**深 圳 大 学 实 验 报 告**

**课程名称： 算法设计与分析**

**实验名称： 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（创新班）**

**报告人： 李文俊 学号： 2023150001 班级： 高性能班**

**同组人： 无**

**指导教师： 杨烜**

**实验时间： 2025/3/28——2025/4/11**

**实验报告提交时间： 2025/4/11**

**教务处制**

### 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。

### 二、问题描述

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**、**实验分析**

0.生成随机数

**0.1 算法描述：**通过随机数算子均匀的生成n个均匀分布的点

**0.2 实现细节：**

1.初始化硬件熵源作为随机种子

2.创建梅森旋转算法随机数生成器

3.定义[0.0, 100000)区间的均匀实数分布

4.生成n个随机点：

a. 每次迭代生成两个独立随机数作为x,y坐标

b. 将坐标对存储为二维点

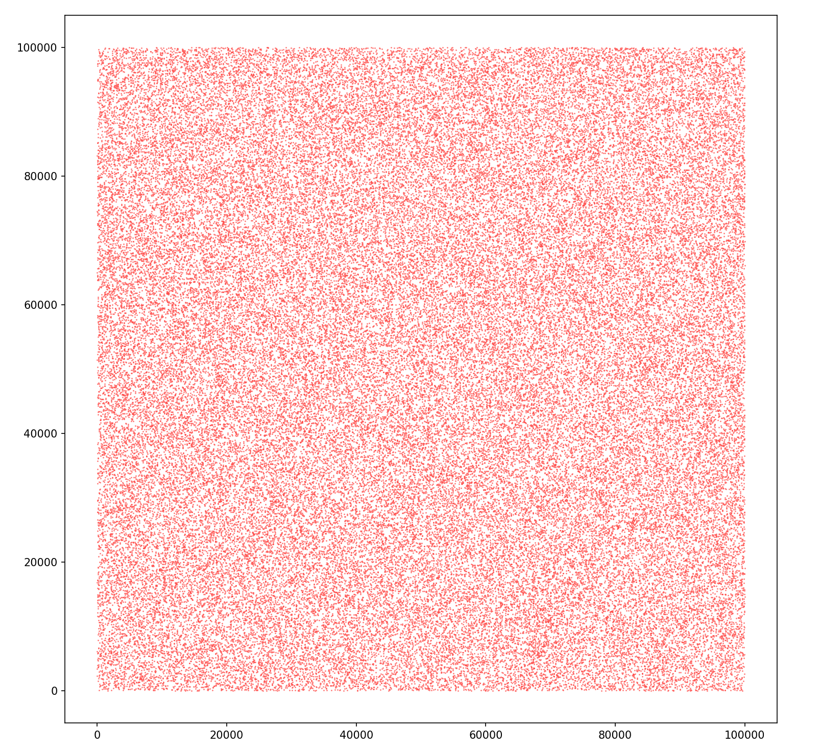
5.返回生成的点集合

文本, 信件

AI 生成的内容可能不正确。

图a 均匀数据生成

**生成10w个点效果如图b.**



图b

1.蛮力法

1.1原理描述

核心思想是**穷举所有可能的候选解**，并通过逐一验证找到符合条件的最优解。在最近点对问题中，蛮力法通过计算所有点对之间的距离，找到其中最小的距离。

电脑萤幕画面

AI 生成的内容可能不正确。

图1 蛮力法图示

**实现细节：**

**1.初始化最小距离**：将最小距离 dmin​ 设为无穷大。

**2.双重循环遍历点对：**外层循环遍历每个点 Pi，内层循环遍历 Pi之后的所有点 Pj（避免重复计算）。

**3.计算距离：**对每对点 (Pi,Pj)，计算其平方距离 distance=sqrt( (xi−xj)^2+(yi−yj)^2 )（避免开平方操作以节省计算时间）。

**4.更新最小值**：若distance<dmin ​，则更新 dmin=distance;

**5.返回结果**：最终返回 dmin作为实际最小距离。

1.2 核心伪代码

**文本, 信件

AI 生成的内容可能不正确。**

图2 蛮力法伪代码

1.3 复杂度分析

每一趟都需要两层循环遍历数组并计算来找到最小元素，需要比较的点数为

所以，**最好最坏和平均时间复杂度都是。**

1.4 效率测试

使用随机数生成，生成了十万到一百万的数据集，每个数据量都进行了10次测试并取平均值。

以输入规模为10万的数据运行时间为基准点，计算输入规模为其他值的理论运行时间，

结果如表1所示。

表1 蛮力法运行时间表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **数据规模** | **100000** | **200000** | **300000** | **400000** | **500000** | **600000** | **700000** | **800000** | **900000** | **1000000** |
| **实际时间s** | **29.346** | **113.75** | **258.791** | **455.885** | **731.772** | **1051.5** | **1478.39** | **1960.44** | **2496.65** | **2989.24** |
| **理论时间s** | **29.346** | **117.384** | **264.114** | **469.536** | **733.65** | **1056.456** | **1437.954** | **1878.144** | **2377.026** | **2934.6** |
| **相对误差%** | **0.0000** | **3.0958** | **2.0154** | **2.9073** | **0.2560** | **0.4691** | **2.8121** | **4.3818** | **5.0325** | **1.8619** |

