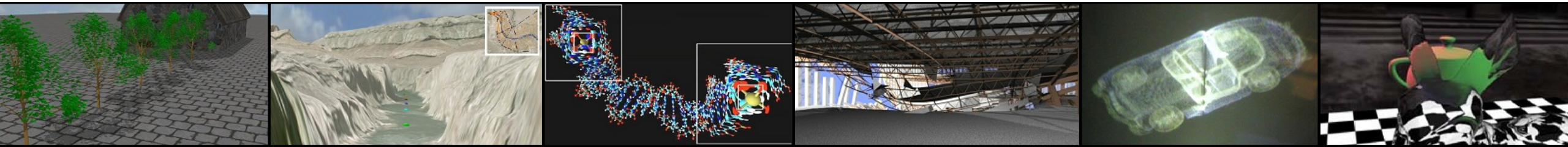


COT 4521: INTRODUCTION TO COMPUTATIONAL GEOMETRY



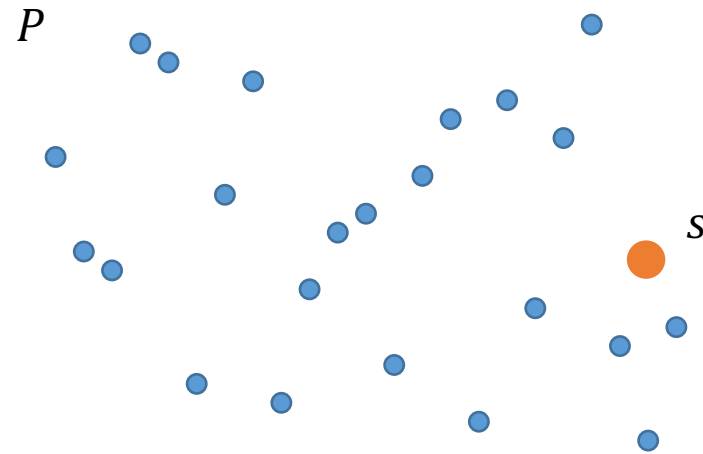
Searches

Paul Rosen
Assistant Professor
University of South Florida

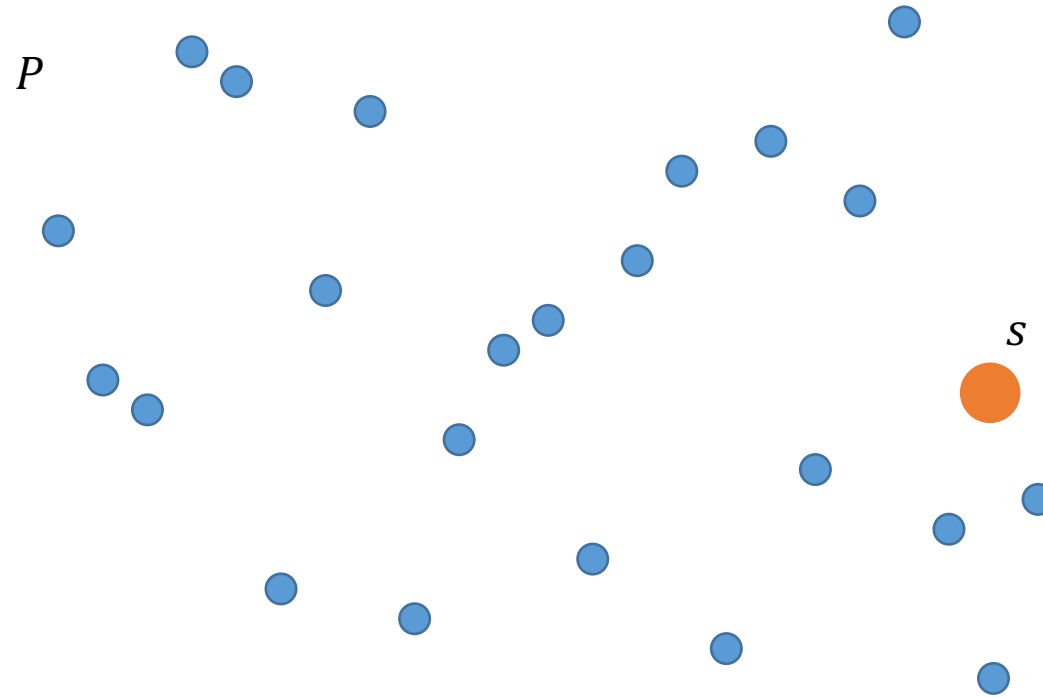


POINT SEARCHES

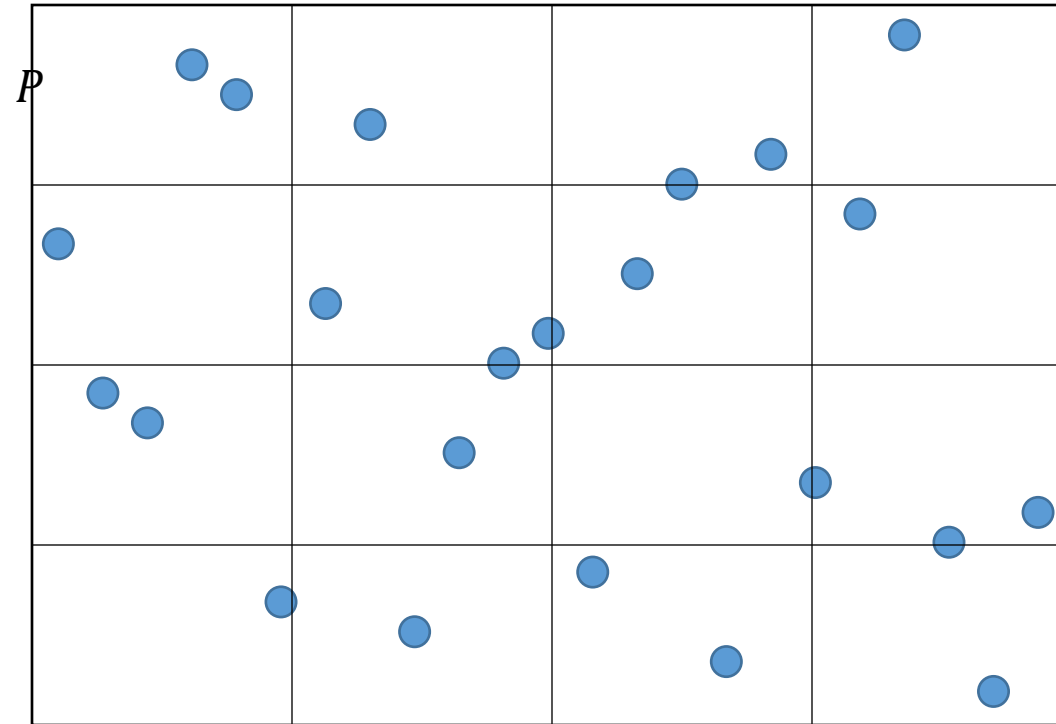
- PROBLEM DEFINITION: GIVEN A SET OF POINTS P IN R^d PROVIDE A DATA STRUCTURE THAT GIVEN AN INPUT SEARCH LOCATION s RETURNS THE k NEAREST POINTS
 - For simplicity we will primarily consider $d = 2$ and $k = 1$



POINT SEARCHES : IDEAS?



