**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： 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： 郑雨婷 学号：2021150122 班级： 高性能**

**实验时间： 2023/3/23——2023/3/30**

**实验报告提交时间： 2023/4/1**

**教务部制**

|  |  |  |
| --- | --- | --- |
| 实验目的与要求：   * + 1. 掌握分治法思想。     2. 学会最近点对问题求解方法。 | | |
| 方法、步骤：  一、问题描述  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。  二、步骤  1.生成N个随机点。  2.用蛮力法编程计算出所有点对的最短距离。  3.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。 | | |
| 实验过程及内容：   1. **预处理**   生成随机点，使用rand（）函数生成两个随机数，作为点的横纵坐标。但这样直接生成的点会出现有重复的点的情况。因此，在生成随机点的时候要排除掉重复的，遍历一遍已经生成的点，若发现有横纵坐标均相同的点，就重新生成。因为生成随机点是一个双重循环，所以在大规模时生成点的运行时间较长，但这不影响我们测试求最近点对的时间。   1. **蛮力法**   **1.1算法原理**  设置初始结果为INF无穷大，依次遍历枚举每两个点之间的距离，当找到小于当前最小距离的，更新最小距离。    图1 蛮力法原理图示  **1.2 伪代码实现**  min=99999;  for i=0 to n-2  for j=i+1 to n-1  dis=sqrt((A[i].x-A[j].x)^2+(A[i].y-A[j].y)^2)  if(dis<min) min=dis  **1.3 算法实际执行时间**  蛮力法各数据规模下运行平均时间如下表1所示：    表1 暴力法在不同规模下实测运行时间  **1.4 算法效率分析**  蛮力法是一个双重循环，共需要计算次距离，所以它的时间复杂度为。    表2 蛮力法理论值于实际值的对比  经过实际运行，我选取100000个数据作为基准，通过之间的比例关系，计算得到各实验数据中算法执行时间的理论值。通过程序执行，求得蛮力法各数据规模理论值与误差如上表2。  由上方图表可以发现，蛮力法的理论曲线和实际曲线几乎重合，说明蛮力法很稳定，并且符合x的平方的趋势。分析其原因，因为这是最简单粗暴的方法，没有进行任何优化，无论随机点重合与否，点离散还是聚集得较近，它计算距离的执行次数不会有任何变化，故整体拟合效果非常好，同时也证明蛮力法的时间复杂度为的结论正确。    图 2 蛮力法不同数据规模下的理论值和实验值   1. **分治法**   **2.1 算法原理**  我们知道，将一个问题分成小问题分而治之是提高效率常用的一种思想。在求最近点对的这个具体问题中，分治法的总体思想是：   1. 分为左侧、右侧两个区域。 2. 求出左侧的最小距离dl和右侧的最小距离dr。 3. 求出跨域的最小距离dd。 4. 选出三者中最小的d。     图3 分治法原理图示  有了总体的算法思想，再来看分治算法实现细节：  (1)如何均匀平分为左右两侧？  将数组对x升序排序，以该数组下标中值分割。下标小于中值下标的点为左侧的点，下标大于中值的点为右侧的点。    (2)如何求左侧的最小距离dl和右侧的最小距离dr？  若区域内有三个及以上个点时，继续分割，在最小问题时求解。  当分割至区域内只剩两个点时，最小距离即为两点间的距离；区域内只有一个点时，其距离设为无穷大INF。    (3)如何求跨界区域的最小距离dd？    从中线向左右两侧扩展d，其中d=min(dl,dr)，得到区域Y。若有跨域的小于d 的距离，一定在Y区域中。求Y区域的方法是遍历所有点，与中点midp的横坐标相差小于等于d的就放入Y区域。  (4)如何对Y进行排序？  我们已经得到了区域Y，下一步就是要找到最小距离，在找最小距离时需要遍历Y中的所有点，此时需要对y坐标进行升序排序。