**深 圳 大 学 实 验 报 告**

**课程名称： 算法设计与实现**

**实验项目名称： 分治法求最近点对**

**学院： 计算机与软件学院**

**专业： 计算机科学与技术**

**指导教师： 杨烜老师**

**报告人： 李若龙 学号：2018171028 班级： 计科02**

**实验时间： 2020/4/4**

**实验报告提交时间：**

**教务处制**

1. 问题描述：
2. 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。

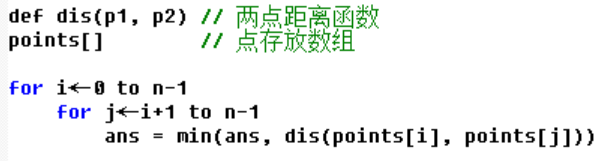
5. 如果能将算法执行过程利用图形界面输出，可获加分。

1. 求解问题的算法原理描述

暴力法：

暴力法的思路相对简单，即枚举所有可能的配对情况，然后一一比对，找到距离最小的点，一共有 Cn2 种组合，也就是 n\*(n-1)/2种，总的时间复杂度是O(n^2)

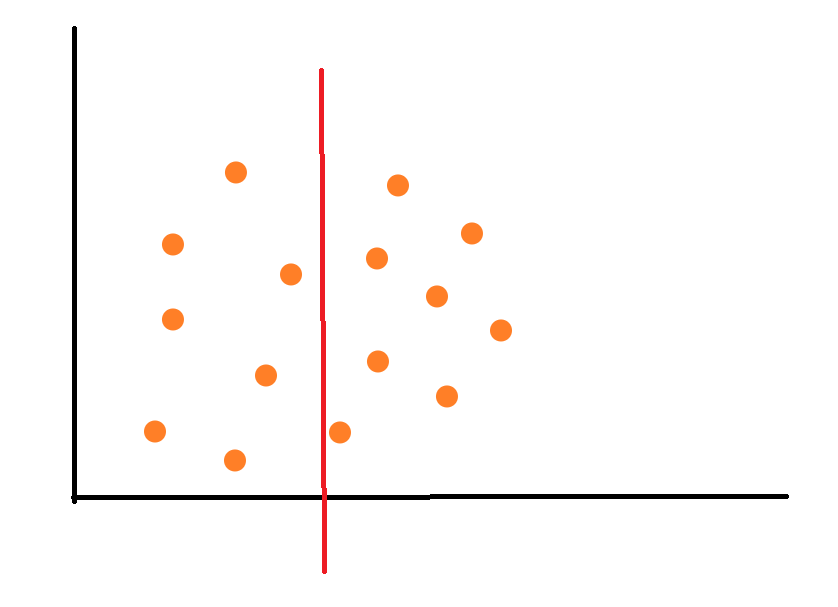
暴力法伪代码：



分治法

**问题分割策略：排序，取中位数**

如果像快速排序，分割的策略与数据有关，那么算法便会退化，不能达到稳定的O(nlogn)，但是如果效仿归并排序，每次选取中点进行分割，那么可以使得递归树的深度稳定的控制在log（n），这里对点做预处理，**按照x升序，x相同则y升序排序**，这样我们每次选取中间下标的点，都能尽量地将点分割成几何上的两半



