**最接近点对算法说明书**

**1. 算法功能**

给定平面上n个点，找其中的一对点，使得在n个点组成的所有点对中该点对的距离最小。

**2. 接口参数**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 名称 | 类型 | 是否必须 | 示例值 | 描述 |
| 1 | s | int [][] | 是 | [[1,2], [2, 3]] | 二维数组，表示n个点的坐标。 |

**3. 接口返回值**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 序号 | 名称 | 类型 | 是否必须 | 示例值 | 描述 |
| 1 | p1 | int [] | 是 | [1,2] | 整型数组，表示点对中的第一个点 |
| 2 | p2 | int[] | 是 | [2,3] | 整型数组，表示点对中的第二个点 |
| 3 | dist | double | 是 | 1.414 | p1与p2的欧氏距离 |

**4. 算法实现**

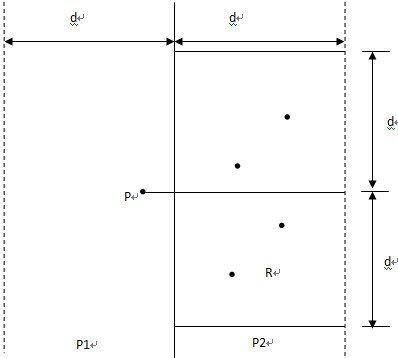
严格来说，最接近点对可能多于1对，为简单起见，只找其中的1对作为问题的解。将每一个点与其他n-1个点比较的时间复杂度为O(n^2)。可以证明该问题的计算时间下界为O(nlogn)，这个下界引导去找问题的O(nlogn)时间算法，很自然地想到用分治法解决问题。

将所给平面上n个点的集合S分成2个子集合S1和S2，每个子集中约有n/2个点。然后在每个子集中递归地求其最接近的点对。关键问题是如何实现分治法中的合并步骤，即由S1和S2的最接近点对，如何求得原集合中的最接近点对。如果组成S的最接近点对的2个点都在S1中或在S2中，则问题容易解决。但是若这2个点分别在S1和S2中，问题就相对复杂。

关键是设计二维最接近点对的合并方法。

P1中所有点与P2中所有点构成的点对均为最接近点对的候选者。在最坏情况下有n^2/4对这样的候选者。

但是P1和P2中的点具有以下的稀疏性质，它使我们不必检查所有这n^2/4对候选者。考虑P1中任意一点p,它若与P2中的点q构成最接近点对的候选者，则必有d(p,q)<d。满足这个条件的P2中的点一定落在一个d\*2d的矩形R中，如下图所示:



包含点q的d\*2d矩形R

由d的意义可知P2中任何2个S中的点的距离都不小于d。由此可以推出矩形R中最多只有6个S中的点。

在分治法的合并步骤中，最多只需要检查6\*n/2=3n对候选者，而不是n^2/4对候选者。将p和P2中所有S2的点投影到垂直线l上。由于能与p点一起构成最接近点对候选者的S2中点一定在矩形R中，所以它们在直线l上的投影点距p在l上投影点的距离小于d。这种投影点最多只有6个。因此，若将P1和P2中所有S的点按其y坐标排好序，则对P1中所有点p，对排好序的点列作一次扫描，就可以找出所有最接近点对的候选者，对P1中每一点最多只要检查P2中排好序的相继6个点。

因此二维情况下的最接近点对的分治算法如下：

1. m=S中各点x坐标的中位数，构造S1和S2
2. d1、d2分别为S1、S2中最接近点对的欧氏距离
3. dm为d1、d2中的较小者
4. P1是S1中距离垂直平分线l的距离在dm之内的所有点的集合；

P2是S2中距离垂直平分线l的距离在dm之内的所有点的集合；

P1和P2依其y坐标值排序；

设X和Y是已经排好序的点列。

1. 通过扫描X以及对于X中的每个点检查Y中与其距离在dm之内的所有点 （最多6个）可以完成合并；

设dl是按这种扫描方式找到的点对间的最小距离。

1. d=min(dm, d1)

return d

**5. 注意事项**

无