lab5 实验报告

-------闫世杰 2020200982

790:二维搜索树

采用 Ranger_Tree 实现

思路:根据二维平面点建立 2D_Ranger_Tree,在 Ranger_Tree 进行 Ranger_Search

数据结构:

```
struct Y_Tree{ //X_Tree 中所包含的 Y 节点
    Point point; //节点值
    Y_Tree *left; //左子树
    Y_Tree *right; //右子树};
struct X_Tree{
    Point point; //节点值
    X_Tree *left; //左子树
    X_Tree *right; //右子树
    Y_Tree *yt; //对应的 Y_Tree
    int max, min//记录当前节点及其子树的 x 坐标的取值范围};
```

实现:i)建树

根据输入依次插入节点来建立 2D_Ranger_Tree,和普通的 BST 类似,但是在寻找插入位置的过程中需要将该节点插入到路径上所有 X_Tree 节点的 X_Tree.yt 上,同时需要判断 point.x 与节点 max min 值的大小关系,来维护树的性质

ii)搜索

先根据节点的 x 维的值进行搜索,根据搜索到的当前节点的结果判断所要进行的操作

1.当前节点及其子树的 x 坐标值都在 ranger 中,直接进入 Y_Tree 进行一维搜索

2.当前节点及其子树的 x 坐标值部分在 ranger 中,递归检查子树

3.当前节点及其子树的 x 坐标值都不在 ranger 中,停止搜索

(判断与 ranger 关系的实现:直接比较节点 max/min 值与 ranger 的关系即可

提交发现 TLE,尝试优化:以 x 为基准 建立平衡二叉树,仍然超时,转换为 KD_Tree,AC(据观察,应该是插入建树所消耗的时间较大,导致超时

数据结构和 791 的 KDT 相同

实现:i)建树

将点存在一数组中,逐渐递归选取中间点作为节点,其余值均等分布在左右子树上,建立比较平衡的 KD 树,保证搜索速度够快

ii)搜索

每次检查当前节点,判断当前节点的 x/y 值与 range 的关系

i)在 range 范围左侧

检查右子树即可

ii)在 range 范围右侧

检查左子树即可

iii)在 range 之间

判断节点是否符合

检查左右两个子树

791:二维平面最近点

采用 KD_Tree 实现

思路:根据二维平面点建立 2 维的 KD_Tree，在 KD_Tree 上进行 Nearest_Neighbor_search

数据结构:

```
struct KDT{
    Point point; //记录该点的值
    KDT *left; //指向左孩子
    KDT *right; //指向右孩子
    int D; //记录当前划分所依据的维度:D=0 means y; D=1 means x;};
```

实现:

i)建树

根据输入依次插入节点来建立 KD_Tree,实现方式和普通的 BST 没有区别,只是在搜索插入位置的时候需要根据节点的 D 的值来判断划分的依据

存在问题:直接根据输入的次序建树,可能会出现最差的情况

ii)搜索:

整个搜索过程是一个递归+回溯的过程

KD_Tree 建树相当于不断把整个空间二分切割划分成更小的区域

先递归查找目标点所在的最小切割区域,在该区域中求出最近距离

然后不断的回溯将区域扩大至整个平面,在回溯过程中依据当前已求出的最近距离来判断是否需要查找目标点不在的另一半区域

当回溯到根节点,平面已经扩展至整个平面,此时的最短距即为所求