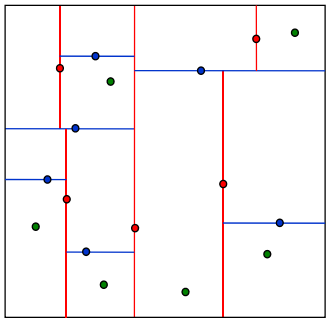
CS 190C
Science Education in Computational Thinking



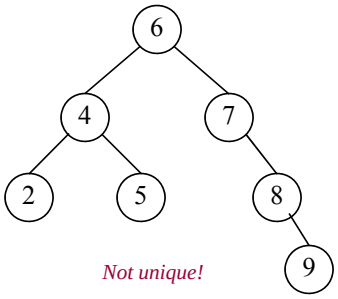
KD – TREES:

- Search trees, 2D case
- kd-Trees, node structure
- Building, Searching
- Range query
- Nearest point query

Hoffmann, 2008

Search Trees

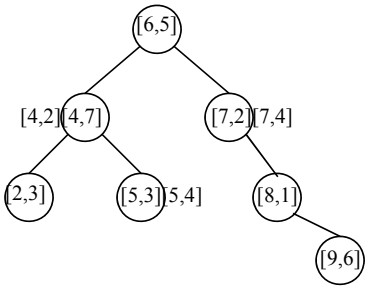
- Fast way to search for keys from a list:
  - Root key value splits the set
  - Smaller keys in left, larger keys in right subtree
  - For instance: [2, 4, 5, 6, 7, 8, 9]



Not unique!

What About 2D?

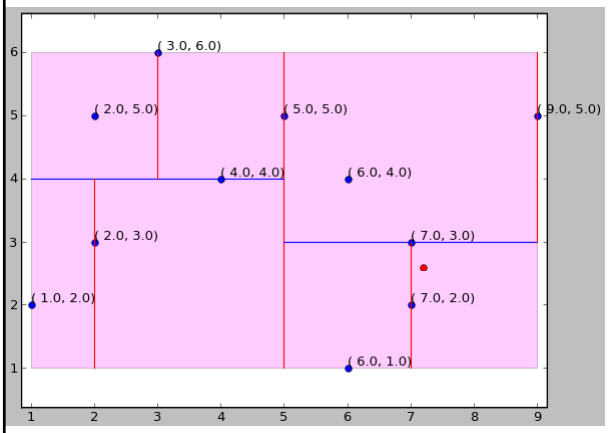
- For instance
  - [[2,3], [5,4], [9,6], [4,7], [8,1], [7,2], [4,2], [5,3], [6,5], [7,4]]
- Could use *x*-value as key, but does not address multiple *y*-values with the same *x*…

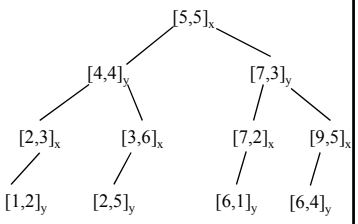


3

Solution

- Split by both *x* and *y* values, but alternating:





4

Creation from Point List

- (Alternate the axis by which to split)

1. Pick a split point, label the root with it
2. Build left subtree from the points with the smaller coordinate values
3. Build the right subtree from the points with the larger coordinate (≥)

Structure and Building the Tree

```
class KDTreeNode:
    def __init__(self, points, depth):
        self.axis = depth % 2
        if self.axis == 0:
            points.sort(key=getx)
        else:
            points.sort(key=gety)
        med = len(points)//2
        self.split = points[med]
        if med > 0:
            self.left = KDTreeNode(points[:med], depth+1)
        else:
            self.left = None
        if med+1 < len(points):
            self.right = KDTreeNode(points[med+1:], depth+1)
        else:
            self.right = None
```

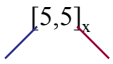
KDTreeNode:
split (key field)
axis (of split)
left (subtree)
right (subtree)

Example (1/6)

- ((1,2), (2,3), (4,4), (5,5), (7,2), (7,3), (9,5))

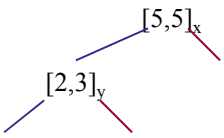
Example (2/6)

