实验（5）：时空数据查询

实验要求：为给定的空间数据建立不同的索引结构(例如：KD树,R树)（二选一）以完成最近邻搜索，对索引性能并进行分析。

数据集：CA ，BJ

**5、时空数据查询**

在进行近邻数据查询中最容易的办法就是线性扫描，也就是穷举搜索，依次计算样本集E中每个样本到输入实例点的距离，然后抽取出计算出来的最小距离的点即为最近邻点。此办法简单直白，但数据量很大时，时间代价无法接受。因此需要构建数据索引，因为实际数据一般都会呈现簇状的聚类形态，因此可为数据建立索引，然后再进行快速匹配。索引树是一种树结构索引方法，其基本思想是对搜索空间进行层次划分。根据划分的空间是否有混叠可以分为Clipping和Overlapping两种。前者划分空间没有重叠，其代表就是KD树；后者划分空间相互有交叠，其代表为R树。**本此实验对这两种基本索引结构进行实现分析，完成基本实验要求后可建立多种空间索引结构进行性能对比**。

1. **KD树索引结构基本原理及实现**

KD树是对数据点在k维空间（如二维(x，y)，三维(x，y，z)）进行划分的一种数据结构，主要应用于多维空间关键数据的搜索。本质上说，KD树就是一种平衡二叉树。KD树是一种空间划分树，把整个空间划分为特定的几个部分，然后在特定空间部分内进行相关搜索操作。KD树按一定的划分规则把这个二维空间划分了多个空间，如下图所示：

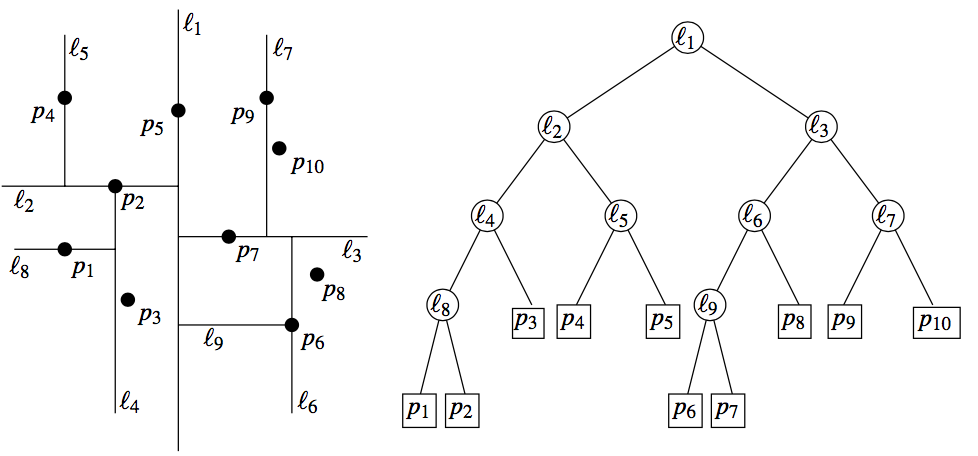


图5 kd树示意图

**KD树的构建过程如下:**

输入：K维空间数据集T={x1, x2, …. xn}，其中 xi={xi(1), xi(2), … xi(k)}，i=1,….N。

输出：KD树

1. 开始：构造根结点，根结点对应于包含T的k维空间的超矩形区域。选择x(1)为坐标轴，以T中所有实例的x(1)坐标的中位数为切分点，将根结点对应的超矩形区域切分为两个子区域。切分由通过切分点并与坐标轴x(1)垂直的超平面实现。由根结点生成深度为1的左、右子结点：左子结点对应坐标x(1)小于切分点的子区域，右子结点对应于坐标x(1)大于切分点的子区域。将落在切分超平面上的实例点保存在根结点。
2. 重复。对深度为j的结点选择x(l)为切分的坐标轴，l=j%k+1，以该结点的区域中所有实例的x(l)坐标的中位数为切分点，将该结点对应的超矩形区域切分为两个子区域。切分由通过切分点并与坐标轴x(l)垂直的超平面实现。由该结点生成深度为j+1的左、右子结点：左子结点对应坐标x(l)小于切分点的子区域，右子结点对应坐标x(l)大于切分点的子区域。将落在切分超平面上的实例点保存在该结点。

