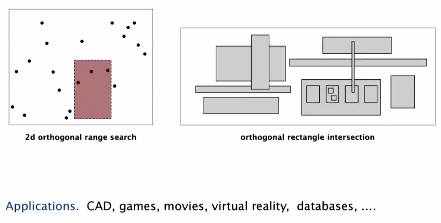
Geometric applications of BSTs

Intersections among geometric objects



Applications: CAD, games, movies, vr, db

Binary search trees provide very efficient solutions to problems of these types

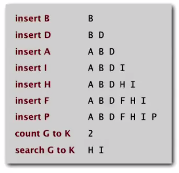
**1D Range Search**

Extension of ordered symbol from before.

Operations:

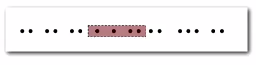
* Insert k/v pair
* Search for key
* Delete key
* + Range search-> find all keys between k1 and k2
* + Range count-> number of keys between k1 and k2

Application: db queries



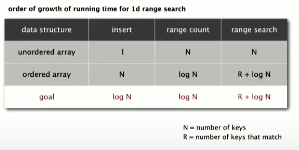
Geometric interpretation

* Keys are points on a line
* Find/count points in a given 1d interval

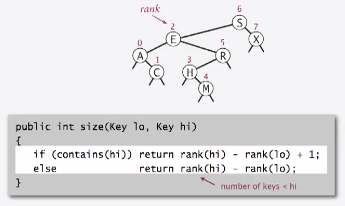


* Unordered array: fast insertion, but slow range search (range count/range search linear time)
* Ordered array: insertion is linear time, but range search and range count are logarithmic time)

Order of growth current vs goal



Range count implementation:

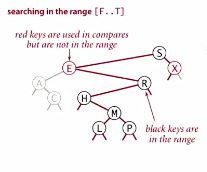


Running time proportional to log N

Range search implementation

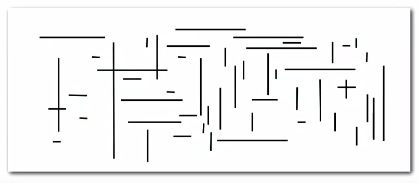
Find all keys between lo and hi:

* recursively find all keys in left subtree if could fall in range
* Check key in current node
* recursively find all keys in right subtree if could fall in range



**Line segment intersection**

Orthogonal line segment intersection search: given N horizontal and vertical line segments, find all intersections (line segments constrained to being either horizontal or vertical)



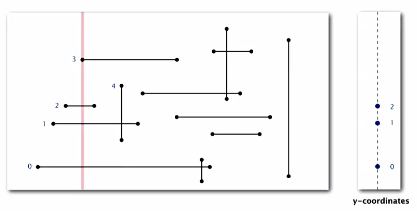
Quadratic algorithm: check all pairs of line segments for intersection

(in these slides: nondegeneracy assumption… all x and y coordinates are distinct)

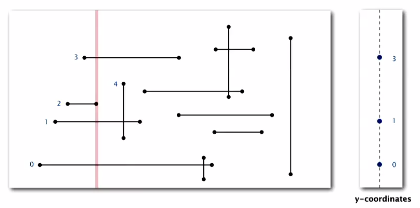
Sweep line algorithm

Vertical line sweeps left to right through data:

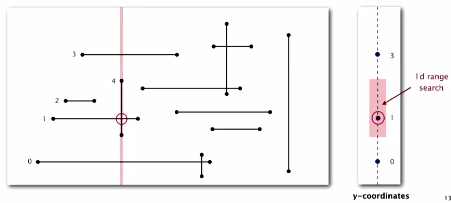
* x coordinates define events
* horizontal-segment (left endpoint) insert y coordinate into BST



* horizontal-segment (right endpoint): remove y coordinate from BST  
  Notice what is now gone below?



* vertical-segment: range search for interval of y endpoints



* Any point in interval (range search) represents a horizontal line segment that is an intersection

Very simple and fast (Linearithmic)

Methods to implement:

* Put x coordinates on a PW (or sort)-> N log N
* Insert y coordinates into BST-> N log N
* Delete y coordinates from BST-> N log N
* Ranges searches in BST-> N log N + R (intersections)

Sweep line reduces 2d orthogonal line segment intersection search to 1d range search