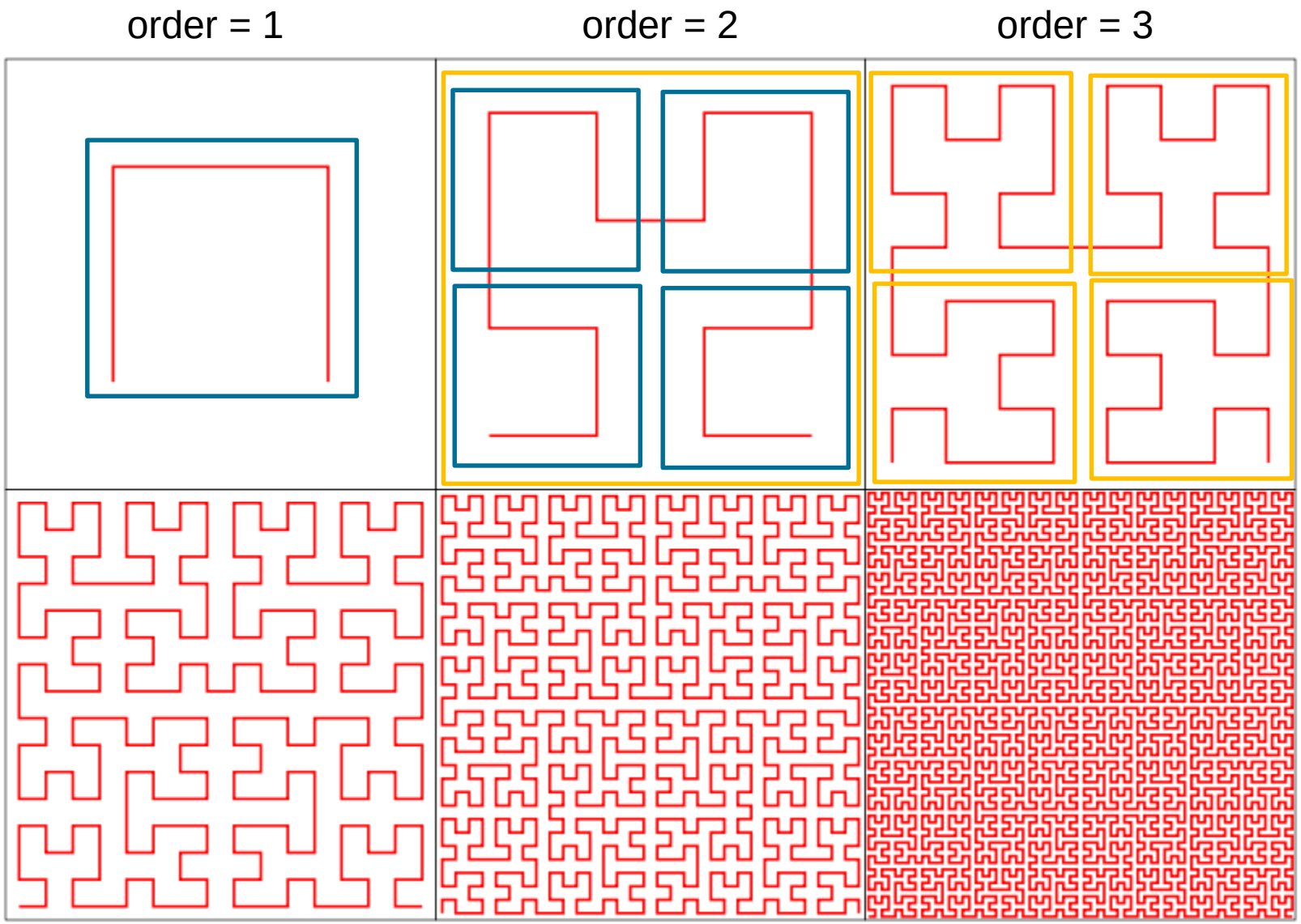
# 实习3

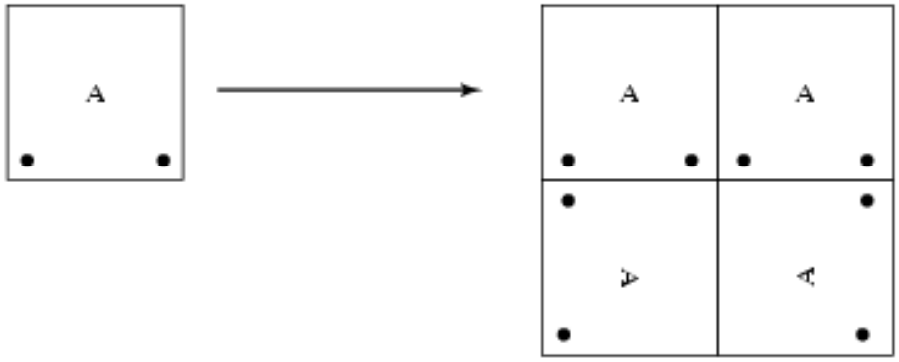
空间索引编程

# Hilbert Curve

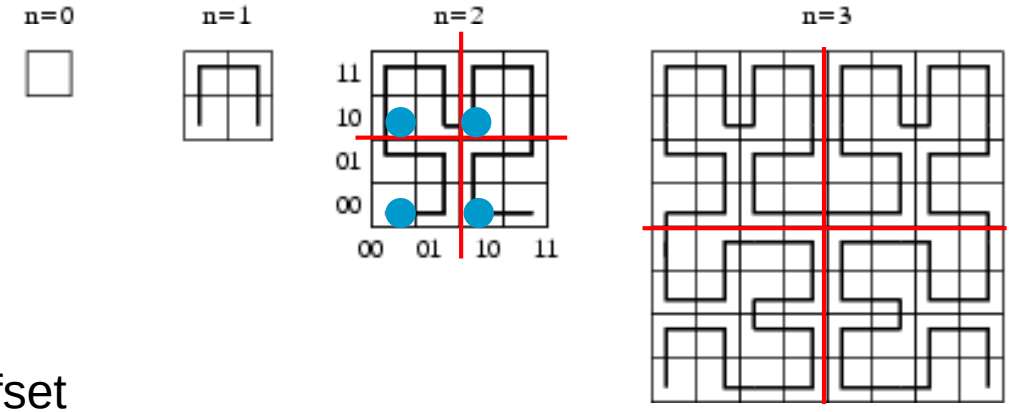



http://en.wikipedia.org/wiki/Hilbert_curve

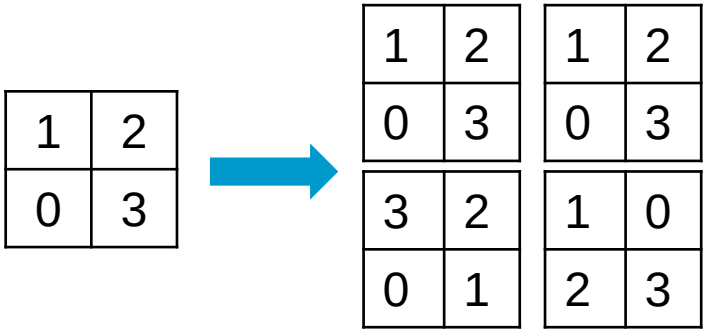
# Hilbert Curve

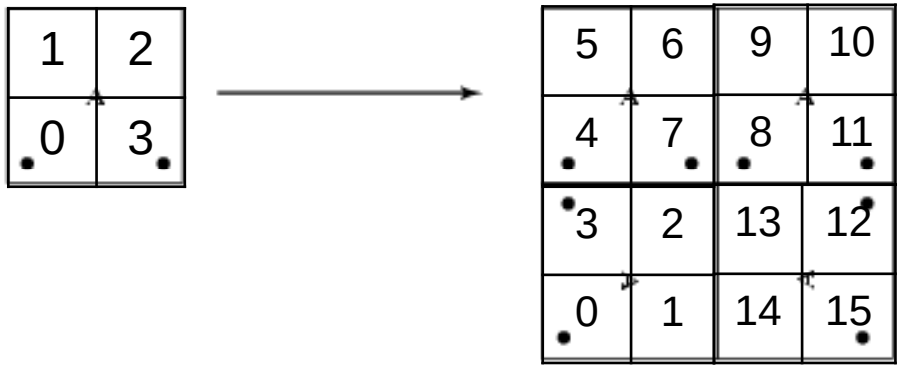



```
int numP = pow(2, order);
int numC = pow(2, order + 1);
int offset  = numP * numP;
for (int y = 0; y < numP; ++y) {
    for (int x = 0; x < numP; ++x) {
      hC[y * numC + x] = ? + 0
      hC[(y + numP) * num + x] = ? + offset
      hC[(y + numP) * num + numP + x] = ? + offset * 2
      hC[y * numP + numP + x] = ? + offset * 3
    }
}
```