GRIDS



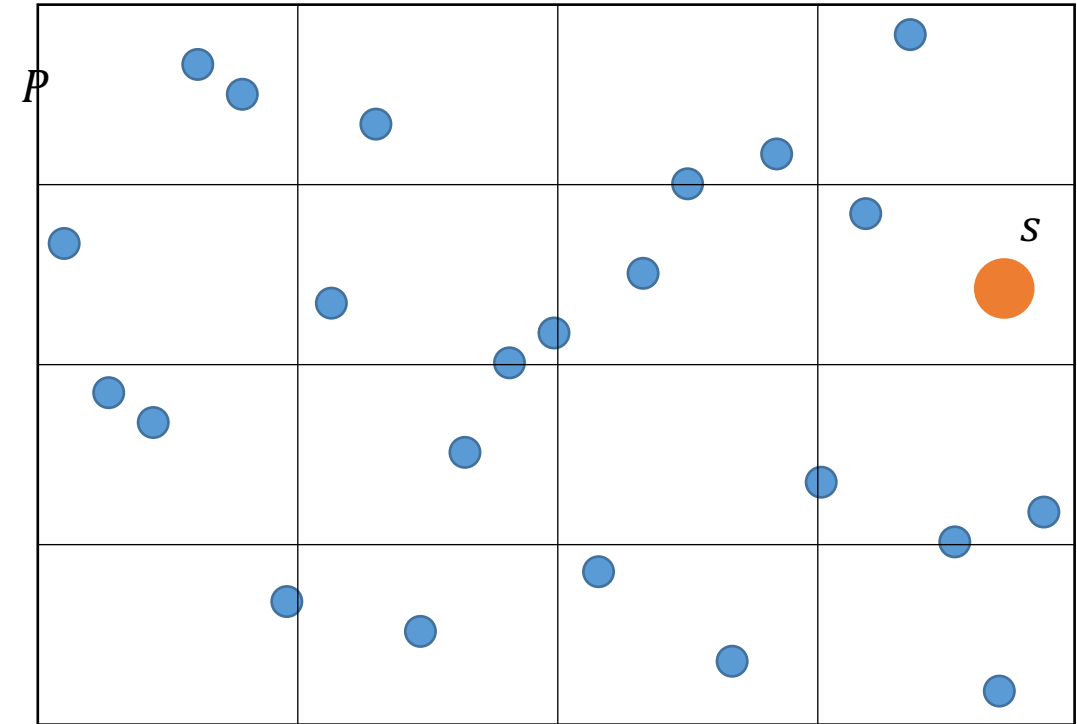
GRID ALGORITHMS

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



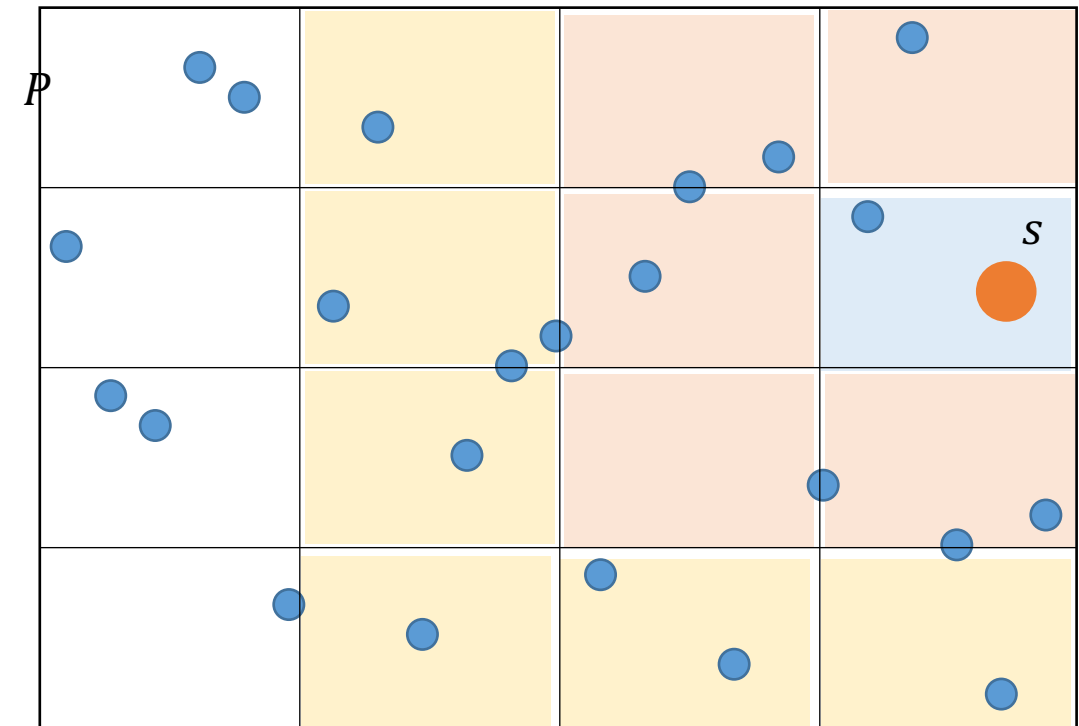
GRID ALGORITHMS

- 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 $O(r^d + n)$

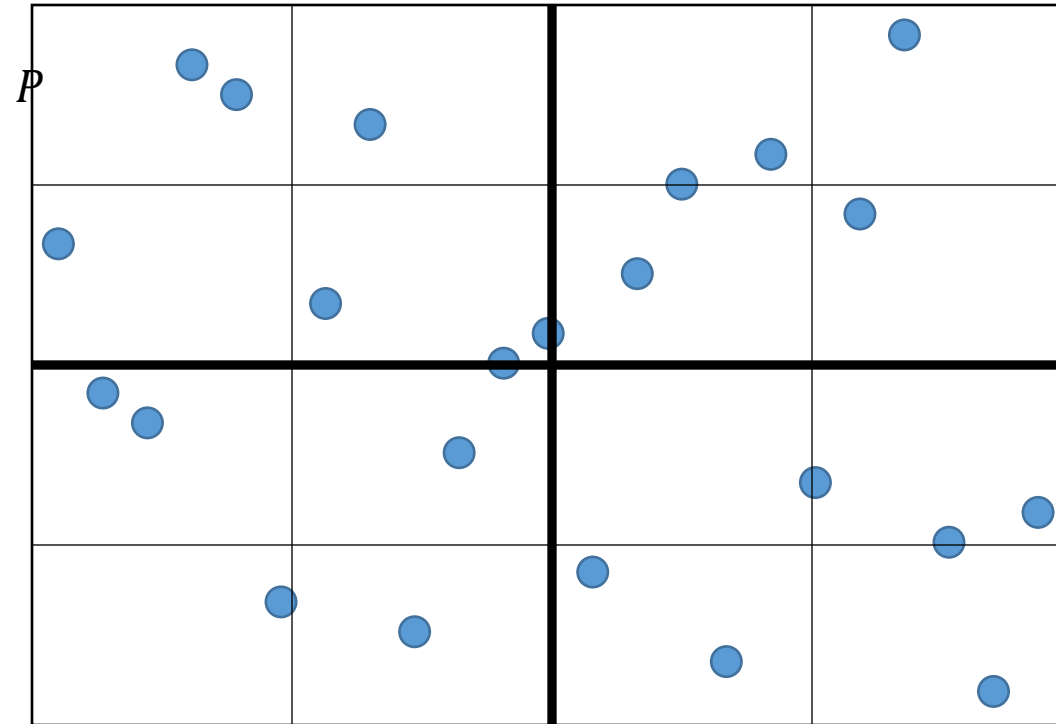


