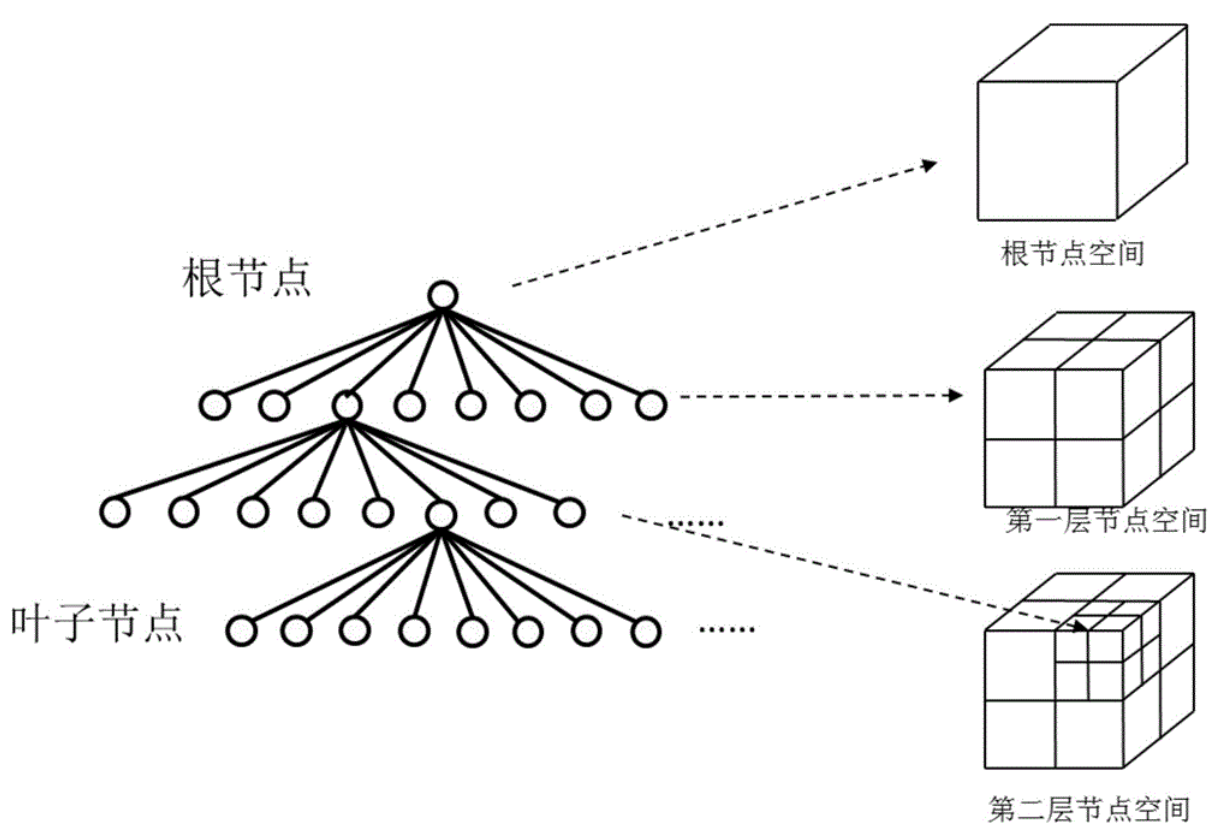
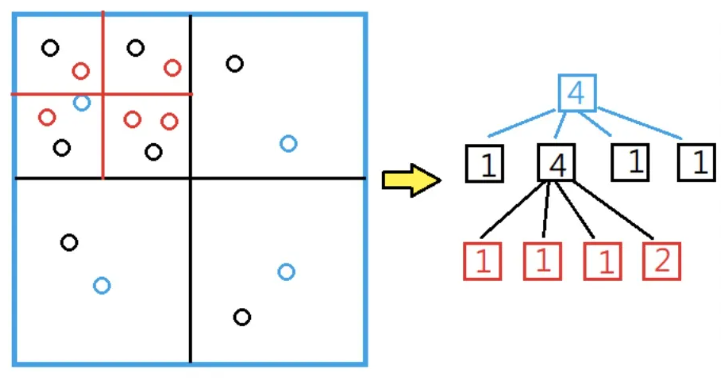
**HW6 KD-Tree小练习**

