**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1 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： 刘刚**

**报告人：黄志鹏 学号：2017303008 班级：计科1班**

**实验时间： 2020年4月15日 - 2020年4月21日**

**实验报告提交时间： 2020年4月21日**

**教务部制**

**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。

1. **实验内容**

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=100,1000,10000,100000，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。

1. **实验步骤与结果**
2. **问题描述**

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

1. **算法原理描述**

利用分治法求最近点对，将时间复杂度降为nlogn。

1. 预处理：创建Point结构（包含x，y信息），随机生成规模为N的点集，根据输入点集Points中的x轴进行归并排序。

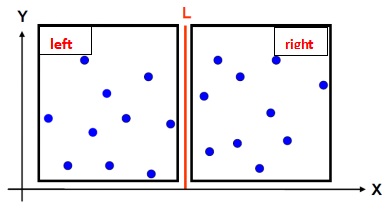
2. 当点集的点数较少时的情形

2.1 只有三个点 -> 返回三点之间的最小距离

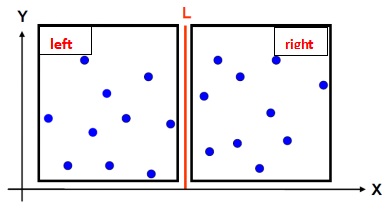
2.2 只有两个点 -> 直接返回两点间的距离

2.3 只有一个点 -> 返回无穷大

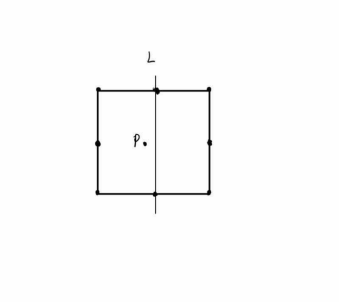
3. 点集Points的点数>3时，将平面点集分割成为大小大致相等的两个子集left和right，选取点集的中位数作为垂直线L作为分割直线。



1. 两个递归调用，分别求出left和right中的最短距离为dl和dr。
2. 取d=min(dl, dr)，在直线L两边分别扩展d，得到边界区域Y，将点集Points按Y轴排序，S是区域Y中的点按照y坐标值排序后得到的点集，S又可分为左右两个集合s\_left和s\_right。



1. 对于S中的每一点，检查它与包围它的八个点的距离，更新所获得的最近距离



d

**3. 算法实现的核心伪代码**

预处理：先将Points点集按x进行排序

double Min\_Dis(Points[], left, right)

if(right - left = 1) 返回两点间的距离

if(right - left = 2) 返回三点点间最短距离

if (right = left) 返回无穷大

int mid = ( right + left)/2

mid\_x = Points[mid].x // 以中位数作垂直线L

dis\_left = Min\_Dis(Points,left,mid) //两次递归求出左右区间最短距离

dis\_right = Min\_Dis(Points,mid+1,right)

mid\_dis = (dis\_left <dis\_right) ? dis\_left : dis\_right // 取d=min(dl, dr)

Merge\_Sort(Points, left, right, cmp\_y) //将点集Points[]按y轴排列

double temp

Point \*s =new Point[100000] //s数组存放中点x-min\_x到中点x+min\_x区间内的点

for i=0 to right

if(abs(Points[i].x - mid\_x)<=sqrt(min\_dis))

取出符合条件的点放入s数组

for i=0 to s.lenght

j=i-4,k=i+4 //j到i表示点下方的四个点，i到k表示点上方的四个点

if j<0 j=0

if k>s.lenght k=s.lenght

for j to k && j!=i

计算点i-4 到 i+4的距离，若比min\_dis更小，则更新min\_dis

delete[] s

return min\_dis

1. **算法测试结果及效率分析**

实验结果输出图（部分）如图4.1所示：

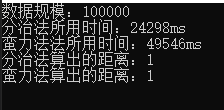


图4.1 实验结果图

分别以N=100 1000 10000 100000 为基准对各种算法进行测试。得到的运行时间表如下：

表4.1 各算法运行时间表(ms)

|  |  |  |
| --- | --- | --- |
| N\ms | 分治法 | 蛮力法 |
| 100 | 4 | 0 |
| 1000 | 60 | 6 |
| 10000 | 726 | 515 |
| 100000 | 24298 | 49546 |

由表4.2可做出效率曲线图如下：

图4.2 效率曲线图

分析：

从以上的数据可以得出，当数据规模较小时，蛮力法和分治法所需时间相差不多，甚至蛮力略优与分治法，可能时分值法具有递归和更多的赋值操作导致的；当数据规模比较大时，蛮力法的效率明显低于分治法，由时间复杂度分析可知：

分治法：T(n) = 2(T(n/2)) + 8\*n // 8表示进行了8次比较

因为a=b=2, 属于第二种情况，所以f(n) = O(n\*logn)

蛮力法：T(n) =an² + bn + c

所以时间复杂度为O(n\*n)

综上可知，当数据规模较大时，使用分治法具有很大的优势。

**四．实验心得**

通过本次实验，加深了我对分治算法的认识，同时也学习了最近点对的求解方法，分析了蛮力法与分治法求最近点对的时间复杂度。

同时，在进行算法设计中，若控制数据读写、修改的操作尽量减少，对于算法性能提升也具有一定效果，合理分析和利用算法能极大的提高效率。

|  |
| --- |
| 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。