GRIDS

- **EXAMPLES OF FAILURE CASES?**

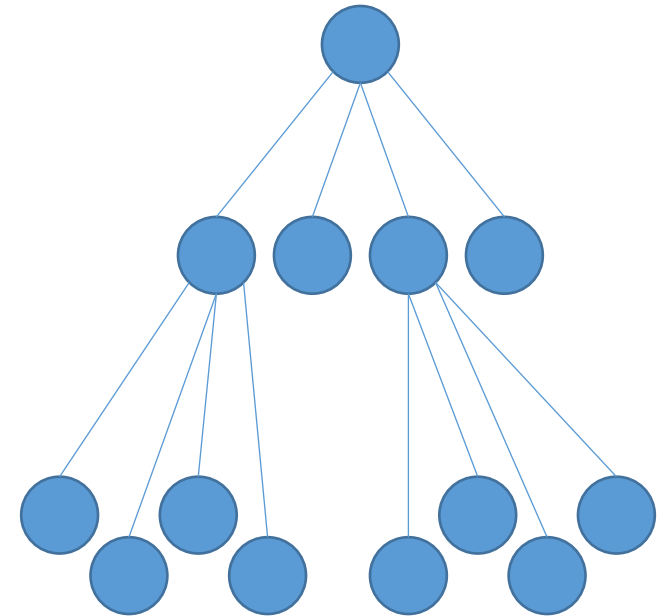


QUADTREES



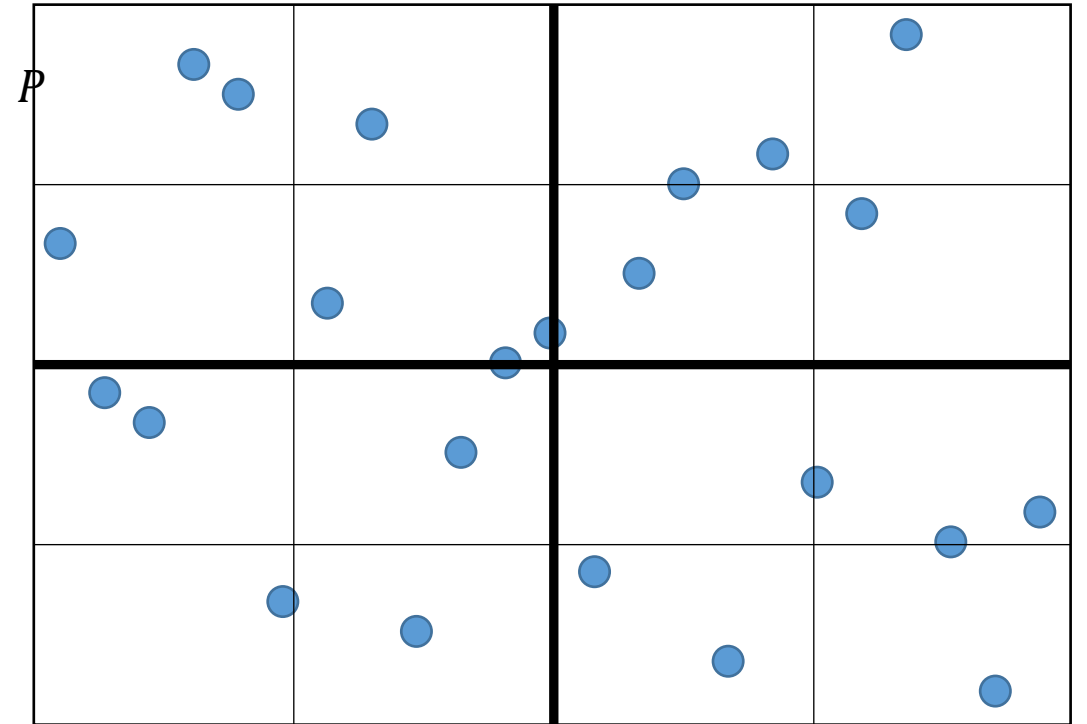
QUADTREE ALGORITHMS

- 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

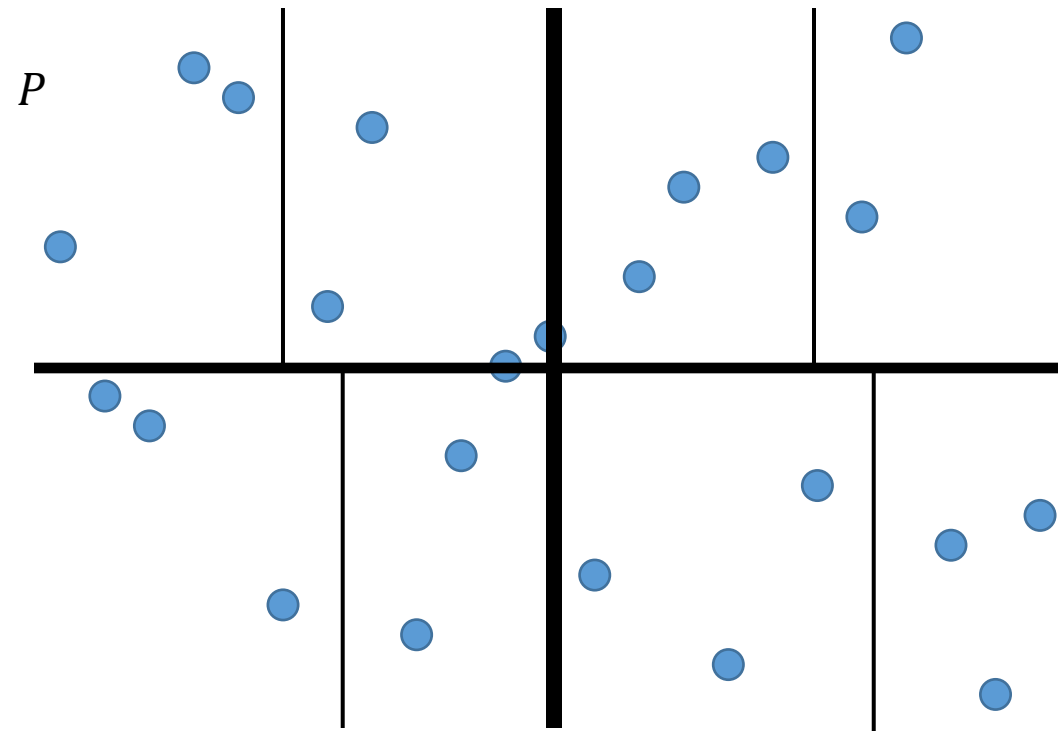


QUADTREES