**子问题存在分析：**

问题是选取两个点，p1，p2，使得他们距离最短，子问题存在于以下空间

1. p1 p2 同时位于左边点集
2. p1 p2 同时位于右边点集
3. p1 p2 一个在左侧一个在右侧

对于子问题1和2，我们可以通过递归直接解决，难点就在于解决子问题3，即两点在两边的情况

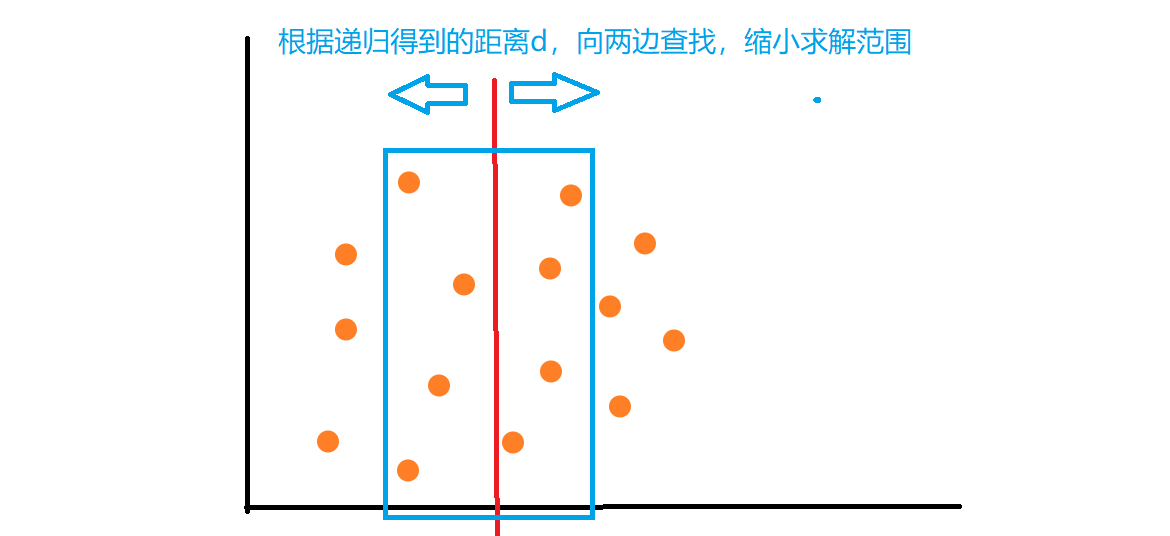
**子问题3求解 缩小空间：**

想起来归并是要【合并】的，如何有效利用递归得到的答案呢？

如果要一一配对找两边点求距离，那么合并效率还是O(n^2)