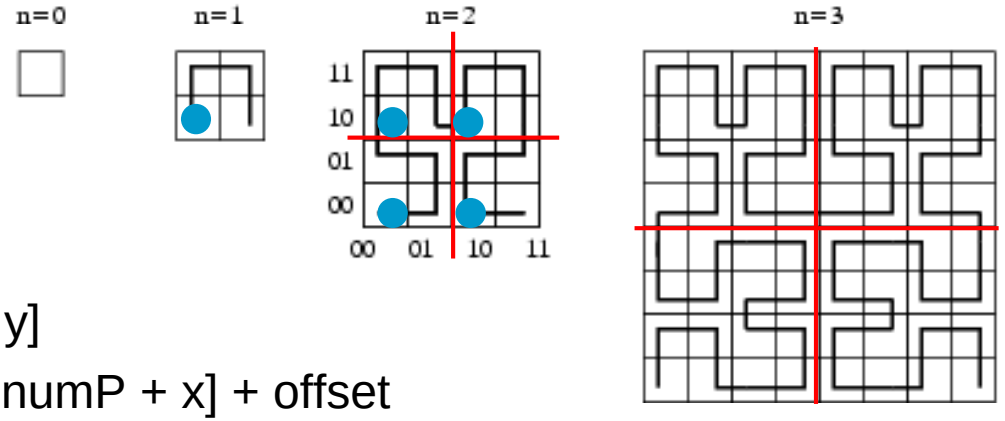


# Hilbert Curve



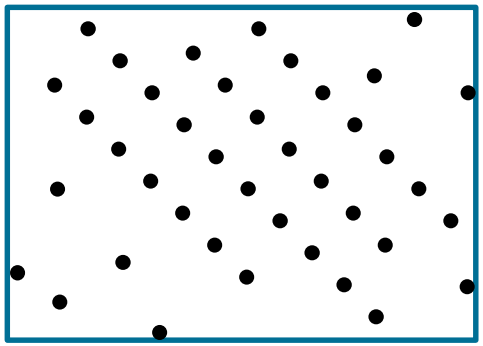

```
int numP = pow(2, order);
int numC = pow(2, order + 1);
int offset  = numP * numP;
for (int y = 0; y < numP; ++y) {
    for (int x = 0; x < numP; ++x) {
      hC[y * numC + x] = hP[x * numP + y]
      hC[(y + numP) * num + x] = hP[y * numP + x] + offset
      hC[(y + numP) * num + numP + x] = hP[y * numP + x] + offset * 2
      hC[y * numP + numP + x] = hP[(numP – 1 – y) * numP + numP – 1 – x] + offset * 3
    }
}
```

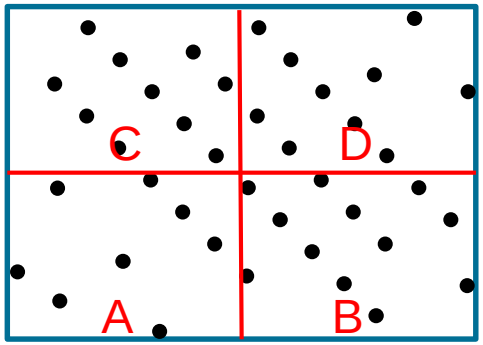


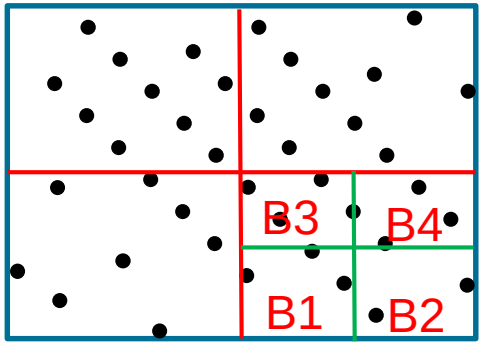
# 空间距离计算

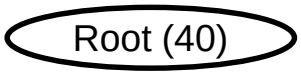
- 点、线、面与面计算距离时，需要考虑点、线和面是否在面内部，内部返回0，不在内部时，通过点、线和外边界计算最小距离

