**求平面上n个顶点的最近点对问题**

**实验报告**

1. **实验目的**

通过实践，加深理解在n 个点的集合Q中寻找最近点对的问题，提高编程能力。

1. **实验内容**

编程实现求平面上n个顶点的最近点对问题。

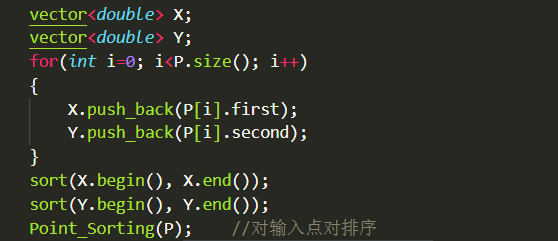
1. **实验步骤**

解决本问题可以通过分治算法递归实现。

**1.分治算法的输入及“预排序”**

算法每一次递归调用的输入为子集PQ以及数组X和Y，每个数组均包含输入子集P的所有点，对数组X中的点排序，使其x坐标单调递增，类似地，对数组Y中的点排序，使其y坐标单调递增。我们对数组X和Y进行“预排序”，这样就无需在每次递归调用中都排序。

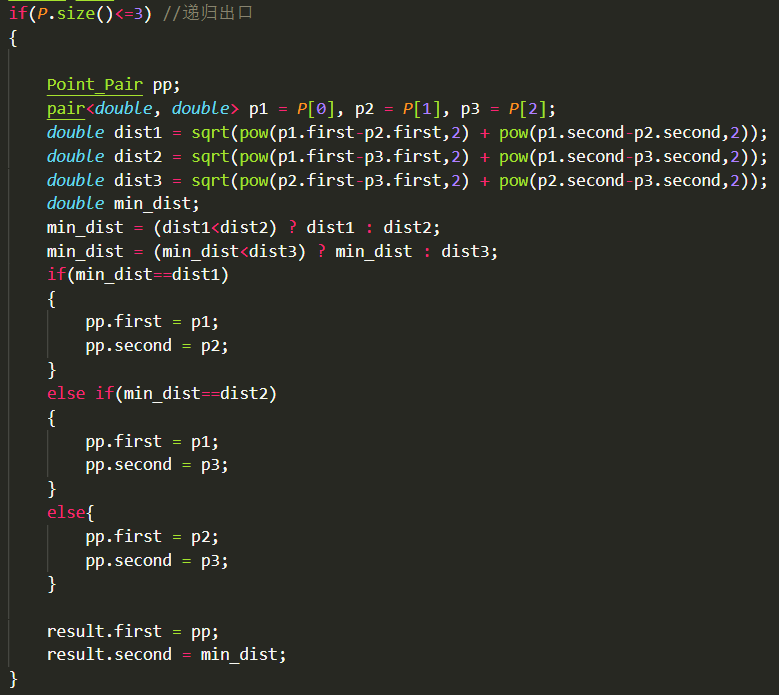
预排序的实现如下所示：



**2.分治算法的具体实现**

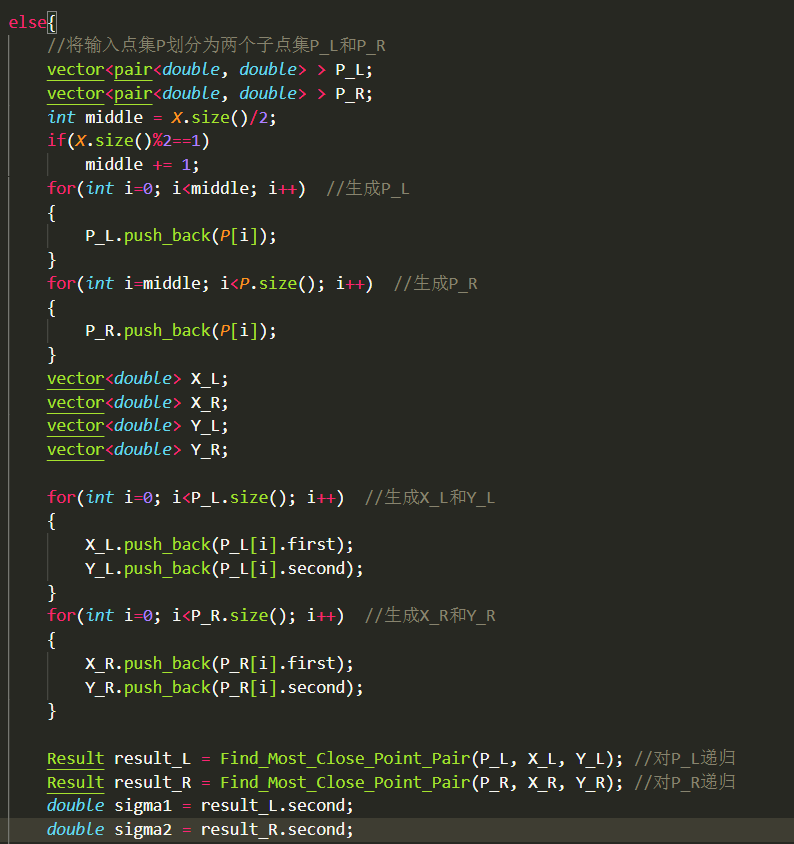
**（1）递归出口**

输入为P、X、Y的递归调用首先检查是否有|P|成立。如果有，则通过暴力搜索实现：对所有个点对进行检查，并返回最近点对。



