2.

def insert(self, p: List[Point], depth=0):

if not p:

return None

dimension = depth % 2

p.sort(key=lambda x: x[dimension])

median = len(p) // 2

left\_subtree = self.insert(p[:median], depth+1)

right\_subtree = self.insert(p[median+1:], depth+1)

node = Node(location=p[median], left=left\_subtree, right=right\_subtree)

return node

def range(self, rectangle: Rectangle, node=None) -> List[Point]:

if node is None:

node = self.\_root

if node is None:

return []

result = []

if rectangle.is\_contains(node.location):

result.append(node.location)

if rectangle.lower.x <= node.location.x:

result += self.range(rectangle, node.left)

if rectangle.upper.x >= node.location.x:

result += self.range(rectangle, node.right)

return result

5.

def nearestneighbor(self, point: Point, node=None):

if node is None:

node = self.\_root

if node is None:

return None

nearest = node.location

while node:

dimension = depth % 2

if point[dimension] < node.location[dimension]:

node = node.left

else:

node = node.right

if node and self.distance(point, node.location) < self.distance(point, best):

nearest = node.location

return nearest