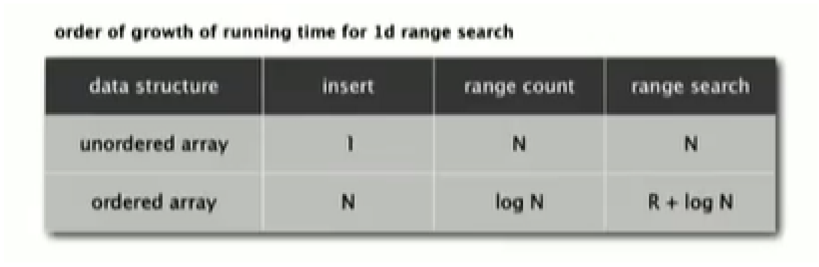
Geometric Application of BSTs

> intersection among geometric objects

> applications---CAD, games, movies, virtual reality, databases

1d range search

- Extension of ordered symbol table
    - insert key-value pair
    - search for key k
    - delete key k
    - range search: find all keys between
    - range count: number of keys between
- application---database queries



order of growth of running time for 1d range search

data structure	insert	range count	range search
unordered array	1	N	N
ordered array	N	log N	R + log N

line segment intersection

Orthogonal line segment intersection search

> Given N horizontal and vertical line segments, find all intersections

Sweep-line Algorithm

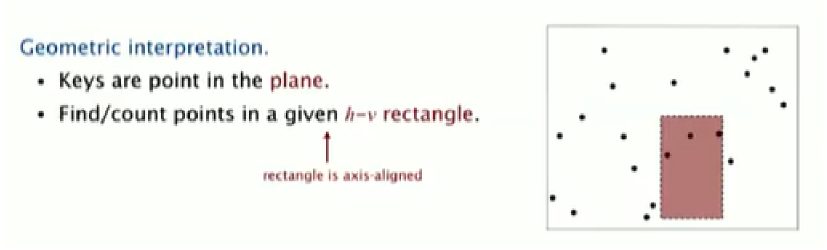
- sweep vertical line from left t right
    - x-coordinates define events
    - h-segment (left end point): insert y-coordinate into BST
    - h-segment (right end point): remove y-coordinate into BST
    - v-segment: range search for interval of y-endpoints

kd trees

2d Tree

Geometric interpretation

> Application---networking circuit design, databases

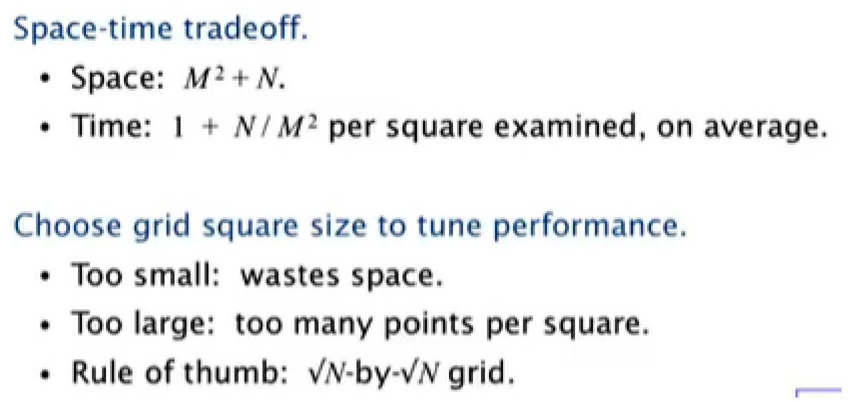


Geometric interpretation.

- Keys are point in the plane.
- Find/count points in a given h–v rectangle.

rectangle is axis-aligned

Grid Implementation



Space-time tradeoff.

- Space: $M^2 + N$.
- Time: $1 + N / M^2$ per square examined, on average.

Choose grid square size to tune performance.