# 四叉树创建

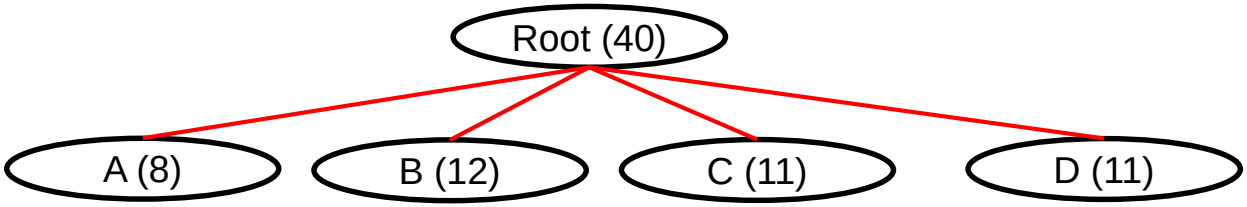


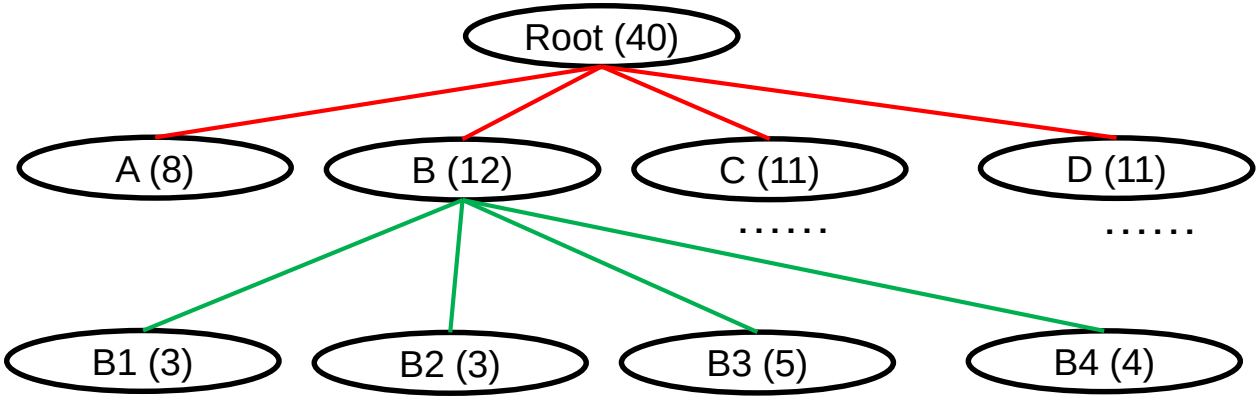


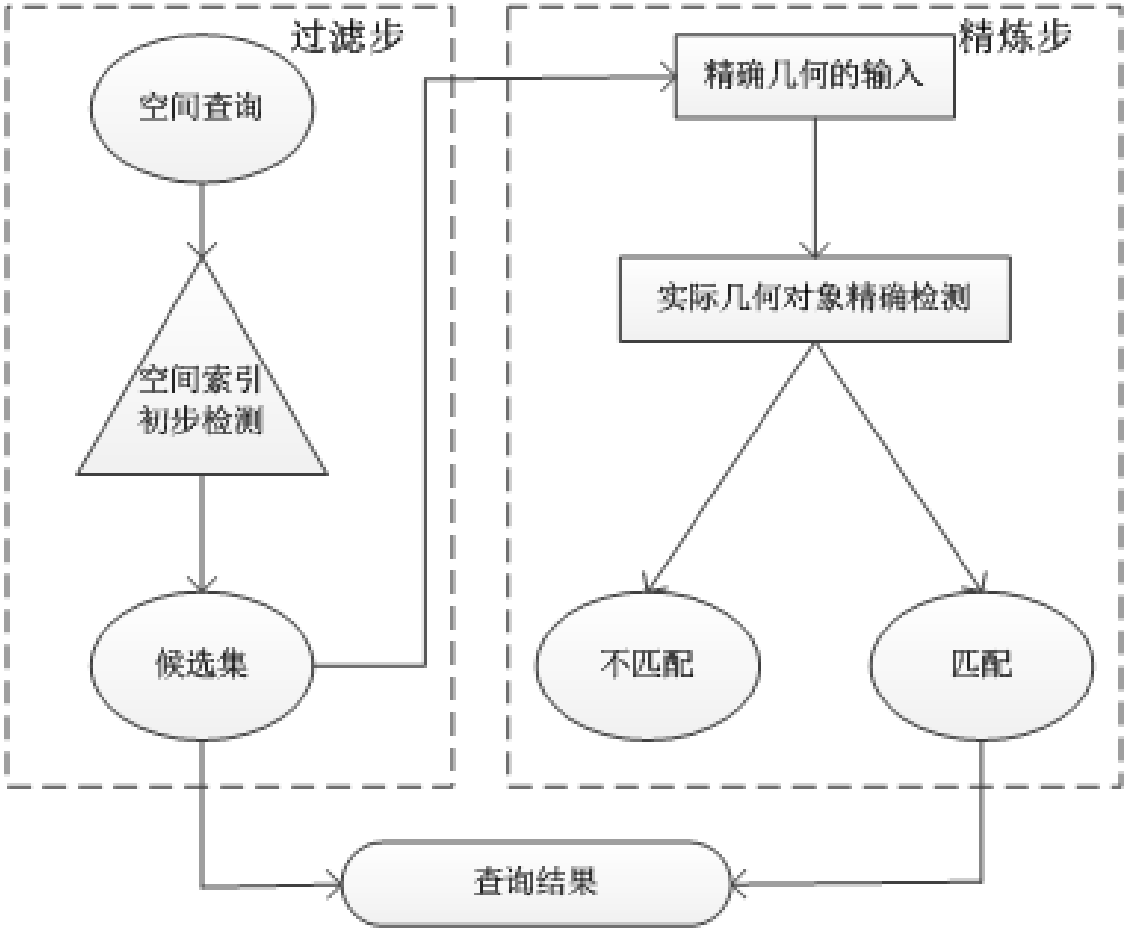

capacity = 10
Feature仅保存在叶节点
内部节点仅保留包围盒和四个子节点



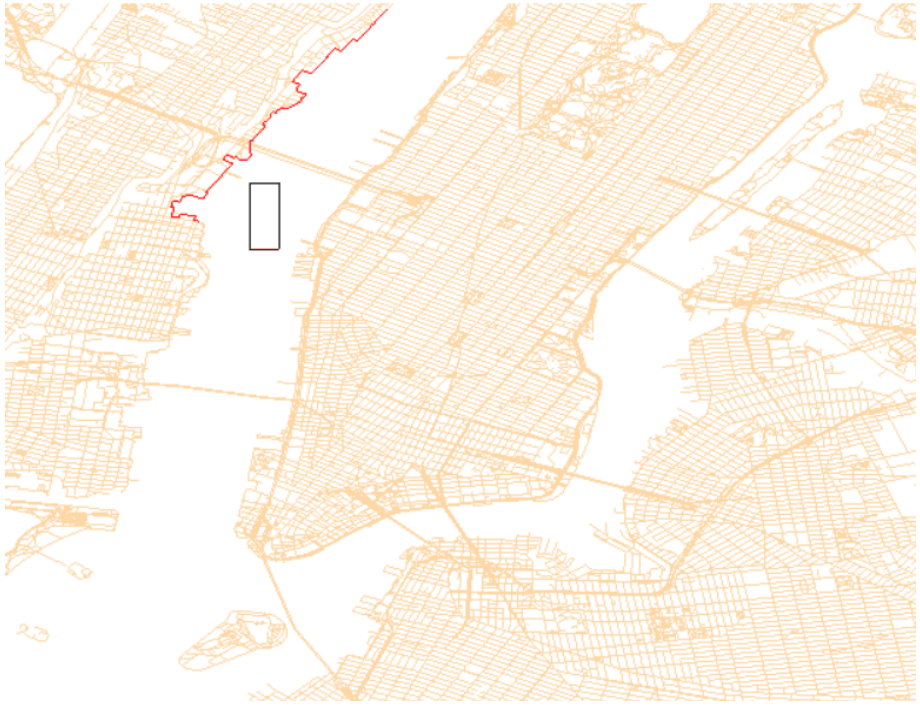
