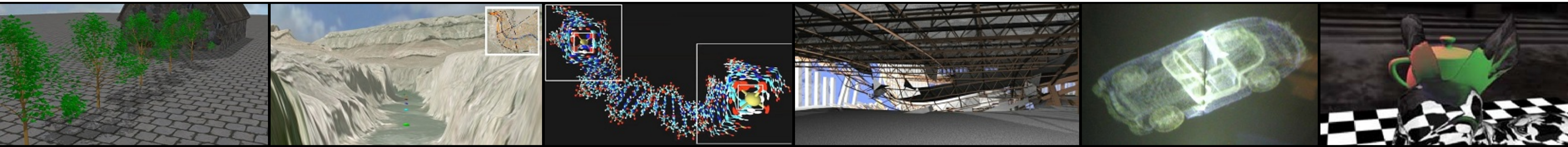
COT 4521: Introduction to Computational Geometry

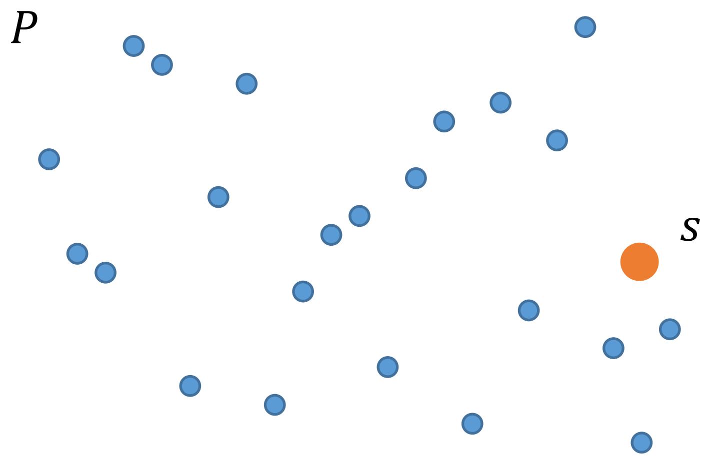
Searches

Paul Rosen
Assistant Professor
University of South Florida

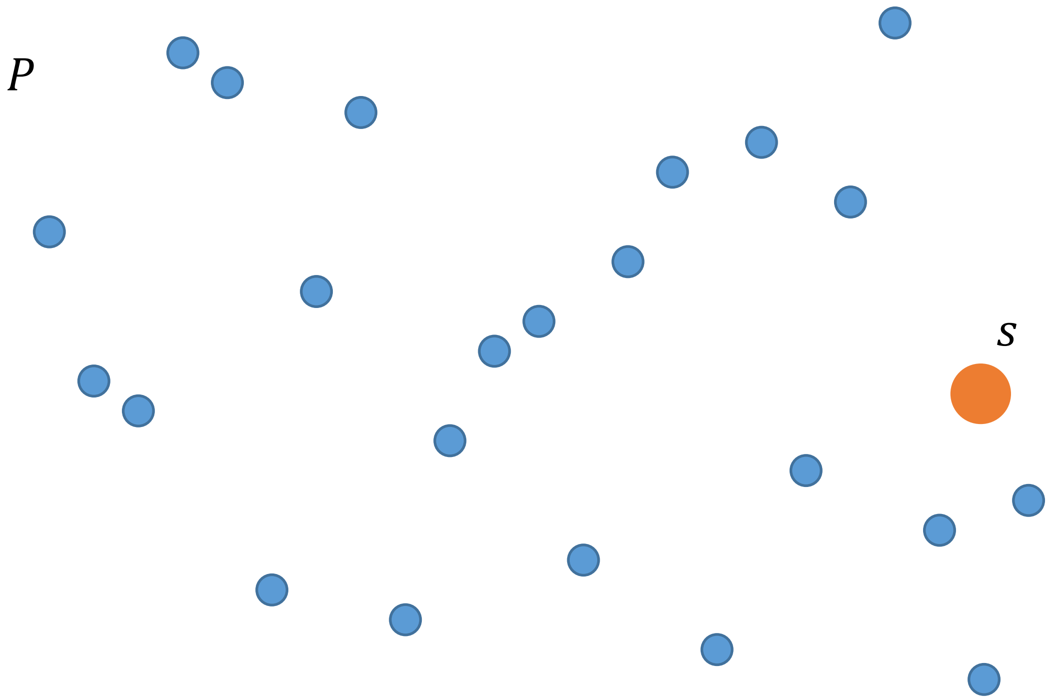


POINT SEARCHES

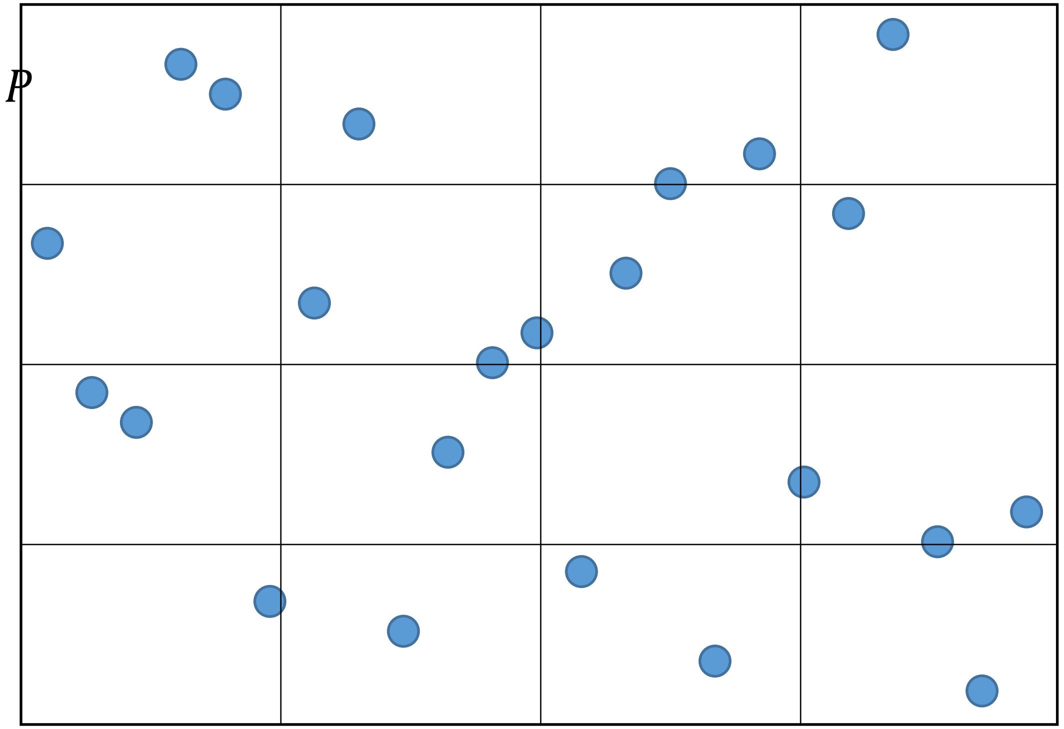
- PROBLEM DEFINITION: GIVEN A SET OF POINTS P IN R^d PROVIDE A DATA STRUCTURE THAT GIVEN AN INPUT SEARCH LOCATION s RETURNS THE k NEAREST POINTS
  - For simplicity we will primarily consider $d = 2$ and $k = 1$



Point Searches : Ideas?