画出不同规模数据的理论运行时间曲线和实测效率曲线，如图3所示。



图3 蛮力法曲线

理论运行时间曲线和实测效率曲线几乎重合，误差在合理范围内，符合O()曲线，经验分析和理论分析一致。

2.分治法

2.1原理描述

**1.分（Divide）​**：

**均匀分割**：首先将所有点按照x坐标进行排序，找到中间位置的x坐标值作为分割线L。将点集划分为两个数量大致相等的子集，左半部分位于L左侧，右半部分位于L右侧。若点数为奇数，允许左右子集相差1个点以确保均匀性。

**目的**：将问题分解为规模更小的子问题，便于递归处理。

​ **2.治（Conquer）​：**

**递归求解**：对左右两个子集分别递归调用最近点对算法，得到左子集的最小距离dL和右子集的最小距离dR。

**终止条件**：**当子集内点数≤3时，直接使用暴力法计算所有点对的距离**，返回最小值。此时时间复杂度为O(1)或O(3²)=O(9)。

图表, 箱线图

AI 生成的内容可能不正确。

图4.1结束标识

**3.合（Combine）​：**

**确定候选距离**：取d = min(dL, dR)，作为当前左右子集的最近距离。

**中间区域处理**：通过**不同方法**找到跨越中线的最小距离D

**4.合并结果**：最终合并后的结果为min(d, D)，即跨越分界线的最近点对可能更新全局最小值。

简述：

​ 分：按x坐标排序后取中点，将点集均匀划分为左右两部分。

​ 治：递归计算左右子集的最近距离dL和dR。

​ 合：取d = min(dL, dR)，在分界线L两侧扩展d形成带状区域，检查区域内点对的最近距离D。

​ 结果：返回min(d, D)作为全局最近距离。

图表, 散点图

AI 生成的内容可能不正确。 图表, 散点图

AI 生成的内容可能不正确。

图表, 散点图

AI 生成的内容可能不正确。

图4.2 分治法图示

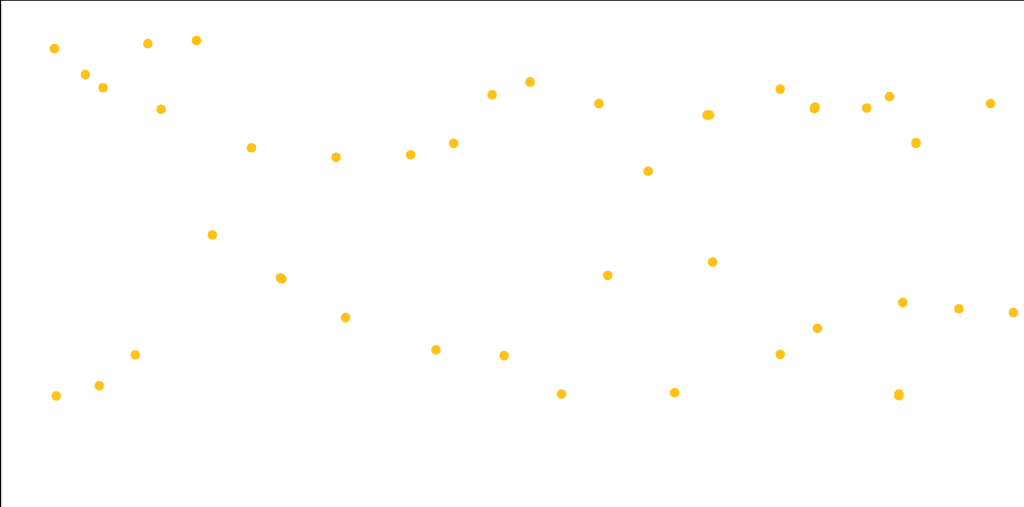


图4.3 分治法可视化

2.2核心伪代码

**实现细节：**

**1.​递归终止条件**

**​ 基础情形：**当点集规模 ≤ 3时，直接暴力枚举所有点对计算最小距离

**​ 数学保证：**3个点构成的平面中，暴力法只需3次距离计算即可找到最近点对

**​ 2.空间分割策略**

**​ 中点选取：**取点集中间位置的x坐标作为垂直分割线（中位点保证递归树平衡）

**​ 子问题划分：**将点集按x坐标分为左右两部分（需预处理x排序）

**​ 3.递归求解子问题**

**​ 左半区计算：**对分割线左侧点集递归调用算法，得到左半区最小距离d\_L

**​ 右半区计算：**对分割线右侧点集递归调用算法，得到右半区最小距离d\_R

**​ 当前最小值：取d = min(d\_L, d\_R)**作为左右区域的临时最小距离

**​ 4.跨区域候选点筛选**

**​ 带状区域定义：**收集距离分割线x坐标不超过d的所有点（即|x-L| ≤ d）

**​ 几何意义：**仅该区域内的点可能形成比当前d更小的跨区域点对（图4.2）

**5.合并方法：**下面三种算法得到最小距离min\_d

文本, 日程表

