**深 圳 大 学 实 验 报 告**

**课程名称：­ 算法设计与分析**

**实验项目名称： 实验二 分治法求最近点对问题**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 杨烜**

**报告人：沈晨玙 学号：2019092121 班级： 19计科国际**

**实验时间： 2021.4.9**

**实验报告提交时间： 2021.4.9**

**教务部制**

### 实验二 分治法求最近点对问题

### 一、实验目的：

* + 1. 掌握分治法思想。
    2. 学会最近点对问题求解方法。

### 二、内容：

1. 对于平面上给定的N个点，给出所有点对的最短距离，即，输入是平面上的N个点，输出是N点中具有最短距离的两点。

2. 要求随机生成N个点的平面坐标，应用蛮力法编程计算出所有点对的最短距离。

3. 要求随机生成N个点的平面坐标，应用分治法编程计算出所有点对的最短距离。

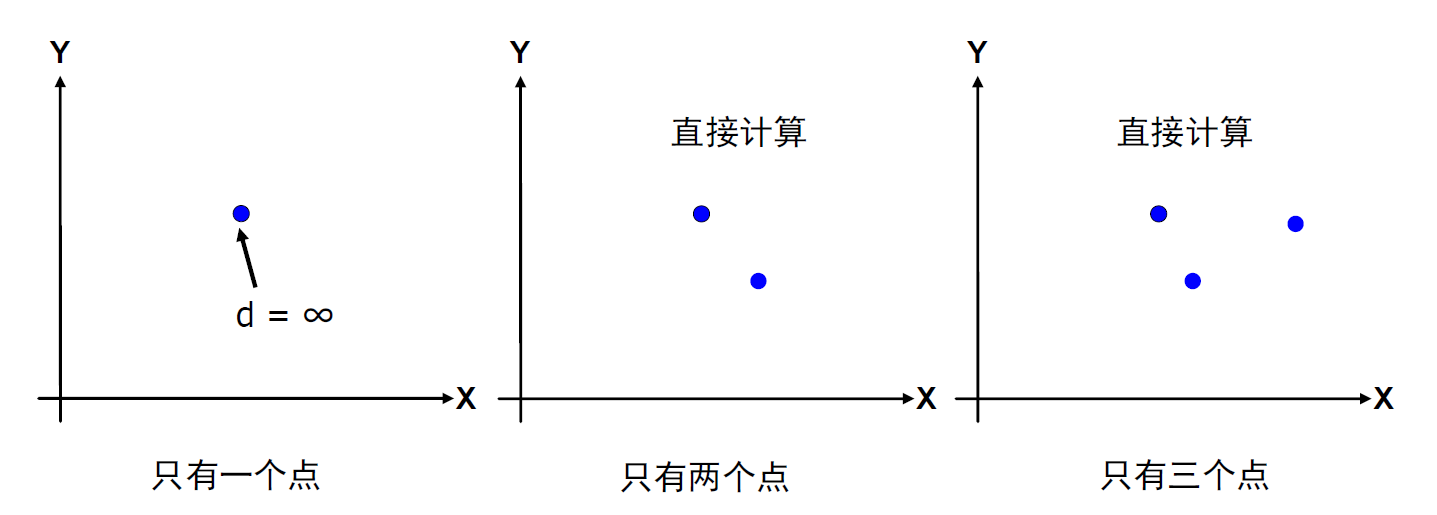
4. 分别对N=100000—1000000，统计算法运行时间，比较理论效率与实测效率的差异，同时对蛮力法和分治法的算法效率进行分析和比较。