- ((1,2), (2,3), (4,4), (5,5), (7,2), (7,3), (9,5))



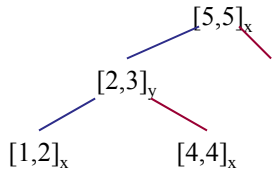
Example (3,6)

- ((1,2), (2,3), (4,4), …



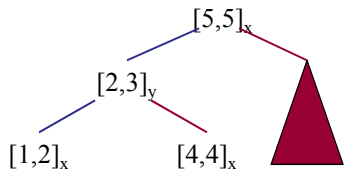
Example (4/6)

- ((1,2), … (4,4), …



Example (5/6)

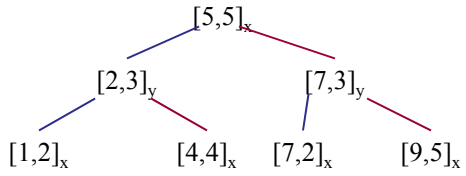
- (…, …(7,2), (7,3), (9,5))



11

Example (6/6)

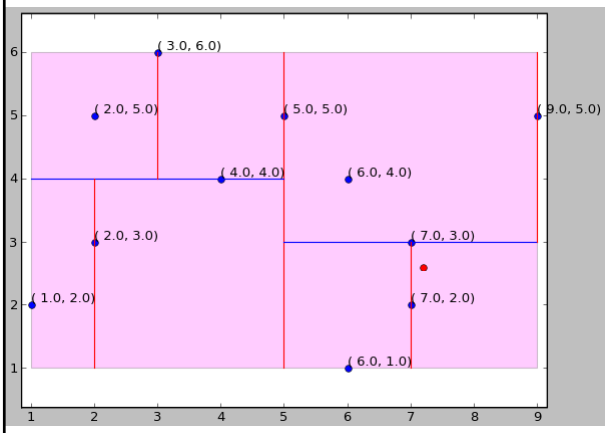
- ((1,2), (2,3), (4,4), (5,5), (7,2), (7,3), (9,5))

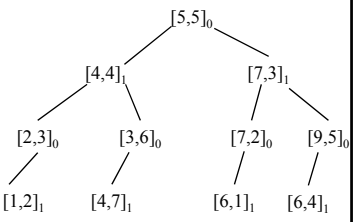


12

Search Examples

Find [4,7] and [6,8] – but what about [5,8]???