递归得到两边点子集的最近点对距离，取最小值d，我们可以**借此距离d来缩小我们查找的空间**

从中间点向外扩散，如果点与中间点的x左边之差，大于d，那么这个点及其往后的点，都不可能是最优的答案，缩小了查找空间

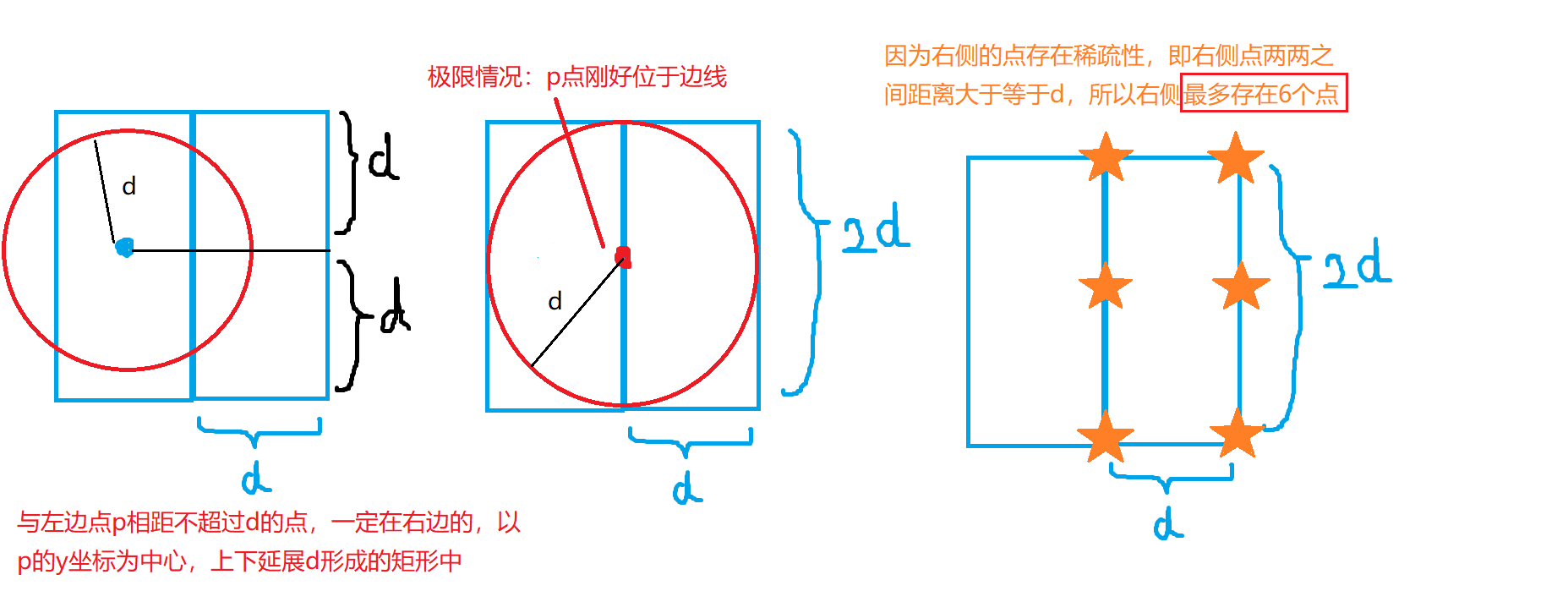


缩小查找空间，其实是不稳定的，可能查找空间和原来一样，所以我们不能暴力枚举查找空间内的点，下面引入一个结论，来帮助我们快速查找

**结论：对于左边区域的每一点p，要想在右边区域找到一点p’使得 pp’距离<=d，这样的点p’必定存在于【以p的y坐标为中心，上下延展d形成的d\*2d的子矩形中】**

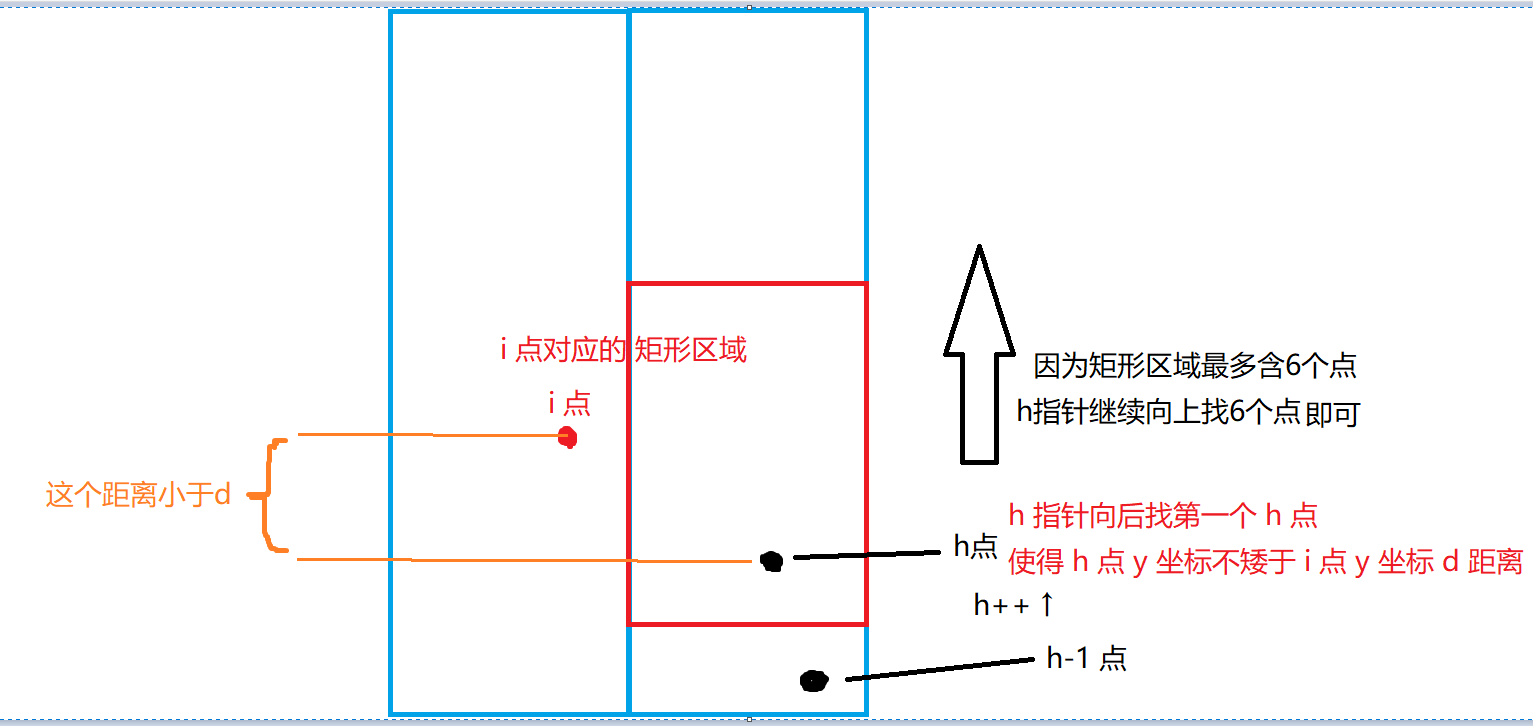
证明：假设极限情况，p在左右边线上，以p为中心半径为d画圆，半圆区域内，都是距p的距离小于d的，半圆是矩形的真子集，故目标点一定存在于矩形中