5. 如果能将算法执行过程利用图形界面输出，可获加分。

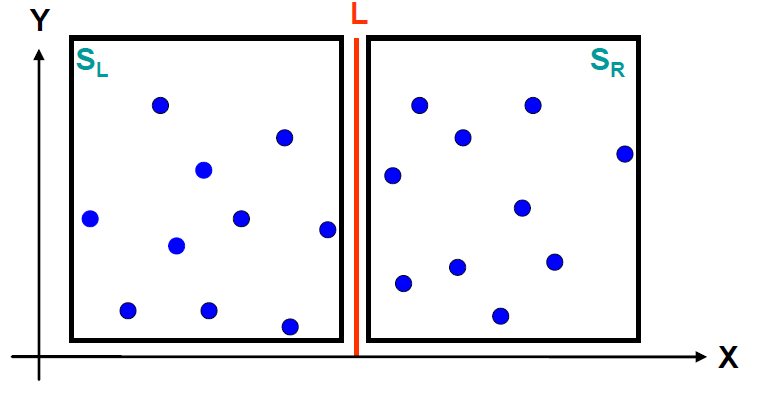
### 三、算法思想提示

1. 预处理：根据输入点集S中的x轴和y轴坐标进行排序，得到X和Y,很显然此时X和Y中的点就是S中的点。

2. 点数较少时的情形

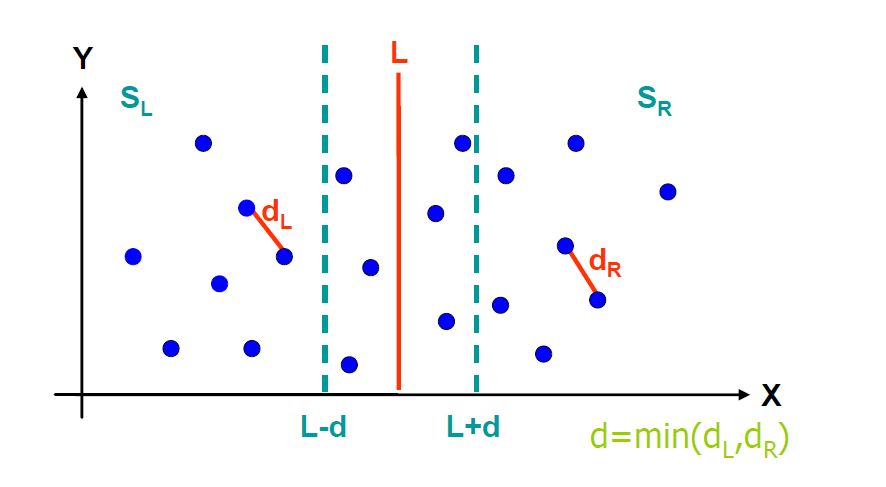


3. 点数|S|>3时，将平面点集S分割成为大小大致相等的两个子集SL和SR，选取一个垂直线L作为分割直线，如何以最快的方法尽可能均匀平分？注意这个操作如果达到效率，将导致整个算法效率达到θ。

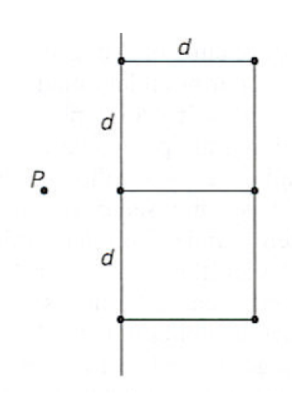


4. 两个递归调用，分别求出SL和SR中的最短距离为dl和dr。

5. 取d=min(dl, dr)，在直线L两边分别扩展d，得到边界区域Y，Y’是区域Y中的点按照y坐标值排序后得到的点集（为什么要排序？），Y'又可分为左右两个集合Y’L和Y’R



6. 对于Y’L中的每一点，检查Y’R中的点与它的距离，更新所获得的最近距离，注意这个步骤的算法效率，请务必做到线性效率，并在实验报告中详细解释为什么能做到线性效率？

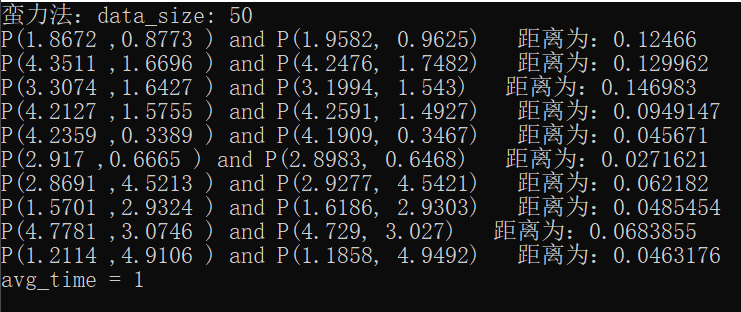


### 四、实验过程及结果

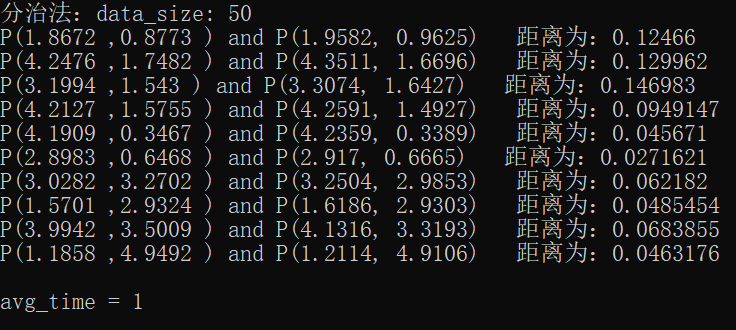
**一：实验结果正确性展示**

随机生成了10组50个点，通过蛮力法与分治法分别求解，验证程序正确性，结果如下。

蛮力法：



分治法：



小规模测试中二者测试结果一致，答案正确。

**二：蛮力法求解**

1. 算法原理：
   1. 存在N个点，那么就存在N（N - 1）/ 2 对点间的距离。穷举所有情况，选出最小值。
2. 伪代码：

*for i = 1 to N - 1*

*for j = i + 1 to N*

*if ( dis(p[i], p[j]) < min )*

*min = dis(p[i], p[j])*

1. 复杂度分析：
   1. 需要遍历N（N - 1）/ 2种情况来找出最小值，最好最坏和平均情况的时间复杂度都为 O(n^2)
   2. 需要一个临时变量用来存储最小值，所以空间复杂度为 O(1)。
2. 数据测试