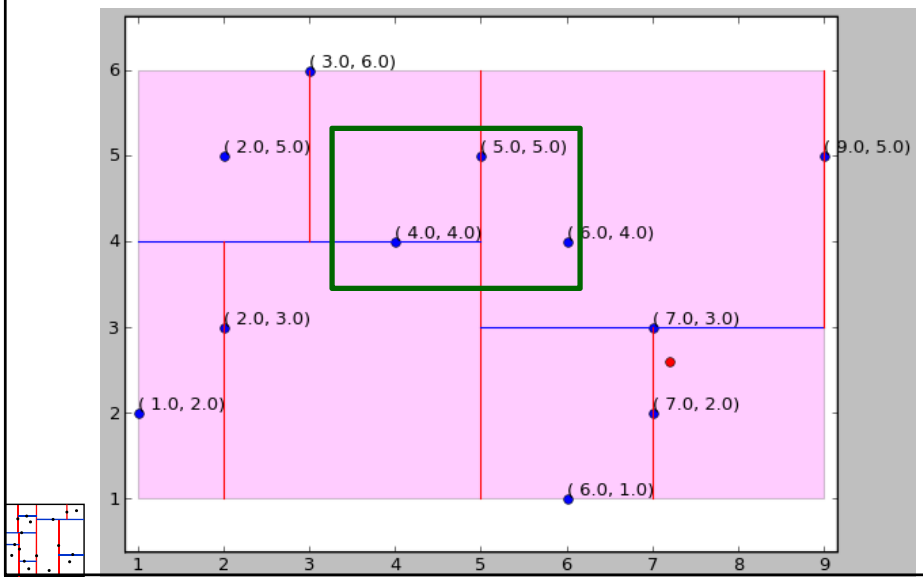
13

Use #2: Range Query

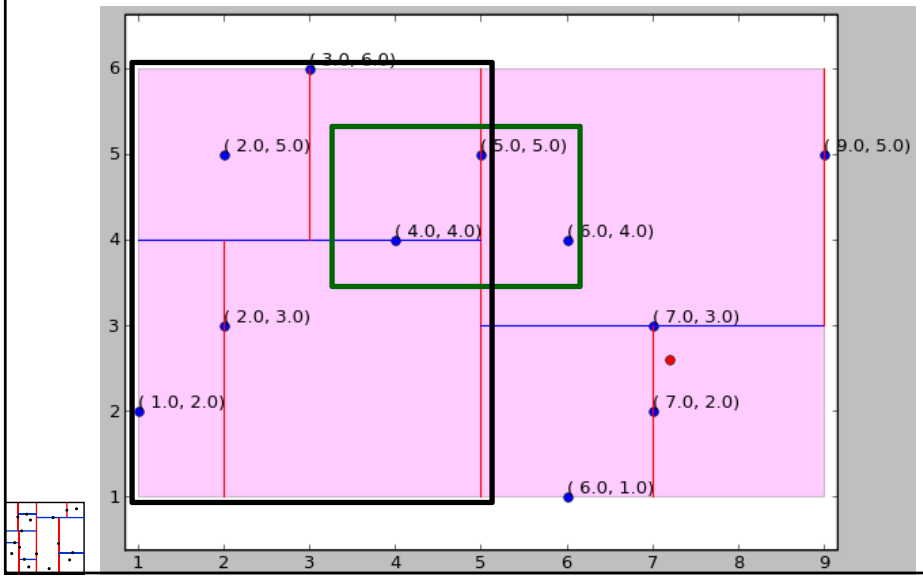
- Report all points in given (axis-aligned) box:
- If tree.split in box, report it
- If tree.left (subtree) overlaps, search left subtree
- If tree.right (subtree) overlaps, search right subtree

14

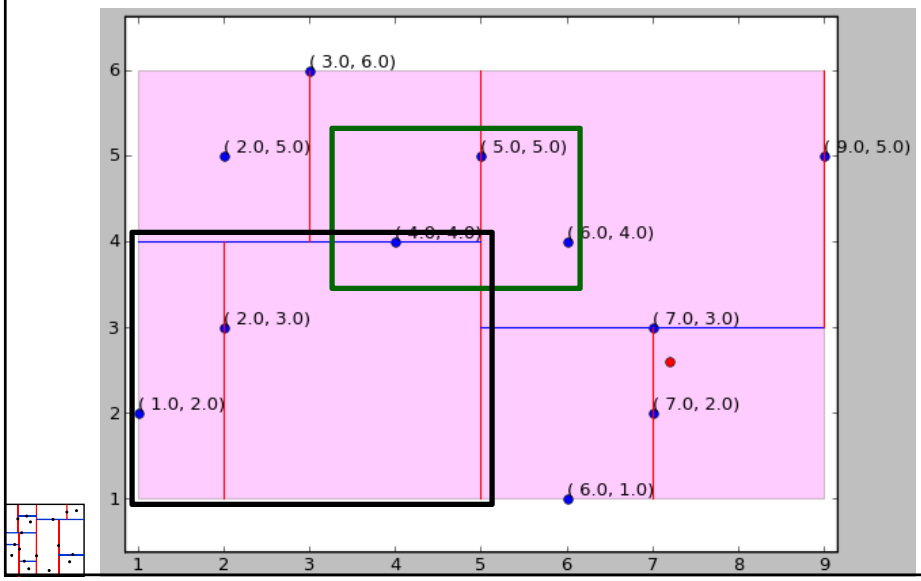
Example (1/9)