# 空间查询

- 空间查询一般分为过滤和精炼两步

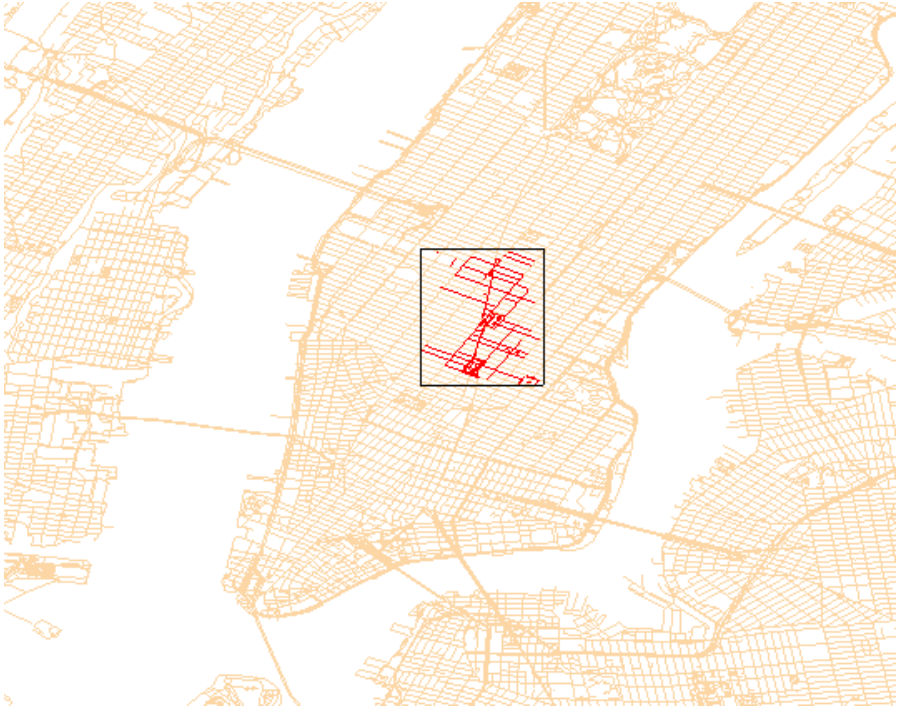


# 区域查询

- 从根节点开始判断节点包围盒是否与查询区域相交，迭代遍历树的内部节点，直到叶节点，对叶节点中每个featrue，判断是否与查询区域相交，相交就作为候选集，最后候选集中每个feature的真实几何与查询区域判断是否真正相交，得到区域查询结果