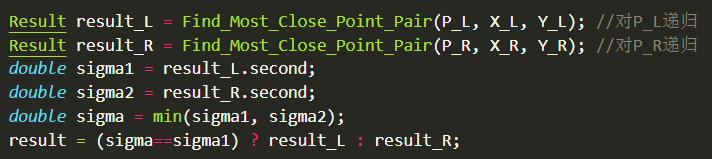
**（2）分治算法的分解**

当|P|>3时，找出一条垂直线,它把点集P对分为满足下列条件的两个集合和:使得,,中的所有点都在直线l上或在l的左侧，中的所有点都在直线l上或在l的右侧。数组X被划分为两个数组和，分别包含和中的点，并按x坐标单调递增的顺序排好序。对数组Y的处理与数组X类似。



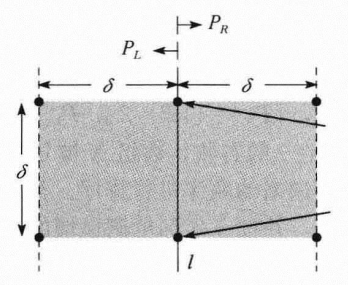
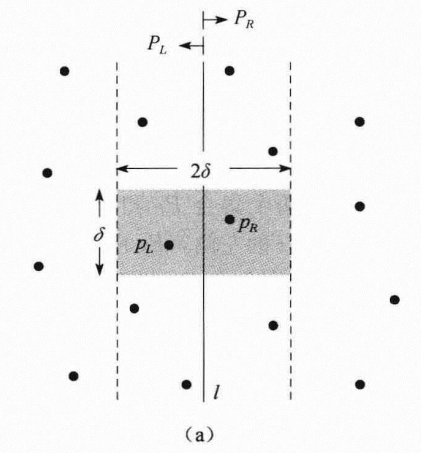
**(3)子问题的解决**

对第(2)步中划分出的和各自进行一次递归调用，一次找出中的最近点对，另一次找出中的最近点对。第一次调用的输入为子集、数组和，第二次调用的输入为子集、数组和。令和返回的最近点对的距离分别为和,并且置=min()。



**（4）分治算法的合并**

最近点对要么是某次递归调用找出的距离为的点对，要么是中的一个点与中的一个点组成的点对。算法确定是否存在距离小于的一个点对，一个点位于中，一个点位于中。注意，如果存在这样的一个点对，则点对中的两个点与直线l的距离必定都在单位之内。因此，它们必定都处于以直线l为中心、宽度为2的垂直带形区域内。