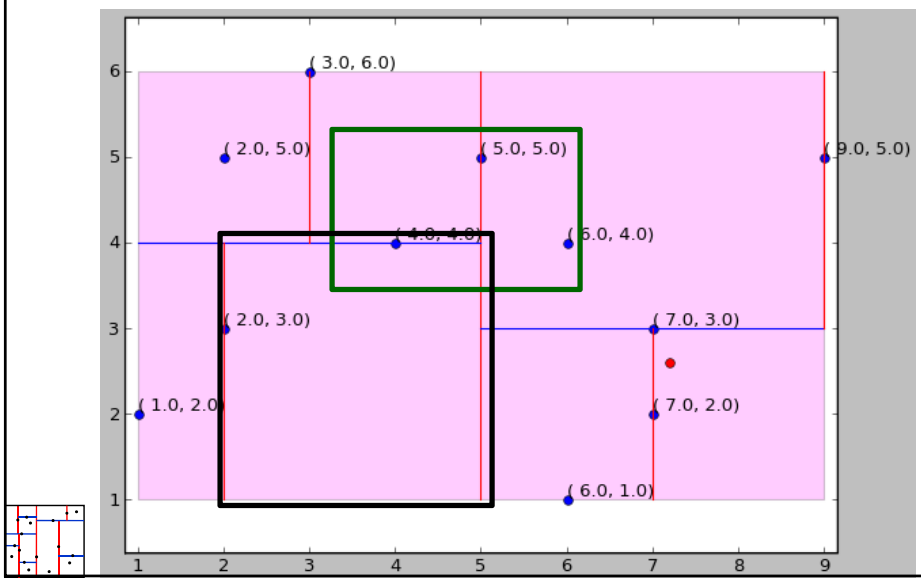
Example (2/9)



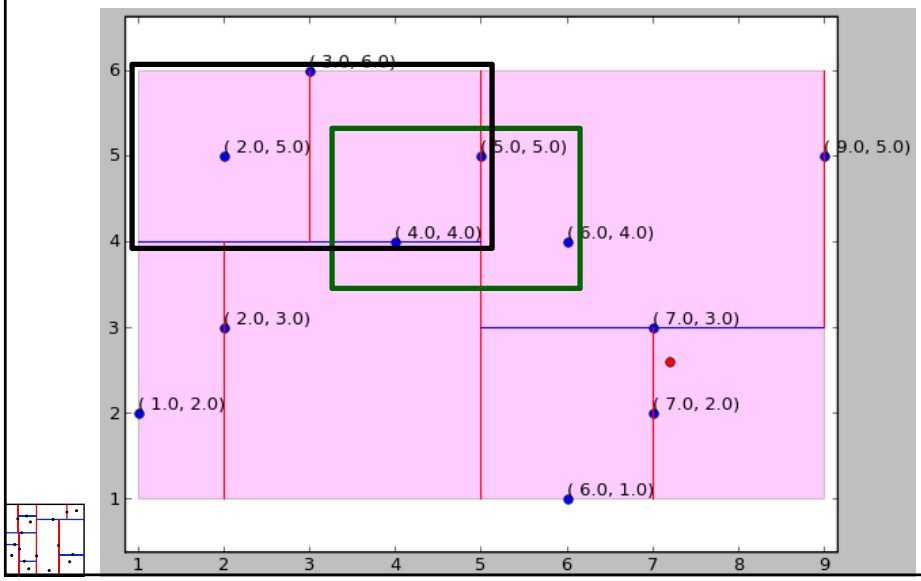
Example (3/9)