**简答（言之有理即可）**

1.在构造KD-Tree时，如何消除多点共垂直、共水平的退化情况？

我认为其中一种方法可以是在构造kd-tree时，给每个点添加微小的随机扰动，但不修改其真实值，这样就能使其在进行划分的时候，避免多点共垂直，共水平的情况出现。

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？



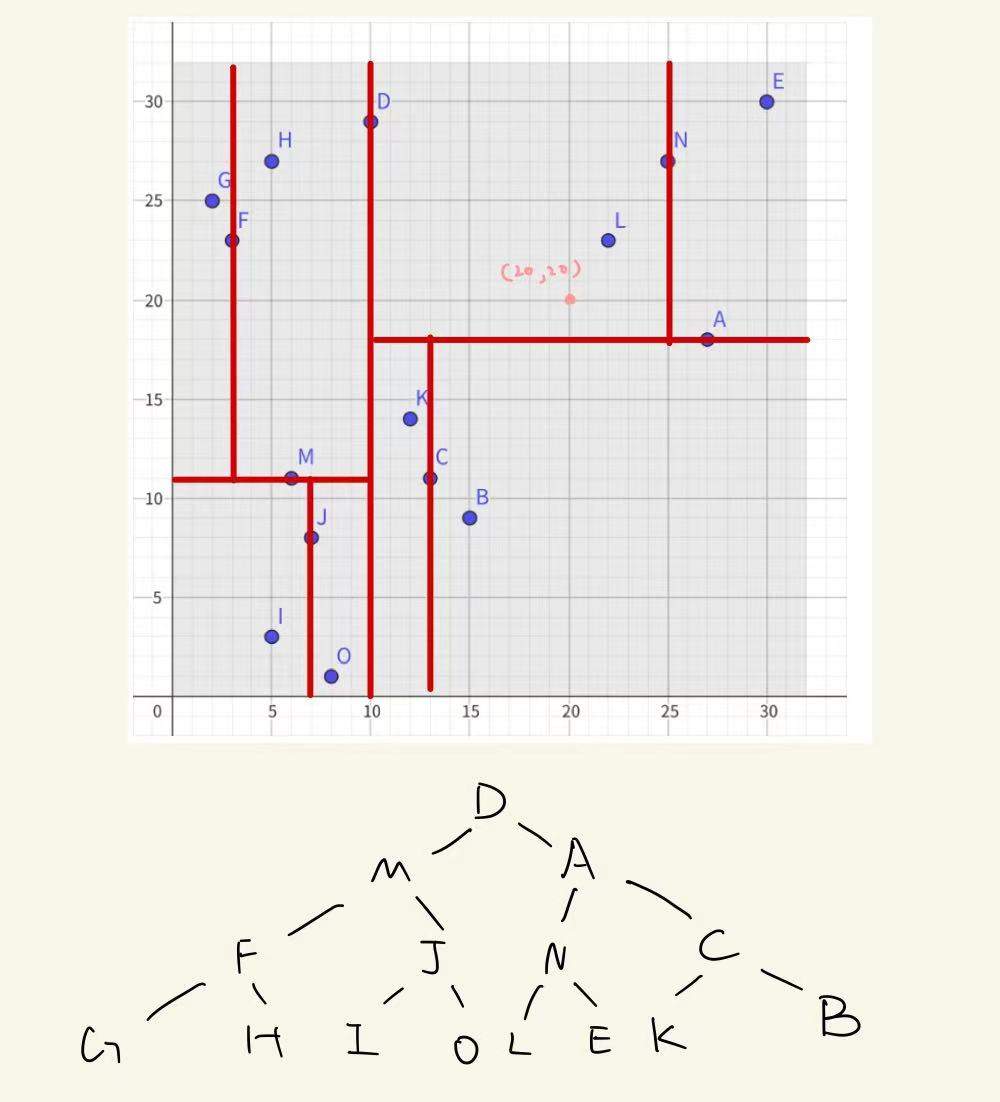
在需要查找最临近节点的情况下，通过最近邻查找算法，在kd-tree中，可以剪枝大量无需遍历的子树，其查找平均时间复杂度可以达到，k为维度。而（二维）四分树、（三维）八分树难以做到。同时，kd-tree的存储比（二维）四分树、（三维）八分树更节省空间，在需要有效利用存储空间时，kd-tree更有优势。

**实践**

在以下 [0,32]×[0,32] 的二维空间中（灰色区域），有15个点。

请回答以下问题：

1.将这15个点建立KD-Tree（要求首先按照横轴划分）。



2.在构造的KD-Tree中，要查找离点 (20,20) 距离最近的点，请给出查找过程。

查找D：发现(20,20)在D右侧，因此查找A分支

查找A：发现(20,20)在A上侧，因此查找N分支

查找N：发现(20,20)在N左侧，因此查找L分支

查找L：发现没有孩子，更新L到(20,20)为最短距离，回退至N

查找N：发现(20,20)到N的x坐标的水平距离大于当前最短距离，回退至A

查找A：发现(20,20)到A的y坐标的竖直距离小于当前最短距离，因此查找C分支

查找C：发现(20,20)在C右侧，因此查找B分支

查找B：发现没有孩子，但是B到(20,20)的距离大于当前最短距离，不更新，回退至C

查找C：发现(20,20)到C的x坐标的水平距离大于当前最短距离，回退至A

查找A：发现左右孩子都查找过了，回退至D

查找D：发现(20,20)到D的x坐标的水平距离大于当前最短距离，终止查找

最终最近的点为L