GRIDS

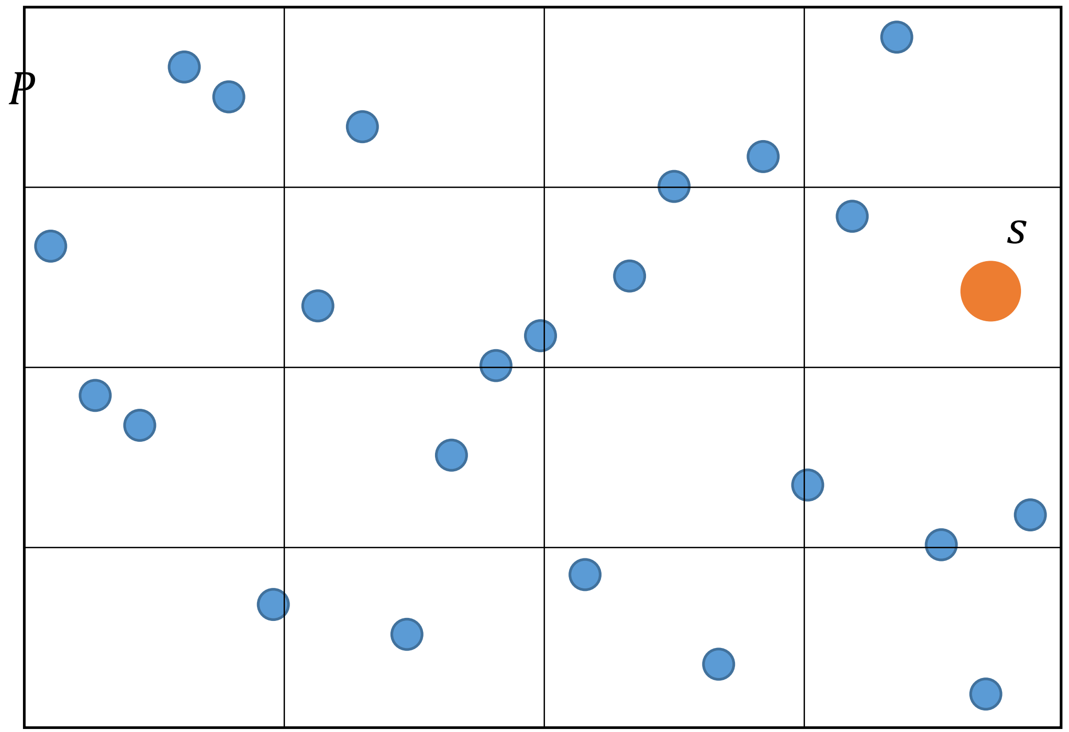


Grid Algorithms

- Construction
  - Find extrema
  - Divide space by predetermined number of size of interval
  - Place points into grid cells

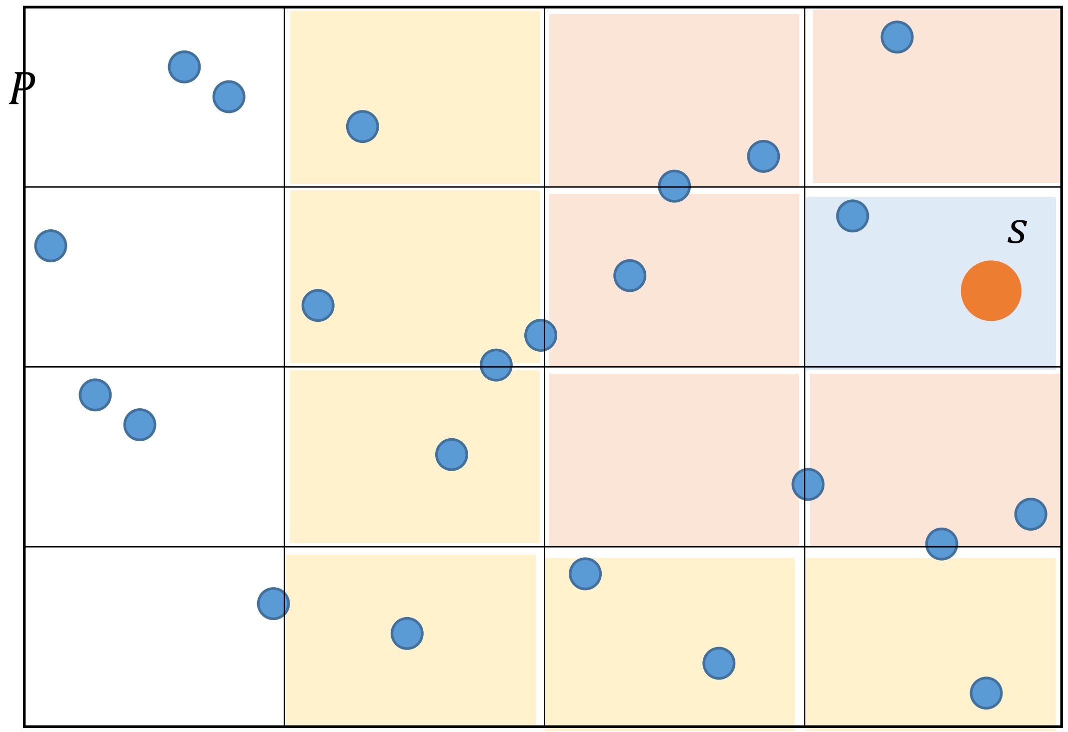
Grid Algorithms

- Search
  - Given a search location
  - Identify closest grid cell
  - Find closest point inside of cell with distance d
  - Perform the same search with neighboring cells in all directions until cell distance is $> \ d$



Grid Algorithms

- Searching for the closest point CP
- Search ring while
  $$d(Ring) < d(CP)$$
- Search Cell if
  $$d(Cell) < d(CP)$$
- For every point p_i in the cell
  - If $d(p_i) < d(CP), CP = p_i$



Finding Distance

- Point-Point Distance?
- Point-cell distance?
- Point-ring distance?

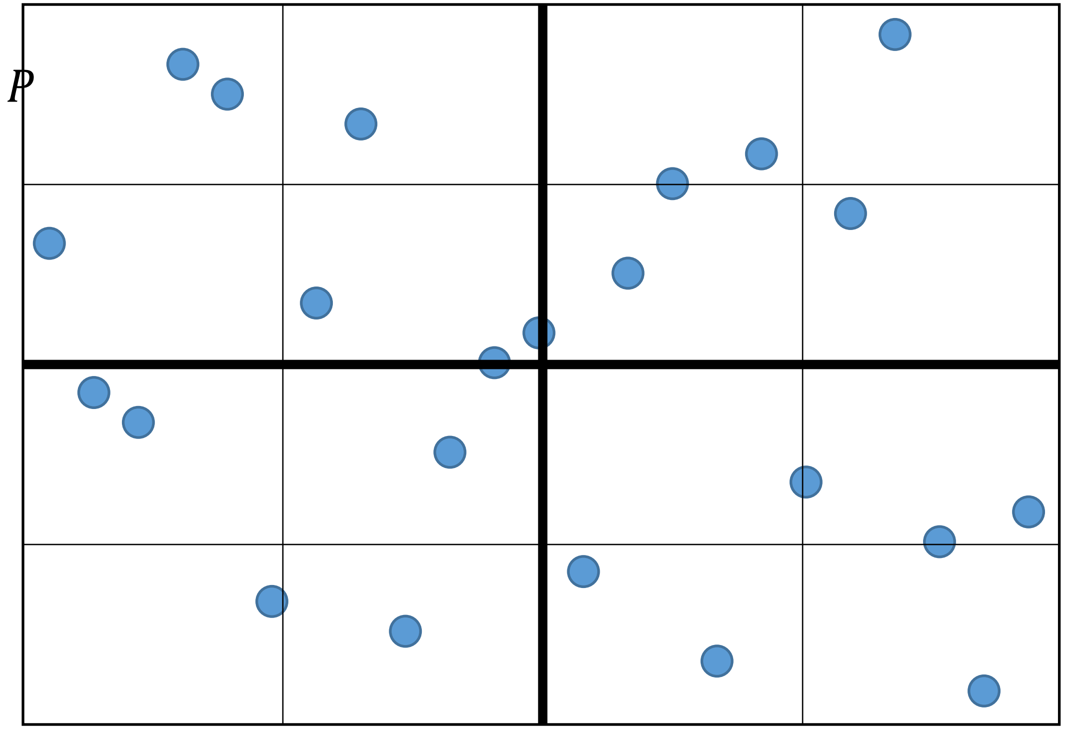
GRIDS

- CONSTRUCTION
  - $O(n)$
- SEARCH
  - Best Case: $O(1)$
  - Worst Case: $O(n)$
  - Average case: 2D $O\left(\frac{n}{r^2}\right) = O(1)$, generally $O\left(\frac{n}{r^d}\right) = O(1)$
- SPACE
  - In 2D $O(r^2 + n)$, generally $O(r^d + n)$

Grids

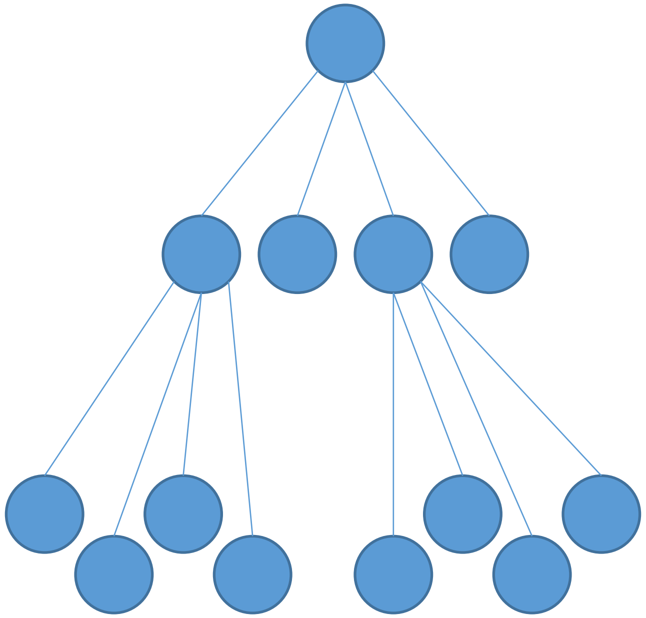
- Examples of failure cases?