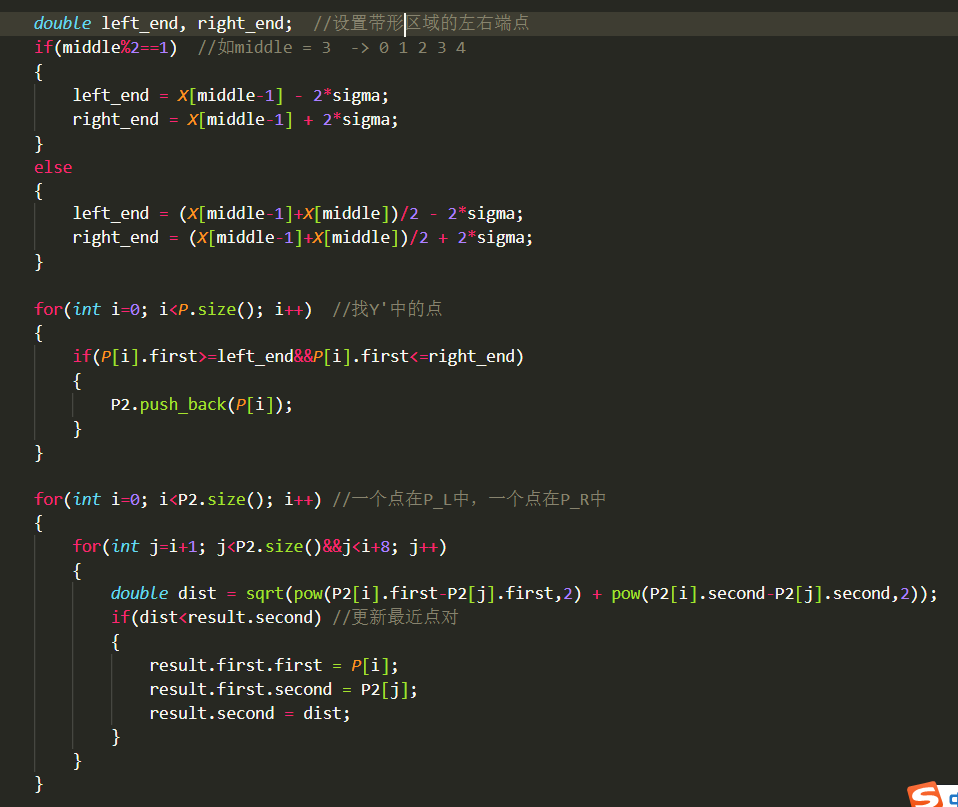
(a)带形区域 (b)P中至多有8个点位于的矩形区域内

为了找出这样的点对(如果存在)，算法要做如下工作：

①建立一个数组，它是把数组Y中所有不在宽度为2的垂直带形区域内的点去掉后得到的数组。数组与Y一样，是按y坐标排序的。

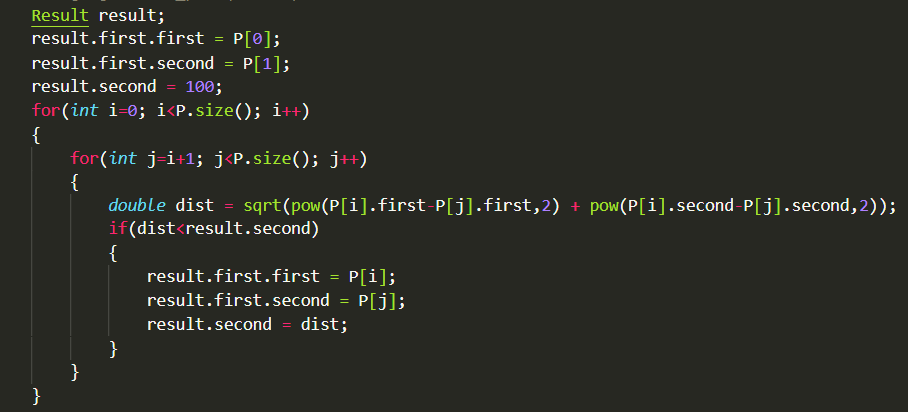
②对数组中的每个点p，算法试图找出中距离p在单位内的点。在中仅需考虑紧随p后的7个点。算法计算出从p到这7个点的距离，并记录下中的所有点对中最近点对的距离。

③如果,则垂直带形区域内的确包含比根据递归调用所找出的最近距离更近的点对，于是返回该点对及其距离。否则，返回函数的递归调用中发现的最近点对及其距离。



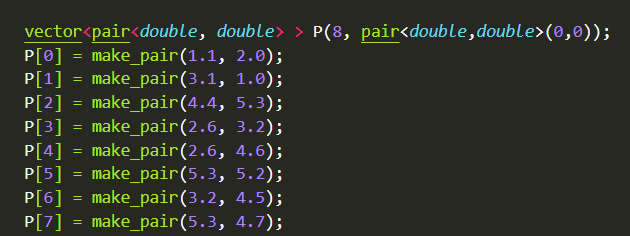
**3.使用暴力搜索算法对分治法的结果进行验证：**

**暴力搜索算法主要过程如下：**

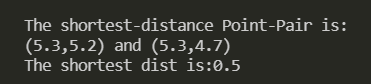


1. **实验结果**

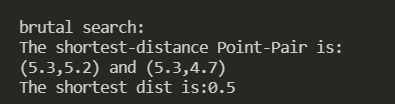
输入点集1：



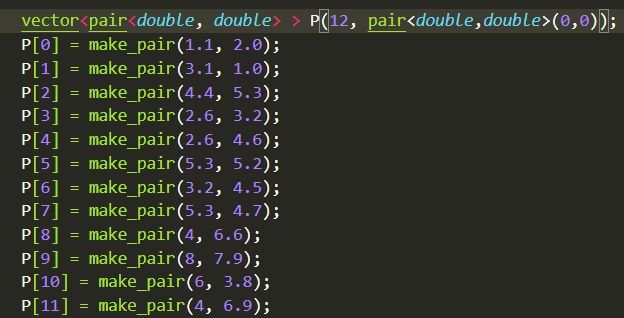
分治法结果：



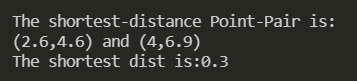
暴力搜索结果：



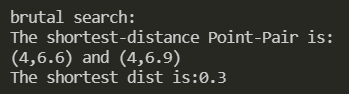
输入点集2：



分治法结果：



暴力搜索结果：



**五、实验心得**

有时候对算法的输入进行一些“预处理”能够给算法的执行效率带来一些不小得提升，例如在上述算法中，对分治算法的每个输入数组和都要按递增排序(和一样)，但是我们在分治算法的第一次输入前就对数组X和Y分别执行了一次“预排序”，使得以后在每次递归调用中子数组和、和均为按递增排序，而不需要我们每次都为其排序，给算法执行效率带来了提升。