AI 生成的内容可能不正确。

图5

2.3 时间复杂度分析

T(n) = 2T(n/2) + f(n), **时间复杂度取决于合并代价**

通过对不断优化，最后达到，得到。

3.分治法-蛮力合并

3.1原理描述

将中间点分成左右两个部分，左边所有点分别对右边所有点计算距离得到中间最小距离D，最后和d比较得到最小距离结果，如图6所示。

图表, 散点图

AI 生成的内容可能不正确。

图6 蛮力合并图示

**实现细节： 合并过程使用蛮力法**

3.2 核心伪代码

文本, 信件

AI 生成的内容可能不正确。

图7 蛮力合并伪代码

3.3 复杂度分析

平均情况（均匀分布）：只有O（√𝑛）个点是在带中 ， T(n) = 2T(n/2) +

平均时间复杂度O()

最坏情况：合并代价为O() ，T(n) = 2T(n/2) + n2

最坏时间复杂度： O()

3.4 效率测试

使用随机数生成，生成了十万到一百万的数据集，每个数据量都进行了10次测试并取平均值。

以输入规模为10万的数据运行时间为基准点，计算输入规模为其他值的理论运行时间，

结果如表2所示。

表2 蛮力合并运行时间表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **数据规模** | **100000** | **200000** | **300000** | **400000** | **500000** | **600000** | **700000** | **800000** | **900000** | **1000000** |
| **实际时间s** | **273.864** | **564.97** | **887.953** | **1225.97** | **1526.51** | **1920.07** | **2163.3** | **2535.25** | **2749.96** | **3156.2** |
| **理论时间s** | **273.864** | **580.7045** | **899.99** | **1227.362** | **1560.743** | **1898.913** | **2241.067** | **2586.63** | **2935.175** | **3286.368** |
| **相对误差%** | **0.00** | **2.71** | **1.34** | **0.11** | **2.19** | **1.11** | **3.47** | **1.99** | **6.31** | **3.96** |