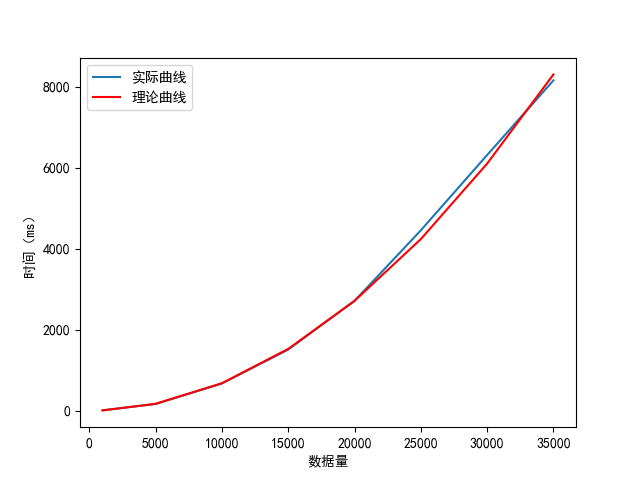
使用随机数生成，均匀分布的生成了1000~35000的数据集。为了减少数据的偶然性，每个数据量都进行了10次测试并取平均值。

为了检验实验是否准确，将实际值将理论值进行对比（基准点为20000）。理论值计算方法如下：



最终结果如下：

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 数据量 | 1000 | 5000 | 10000 | 20000 | 30000 | 40000 | 50000 |
| 单次时间（ms） | 2 | 36 | 138 | 540 | 1199 | 2130 | 3416 |
| 理论时间（ms） | 1.38 | 34.5 | 138 | 552 | 1242 | 2208 | 3450 |



