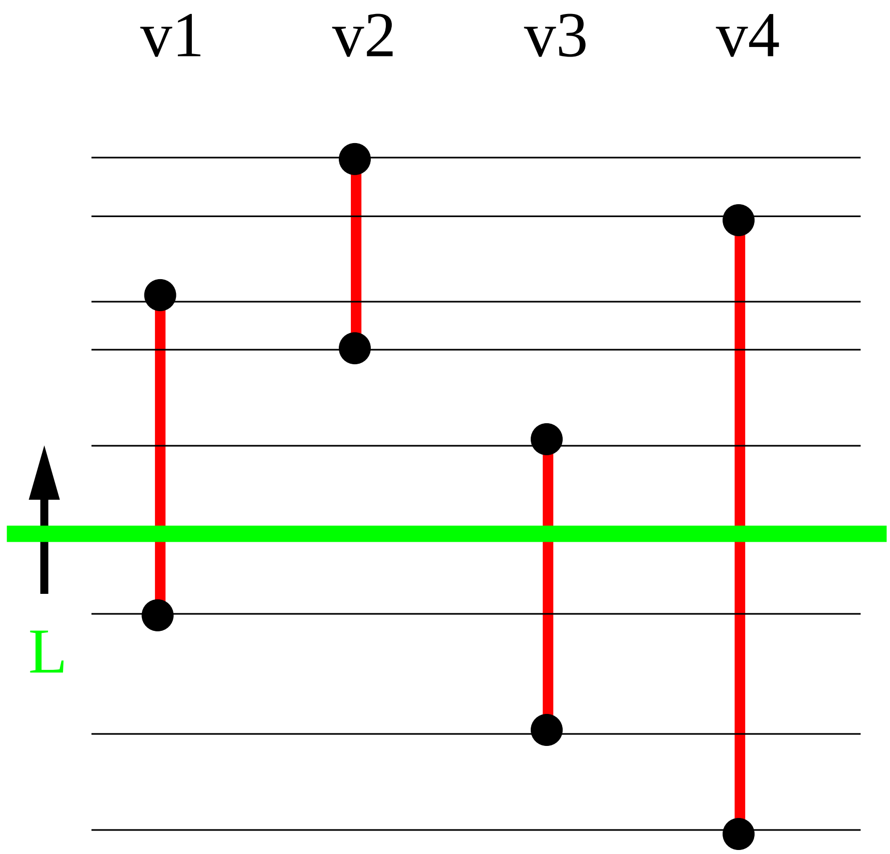
- This algorithm runs in time $O(N_H \cdot N_V) = O(N^2)$
- But the number of intersections could be $<< N^2$.
- We want an output sensitive algorithm:
  Time = $f(N, K)$, where K is the number of intersections.

Plane Sweep Technique

- Horizontal sweep-line L that translates from bottom to top
- Status(L), the set of vertical segments intersected by L, sorted from left to right
  - A vertical segment is **inserted** into Status(L) when L sweeps through its **bottom endpoint**
  - A vertical segment is **deleted** from Status(L) when L sweeps through its **top endpoint**



Evolution of Status in Plane Sweep



Status(L)