而且因为右侧的点具有稀疏性，即**两两之间距离大于等于d**， d\*2d的矩形区域，矩形区域内最多存在6个点（如果圆形区域则是四个）

****

我们遍历左边区域的所有点，划分出d\*2d的矩形区域，然后检查6个点，更新答案即可，可是如何用短时间找到d\*2d的矩形区域？

**对左右的子区域的所有点按照y升序排序**（左右分开排序，不是混合在一起），在指针i升序遍历左边的所有点的时候，设置指针h在右边的点中向后查找，找到第一个不矮于i点y坐标d高度的点h（即h和i点高度差小于d），从h往后找6个点即可



因为左右点集都是升序，这表明，i++之后，下一个i点的d\*2d矩形的下边界一定会提升，而h指针随着做出更新即可，因为i点矩形的下边界是递增的，h指针不走回头路，h指针最多从0增加到右边区域点的个数，最多n/2次，所以遍历左边点集，h迭代的次数，**均摊到每一次i迭代**，就是均摊一次，O(1)，而查找矩形区域总共是6个点，也是常数时间可以完成，所以总的复杂度仍然是O(n)

分治法的改进：

因为每次递归之后，还要对左右按d划分的区域排序，总复杂度是O(2\*n/2log(n/2))， 即 O(nlog(n/2))，合并代价，不是线性效率

效仿归并排序，每次二分递归之后，我们对数组归并，因为递归排好了左右子区间，这样归并后，数组就是对y有序的了，不用消耗额外的时间再排序了，可以使合并子问题的代价变为O(n)级别

测试流程

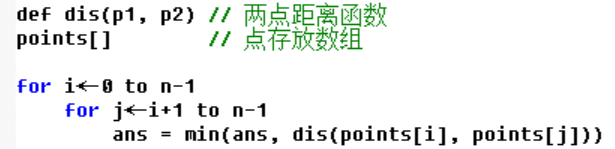
测试之前做了270万（2715633）次验证，生成随机数据，比对两种分治法与暴力法的结果（暴力法保证正确），结果三种方法结果全部相等，排除算法bug导致时间误差

每一次测试随机生成50组规模为batch的数据，利用哈希set确保数据中没有重复的点，点坐标的范围是 [0，sqrt(batch)\*3] 保证随着规模的变化，数据的密集程度不会有太大的改变，排除点稀疏程度导致的时间误差

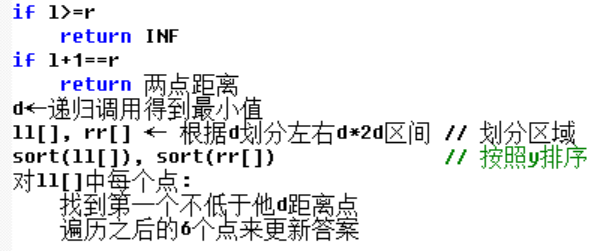
生成的每组数据被复制若干份，保证每次每个算法用到的数据是同一批，排除数据随机生成导致的时间误差