图像上符合O(n^2) 二次曲线，并且理论值与实际值误差较小。

**三：分治法求解**

1. 分治法基本思路

对于本题而言，可以转化为：

分治法基本思路：

分 -- 将问题分解为规模更小的子问题；

治 -- 将这些规模更小的子问题逐个击破；

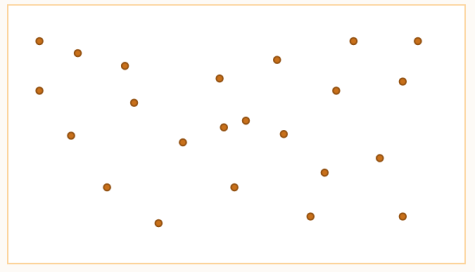
合 -- 将已解决的子问题合并，最终得出“母”问题的解；

本题思路：

分 -- 将整体分为左右两个区域；

治 -- 递归计算左右两区域的最短距离；

合 -- 合并左右区域，并求合并后的最短距离；

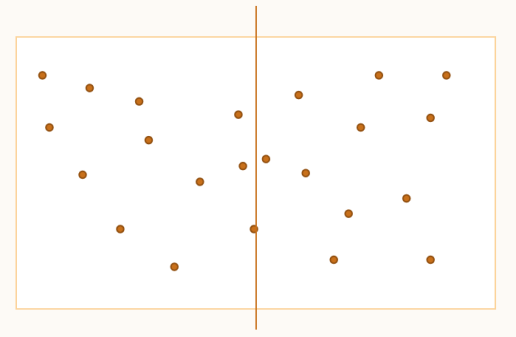


* 1. 分 -- 将整体分为左右两个区域；

将所有点根据x坐标进行排序，取中间点。所以算法时间复杂度下限：O（nlogn）

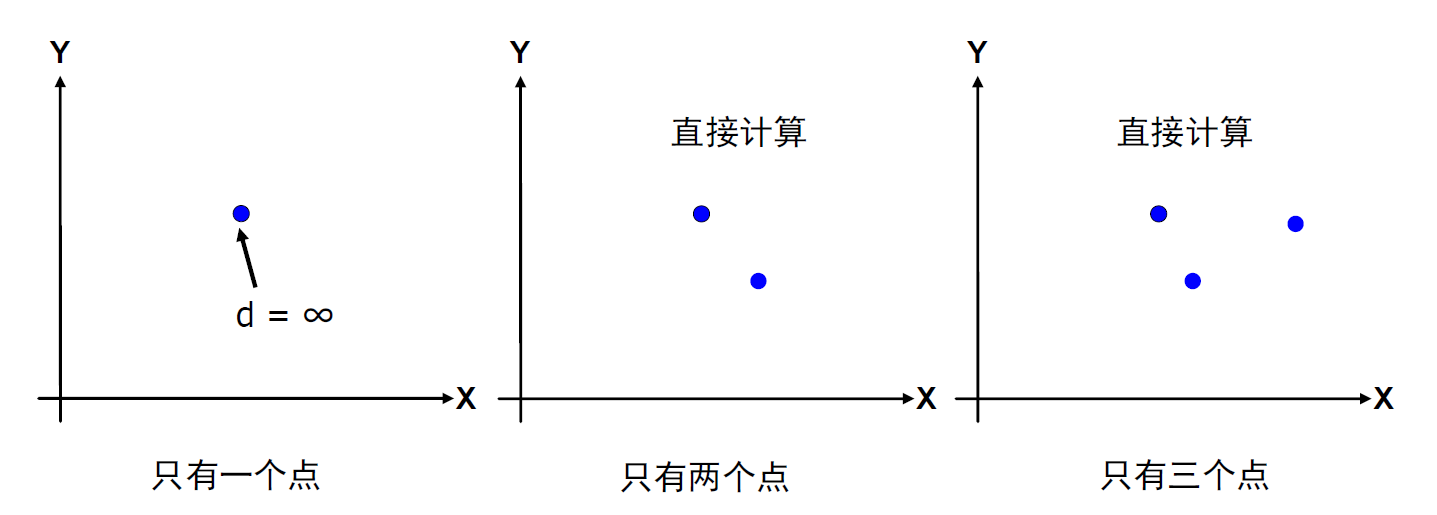
*mid = (l + r) / 2*

做到左右区域点集数目基本相同，降低数据随机性带来的影响



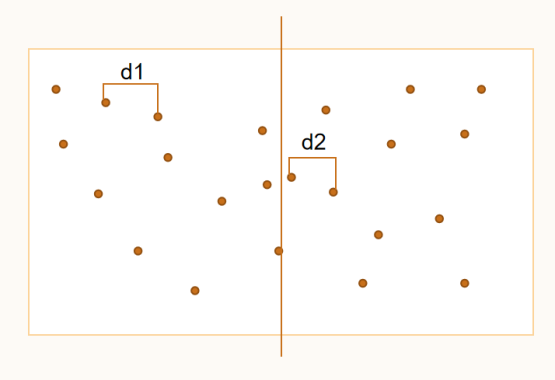
* 1. 治 -- 递归计算左右两区域的最短距离

子问题最小规模



递归调用函数，即可获取左右两区域的最短距离

* 1. 合 -- 合并左右区域，并求合并后的最短距离



问题转化为：已知左右区域各自最短距离，求合并后的最短距离。

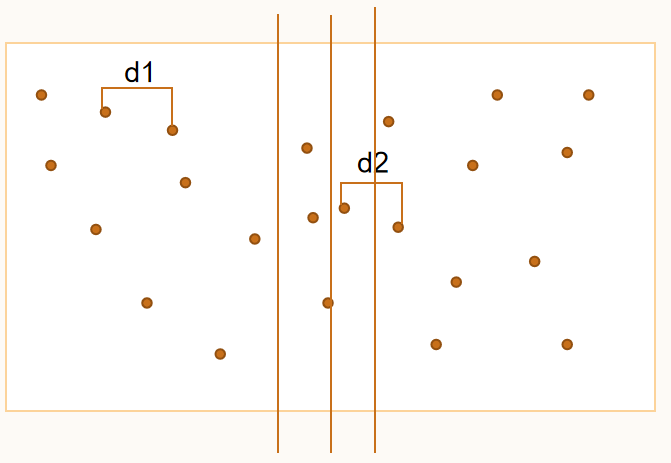
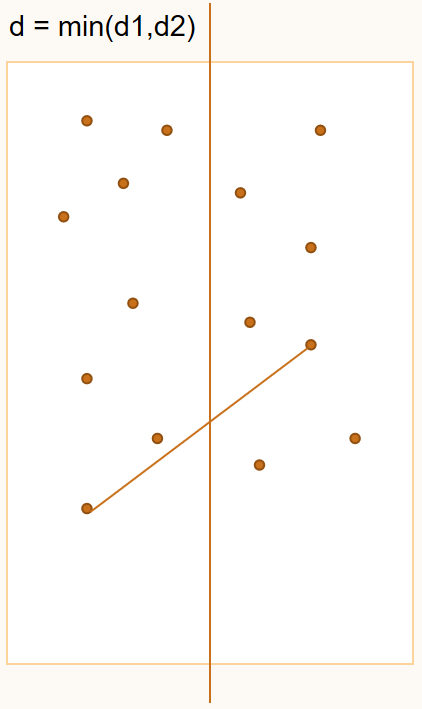
合并之后最短点对的选择一共有三种情况 ：左+右、左+左、右+右

对于左+左、右+右的情况，利用第二步中的递归调用即可获取。

所以主要问题在于，如果最短点对来自于左+右的合并操作。

解决思路：

两点必定来自于中轴线左右两侧附近，并且两点距离小于min(d1,d2)