QUADTREES



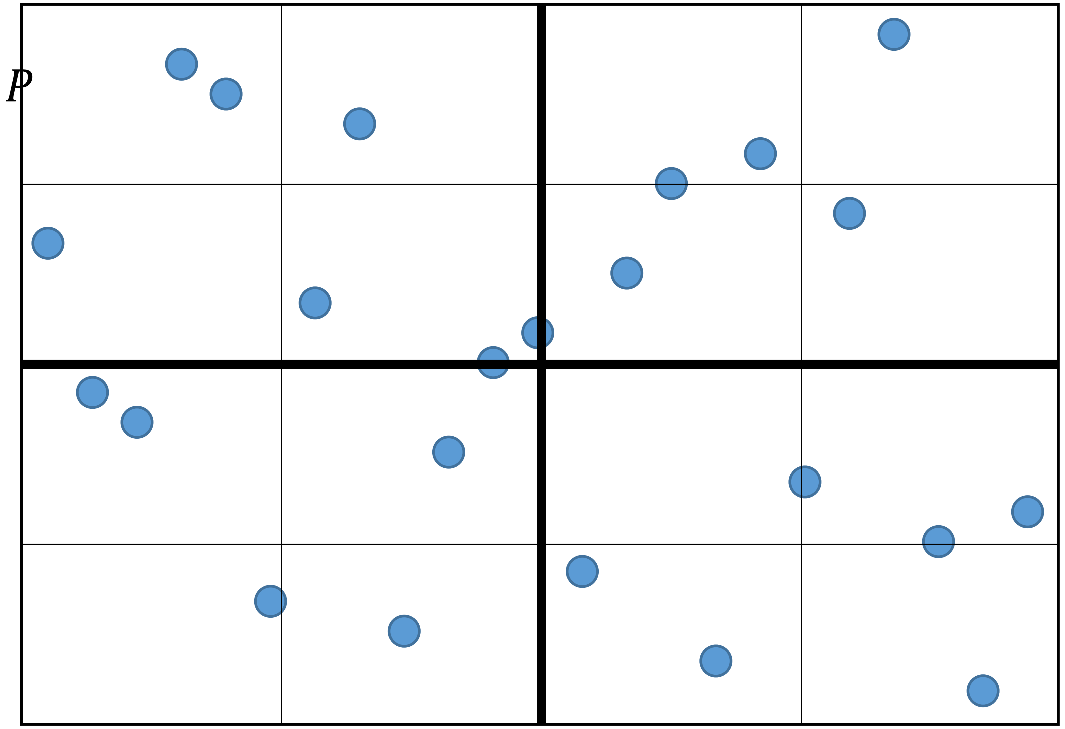
QUADTREE ALGORITHMS

- CONSTRUCTION
  - Find extrema
  - Divide space in half both horizontally and vertically
    - Ideas?
  - Place points into appropriate quadrant
  - Recurse until termination condition
    - Ideas?



QUADTREE

- SEARCH
  - Given a search location
  - Recursively identify closest leaf
  - Find closest point inside of leaf with distance d
  - Perform the same search with neighboring leaves



QUADTREE

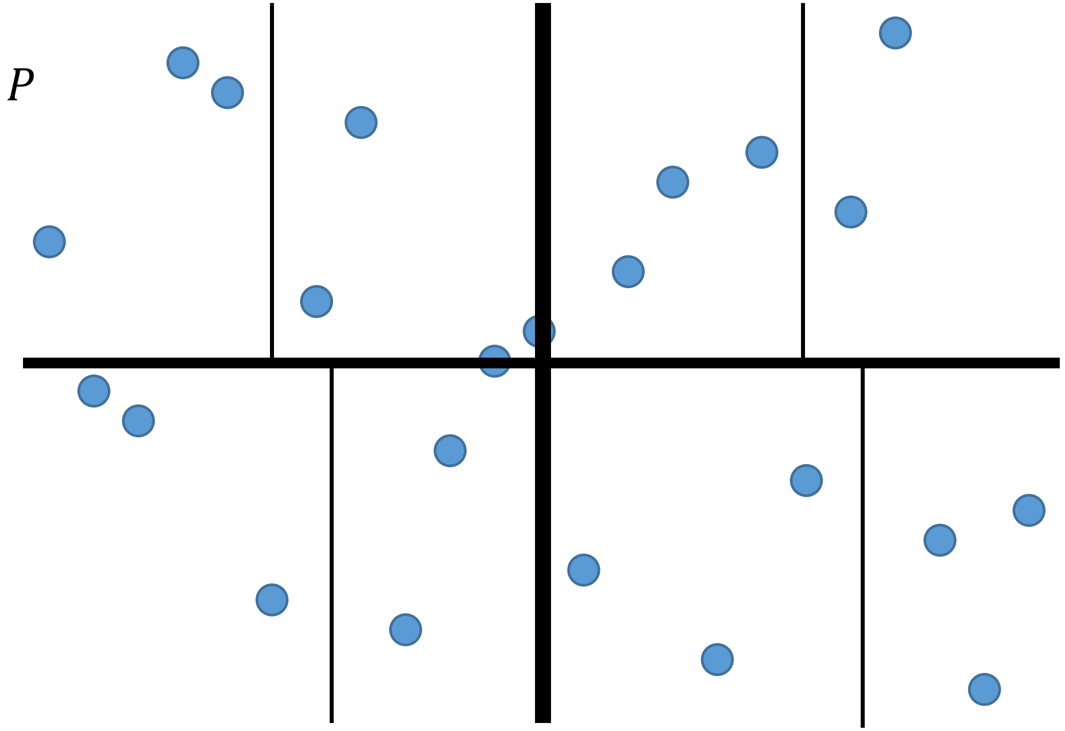
- CONSTRUCTION
  - $O(n \log n)$
- SEARCH
  - Average case: $O(\log n + k)$
    - where k is the size of the leaf nodes
  - Worst Case: $O(n)$
- SPACE: IN $O(n)$

*Octree—3D version of quadtree

QUADTREES

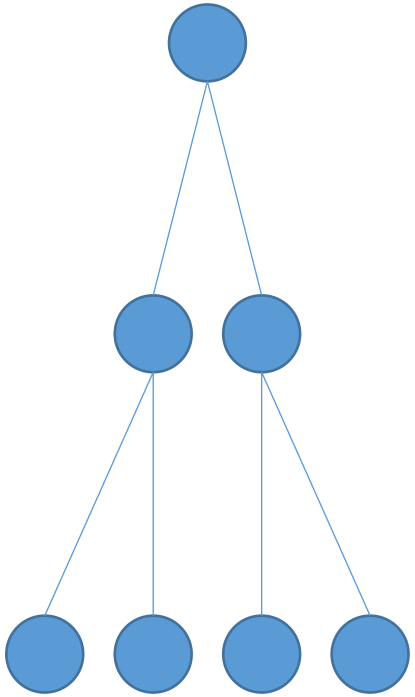
- EXAMPLES OF FAILURE CASES?