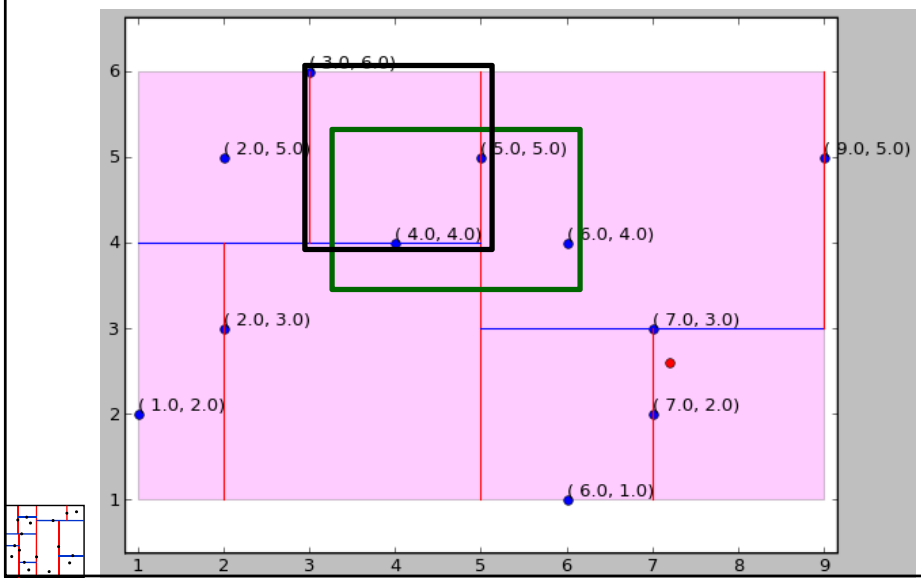
Example (4/9)



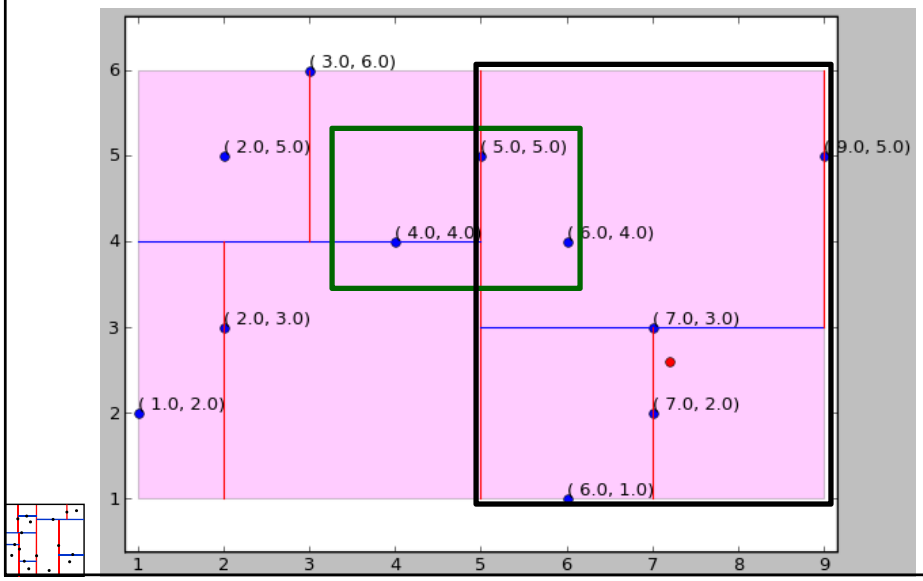
Example (5/9)



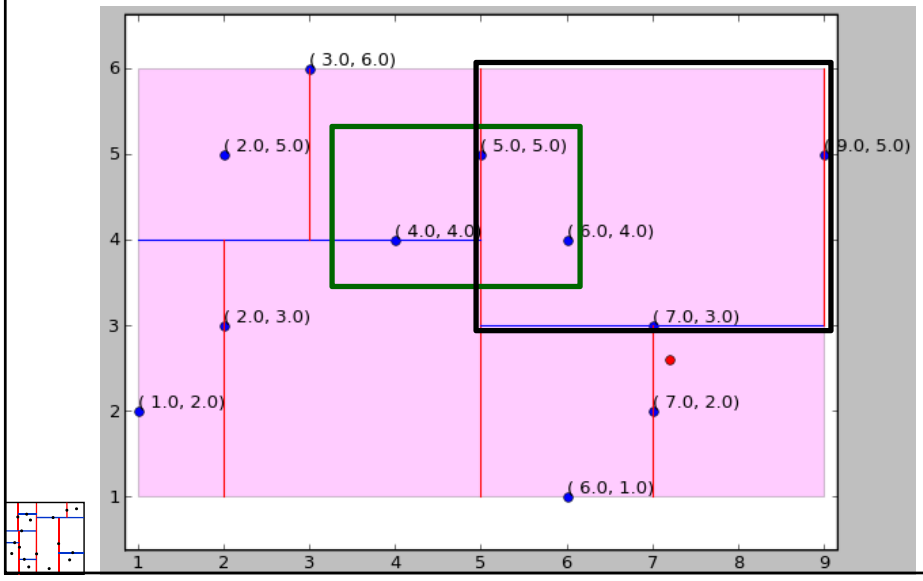
Example (6/9)