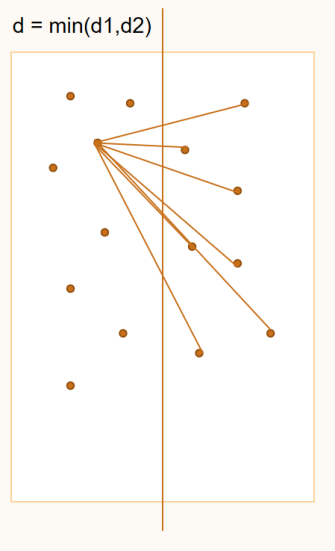
 

如左图所示，可以在中轴线附近取带状区域，其左右宽度为min(d1,d2)。在如此带状区域内，左右两点距离必定小于min(d1,d2)，极限状态为一侧宽度长度。

问题转化为：从左右区域内各取一点，与d1，d2比较，取最短距离。

解决方法有如下展示三种。

1. 分治——部分蛮力
   1. 算法原理：



遍历左右带中的所有点，比较获得最短距离。

* 1. 伪代码：

ans = min(d1,d2)

for i = 1 to left.size

for j = 1 to right.size

if ( dis(left[i], right[j]) < ans )

ans = dis(left[i], right[j])

* 1. 复杂度分析：

对于均匀分布的大型点集，预计位于该带中的点的个数是非常少的。事实上，容易论证平均只有O（√n）个点是在这个带中的。

-----《数据结构与算法分析-C语言描述》P280

查询资料可知，对于均匀分布的点集而言，带中的元素个数为√n

递推公式为：

平均情况 T(n) = 2T(n/2) + (√n)^2

总体时间复杂度O(nlogn)

对于非均匀分布的点集，最差情况带中的元素个数为n

递推公式为：

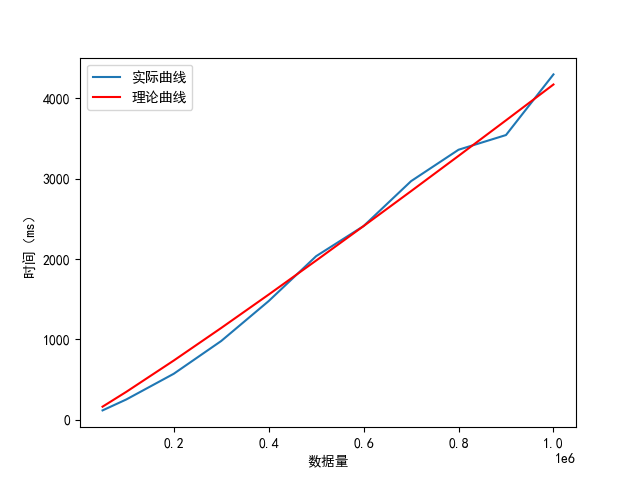
最坏情况 T(n) = 2T(n/2) + (n)^2

总体时间复杂度O(n^2)

* 1. 数据分析：

最终结果如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据量 | 50000 | 100000 | 400000 | 500000 | 800000 | 1000000 |
| 单次时间（ms） | 117 | 253 | 1478 | 2037 | 3362 | 4300 |



图像上符合O(nlogn) 曲线，并且理论值与实际值误差较小。

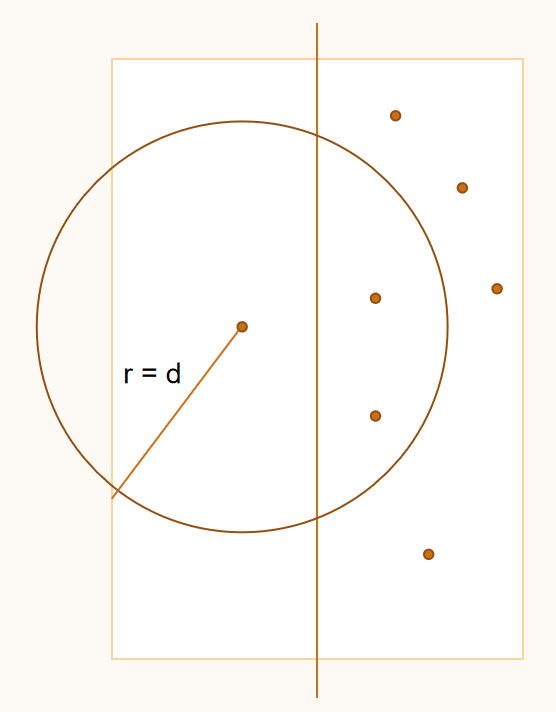
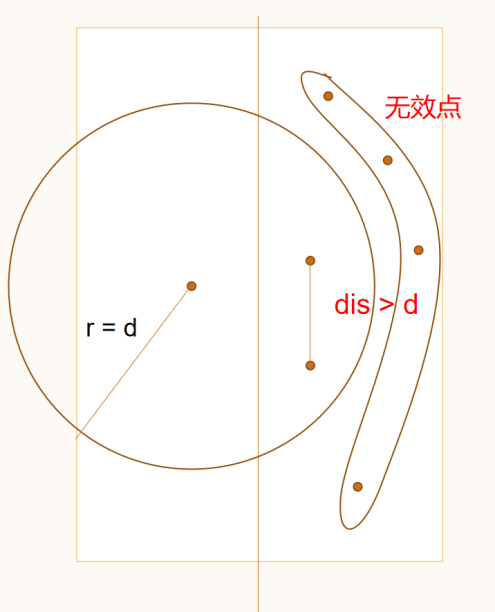
显然对于最差情况仍然为O(n^2)，需要改进。