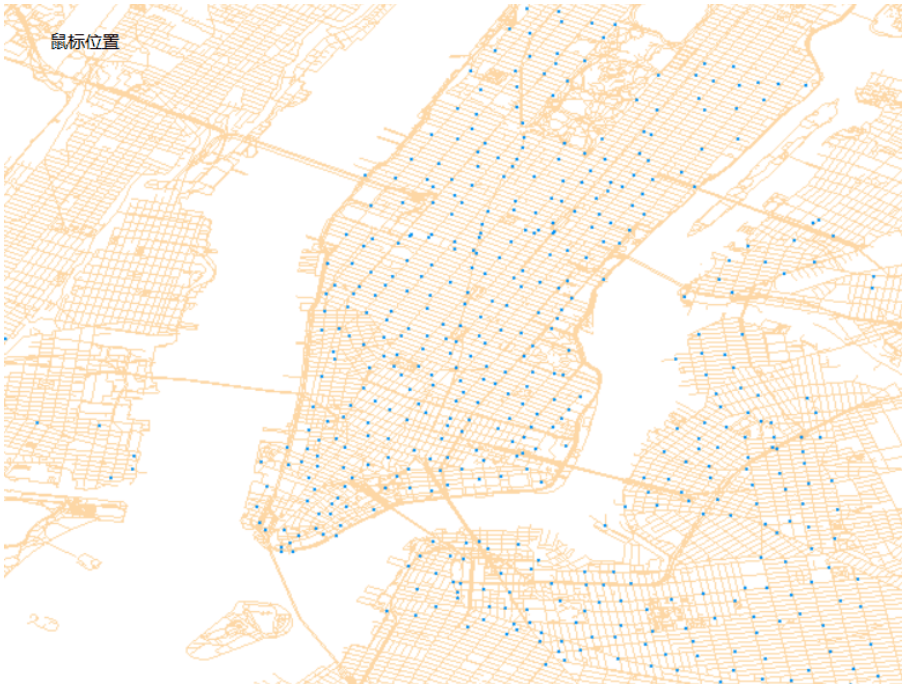
错误：没有判断和查询区域是否真正相交



错误：区域查询是intersect，不是within

# NN查询

- 遍历四叉树获得查询点所在的叶节点，判断查询点到叶节点中feature最大距离的最小值，获得区域查询边长，再按区域查询获得相交feature集合，依次与查询点计算最小距离，获得最近feature
  - 当叶节点没有feature时，如何合理设置区域查询边长？
    - 太大，效率较低
    - 太小，查询错误
    - 叶节点边长 * 2