- **EXAMPLES OF FAILURE CASES?**

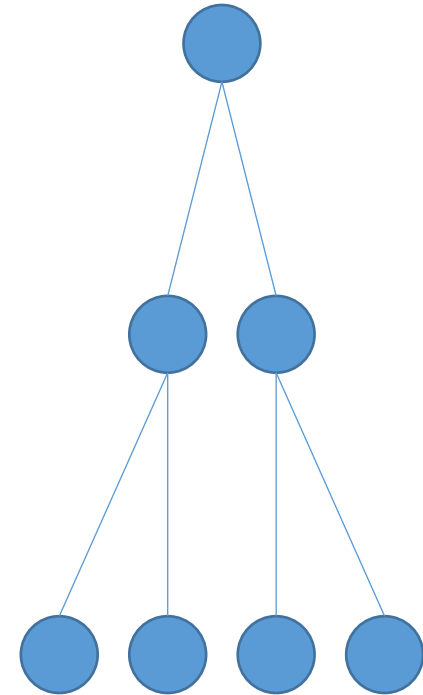


KD-TREE



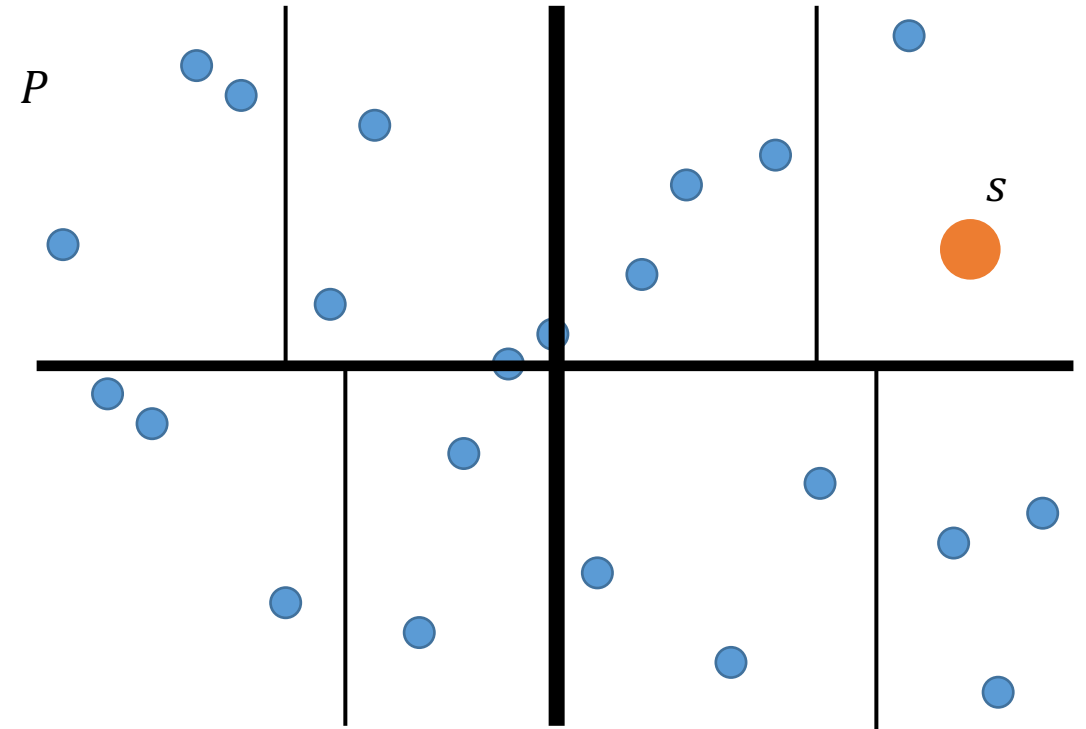
KD-TREE ALGORITHMS

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

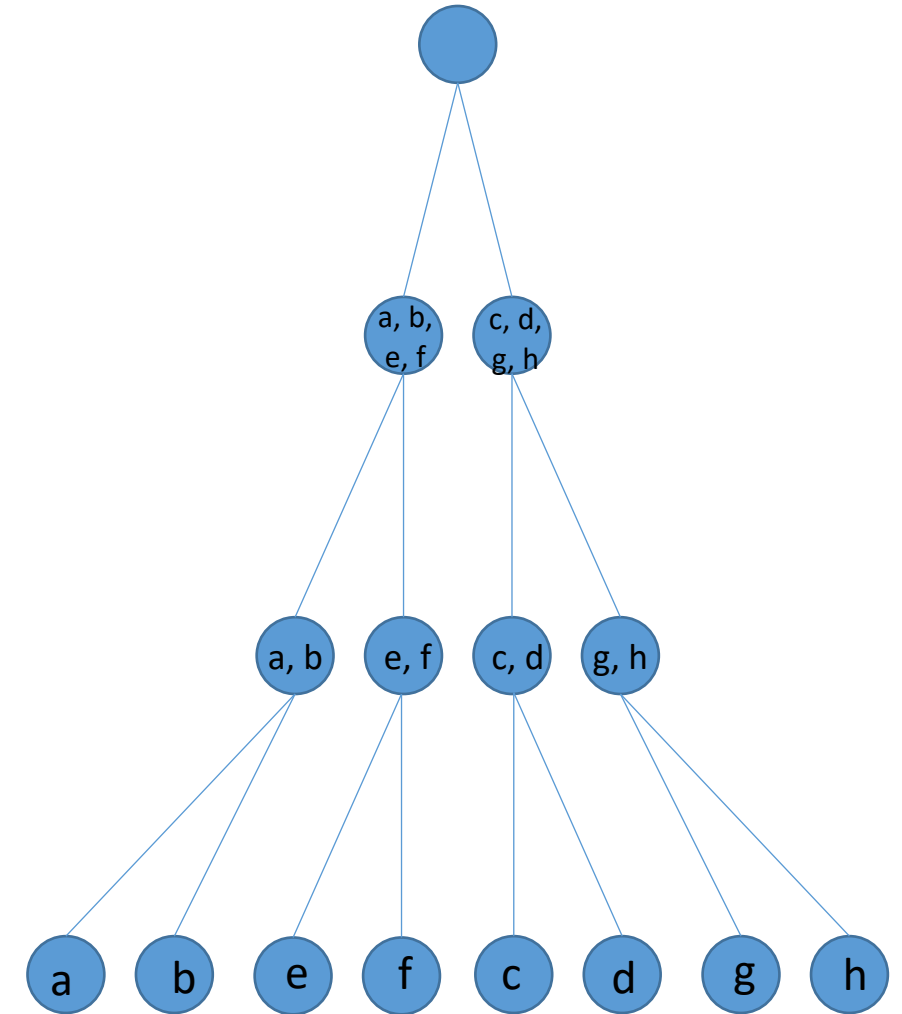
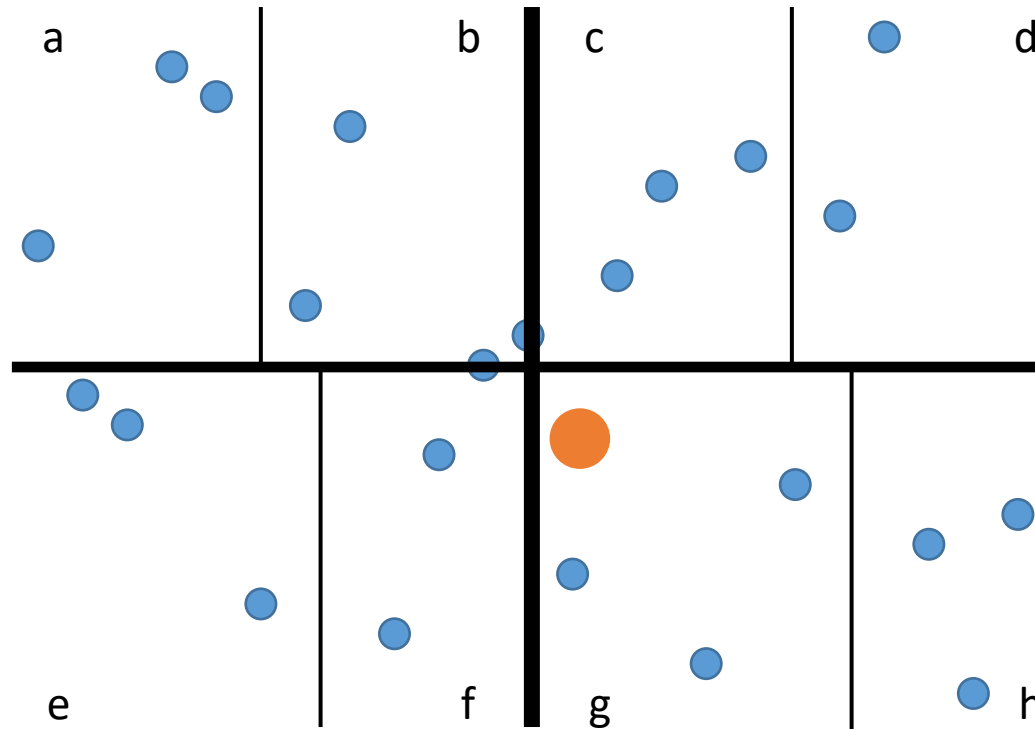