算法核心伪代码

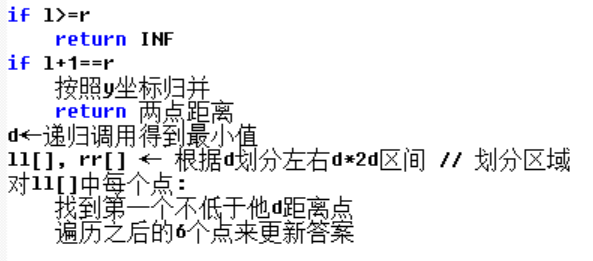
暴力法



单纯分治法



递归+归并法



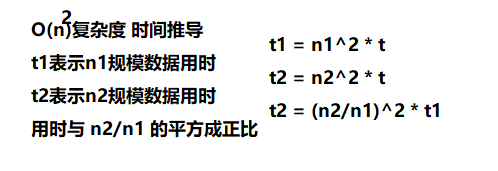
算法测试结果及效率分析

暴力法：显而易见时间复杂度为O(n^2)

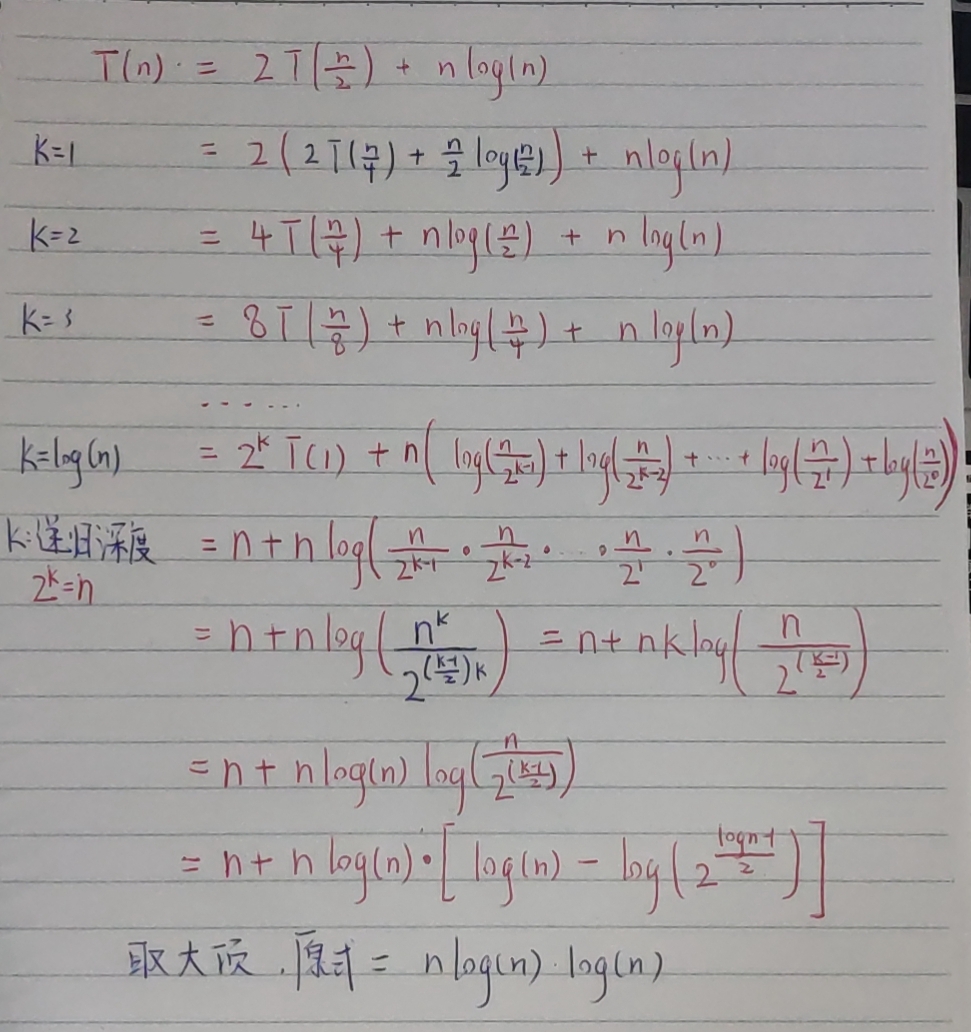
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 暴力穷举 | 10000 | 20000 | 30000 | 40000 | 50000 |
| 实际 | 1.49172 | 6.76564 | 13.4316 | 23.368 | 36.1698 |
| 理论 | 1.49172 | 5.96688 | 13.42548 | 23.86752 | 37.293 |