KD-TREE



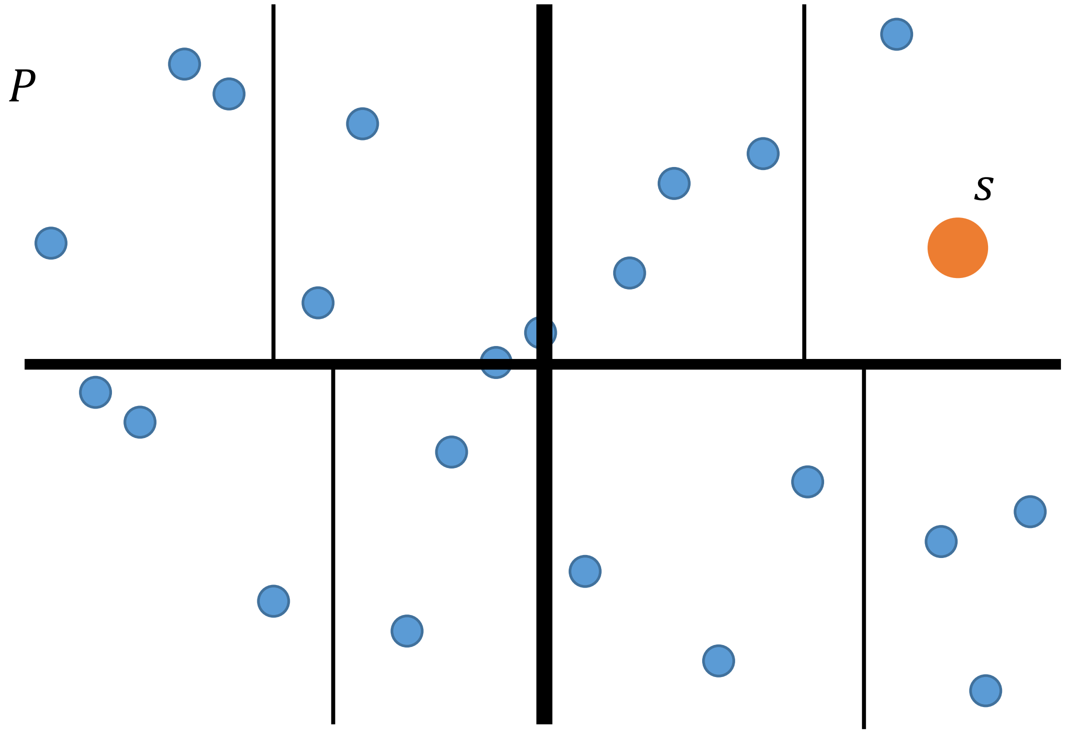
KD-Tree Algorithms

- Construction
  - Find extrema
  - Divide space in half
    - Ideas?
  - Place points into appropriate half
  - Recurse until termination condition
    - Ideas?

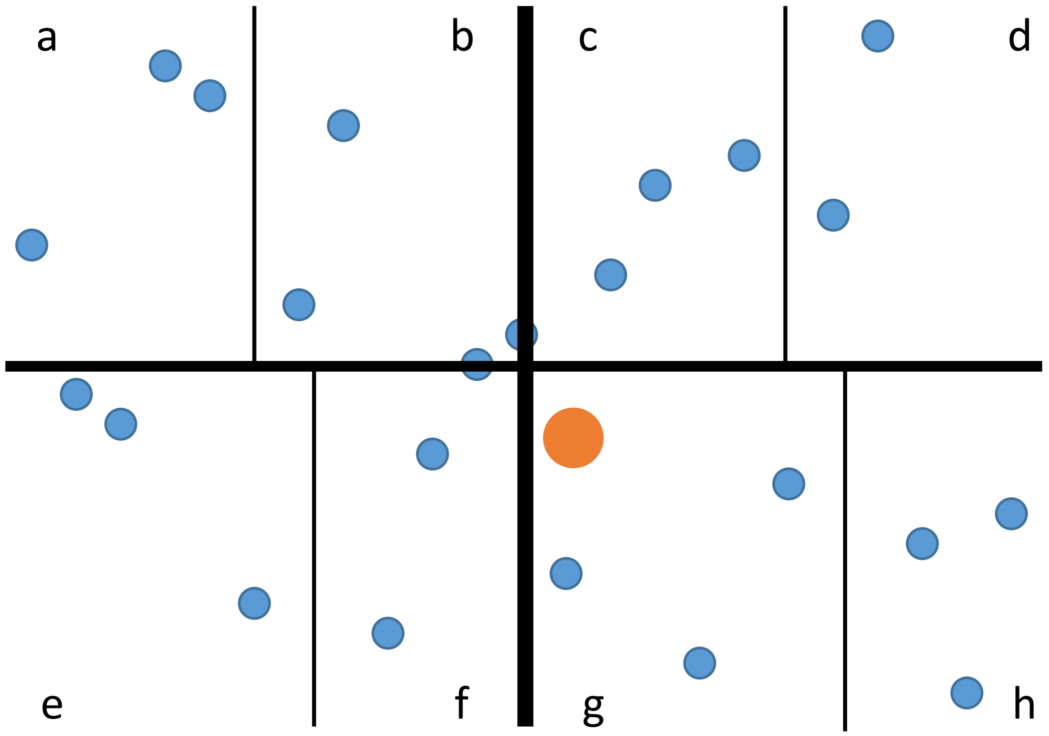


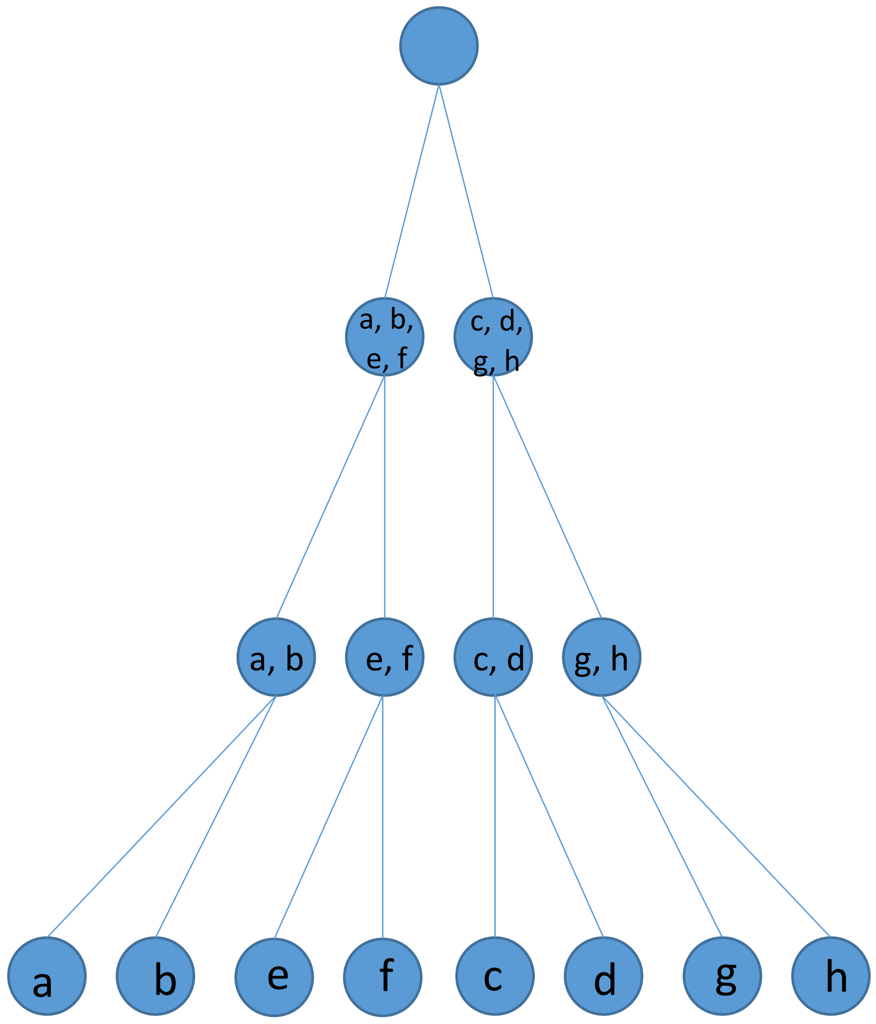
KD-Tree Algorithms

- Search
  - Given a search location
  - Recursively identify closest leaf
  - Find closest point inside of leaf with distance d
  - Perform the same search with neighboring leaves