KD-TREE ALGORITHMS

- 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:
 - $O(n)$

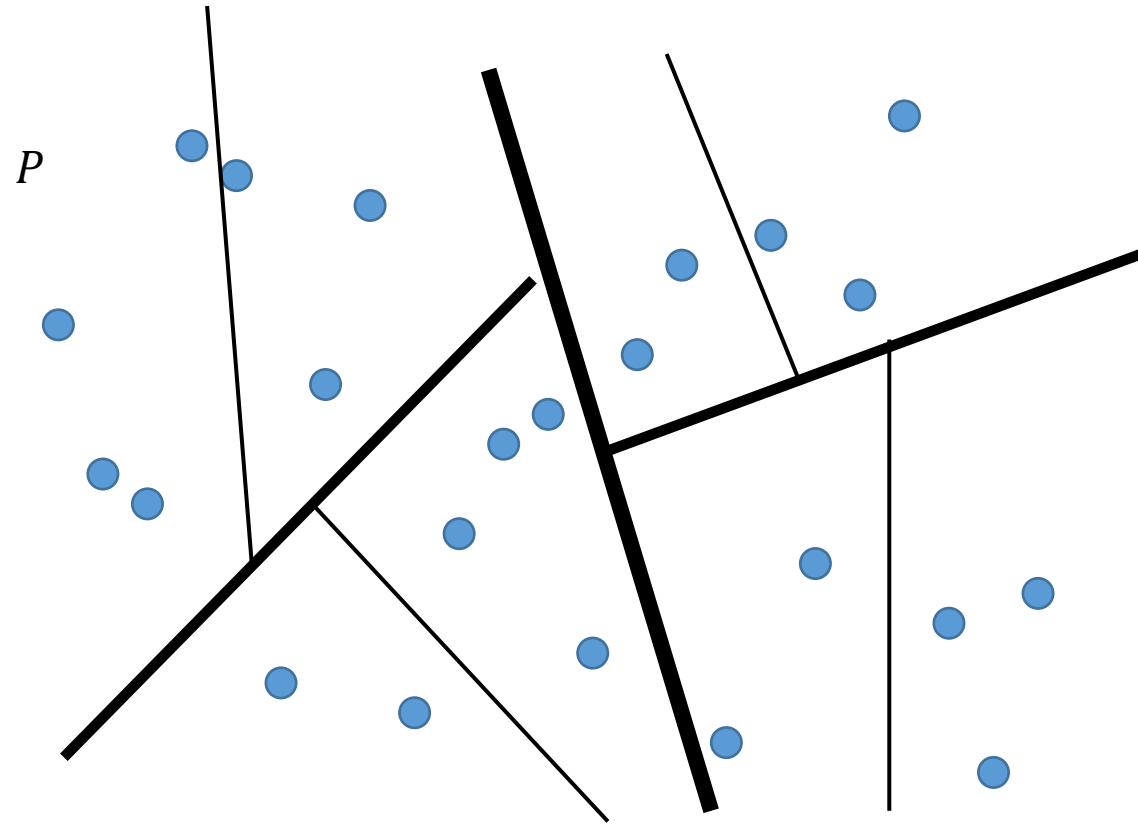


KD-TREE

- EXAMPLES OF FAILURE CASES?

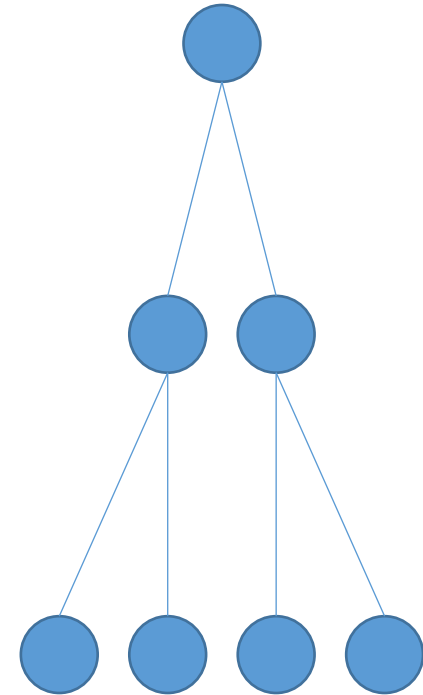


BINARY SPACE PARTITION (BSP)



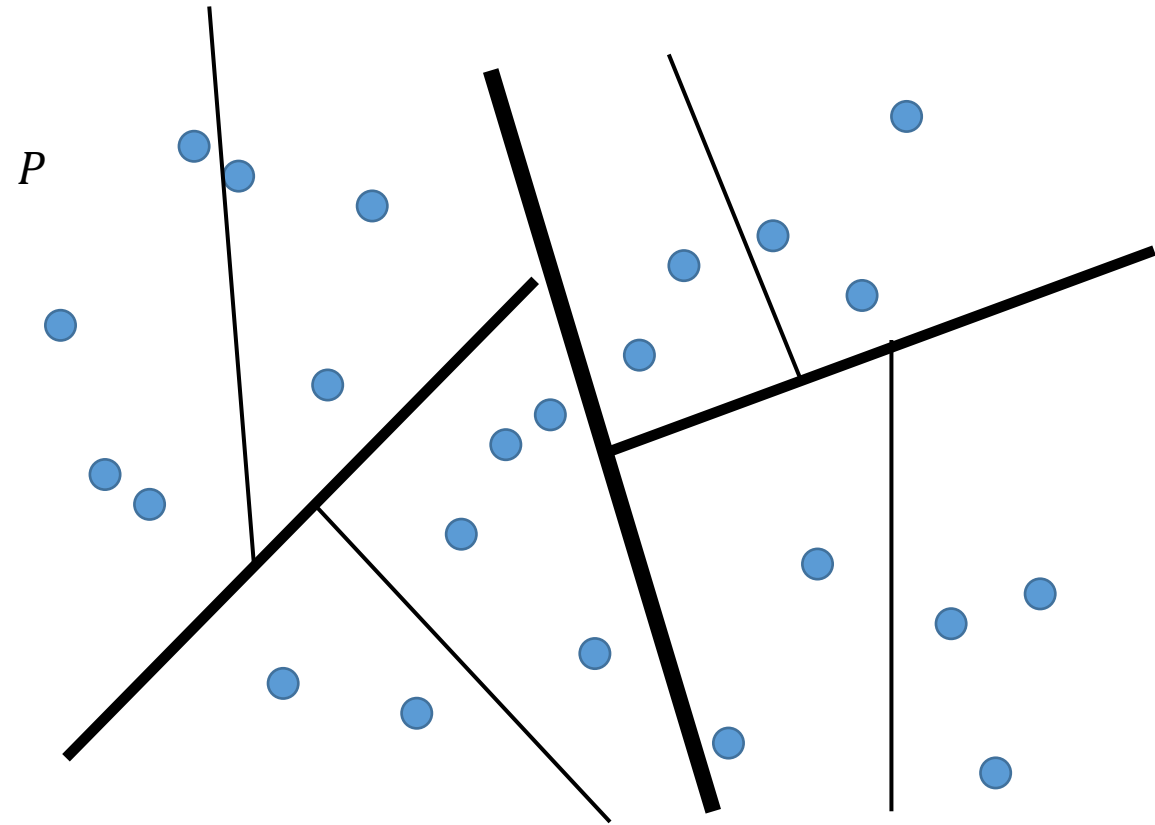
BSP ALGORITHMS

- 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?



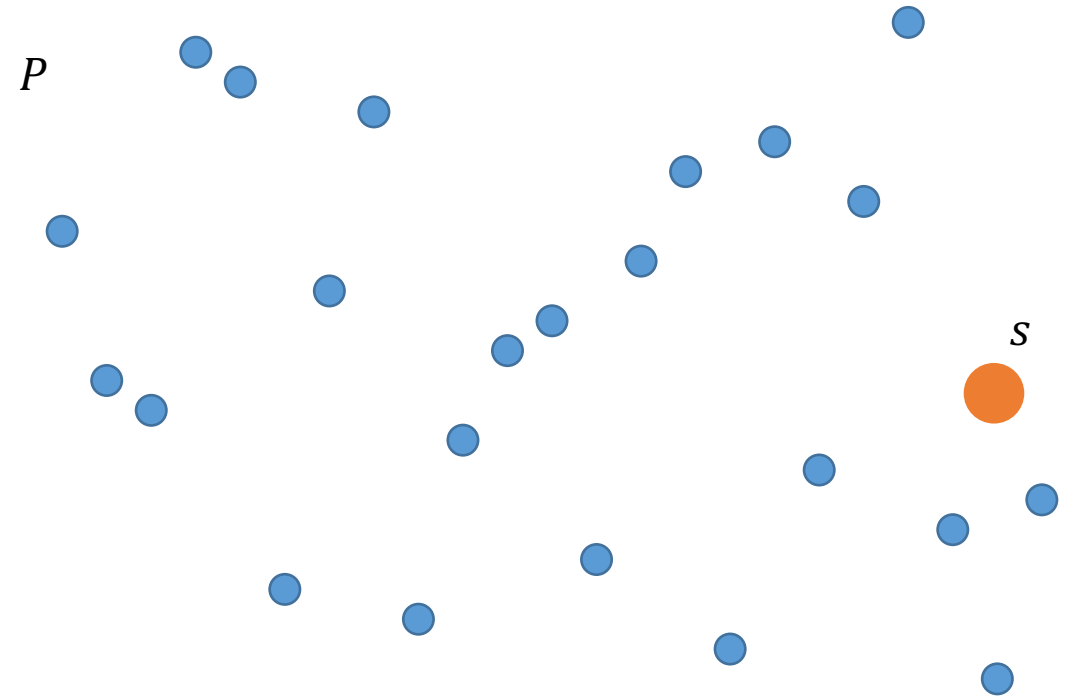
TASKS

- CLOSEST POINT SEARCH
 - Task covered thus far
- K-NEAREST NEIGHBORS SEARCH (NEXT)
- RANGE SEARCH
- CLUSTERING (WE'LL TALK ABOUT THIS NEXT LECTURE)