1. 分治——多趟查询
   1. 算法原理：
      1. 遍历左带内的所有点，并与右带内所有符合条件的点进行长度比较。

右侧符合条件点集筛选过程及原理：

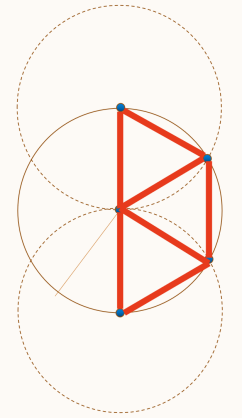
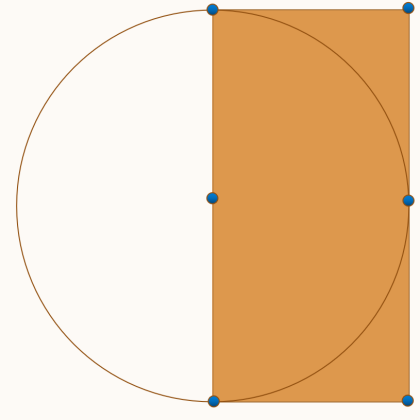
结论：右侧点位于以左侧点为中心，上下高度d，右侧宽度d的d\*2d的范围内。且右侧点的个数存在上限。

证明如下：

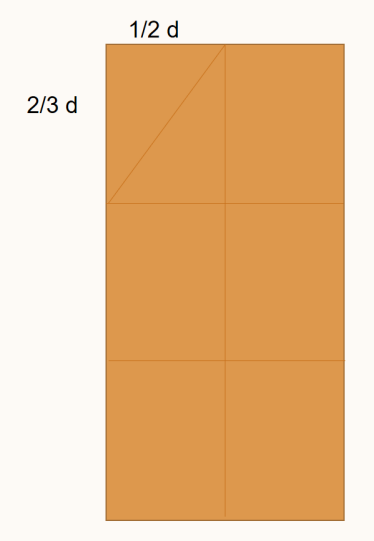
对于左侧的点，只有两点距离小于d才有合并时减小d的可能，所以右侧点必定在以左侧点为圆心、d为半径的圆内（上左图所示），其余点为无效点（上右图所示）。

又因为右侧区域内任意两点距离最小值为d2 >= d, 所以圆内两点距离大于等于半径。

因为右侧点一定位于圆心右侧，最大可覆盖区域为右半圆，又因为两点距离大于半径，所以最多可能存在5个点满足条件（上左图所示）。

在计算机中对点坐标的排序为x或y坐标排序，难以对圆形区域进行判断，所以将半圆扩展为矩形便于计算机运算。改矩形范围内最多可能存在6个点满足条件（上左图所示）。



将矩形分为6个等大的2/3d\*1/2d的小矩形，假设存在7个点，则必定有一个矩形内有两个点。

同一小矩形内两大距离最大值为对角线，即0.8333d < d 与题意不符，所以不能有7个点。而6个点的情况如上右图所示。

综上所述，右侧点位于以左侧点为中心，上下高度d，右侧宽度d的d\*2d的范围内。且右侧点的个数存在上限。

所以在查找过程中，只需要找到第一个属于左侧点对应矩形范围内的点，并之多向上查找6次，即可完成查询。

依次遍历左带中的点，并自下而上的遍历右带直到遇到第一个符合条件的点。

* 1. 伪代码：

for i = 1 to left.size

for j = 1 to right.size

if right[j] 在相应的矩形内

for k = j to j + 6 and k < right.size

ans = min(ans, dis(left[i], right[j]))

* 1. 复杂度分析：

对于均匀分布的大型点集，预计位于该带中的点的个数是非常少的。事实上，容易论证平均只有O（√n）个点是在这个带中的。

-----《数据结构与算法分析-C语言描述》P280

递推公式

平均情况 T(n) = 2T(n/2) + (√n)^2

总体时间复杂度O（nlogn)

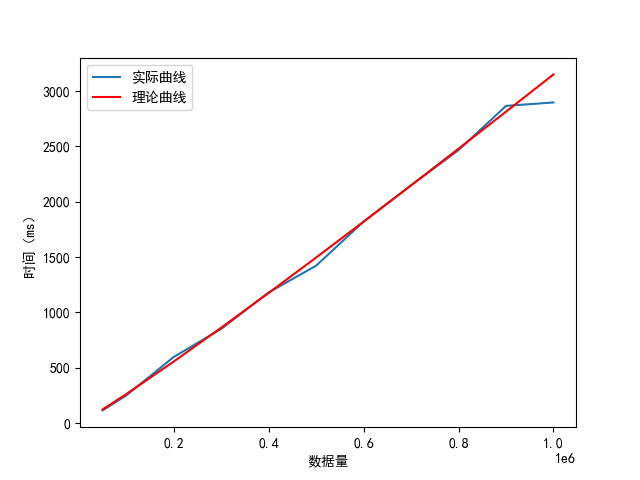
最坏情况 T(n) = 2T(n/2) + (n)^2

总体时间复杂度O（n^2)