KD-Tree Search Example





KD-Tree

- Construction:
  - $O(n \log n)$

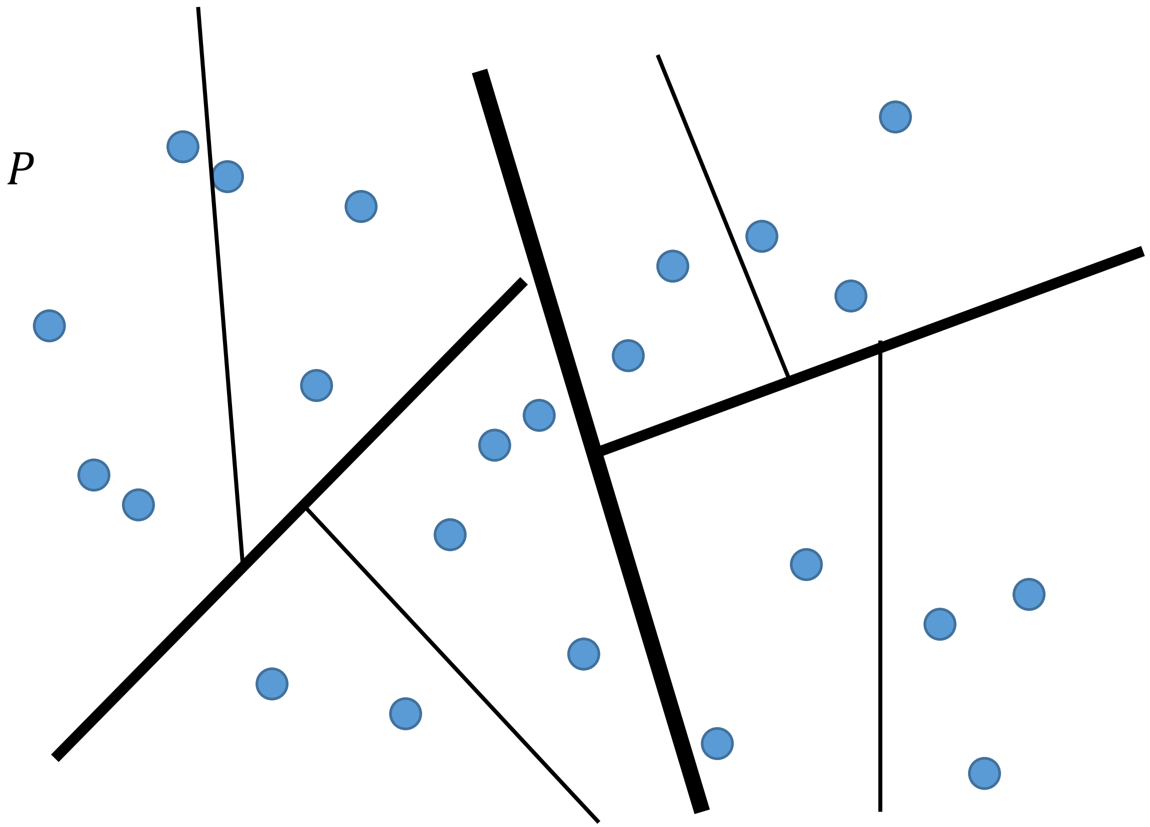
- Search
  - Average case: $O(\log n + k)$, where k is the size of the leaf nodes
  - Worst Case: $O(n)$

- Space:
  - $O(n)$

KD-Tree

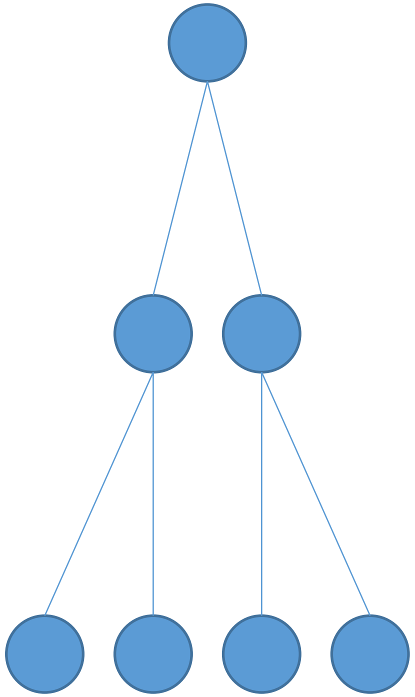
- Examples of failure cases?

Binary Space Partition (BSP)



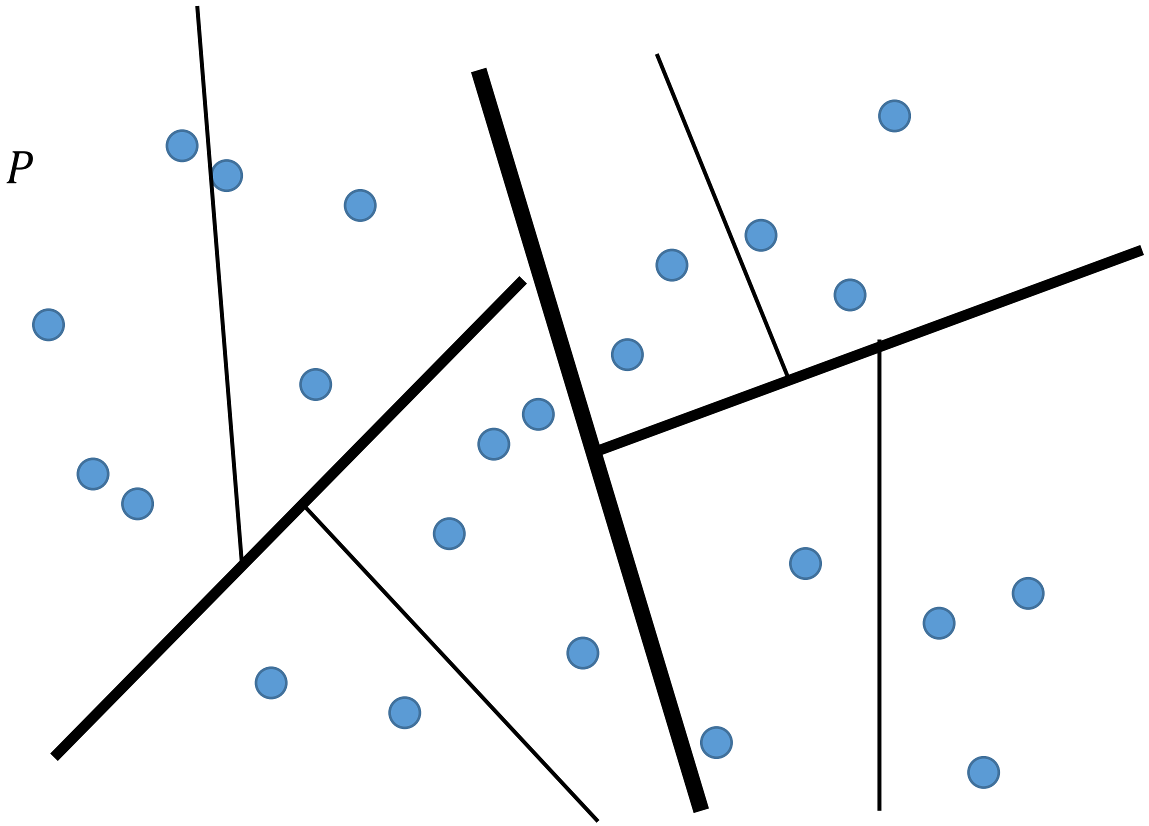
BSP ALGORITHMS

- CONSTRUCTION
  - Find extrema
  - Divide space in half
  - Place points into appropriate half space
  - Recurse until termination condition



BSP ALGORITHMS

- SEARCH
  - Given a search location
  - Recursively identify closest leaf
  - Find closest point inside of leaf with distance d
  - Perform the same search with neighboring leaves



Binary Space Partition

- Construction
  - $O(n \log n)$

- Search
  - Average case: $O(\log n + k)$
    - where k is the size of the leaf nodes
  - Worst Case: $O(n)$

- Space:
  - $O(n)$

Binary Space Partition

- Examples of failure cases?

TASKS

- CLOSEST POINT SEARCH
  - Task covered thus far
- K-NEAREST NEIGHBORS SEARCH (NEXT)

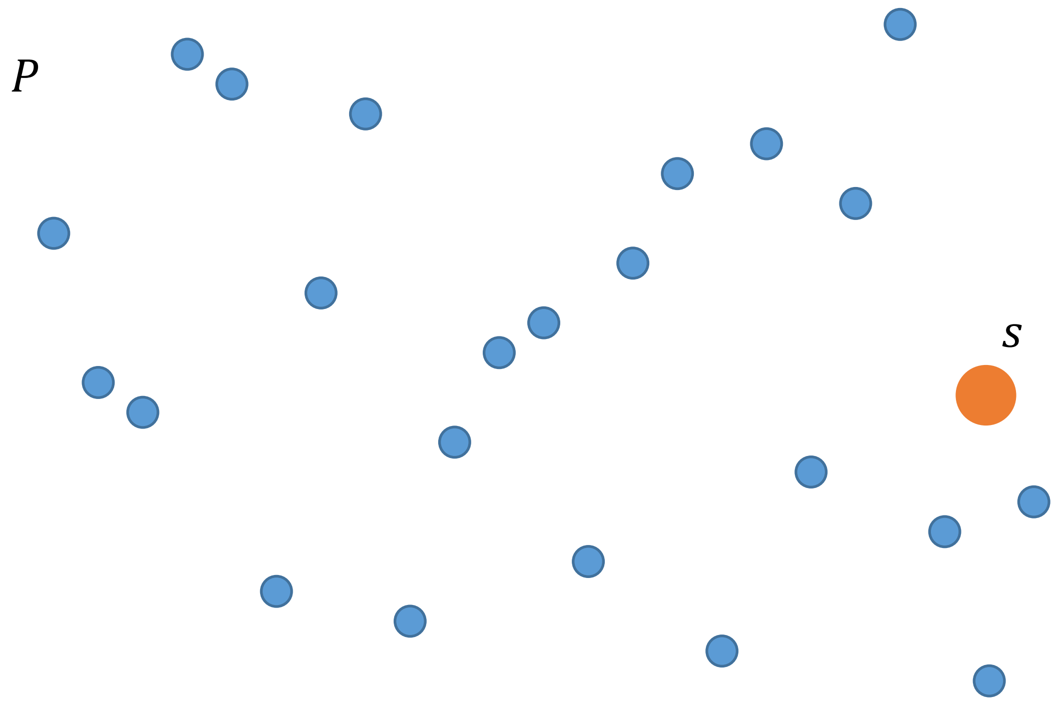
- RANGE SEARCH

- CLUSTERING (WE'LL TALK ABOUT THIS NEXT LECTURE)