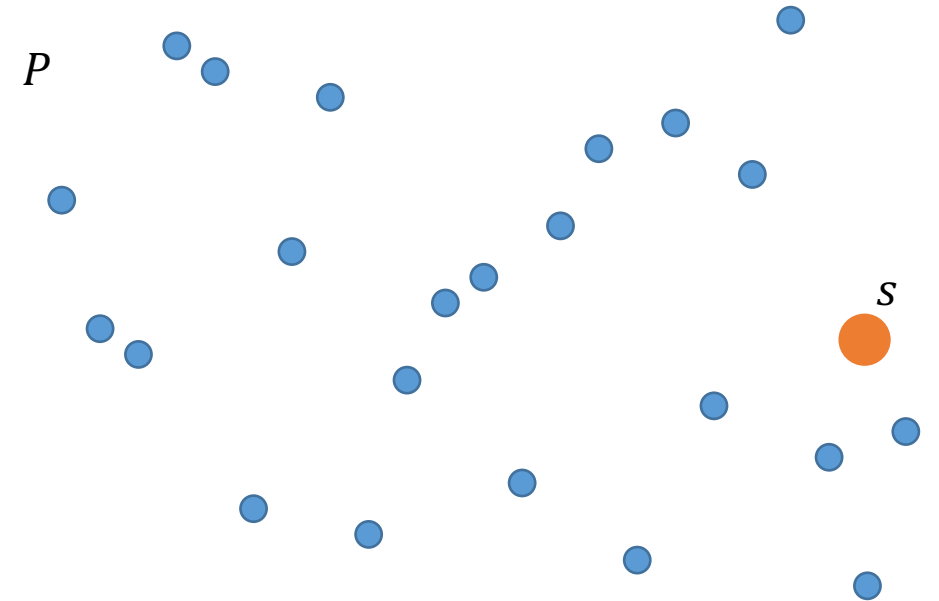
K-NEAREST NEIGHBOR SEARCH

- PROBLEM: GIVEN A SET OF POINTS P , FIND THE K-NEAREST NEIGHBORS EFFICIENTLY
- IDEAS?



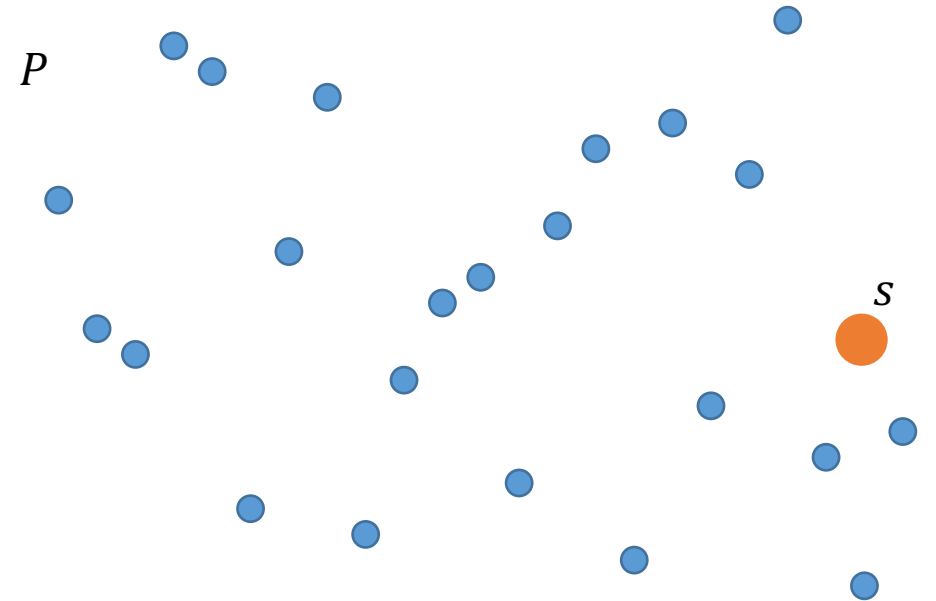
K-NEAREST NEIGHBOR SEARCH

- 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?