直接用sort()函数，时间复杂度为O(nlog2n)。  想要得到线性效率的话，可以在最小问题（两个点）时顺便对y排好序，这样左右两侧就都为y升序的了，这时候再用inplace\_merge()函数即可实现线性效率O(n)。因此，我们要改变之前的思路，不是先找出Y区域再排序，而是先对y排序后再找出Y区域。最小规模两点时的伪代码如下：  if(left+1==right)  if(A[left].y>A[right].y) swap(A[left],A[right])  return dis(A[left],A[right])  合并了之后往左右扩展d得到Y区域依旧是遍历所有点，时间复杂度为O（n）。  (5)如何找到Y中的最小距离？  如果继续用暴力法对Y中每个点进行检测的话，并不能降低时间复杂度。  细想可以发现，每个点并不需要把其他剩余所有点都检测一遍，因为Y区域的点之间的最小距离是d。我们只需对每个点测试有限个点即可。  因为我们从下往上遍历，对每个点只需测试其上方的有限个点即可。对点p，可能出现的使得距离小于d的点一定在这个Y区域中的d\*2d的矩形中。    因为Y区域的点之间的最小距离是d，所以这个矩形中最多有6个点。证明如下：将整个矩形分为8块,每一块的长为1/2d,宽为1/2d，经计算得对角线距离为。是小于d的，又因为最小左右两侧两点间的点距离最小为d,所以每一块中最多只能有一个点，这样一整个矩形中最多只能有8个点。但是当8个点的情况时中间有两点重合，我们已经在生成点集时预处理了没有重复的的，因此最多有6个点    因此，每个点只需要向上寻找6个点即可。  **2.2 伪代码实现**  double divide（A,left,right）  if(left>=right) return INF  if(left+1==right) 对y升序排序，return dis(A[left],A[right])  dl=divide(A,left,mid)  dr=divide(A,mid+1,right)  d=min(dl,dr)  inplace\_merge(A+left,A+mid+1,A+right+1)  得出Y区域，得到Y中有m个点  Y区域中m个点每个点都向上找六个点  **2.3 算法实际执行时间**  各数据规模下运行平均时间如下表3所示：    表3 分治法在不同规模下实测运行时间  **2.4 算法效率分析**  分治法的时间复杂度主要是看合并代价，在左右两侧合成一个对y有序的集合时，效率为O(n)、在向左右扩展出Y区域是效率是O(n)，在Y区域中每个点都寻找6个点时效率为O（m）(m为Y区域中点的个数，小于n)。总合并代价O(n)+ O(n)+ O(m)，依旧为O（n）。每次递归操作都是均等分，所以其递归树为 logn 层，所以总的时间复杂度即为：。  分治法时间复杂度平均值为，经过实际运行，我选取排序100000个数据作为基准，通过之间的比例关系，计算得到各实验数据中算法执行时间的理论值。通过程序执行，求得分治法各数据规模理论值与误差如下表4。    表4 分治法理论值于实际值的对比    图 4 分治法不同数据规模下的理论值和实验值  图像上符合 曲线，并且理论值与实际值误差较小。验证了时间复杂度为的正确性。上图实际运行时间的数据是50组的平均值，因此拟合效果较好。 |
| 数据处理分析：   1. 蛮力法与分治法比对及误差分析     大规模时分治法的效率远远高于蛮力法，与所求出的时间复杂度一致。    小规模时蛮力法的效率高于分治法，与分析出来的时间复杂度不符，这是因为，在小规模时，分治法要开辟内存空间并且通过递归栈实现递归条用的时间都不可忽略。 |

|  |
| --- |
| 实验结论：  在这次实验中，我主要研究了如何解决最近点对问题。蛮力法和分治法这两种方法都有自己的优缺点，蛮力法简单直接，在小规模数据下效率高于分治法，但是在大规模数据时效率太低。分治法在设计上有一定的难度，在大规模数据时的效率远远高于蛮力法，有绝对的优势，但是在小规模数据时因为要开辟内存空间并且通过递归栈实现递归，导致实际运行实际大于蛮力法。 |
| 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。