画出不同规模数据的理论运行时间曲线和实测效率曲线，如图8所示。



图8 蛮力合并曲线

理论运行时间曲线和实测效率曲线几乎重合，误差在合理范围内，符合O**(**)曲线，经验分析和理论分析一致。

4.分治法-多次排序

4.1原理描述和实现细节

**①在[L-d,L+d]的范围内，对左边的一个点可以将其搜索的范围缩小为右边的六个点：**

**1.空间划分原理**：当左右两个子集的最小距离d确定后，合并阶段只需检查中线两侧各d宽度的带状区域（总宽度2d）

**2.​栅格划分策略**：将该带状区域划分为多个 2d/3 × d/2 的小方格，因为对角线距离为5d/6,所以每个方格中至多包含1个点（否则该方格内会出现比d更小的距离，与d的最小性矛盾）

**3.鸽巢原理应用**：对于任意左半区点，其右侧的有效区域为 d × 2d 矩形。将该矩形纵向划分为6个 d2/3 × d/2 的子区域，因此最多存在6个右半区点可能产生更小距离

图示

AI 生成的内容可能不正确。

图9 缩小范围图示

**②左边的每个点对这六个点进行搜索：**

​ 1.**预排序优化**：将右半区候选点按y坐标排序（cmp2比较函数），使纵坐标有序可快速定位

**如果不对y排序，就要遍历右侧所有点，效率和蛮力合并本质相同**

**2.滑动窗口筛选**：

使用while循环**跳过纵坐标差距≥d的点**（right\_points[j].y - left\_points[i].y >= d）

对剩余点**最多检查6个相邻点**（k < j+6），利用排序后的局部性原理

图表

AI 生成的内容可能不正确。

图10 多次排序查找演示

4.2 核心伪代码

文本, 信件

AI 生成的内容可能不正确。

图11多次排序伪代码

4.3 复杂度分析

对右侧点排序需要O(m)，查找到最低点然后计算6个点O()，遍历左边元素O()

**平均情况（均匀分布下m=） T(n) = 2T(n/2) + ，平均时间复杂度O**

**最坏情况 T(n) = 2T(n/2) +，最坏时间复杂度O**

4.4 效率测试

使用随机数生成，生成了十万到一百万的数据集，每个数据量都进行了10次测试并取平均值。

以输入规模为10万的数据运行时间为基准点，计算输入规模为其他值的理论运行时间，

结果如表4所示。

表4 多次排序运行时间表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **数据规模** | **100000** | **200000** | **300000** | **400000** | **500000** | **600000** | **700000** | **800000** | **900000** | **1000000** |
| **实际时间s** | **207.716** | **461.397** | **692.343** | **881.949** | **1174.07** | **1392.73** | **1595.3** | **1811.99** | **2050.59** | **2365.51** |
| **理论时间s** | **207.716** | **440.4435** | **682.611** | **930.91** | **1183.767** | **1440.257** | **1699.769** | **1961.866** | **2226.225** | **2492.592** |
| **相对误差%** | **0.00** | **4.76** | **1.43** | **5.26** | **0.82** | **3.30** | **6.15** | **7.64** | **7.89** | **5.10** |

画出不同规模数据的理论运行时间曲线和实测效率曲线，如图12所示。



图12 多次排序曲线

理论运行时间曲线和实测效率曲线几乎重合，误差在合理范围内，符合O**(**)曲线，经验分析和理论分析一致。

5.分治法-一次排序

5.1原理描述

**核心创新点：单次预处理排序**

​**1.预排序策略：预处理时仅需一次全排序**（时间复杂度O(n log n)）

**1）按x坐标排序生成Px数组**

**2）按y坐标排序生成Py数组**

​**2.数据结构维护**：

1）Px数组用于均匀平分左右两部分

2）Py数组用于在递归过程中通过数组分割自然保持y有序性，​无需再对分割出来的数组显式排序，因此在对数据进行筛选时，筛选的对象是Py数组，Px数组在整个函数中不需要改变。