从图像上，图像符合n^2函数的趋势

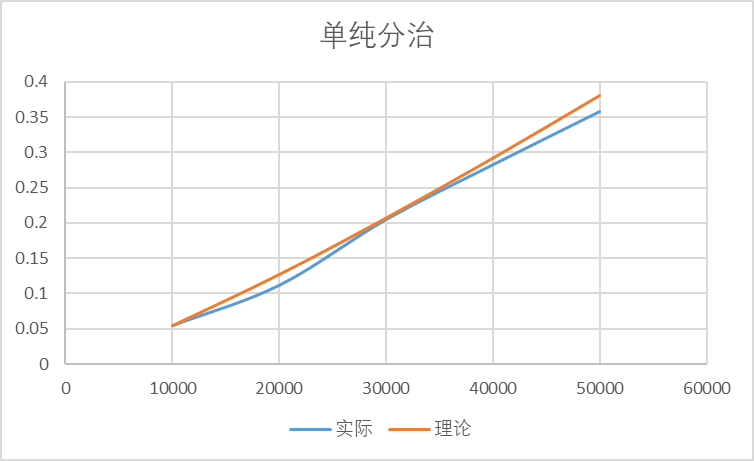
数据上，规模扩大k倍时，时间开销扩大k^2倍

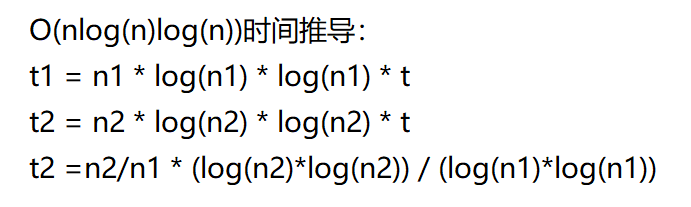


单纯的分治法：如果我们直接在合并的时候排序，那么相当于**合并代价是nlog(n)**，根据递推法，我们可以很快得到，单纯的分治法，递推式如下：T(n) = 2T(2/n) + nlog(n)