**KD树近邻搜索算法流程：**

1. 在kd树中找出包含目标点x的叶结点：从根结点出发，递归地向下搜索kd树。若目标点x当前维的坐标小于切分点的坐标，则移动到左子结点，否则移动到右子结点，直到子结点为叶结点为止。
2. 以此叶结点为“当前最近点”。
3. 递归的向上回溯，在每个结点进行以下操作：  
   （a）如果该结点保存的实例点比当前最近点距离目标点更近，则更新“当前最近点”，也就是说以该实例点为“当前最近点”。  
   （b）当前最近点一定存在于该结点一个子结点对应的区域，检查子结点的父结点的另一子结点对应的区域是否有更近的点。具体做法是，检查另一子结点对应的区域是否以目标点位球心，以目标点与“当前最近点”间的距离为半径的圆或超球体相交：  
   如果相交，可能在另一个子结点对应的区域内存在距目标点更近的点，移动到另一个子结点，接着，继续递归地进行最近邻搜索；  
   如果不相交，向上回溯。

在实验中，可自由选择编程语言（如C、Java、Python…）按给定的示例数据CA及BJ完成索引建立及近邻搜索，并能够对程序运行过程进行讲解。

**示例：**

INPUT: 节点50；OUTPUT: 该节点的最近邻节点以及算法构建查询时间。

1. **基于R树的近邻搜索基本原理及实现**

R树运用了空间分割的理念，R树采用MBR(Minimal Bounding Rectangle)方法。从叶子结点开始用矩形（rectangle）将空间框起来，结点越往上，框住的空间就越大，以此对空间进行分割。

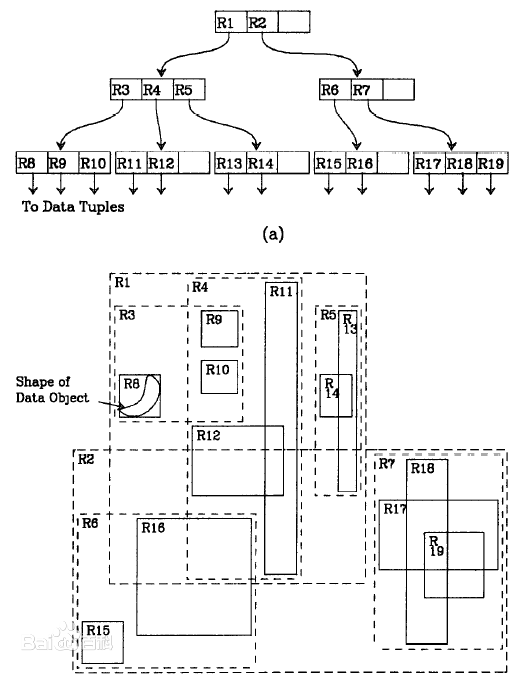


图6 映射示意图

首先我们假设所有数据都是二维空间下的点，图中仅仅标志了R8区域中的数据，为了实现R树结构，用一个MBR恰好框住该不规则区域：R8。R8刚好框住所有在此区域中的数据。其他实线包围住的区域，如R9，R10，R12同理，共得到了12个最基本的MBR，这些矩形被存储在子结点中。

下一步操作就是进行高一层次的处理。R8，R9，R10三个矩形距离最为靠近，因此就可以用一个更大的矩形R3恰好框住这3个矩形。同理，R15，R16被R6恰好框住，R11，R12被R4恰好框住。所有最基本的MBR被框进更大的矩形中之后，再次迭代，用更大的框去框住这些矩形。下面就可以把这些矩形存入R树中。根结点存放的是两个最大的矩形，这两个最大的矩形框住了所有的剩余的矩形，当然也就框住了所有的数据。下一层的结点存放了次大的矩形，这些矩形缩小了范围。每个叶子结点都是存放的最小的矩形，这些矩形中可能包含有n个数据

**基于R树的近邻搜索算法流程：**

描述：假设T为一棵R树的根结点，查找所有搜索矩形S覆盖的记录条目。

S1:[查找子树] 如果T是非叶子结点，如果T所对应的矩形与S有重合，那么检查所有T中存储的条目，对于所有这些条目，使用Search操作作用在每一个条目所指向的子树的根结点上（即T结点的孩子结点）。

S2:[查找叶子结点] 如果T是叶子结点，如果T所对应的矩形与S有重合，那么直接检查S所指向的所有记录条目。返回符合条件的记录。

在实验中，可自由选择编程语言（C、Java、Python…）按给定的步骤完成数据索引的建立以及近邻查询过程，对程序运行过程进行分析。

**示例：**

INPUT: (提供数据集)指定查询节点 100；

OUTPUT: 输出该节点的最近邻信息及算法构建查询时间。