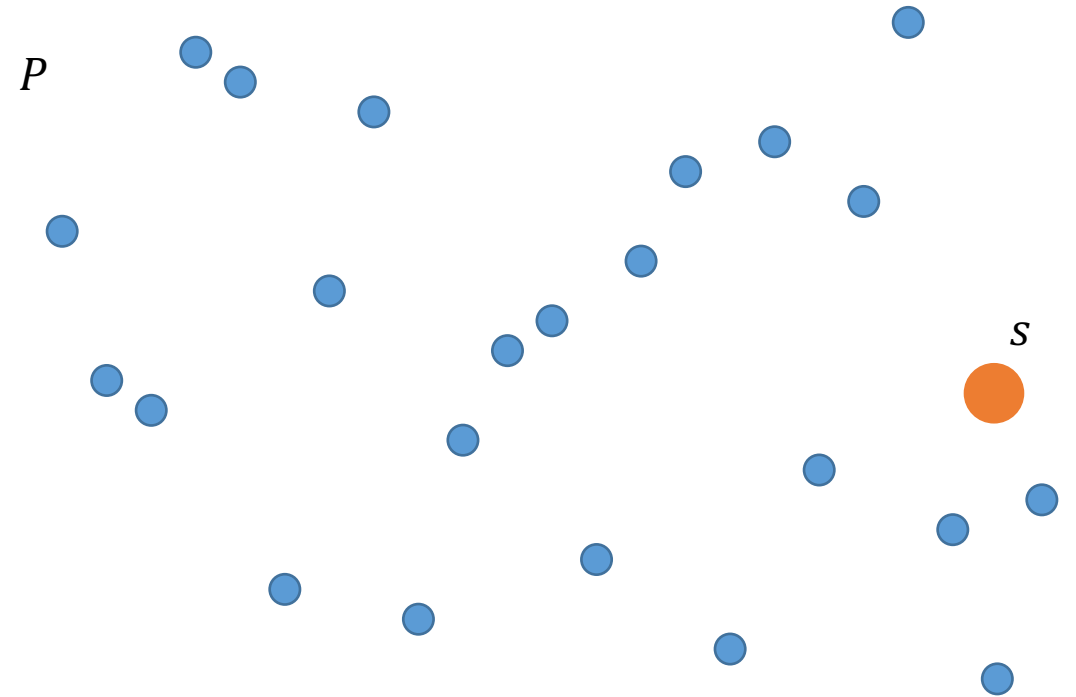
K-NEAREST NEIGHBOR SEARCH

- 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



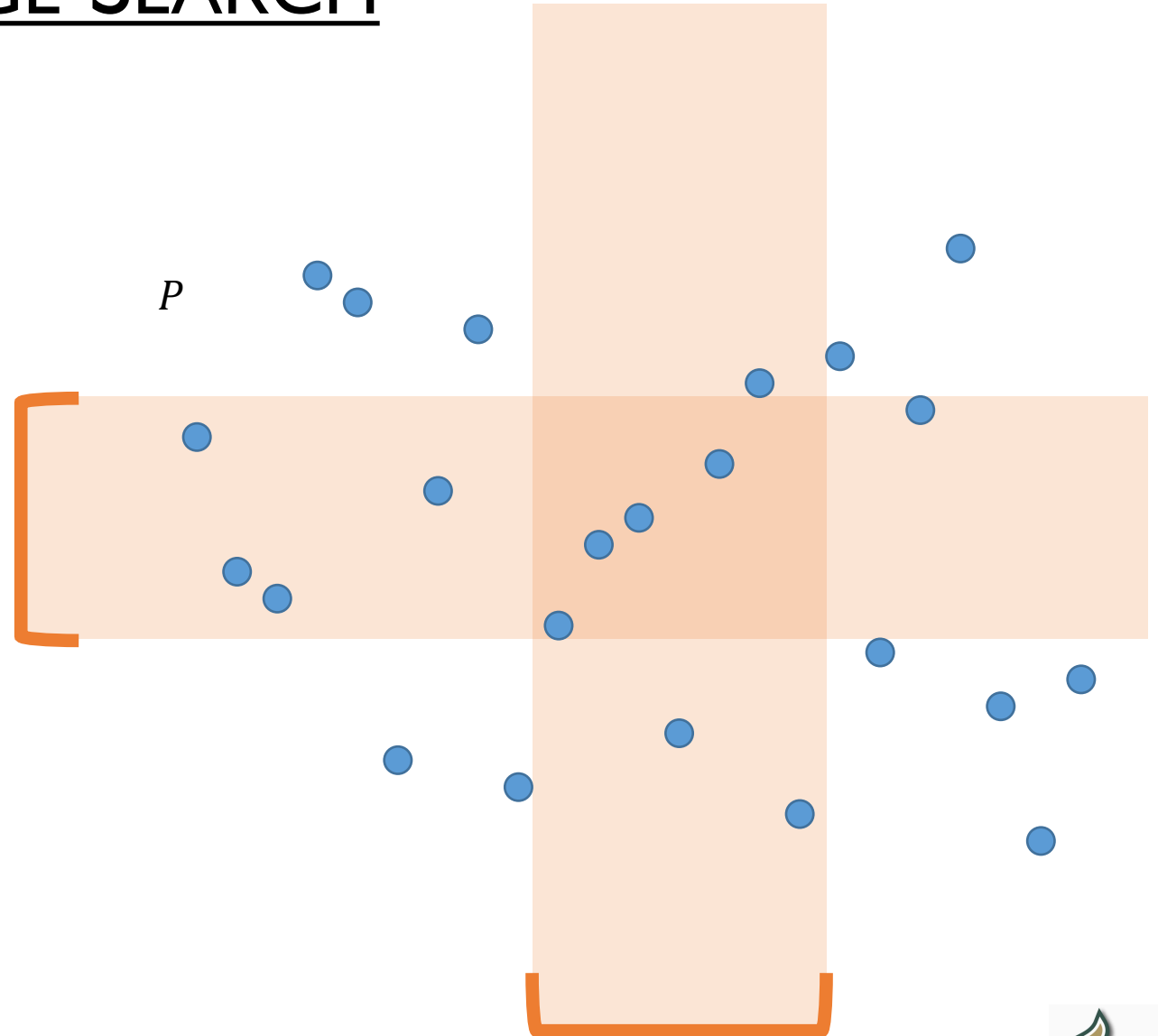
K-NEAREST NEIGHBOR SEARCH

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



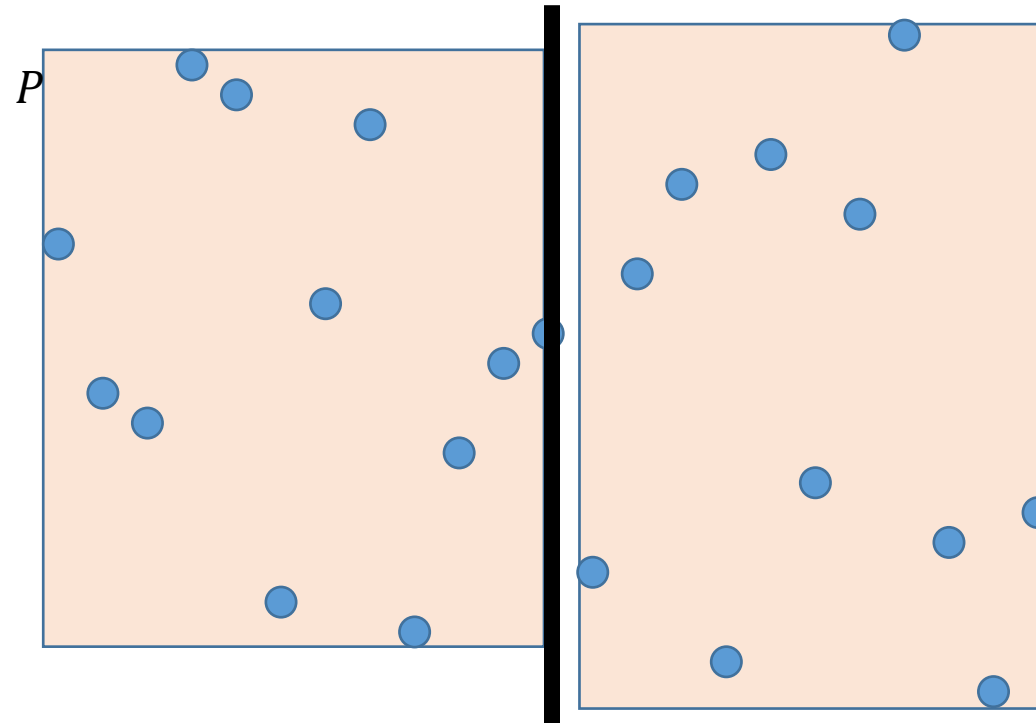
RANGE SEARCH

- PROBLEM: GIVEN A SET OF POINTS P , EFFICIENTLY FIND THE SET OF POINTS WITHIN A SPECIFIED RANGE
- IDEAS?



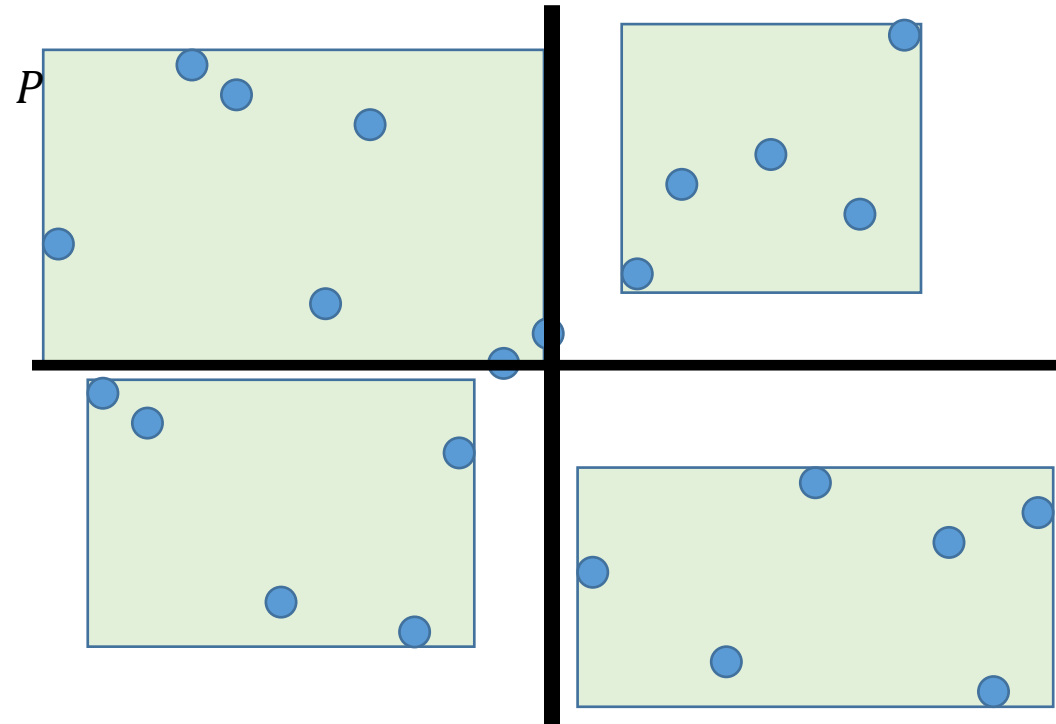
BOUNDING VOLUME HIERARCHY

- VARIATION ON PREVIOUS APPROACHES THAT STORES THE VOLUME COVERED AT EACH LEVEL OF THE TREE.