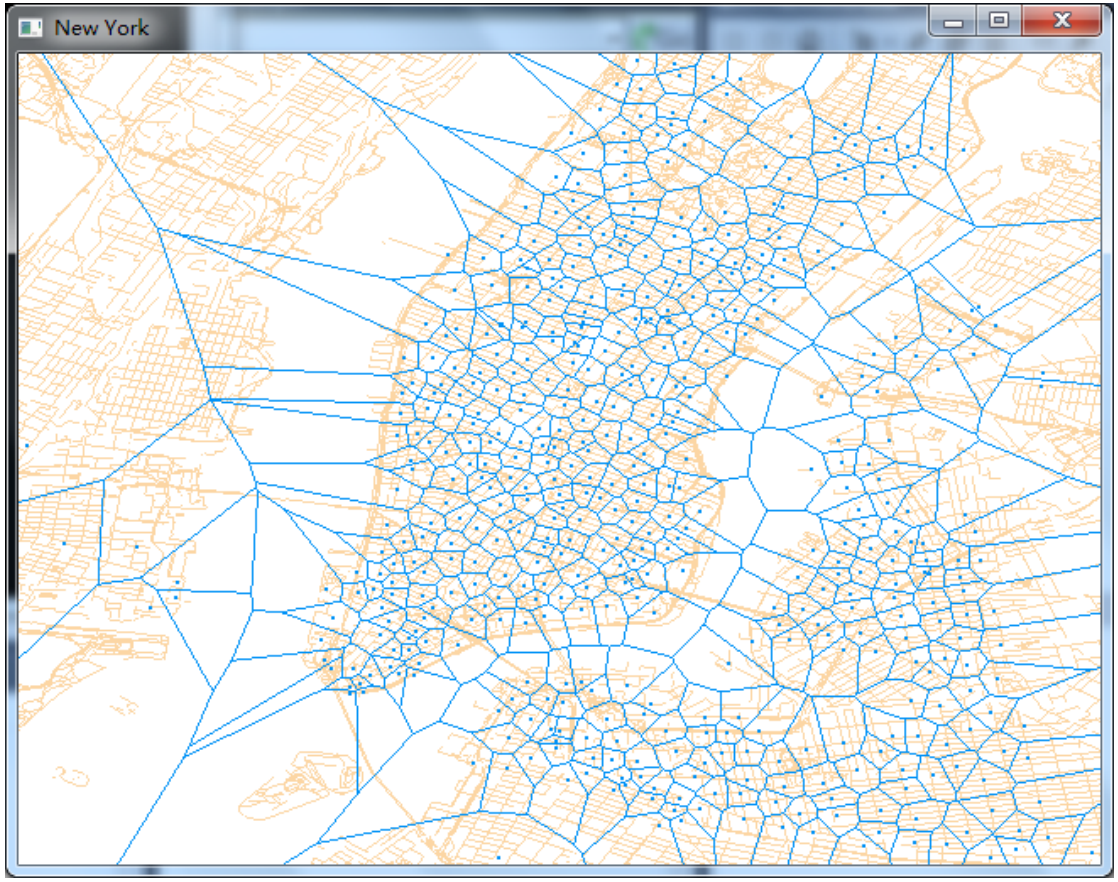


# 四叉树性能分析

- Capacity从70到200，查询效率分析
  - 树的高度降低，节点数目变少
  - 但效率基本没变，甚至越有降低（为什么？）
- Capacity为50
  - 递归调用层次过高，堆栈溢出（如何解决？）
- 原因分析
  - 最佳capacity在70以内，因此70-200时查询效率没有提高
  - 点和点的计算较快，使得计算次数影响不大

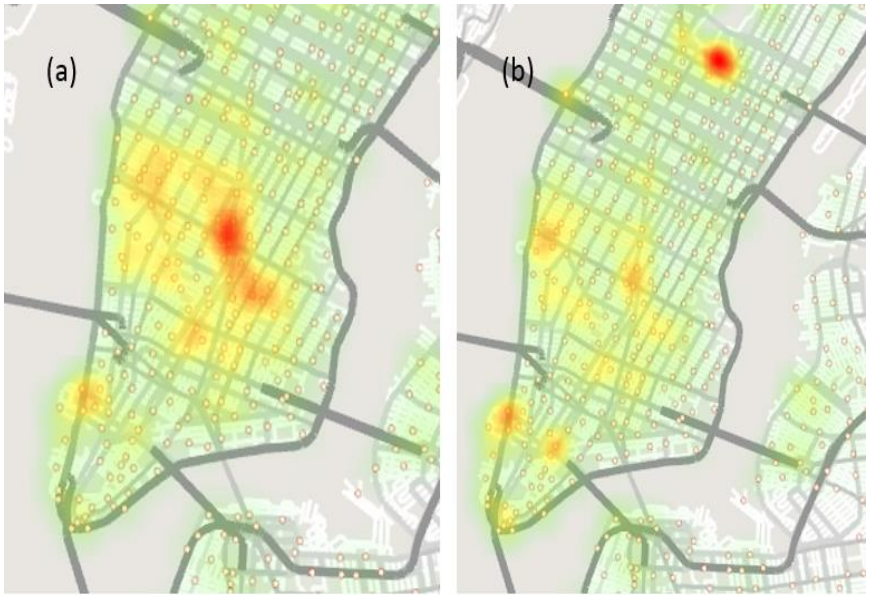
# Voronoi Diagram

- Allocation (service centers, customers)
  - Map service area for each service center
  - https://en.wikipedia.org/wiki/Voronoi_diagram



# Heat Map

- Taxi数据绘制打车位置热力图
  - 构建屏幕大小的二维数组
  - 计算每个打车位置所在的像素，统计每个像素的打车数目
    - 可以通过高斯滤波提高区域的平滑性
  - 将数值数目映射为颜色并通过OpenGL绘制
    - 颜色映射有一定的原则，参考http://colorbrewer2.org/
    - OpenGL可以通过纹理的方式绘制图像



# 何时选择属性索引？

- Make some attribute K a search key if the WHERE clause contains
  - An exact match on K
  - A range predicate on K
  - A join on K
  - Order by / group by
- 如何选择非空间索引？
  - 看where，group by，order by使用的属性，根据属性的特点(唯一或非唯一)和用法(=，>或<，<>)创建索引
  - 当同一个关系使用多个属性的等值判断，可以创建多属性索引加速

# 何时选择空间索引？

- 空间函数和空间索引 – 减少不必要的两两空间处理
  - where子句调用空间函数时，可能会使用空间索引
    - 是否真的调用，取决于查询规划器得到的全表扫描和使用索引扫描的cost值
  - select子句调用空间函数时，通常不会使用空间索引
    - select子句中所有空间操作都是必要的
  - 索引是建在基表上，通过不会在查询结果上使用空间索引
- 空间索引只能创建在空间数据，尽可能创建在静态空间数据，基于几何要素的包围盒Envelope通过overlap判断，获得需要真正几何操作的候选集(filter step)