* 1. 数据分析：

最终结果如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据量 | 50000 | 100000 | 400000 | 500000 | 800000 | 1000000 |
| 单次排序时间（ms） | 115 | 250 | 1181 | 1423 | 2467 | 2898 |



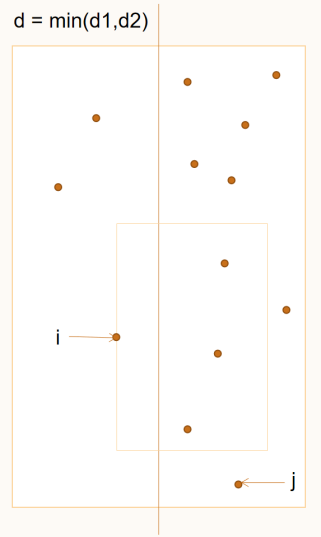
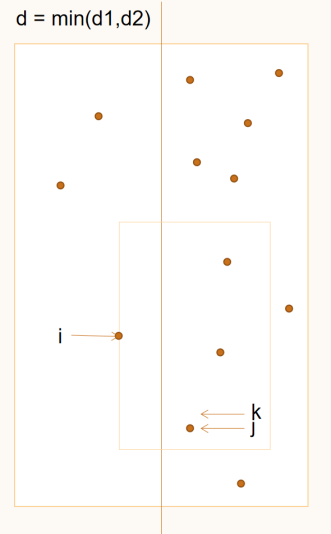
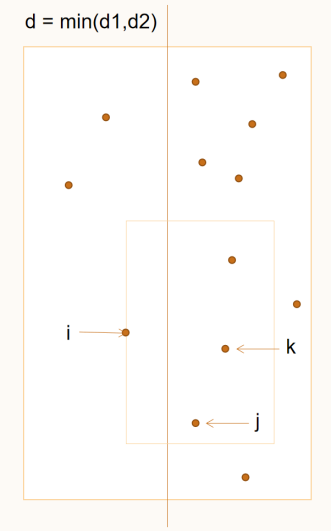
图像上符合O(nlogn) 曲线，并且理论值与实际值误差较小。

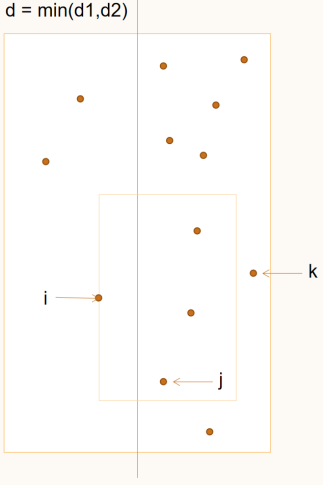
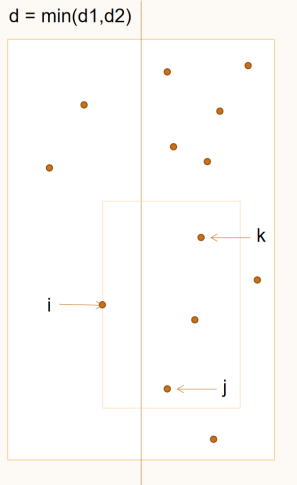
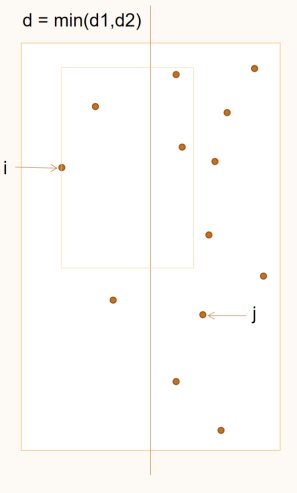
显然对于最差情况仍然为O(n^2)，需要改进。

1. 分治——一趟查询
   1. 算法原理：

此算法在上一部分的基础上进行了部分改进，达到了线性的查找效率。

方法是对于左右侧的点都自下而上进行查找。因为矩形区域是固定的，所以随着左侧点的y坐标增大，矩形的最低点也是逐渐增大，右侧符合条件的第一个点的y也会逐渐增大，并不需要返回重新查找。（可以结合伪代码查看动图中指针的变化过程。具体过程在演示中展示。）

* 1. 伪代码：

j = 0

for i = 1 to left.size

while j < right.size and right[j].y < left[i].y - d

j += 1

for k = j to j + 6 and right[k].y < left[i].y + d

ans = min(ans, dis(left[i], right[j]))