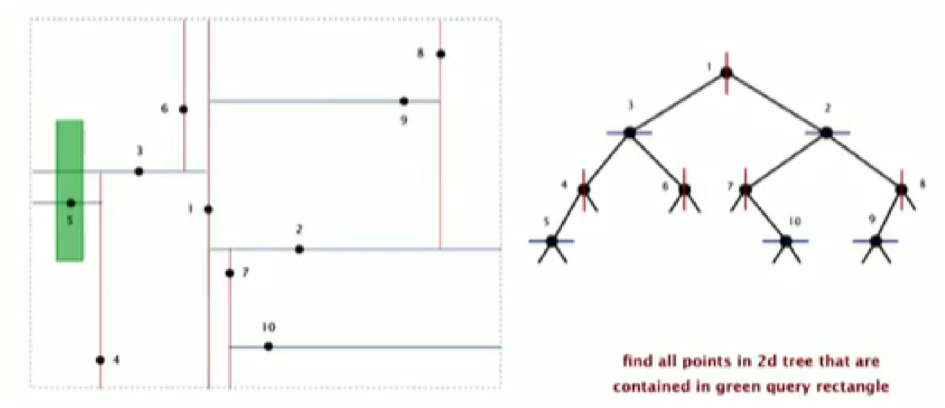
- Too small: wastes space.
- Too large: too many points per square.
- Rule of thumb: $\sqrt{N}$-by-$\sqrt{N}$ grid.

- problem
  - clustering---lists are too long, even though average length is short

Space-partitioning trees

> recursively partition plane into 2 halfplanes

Tree

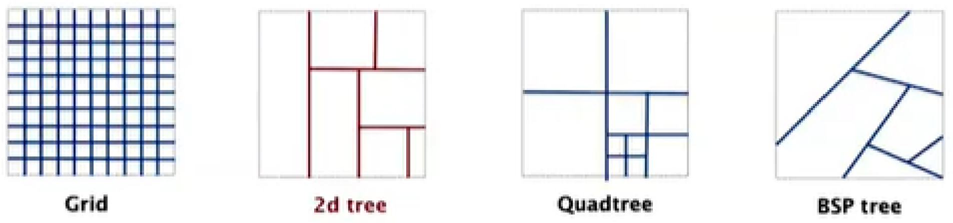


find all points in 2d tree that are contained in green query rectangle

Range search

- Goal---find all points in a query axis-aligned rectangle
- Step
  - check if point in node lies in given rectangle
  - recursively search left/bottom (if any could fall in rectangle)

    - recursively search right/top (if any could fall in rectangle)



- typical case---$R + logN$
- worst case---$R + \square N$

Nearest neighbor search

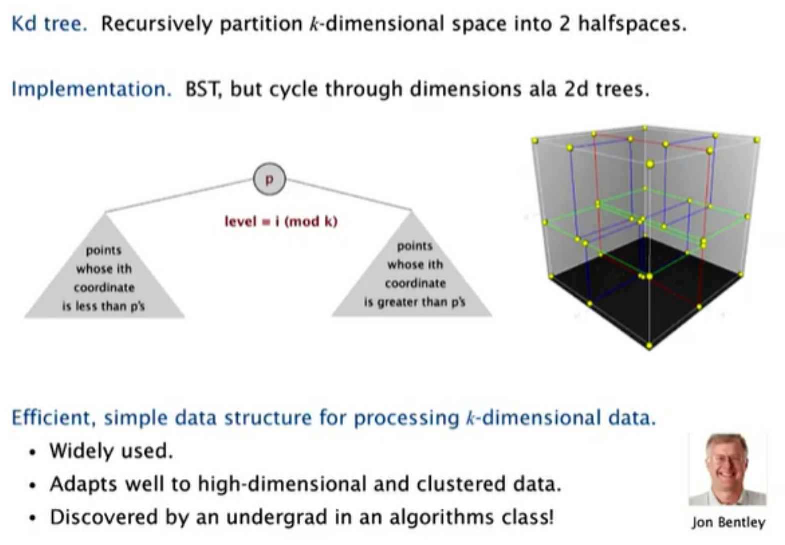
- Goal---find closest point to query point
- step
    - check distance from point in node to query point
    - recursively search left/bottom (if it could contain a closer point)
    - recursively search right/top (if it could contain a closer point)
    - organize method so that it begins by searching for query point

Blocking birds

- Boids---three simple rules lead to complex emergent flocking behavior
- step
    - collision avoidance: point away from k nearest boids
    - flock centering: point towards the center of mass of k nearest boids
    - velocity matching: update velocity to the average of k nearest boids

kd tree

Basics


Kd tree. Recursively partition k-dimensional space into 2 halfspaces.