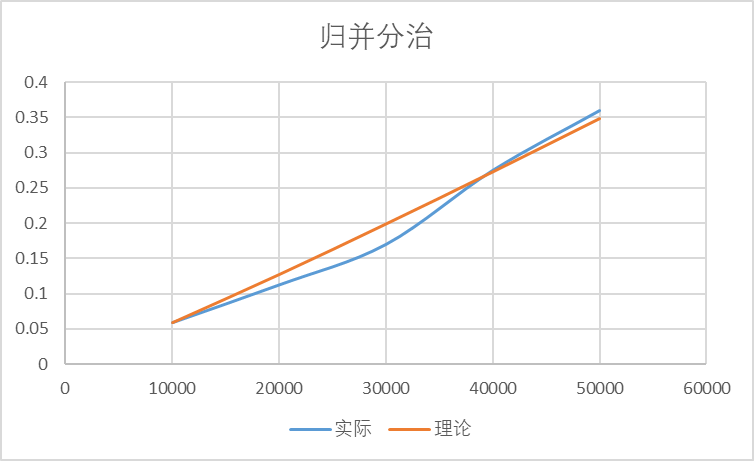
图像上基本符合二次增长，而时间上也是符合上述的推导

增大测试规模

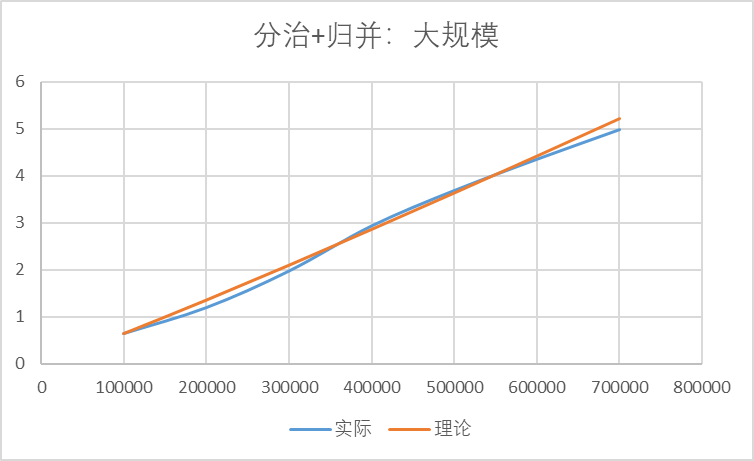


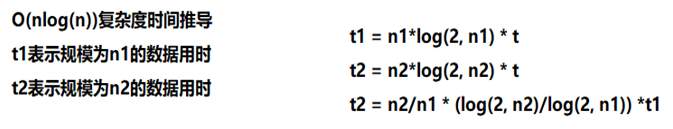
归并+分治： 效仿归并排序，如果我们在分治的同时按照y升序进行归并排序操作，递归后有序，不需要额外在递归中调用排序，**合并代价**从O(nlog(n))变为O(n)，总体时间复杂度变为O(nlog(n))







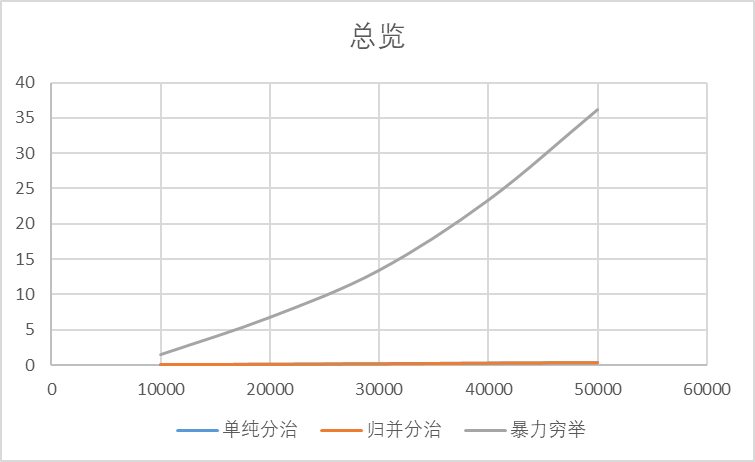




在图像上基本符合二次增长，而时间上也是符合上述的推导，因为每次需要遍历数组划分d\*2d的区间，导致合并代价稳定在O(n)，总体看来曲线比较稳定

总览与对比





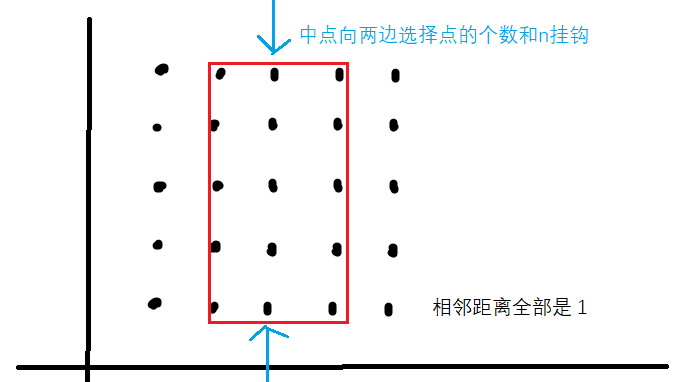
可以看到不论是单纯的分治还是归并的分治，效率都要远远高于暴力法



结论：可以看到虽然时间复杂度不同，但是两种方法消耗的时间相差无几，可能是log(n)在十万级别规模时，带来的影响很小，而且数据也存在随机性

分治法：特殊情况的测试

使用相邻点等距离的密集点来做数据，测量结果如下







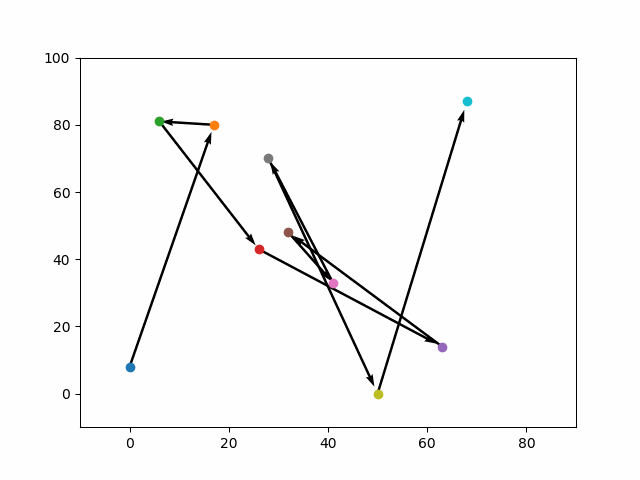
因为不论是归并还是分支+归并，因为合并代价都比较稳定，在极端的数据情况下，仍然能有好的表现