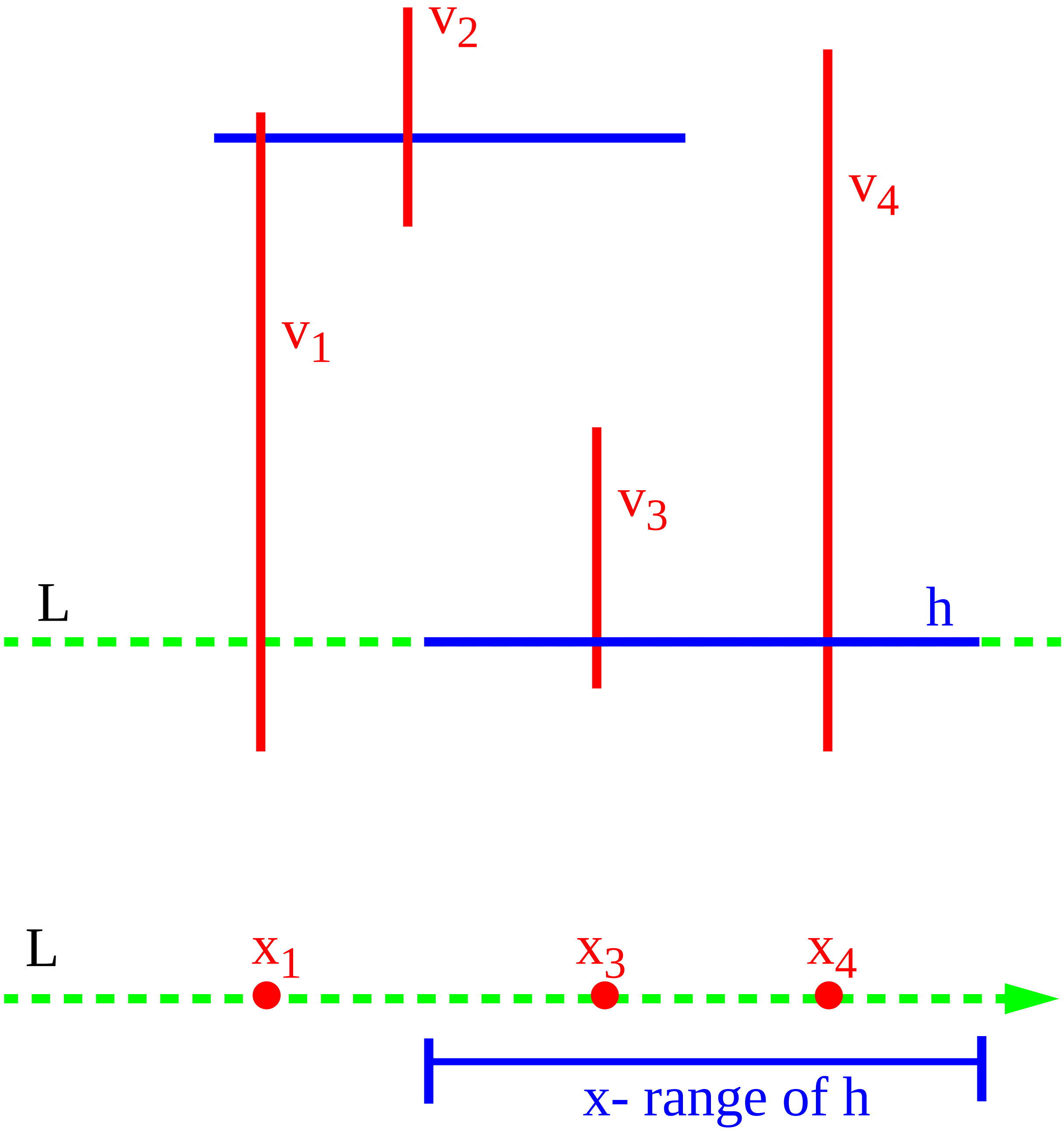
( )
(v2)
(v2 v4)
(v1 v2 v4)
(v1 v4)

(v1 v3 v4)

(v3 v4)

(v4)
( )

Range Query in Sweep



Events in Plane Sweep

- **Bottom endpoint of v**
  - Action:

    insert v into Status(L)

- **Top endpoint of v**
  - Action:

    delete v from Status(L)

- **Horizontal segment h**
  - Action:

    range query on Status(L) with x-range of h

Data Structures

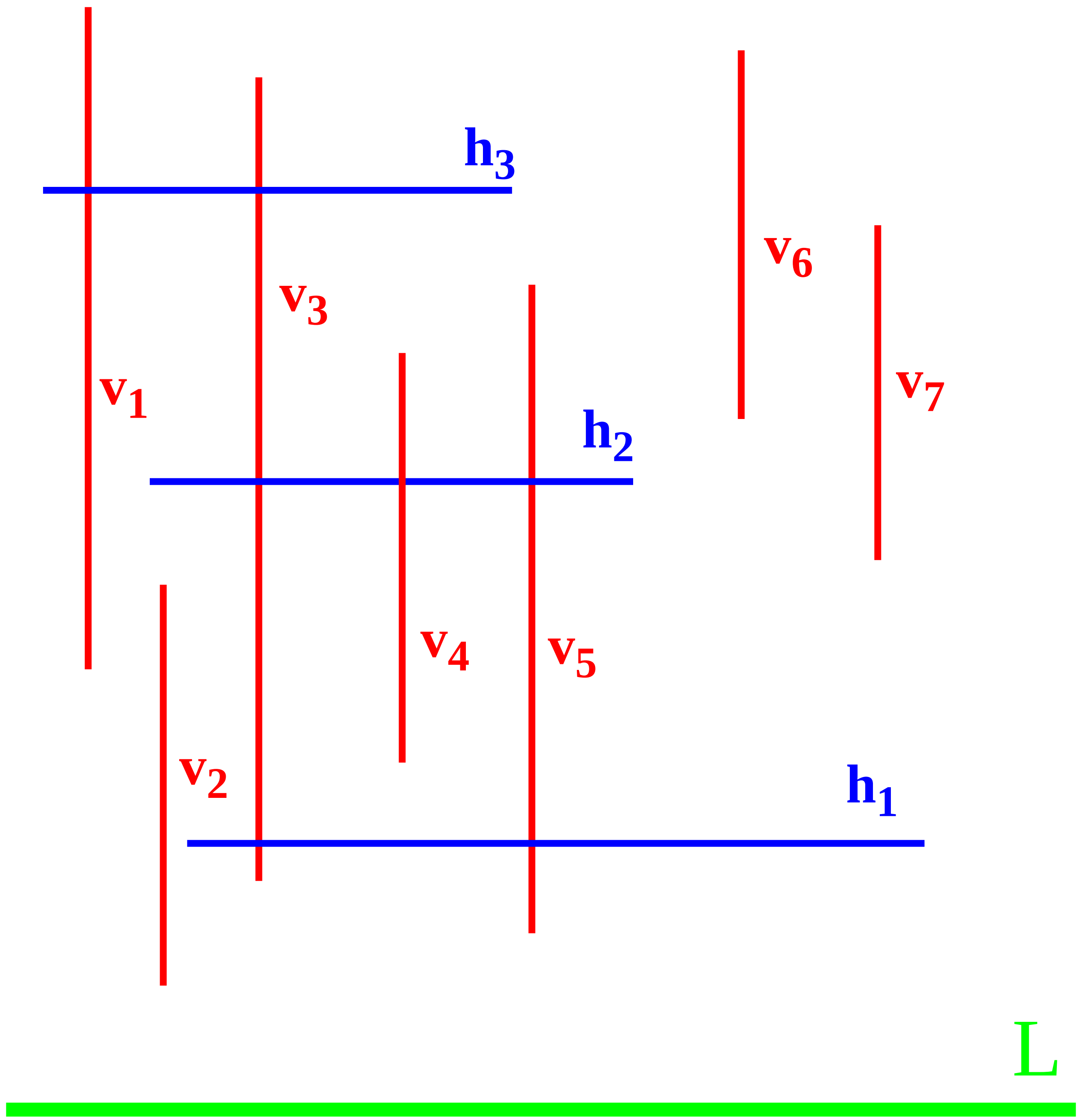
- **Status:**

  - Stores vertical segments
  - Supports insert, delete, and range queries
  - Solution: **AVL tree** or **red-black tree** (key is x-coordinate)

- **Event Schedule:**

  - Stores y-coordinates of segment endpoints, i.e., the order in which segments are added and deleted
  - Supports sequential scanning
  - Solution: **sequence** realized with a sorted array or linked list

Example



Time Complexity

- **Events:**
  - **vertical segment, bottom endpoint**
    - number of occurences: $N_V \leq N$
    - action: insertion into status
    - time: $O(\log N)$
  - **vertical segment, top endpoint**
    - number of occurences: $N_V \leq N$
    - action: deletion from status
    - time: $O(\log N)$
  - **horizontal segment h**
    - number of occurences: $N_H \leq N$
    - action: range searching
    - time: $O(\log N + K_h)$
      K_h = (# vertical segments intersecting h)
- **Total time complexity:**

$$O(N \log N + \sum_h K_h) = O(N \log N + K)$$