Implementation. BST, but cycle through dimensions ala 2d trees.

p
level ≡ i (mod k)
points whose ith coordinate is less than p's
points whose ith coordinate is greater than p's

Efficient, simple data structure for processing k-dimensional data.
- Widely used.
- Adapts well to high-dimensional and clustered data.
- Discovered by an undergrad in an algorithms class!

Jon Bentley

N-body simulation

- Goal---simulate the motion of N particles,mutually affected by gravity
- Build 3d tree with N particles as nodes

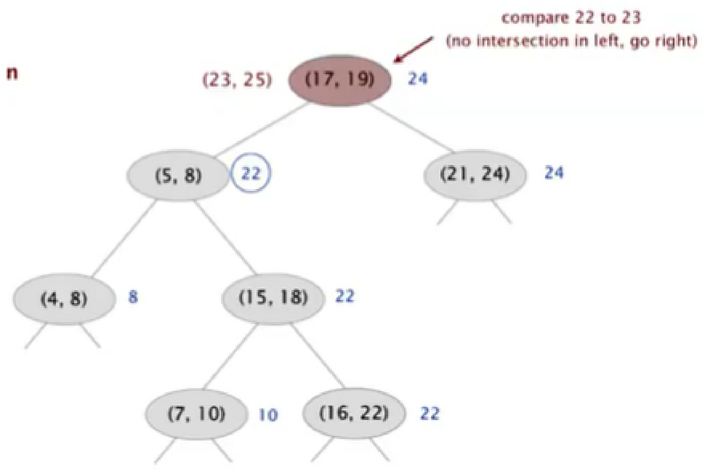
interval search trees

1d interval search

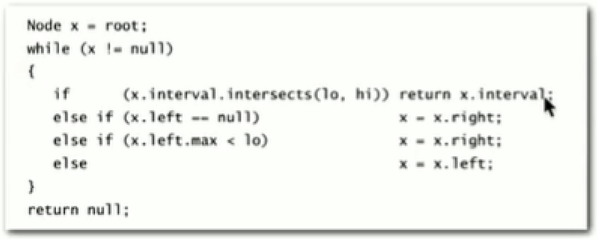
> Data structure to hold set of (overlapping) intervals

create BST

- use left endpoint as BST key
- store max endpoint in subtree rooted at node

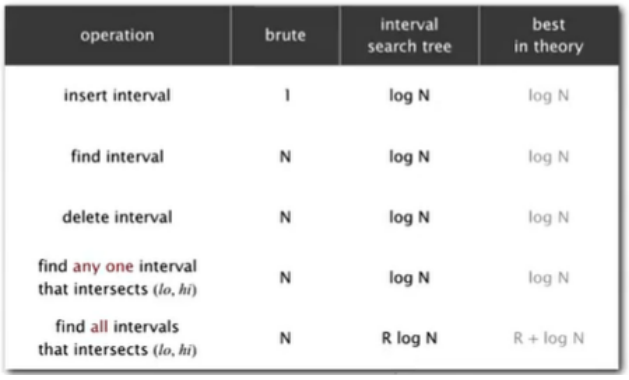


Search



```
Node x = root;
while (x != null)
{
   if      (x.interval.intersects(lo, hi)) return x.interval;
   else if (x.left == null)                x = x.right;
   else if (x.left.max < lo)               x = x.right;
   else                                    x = x.left;
}
return null;
```