* 1. 复杂度分析：

在此过程中，只需要对左右带中的点进行依次遍历，即可找到最短距离，最多比较6n次即可，达到了线性效率。

递推公式

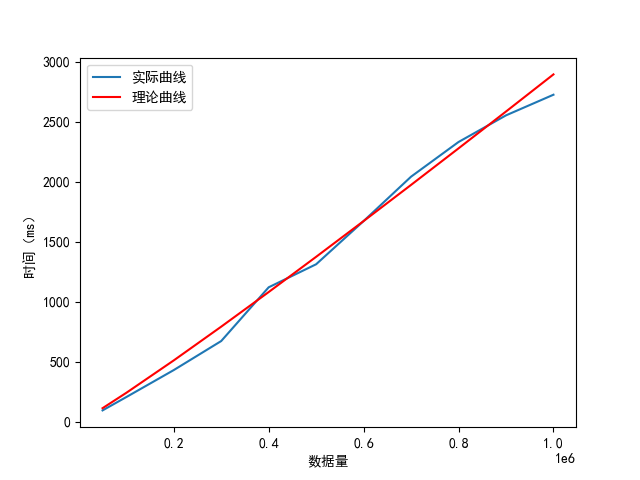
T(n) = 2T(n/2) + n

总体时间复杂度O（nlogn)

* 1. 数据分析：

最终结果如下：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 数据量 | 50000 | 100000 | 400000 | 500000 | 800000 | 1000000 |
| 单次排序时间（ms） | 95 | 206 | 1121 | 1313 | 2332 | 2726 |



图像上符合O(nlogn) 曲线，并且理论值与实际值误差较小。

1. 问题思考：

上述合并的方法是利用自下而上的遍历左右带中的点集，所以需要对带中的点以y坐标进行排序。

前提条件：左右区域内的点是依据y坐标进行排序的。

实际条件：左右区域内的点是依据x坐标进行排序的。（获取中轴线时，已排序）

所以每次递归的过程之中都需要对y进行排序。

对于均匀分布的大型点集，预计位于该带中的点的个数是非常少的。事实上，容易论证平均只有O（√n）个点是在这个带中的。

-----《数据结构与算法分析-C语言描述》P280

递推公式

平均情况 T(n) = 2T(n/2) + √n∙log√n < 2T(n/2) + n

合并效率小于O(nlogn)，又因为一开始对x排序，O(nlogn)为时间效率下限。

总体时间复杂度O(nlogn)

最坏情况 T(n) = 2T(n/2) + nlogn

总体时间复杂度O(nlognlogn)

虽然最坏情况下为O(nlognlogn)，但相较于O(n^2)有较大提升。

为了解决这一问题，引入以下方法。

解决方案：

我们将保留两个表。一个按照x坐标排序，一个按照y坐标排序，成为P和Q。Pl和Ql传递给左半部分递归调用，Pr和Qr传递给右半部分递归调用。一旦分割线已知，我们依序转到Q，把每一个元素放入相应的Ql或Qr。容易看出，Ql和Qr将自动地按照y坐标排序。当递归调用返回时，我们扫描Q表并删除其x坐标不在带内的所有的点。此时Q只含有带中的点，而这些点保证是按照它们的y坐标排序的。

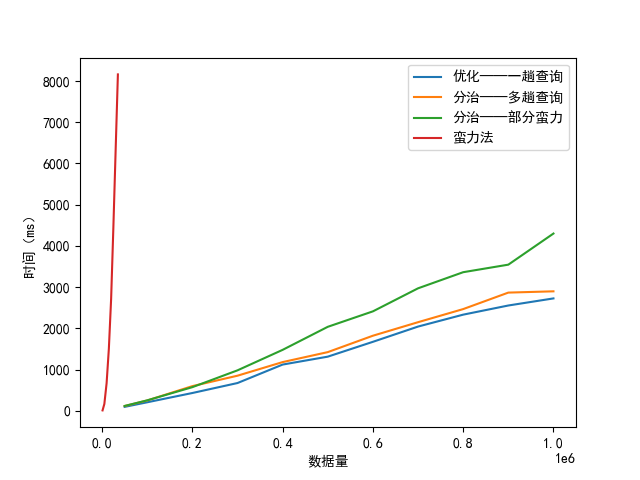
-----《数据结构与算法分析-C语言描述》P281

递推公式

T(n) = 2T(n/2) + n

总体时间复杂度O(nlogn)

1. 综合分析



显然，分治法的时间效率是要远远优于蛮力法。

每一次的优化，都对于时间效率有着小幅度的提升。

### 经验总结

经过本次实验，感受到了分治法合并效率对于整体效率有着极大的影响。一开始写代码的时候可能因为代码过于复杂导致总体效率较低，在多次修改代码细节后达到了理想目标。

另外，首先确定实验正确性也是十分重要的一件事。有些时候修改一处代码后，结果与蛮力法求解不同，这时需要首先考虑代码的正确性而非优化。

|  |
| --- |
| 指导教师批阅意见：  成绩评定：  指导教师签字：  年 月 日 |
| 备注： |

注：1、报告内的项目或内容设置，可根据实际情况加以调整和补充。

2、教师批改学生实验报告时间应在学生提交实验报告时间后10日内。