K-nearest neighbor Search

- Problem: Given a set of points P, find the k-nearest neighbors efficiently

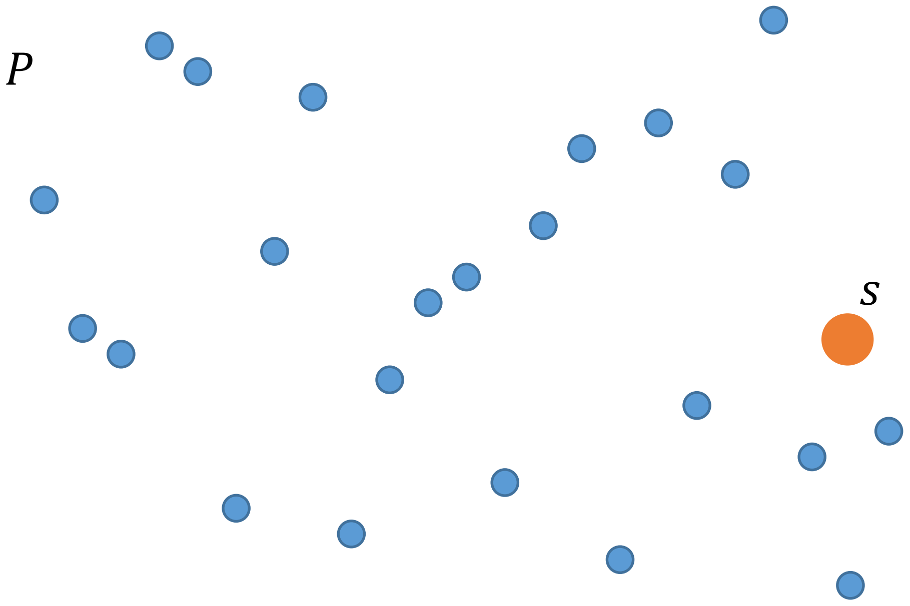
- Ideas?



K-nearest neighbor Search

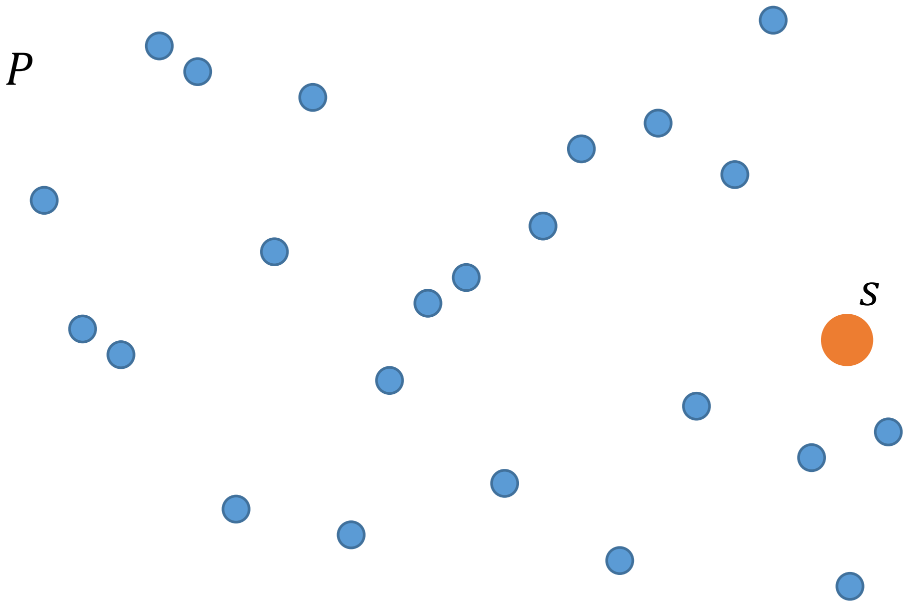
- Use spatial partitioning of your choice
- Keep a list of k length for the closest points
- Search performed similarly to closest point search, except that our stopping condition is on the furthest point in the list

- How do we store the k points most efficiently?



K-nearest neighbor Search

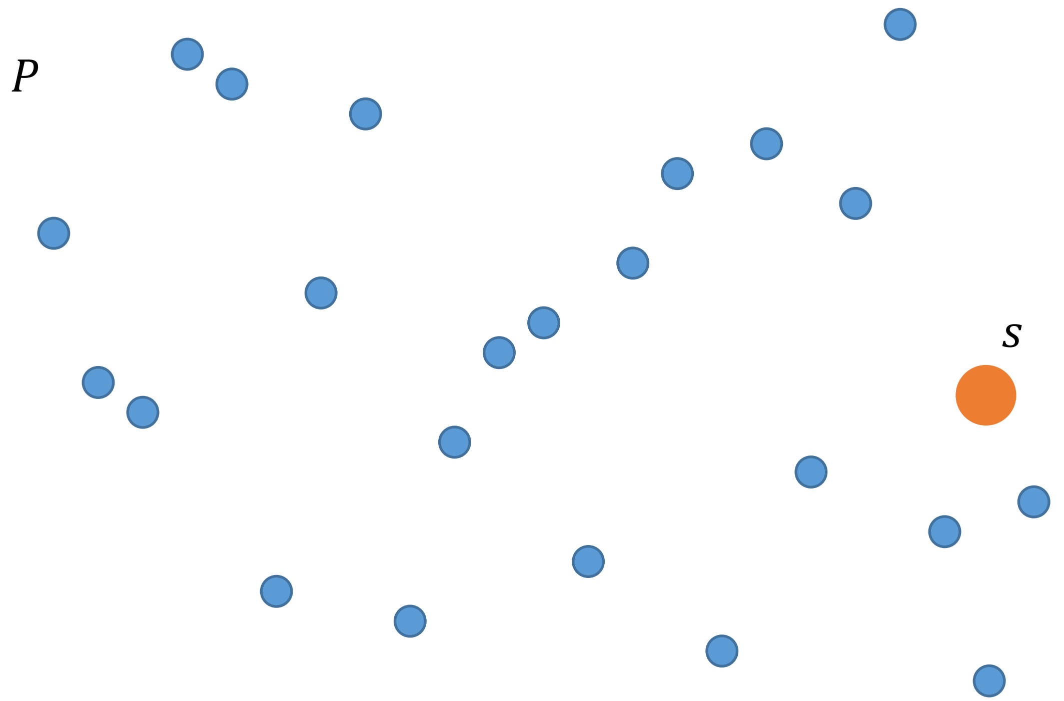
- Make the list efficient by keeping it sorted
  - use a balanced binary tree—O(log k) insertion costs
  - Or insertion sort—O(k) insertion cost