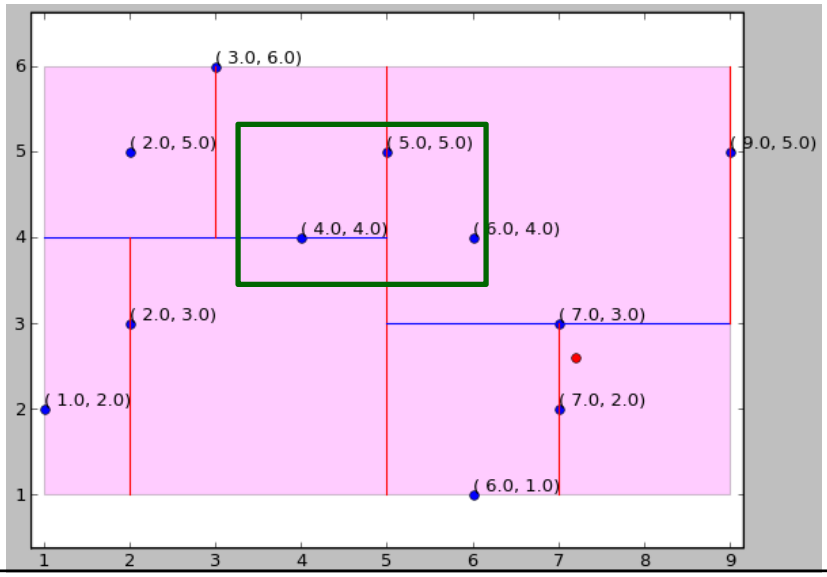
Example (7/9)



Example (8/9)



Example (9/9)



Code: Range Query

```
1.  def  range_query(tree, ll, ur):
2.      if tree.split is in query box [ll,ur]:
3.          add tree.split to solution
4.      if query box overlaps left subtree:
5.          add to solution points from left subtree
6.      if query box overlaps right subtree:
7.          … (analogous)
8.      return solution
```

24

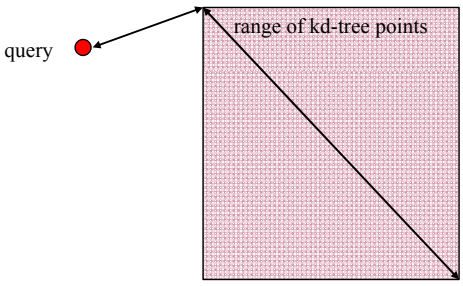
Use #3: Find Nearest Point

- Set closest distance *d* to ∞
- If *d* > distance(query, split) then reduce distance and keep root point as candidate
- Compare query[axis] with split[axis], see whether the query would be located to the left or the right.
- Explore that subtree.
- Upon return, determine whether min distance overlaps the other subtree:
  If so, explore the other subtree as well

25

Example

- Setting initial distance to ∞:
  - Use distance to corner + diagonal
- Program example…



26

Code: Nearest Neighbor

```
1.  nneighbor(tree, qpoint, dist):
2.      d = |tree.split − qpoint|**2
3.      answer = min(dist, (d, tree.split))
4.      d2split = (tree.split − qpoint)[tree.axis]**2
5.      if point[tree.axis] < tree.split[tree.axis]:
6.          first,  second = tree.left,  tree.right
7.      else:
8.          first,  second = tree.right, tree.left
9.      if first:
10.         answer = min(answer, nneighbor(first, qpoint, answer))
11.     if d2split < answer[0] and second:
12.         answer = min(answer, nneighbor(second, qpoint, answer))
13.     return answer
```

The Other Subtree…

Tree((-3,0), (3,3), (4,1)) Query point (2,1)