HW6

522031910213 朱涵

April 7, 2024

1 简答

1.1 问题1：在构造KD-Tree时，如何消除多点共垂直、共水平的退化情况？

查询了一些相关资料后，我认为以下几种方法可能可以减缓产生多点共线的情况：

1. **添加随机噪声**：在划分时给数据点添加一些随机扰动，这样可以大大降低多点共线的可能性，当然在查询和存放时用的是真实值。
2. **使用超平面分割**：考虑不用垂直或水平线进行划分，而是用超平面进行划分，从而避免KD树的退化情况

1.2 问题2：KD-Tree相对于（二维）四分树、（三维）八分树，在什么情况下有什么优势？

查询了一些相关资料后，我认为KD树有以下几种优势：

1. 适用于**高维数据**： KD树能够将数据空间按照每个维度进行递归地划分，从而形成一个多层的树结构。相比之下，如果需要三维以上的数据存储，四分树和八分树会面临指数级增长的划分数（变成十六分树等等），效率较低。
2. 更适合**最近邻搜索**： 对于最近邻搜索问题，KD树可以通过递归+剪枝达到$O(n^{1-\frac{1}{k}})$的平均时间复杂度（k是维度），相比之下四分数八分树无法就要慢许多。

2 实践

2.1 建立KD树

根据15个点建立 的KD树如下图。

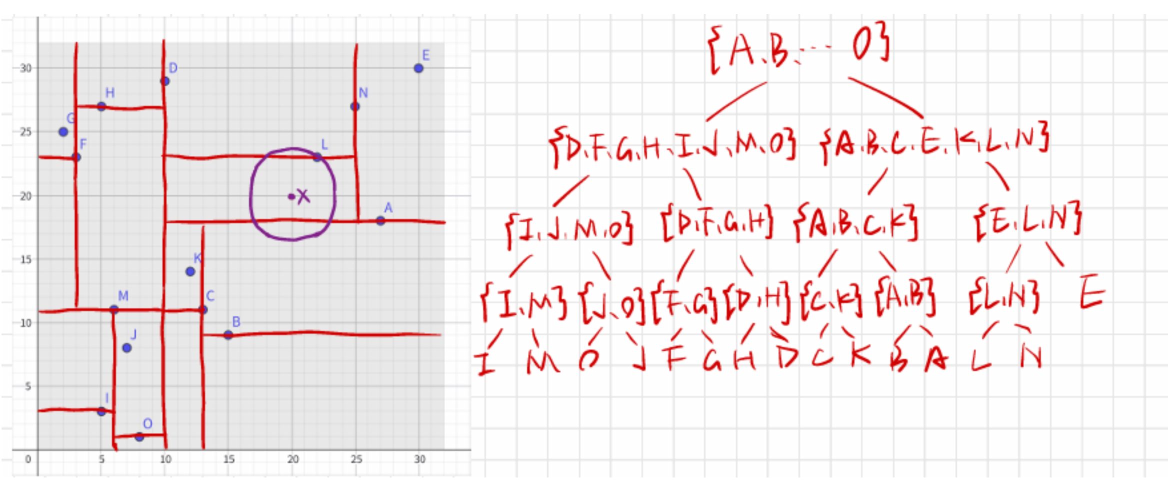

Figure 1: KD树

2.2 K近邻查询

KD树的最近邻查询分为两步：**查询叶节点**及**递归回溯**。下面设点（20,20）为X。

2.2.1 查询叶节点

从根开始查询，由于点X横坐标大于划分点10，因此进入右儿子节点。以此¿类推，查询路径为：根=> 右儿子=>右儿子=>左儿子=>左儿子，最终到达叶节点L。点X所落在的区域的确是L节点的区域，此时把L节点作为点X的最近邻节点，欧氏距离为$\sqrt{13}$。

2.2.2 回溯搜索路径

回溯到路径上一个节点，即判断L节点的兄弟N节点区域是否存在更近的节点。以$\sqrt{13}$为半径，X为圆心画圆，发现与父亲节点划分面y=23有交集，说明需要进入兄弟节点进行查找。把左子空间N加入搜索路径中，计算得距离大于当前最近距离$\sqrt{13}$，不用更新。继续回溯到父节点划分面x=25（节点ELN），发现圆与此划分面无交集，无需查询兄弟区域。同理回溯到划分面y=18（节点ABCEKLN），因为有交集所以查询兄弟区域，搜索到叶节点A，计算得距离大于当前最近距离，无需更新。继续回溯到根节点，划分面无交集，查询结束。最终得到最近邻点为**点L**，最近距离为$\sqrt{13}$。