K-NEAREST NEIGHBOR SEARCH

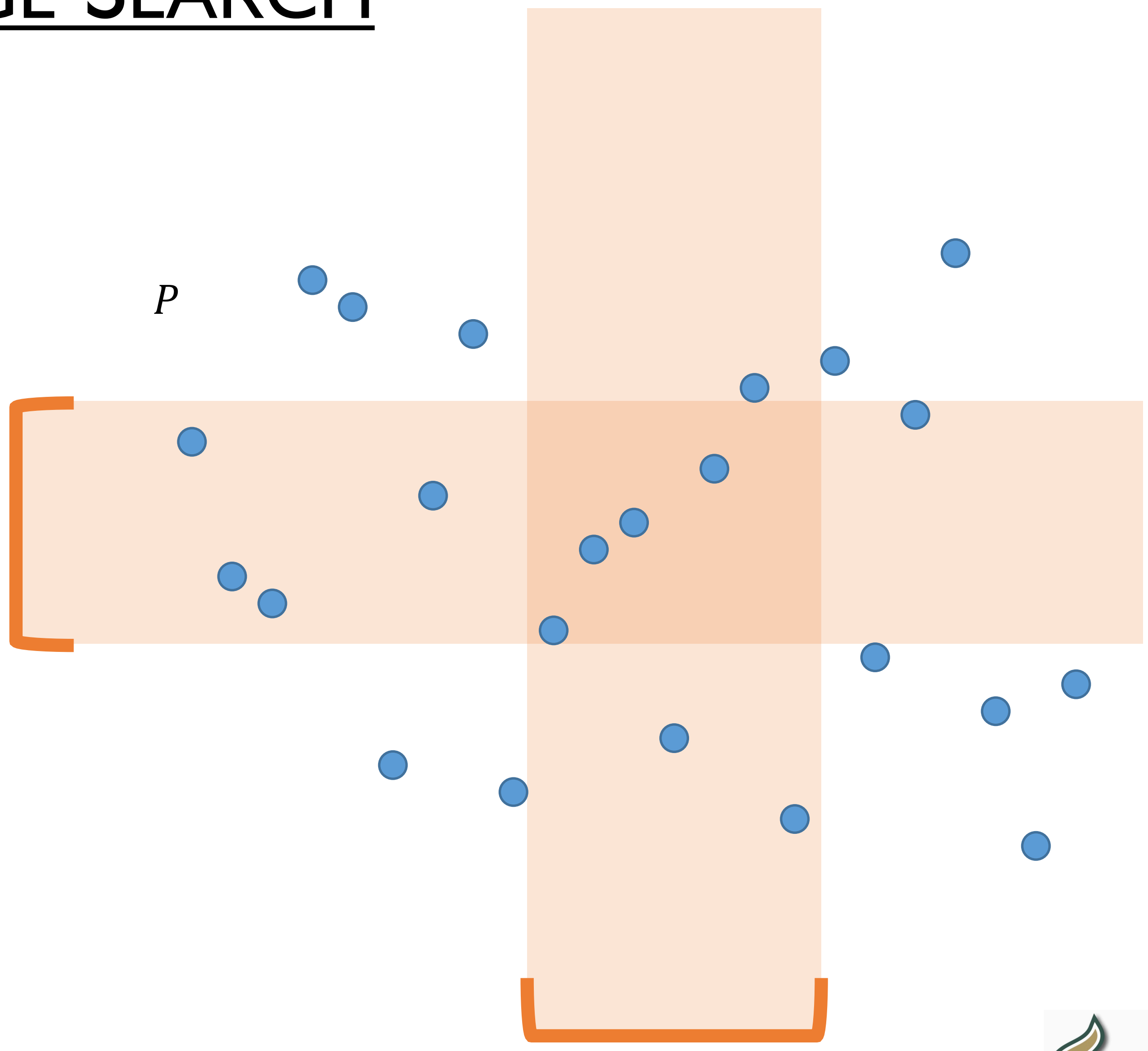
- PROBLEM: GIVEN A SET OF POINTS P, FIND THE POINTS WITHIN INTERVALS IN BOTH DIRECTIONS EFFICIENTLY

- IDEAS?



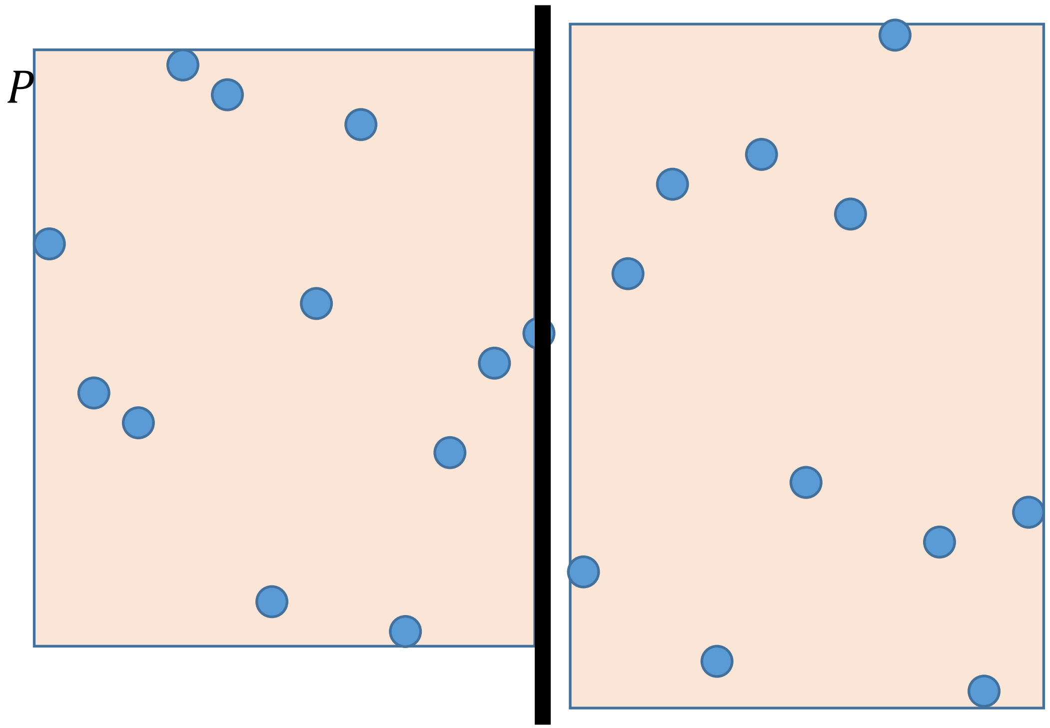
Range Search

- Problem: Given a set of points P, efficiently find the set of points within a specified range
- Ideas?



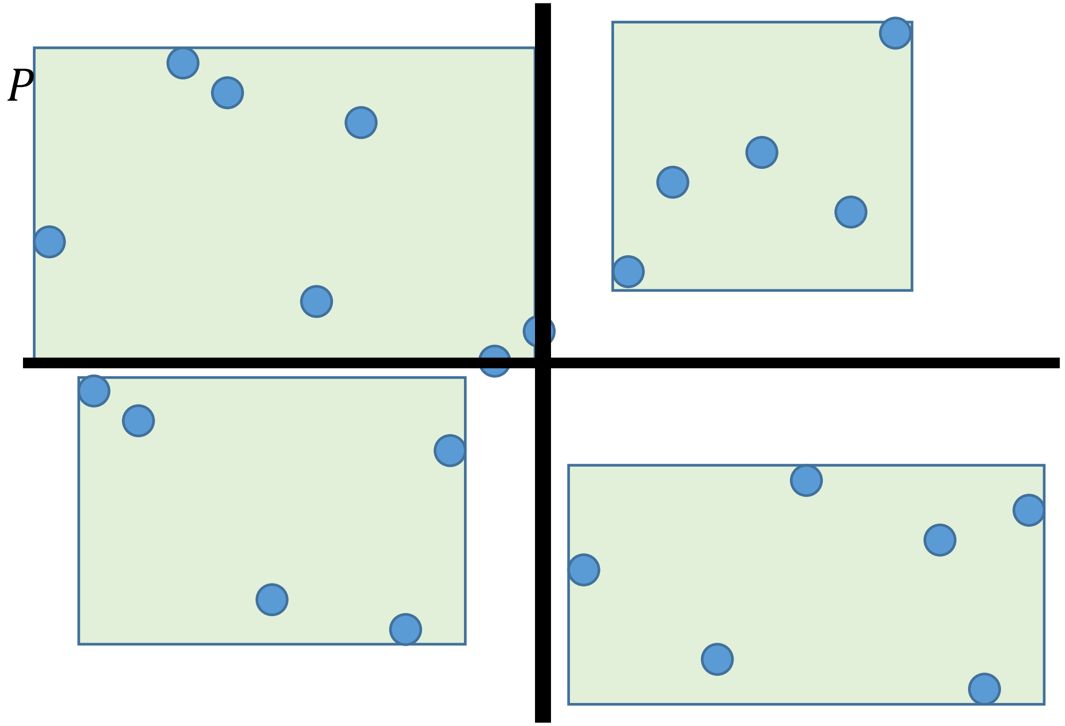
Bounding Volume Hierarchy

- Variation on previous approaches that stores the volume covered at each level of the tree.