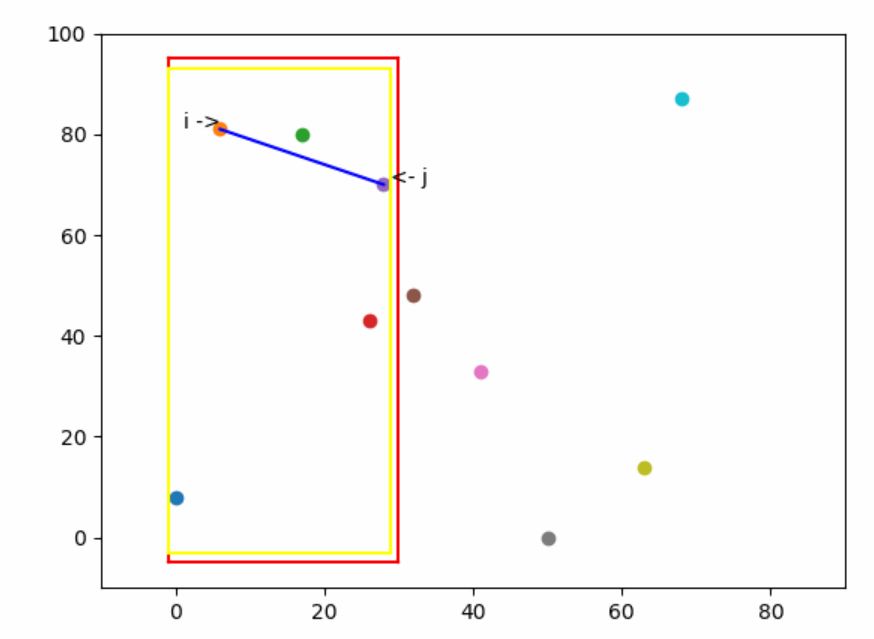
**图形演示**

通过python的matplotlib库的pyplot子库的animation对象的子对象ArtistAnimation进行图形的绘制，在递归的同时记录栈状态，递归结束之后使用图形的方式输出递归的过程，比如边界的划分，子问题的合并，以及点的选择

Word放动图不会动，更加详细的信息请老师查看我的实验课演示，或者移步至我的GitHub查看详细的原理与演示

<https://github.com/AKGWSB/graphic-demo-of-closest-point-pair-algorithm->





四、对求解这个问题的经验总结

（此处写你的过程，比如遇到的错误，以及解决方法，你的所想、所得）

点的坐标是int存储的，而距离则是double，因为二进制存储规则不同，两者之间的传递一定要加上强制类型转化，否则容易出现截断或者是转换不当等其他异常情况

在测试递归程序的时候，一定要保证合并操作的正确性，递归的答案可以先用暴力法代替，然后观察合并的结果是否和暴力法结果一致

在测试开销和递归结果有关的递归程序的时候，一定要大量测试保证代码的正确性，再测试时间，我之前代码有bug的时候，递归算出来的d不是最小的，测量的结果受影响

遇到问题的时候要想，能否通过数学的方法，压缩问题的解，使得求解范围缩小，甚至变为常数，比如通过划分d\*2d的矩形来压缩求解的空间

遇到一些分治递归的问题的时候，合并问题的时候需要考虑：如果左右已经有序了，会不会减少合并的代价？比如**逆序对问题**，也是通过同样的思想，分治的同时归并，然后利用数据的有序性，将合并问题的代价由O(n^2)减少到O(n)

代码的常数开销也很重要，代码常数开销过大，可能会导致预测的时间出现偏差，编程时应该尽量减小常数的开销

|  |
| --- |
| 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。