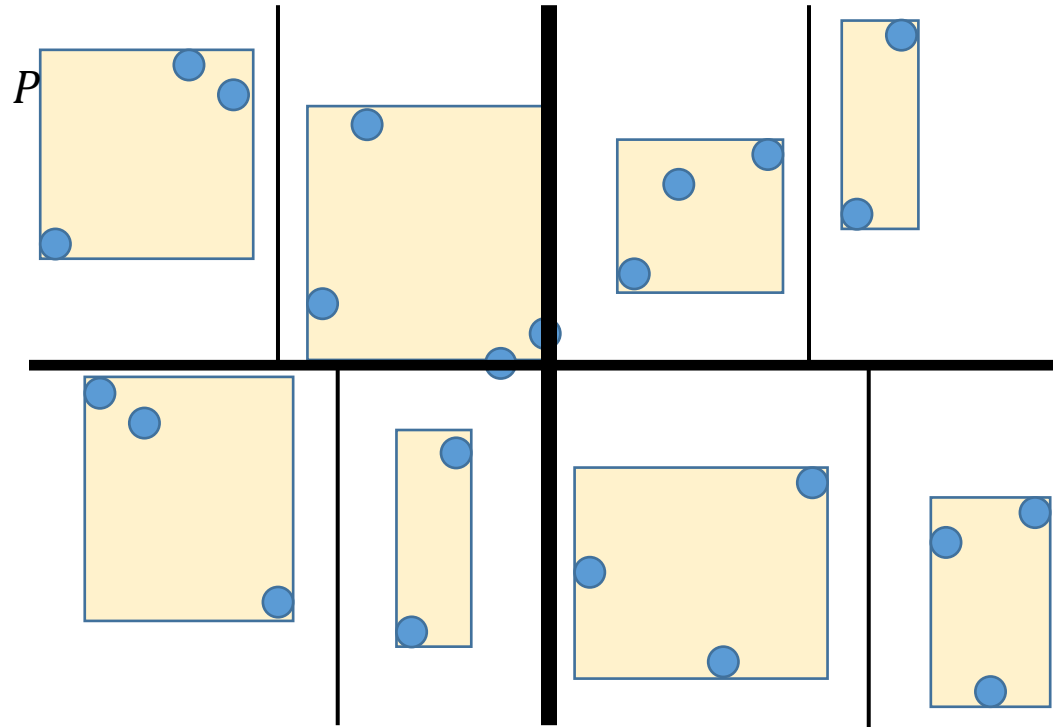
BOUNDING VOLUME HIERARCHY

- VARIATION ON PREVIOUS APPROACHES THAT STORES THE VOLUME COVERED AT EACH LEVEL OF THE TREE.



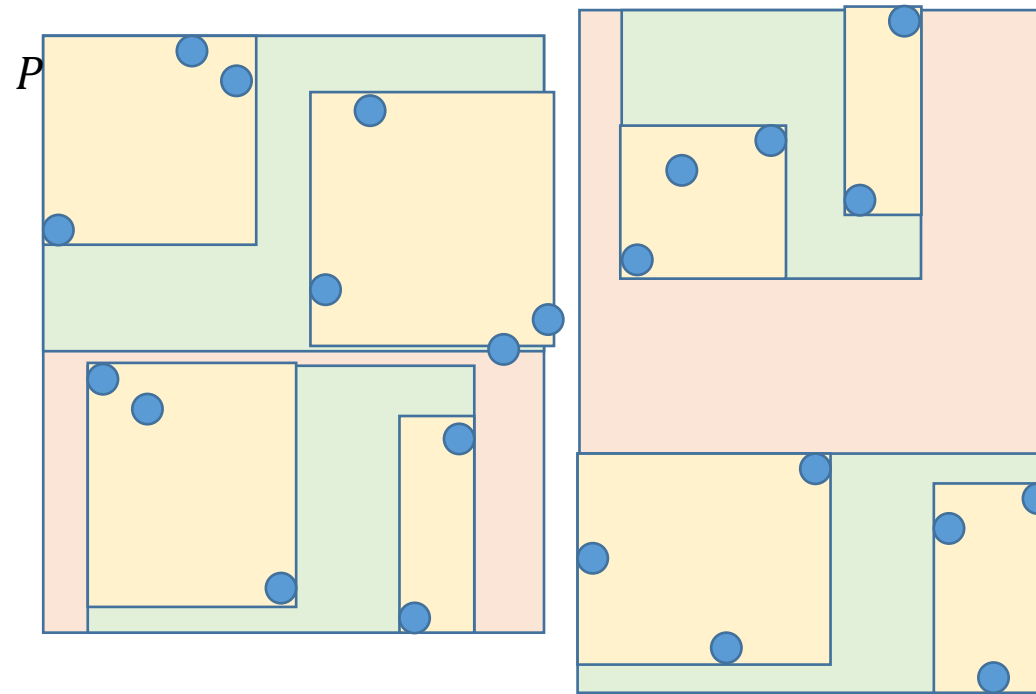
BOUNDING VOLUME HIERARCHY

- VARIATION ON PREVIOUS APPROACHES THAT STORES THE VOLUME COVERED AT EACH LEVEL OF THE TREE.



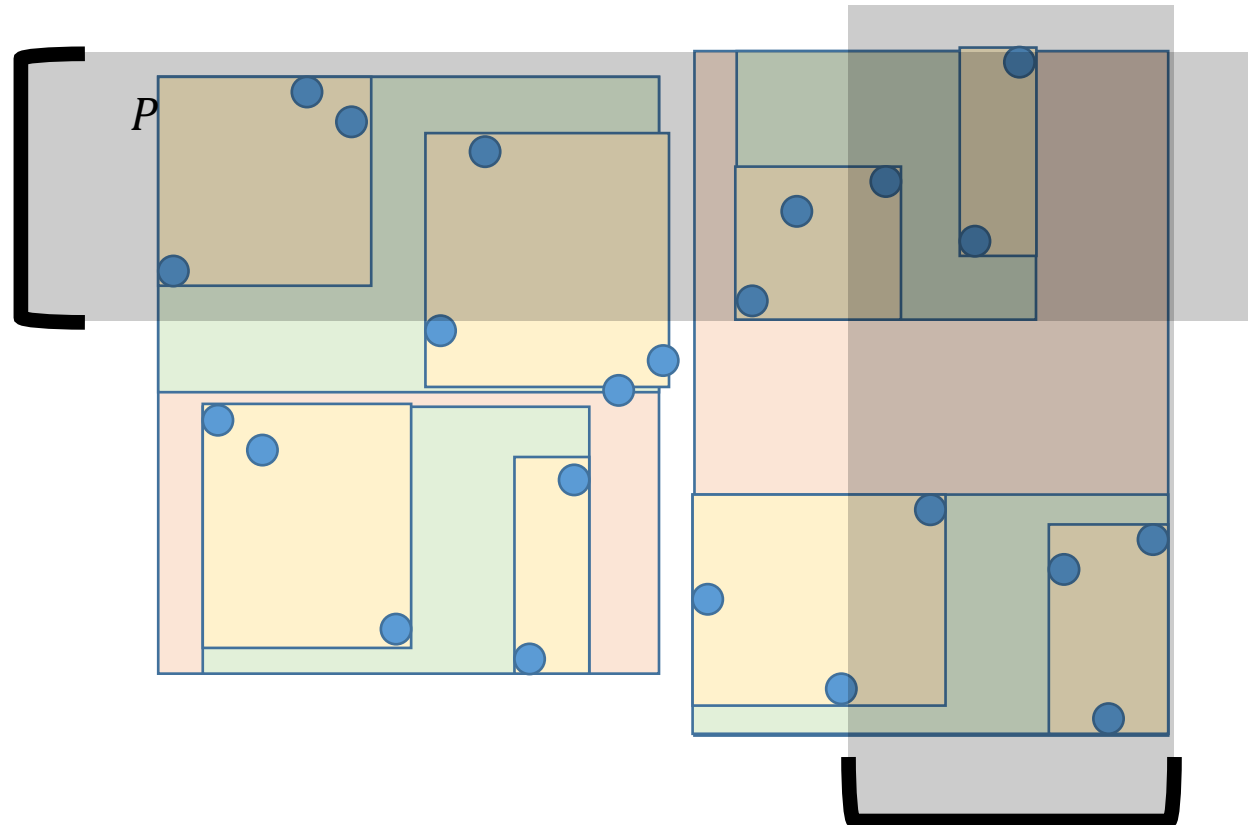
BOUNDING VOLUME HIERARCHY

- 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 $O(n)$