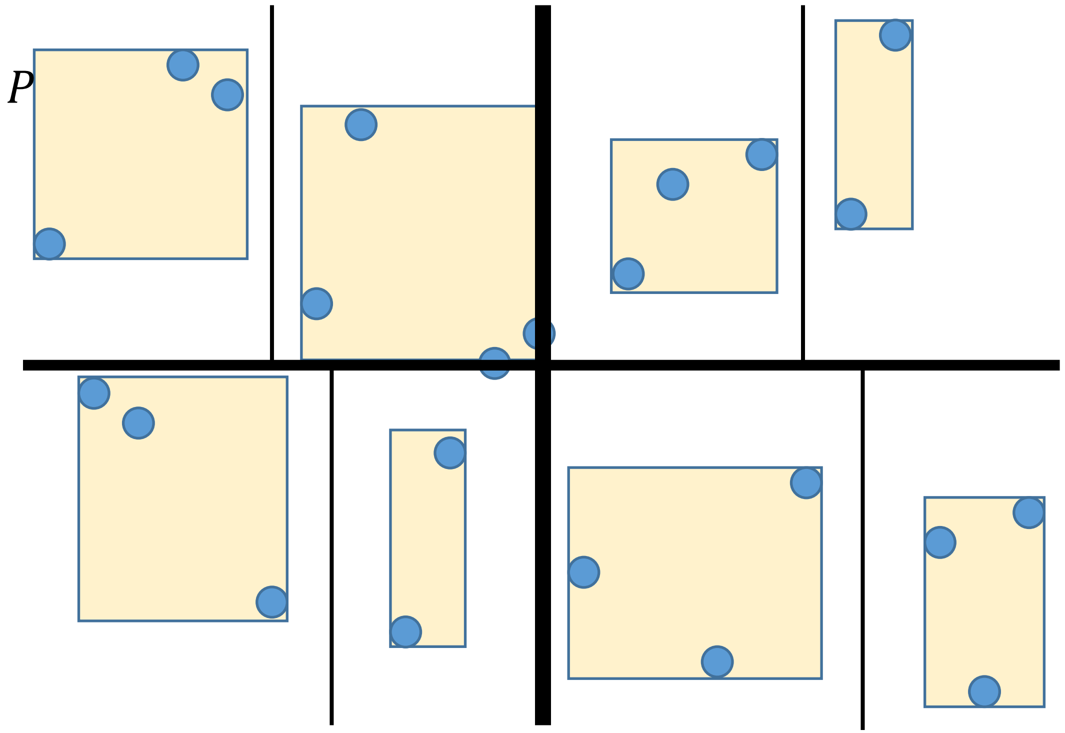
Bounding Volume Hierarchy

- Variation on previous approaches that stores the volume covered at each level of the tree.



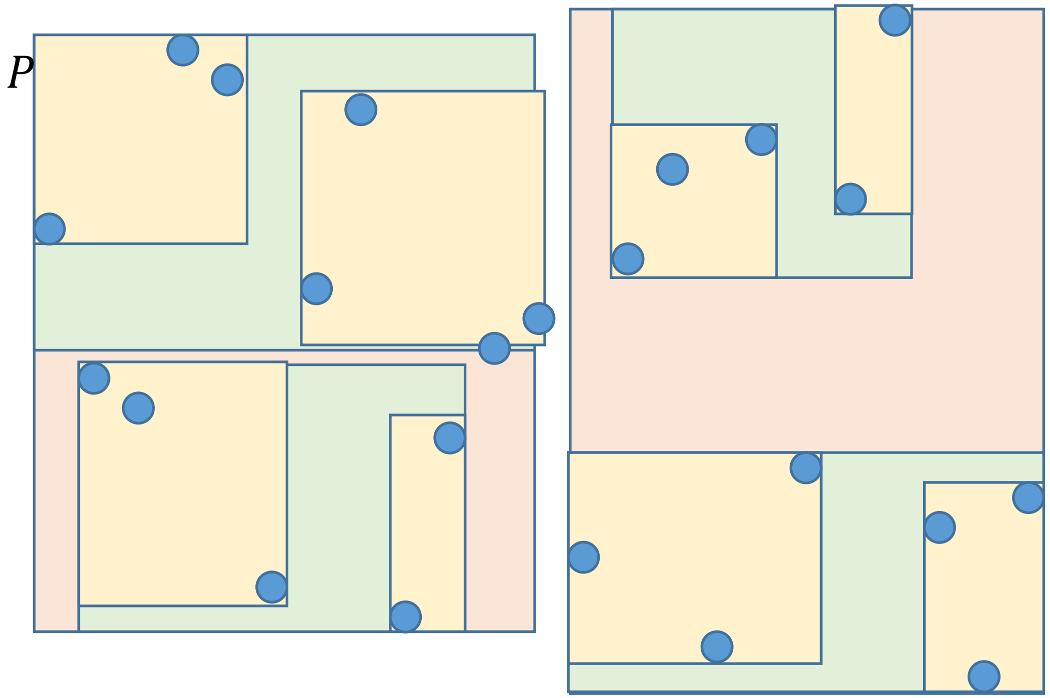
Bounding Volume Hierarchy

- Variation on previous approaches that stores the volume covered at each level of the tree.



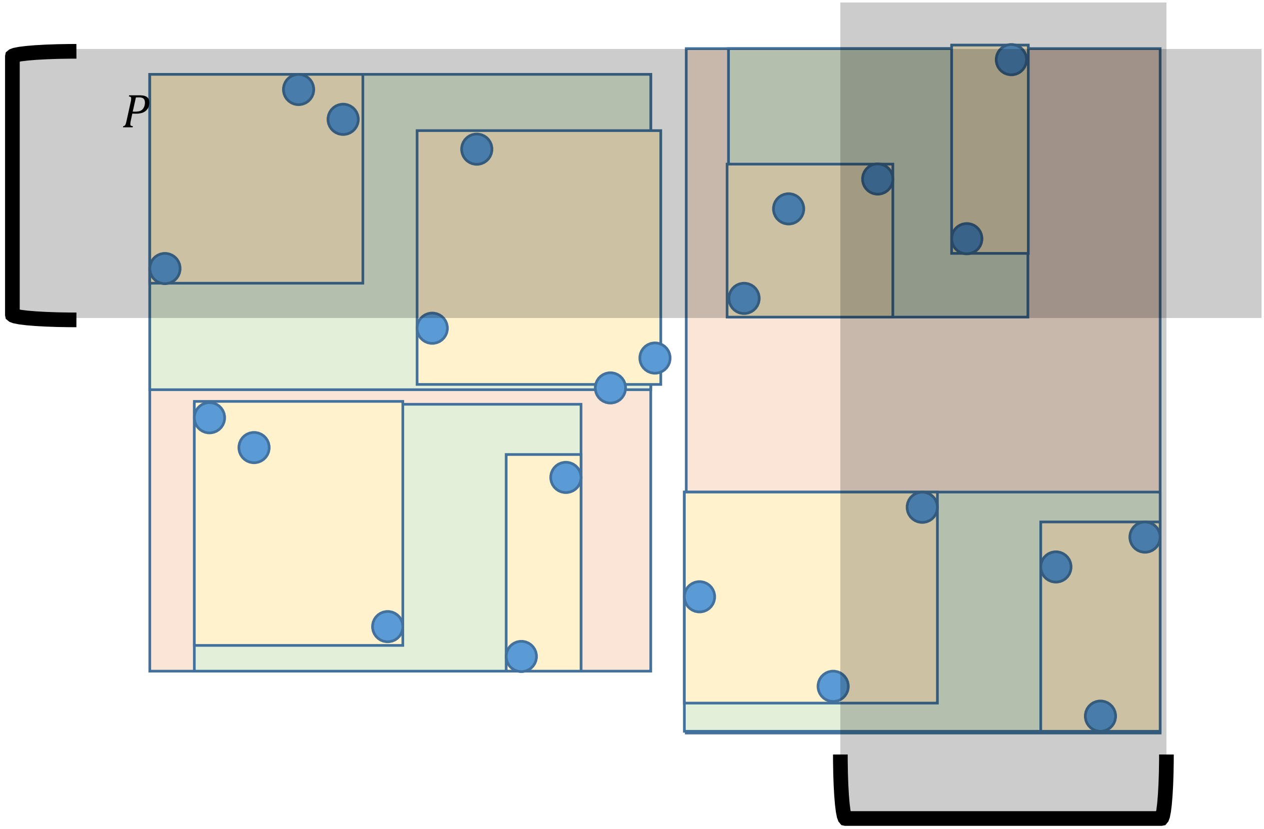
Bounding Volume Hierarchy

- Variation on previous approaches that stores the volume covered at each level of the tree.



Range Search Bounding Volume Hierarchy

- Variation on previous approaches that stores the volume covered at each level of the tree.



Bounding Volume Hierarchy

- Building Data Structure: $O(n \log n)$
- Search
  - Average case: $O(\log n + k)$
    - where k is the size of the leaf nodes
  - Worst Case: $O(n)$
- Space: in $O(n)$