​3.**几何优化原理**：对Py中候选点按y排序后，任意点的有效候选点最多6个

5.2 核心伪代码

文本, 信件

AI 生成的内容可能不正确。

图13一次排序伪代码

5.3 复杂度分析

**主函数：预处理排序：O(n log n) 单次全排序（主要成本）**

**分治法函数：**

**递归分割：O(n) 数组分割操作（线性时间）**

**递归合并：O(n) 带状区域筛选**（线性遍历）**+ 候选点检查**（常数因子7）

​ **总时间复杂度 ​O(n log n)** 由主定理 **T(n) = 2T(n/2) + O(n)**得到

5.4 效率测试

使用随机数生成，生成了十万到一百万的数据集，每个数据量都进行了20次测试并取平均值。

以输入规模为10万的数据运行时间为基准点，计算输入规模为其他值的理论运行时间，

结果如表5所示。

表5 一次排序运行时间表

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **数据规模** | **100000** | **200000** | **300000** | **400000** | **500000** | **600000** | **700000** | **800000** | **900000** | **1000000** |
| **实际时间s** | **181.40** | **383.29** | **581.71** | **760.76** | **967.21** | **1201.29** | **1349.65** | **1536.50** | **1737.70** | **1985.24** |
| **理论时间s** | **181.40** | **384.64** | **596.13** | **812.97** | **1033.79** | **1257.79** | **1484.42** | **1713.31** | **1944.18** | **2176.80** |
| **相对误差%** | **0.00** | **0.35** | **2.42** | **6.42** | **6.44** | **4.49** | **9.08** | **10.32** | **10.62** | **8.80** |

画出不同规模数据的理论运行时间曲线和实测效率曲线，如图14所示。



图14 一次排序曲线

理论运行时间曲线和实测效率曲线几乎重合，误差在合理范围内，符合O**(**)曲线，经验分析和理论分析一致。

**四．实验结果和分析**

1. 蛮力法和分治法对比

**运行时间：蛮力法>>分治法**，如图15所示



图15

分析：

**分治法通过递归分治将问题规模指数级缩小，而蛮力法需检查所有可能的点对**

**大规模数据下，分治法时间优势明显**

**数据规模极小（n≤100），蛮力法更快，分治法的递归开销超过计算收益**

2.分治法不同合并算法对比



图16

分析：

**分治法不同方法的平均时间复杂度都是O(𝑛 log𝑛)，通过优化合并方法**

**1.合并阶段剪枝——减少候选点比较次数**

**2.预处理优化———减少运行时排序成本**

**将合并效率降低到线性**

**五．经验总结**

1. 时间复杂度对比验证

蛮力法的时间复杂度为O(n²)，在n>10⁴量级时效率急剧下降，验证了平方级复杂度对计算资源的消耗不可忽视。

分治算法通过递归分割将时间复杂度降至O(n log n)，其对数线性特性在n=10⁶时仍保持可行性，体现了分治策略对问题规模的分解有效性。

2. 分治优化的核心突破点

**区域划分策略**：将点集按x坐标分割为左右子区间，确保递归过程中子问题的空间隔离性，使合并阶段的跨区检测范围缩小至δ宽度的垂直带区域。

**合并阶段优化**：将候选点集按y坐标排序后，利用几何特性证明每个点只需比较后续6个相邻点，将合并操作的时间复杂度压缩至O(n)，突破合并瓶颈。

**预排序预处理**：在主函数初始阶段完成全局坐标排序，使递归过程直接继承有序结构，避免重复排序产生的O(n log n)开销，整体保持算法复杂度纯度。

3. 算法鲁棒性表现

在均匀分布点集中，分治法通过空间分割能快速收敛候选区域；在聚类分布场景下，递归分割仍能有效隔离密集区域，较蛮力法展现更稳定的性能表现。

对重复点、共线点等边界情况，通过预排序机制和最小距离动态更新策略保证计算正确性，体现分治结构对异常数据的包容性。

4. 实际工程启示

**空间换时间策略**：维护两个坐标轴的有序副本，以约2n的存储代价换取O(1)时间的中位点选取和O(n)时间的区域划分。

**剪枝优化原则**：在合并阶段引入早期终止条件，当检测到比当前δ更小的距离时立即更新阈值，减少无效计算。

​5. 算法正确性验证

​**计数器注入技术**：在分治合并循环内嵌入比较次数计数器，通过大规模随机测试验证其统计分布是否符合“每个点最多比较6个点”的理论上限，从而确认算法实现的正确性。

​**复杂度可视化**：在小规模数据下模拟算法的实际运行过程，可以验证算法是否能达到期望的时间复杂度和正确性。

|  |
| --- |
| 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。