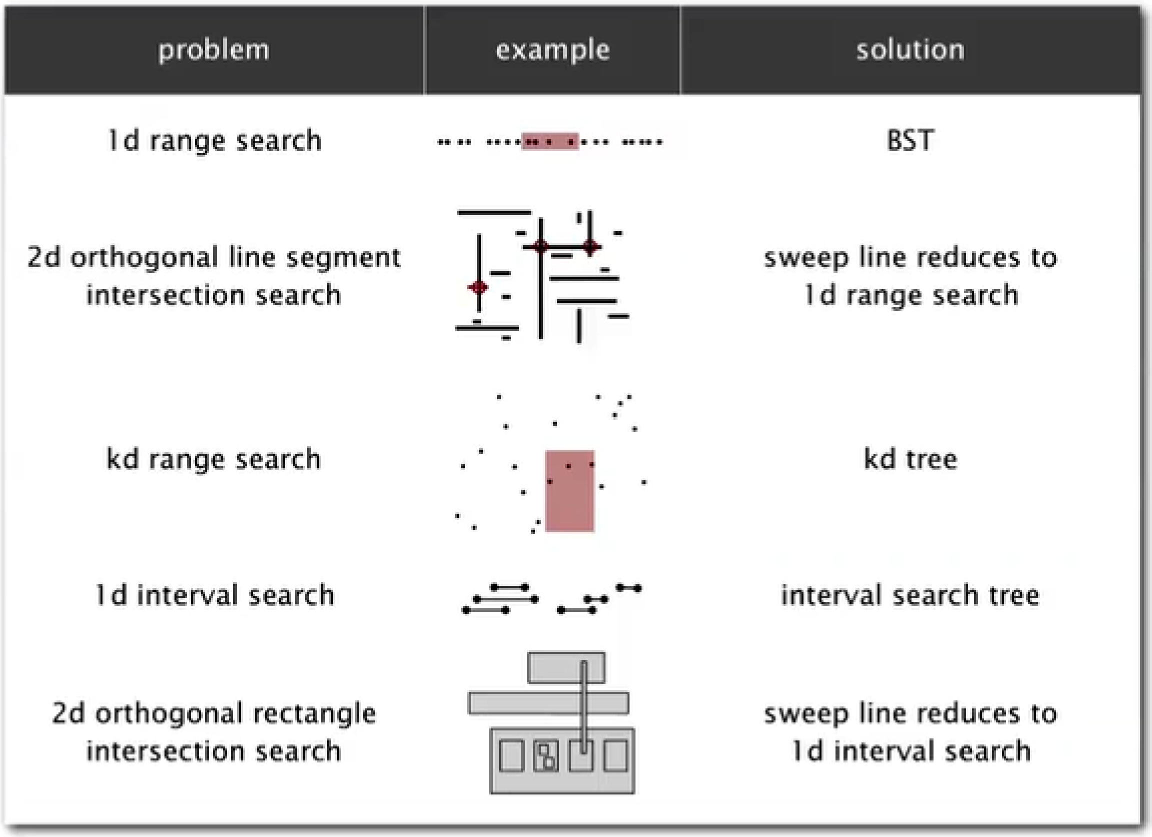
Complexity

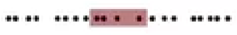
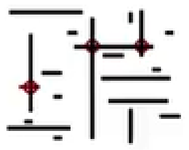
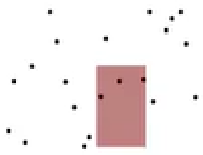
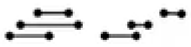
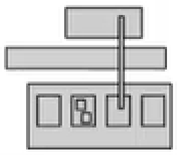


operation	brute	interval search tree	best in theory
insert interval	1	log N	log N
find interval	N	log N	log N
delete interval	N	log N	log N
find any one interval that intersects *(lo, hi)*	N	log N	log N
find all intervals that intersects *(lo, hi)*	N	R log N	R + log N

rectangle intersection

Summary



problem	example	solution
1d range search		BST
2d orthogonal line segment intersection search		sweep line reduces to 1d range search
kd range search		kd tree
1d interval search		interval search tree
2d orthogonal rectangle intersection search		sweep line reduces to 1d interval search