**实验报告**

目录

[实验环境 1](#_Toc20753485)

[实验结果 2](#_Toc20753486)

[实验分析 5](#_Toc20753487)

[小结 5](#_Toc20753488)

## 实验环境

#### 硬件环境

轻薄笔记本：联想小新Air13.3

CPU：Intel酷睿i5 8265U

内存：8GB

硬盘：256GB SSD。两个分区，C盘87GB，23.8GB可用。D盘149GB，46.6GB可用

电池：平衡模式，满电

外接设备：电源，USB台灯，圆孔耳机

#### 软件环境

操作系统：Windows 10 1903 家庭中文版

IDE：IDEA

编程语言：JAVA

JDK版本：JDK 8u191。下载来源：腾讯软件中心

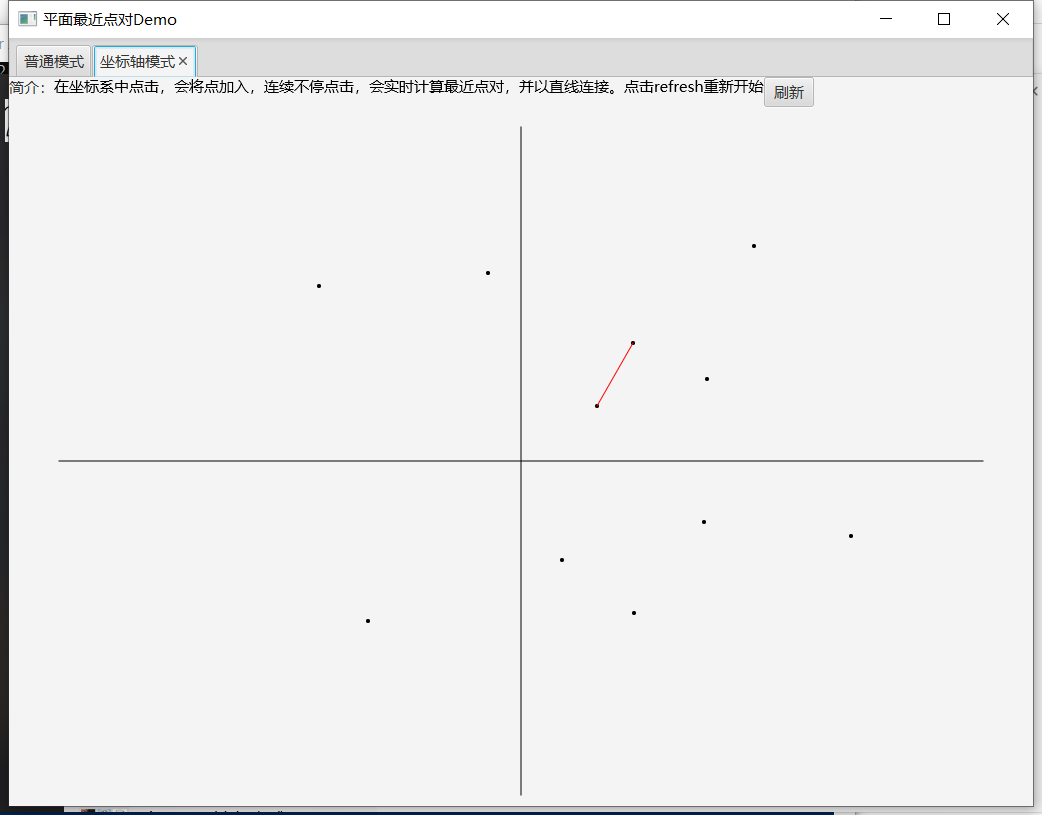
编译器：javac 1.8.0\_191

编译选项：没找到= =，idea不把编译选项输出到控制台，只返回build successful，哎。

运行方式：java -jar java\out\artifacts\JavaFXApp\JavaFXApp.jar

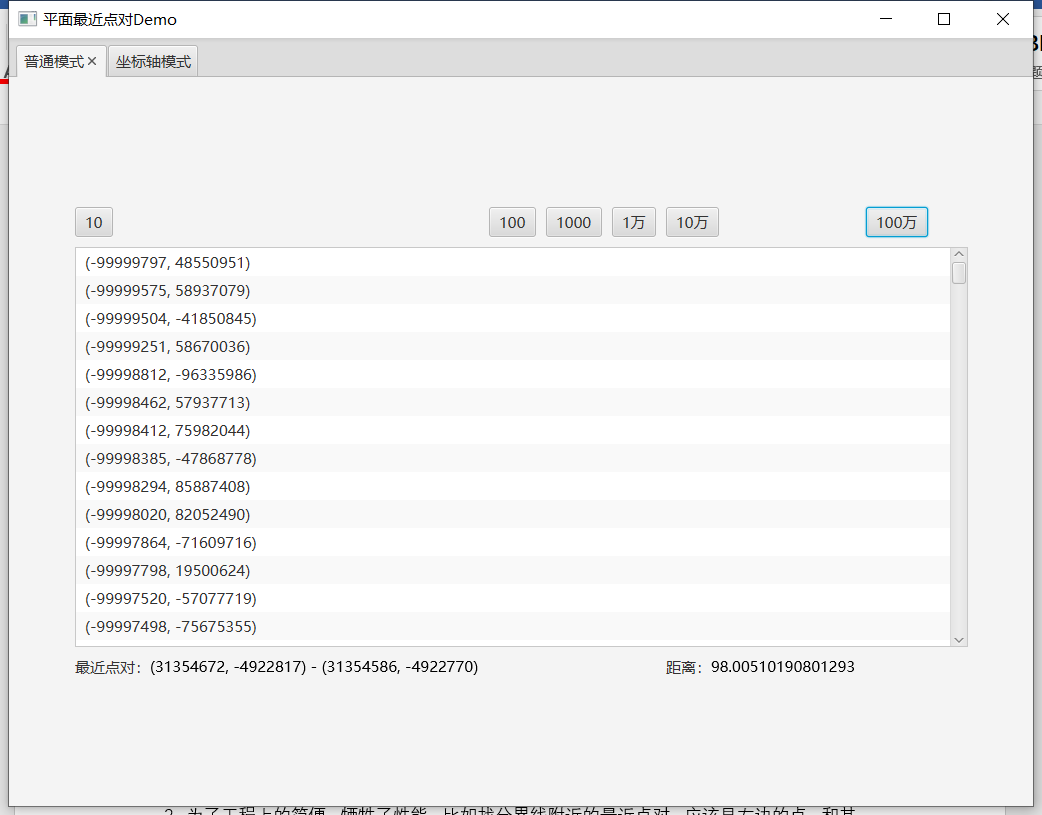
## 实验结果

##### 鼠标输入最近点对



如图，在坐标系中单击鼠标左键，会在相应位置画一个黑点，并实时计算最近点对，用红线连接。

##### 随机100万个点对



如图，点击“100万”按钮，会随机生成100万个点，用分治算法输出最近点对。坐标的范围【-1亿，1亿】

##### 不同输入规模下的运行时间

这部分实验没有放在UI中，直接用控制台跑，一共跑10次，可查看源代码，下面附表格和图。

|  |  |  |  |
| --- | --- | --- | --- |
| 编号 | 输入规模(单位:点数) | 消耗时间 | |
| O(nlgn) 分治法 | O(n^2) 枚举法 |
| 1 | 10 | 427.20us | 17.90us |
| 100 | 538.00us | 466.40us |
| 1000 | 5.07ms | 13.96ms |
| 10000 | 7.62ms | 426.67ms |
| 100000 | 79.27ms | 43.19s |
| 2 | 10 | 5.10us | 800.00ns |
| 100 | 27.20us | 40.60us |
| 1000 | 326.70us | 4.12ms |
| 10000 | 5.11ms | 426.88ms |
| 100000 | 71.66ms | 1.38m |
| 3 | 10 | 4.20us | 700.00ns |
| 100 | 25.60us | 40.50us |
| 1000 | 320.80us | 4.10ms |
| 10000 | 5.61ms | 430.18ms |
| 100000 | 72.51ms | 1.38m |
| 4 | 10 | 4.50us | 600.00ns |
| 100 | 24.50us | 40.40us |
| 1000 | 438.10us | 4.27ms |
| 10000 | 4.75ms | 432.31ms |
| 100000 | 71.74ms | 1.38m |
| 5 | 10 | 4.50us | 1.00us |
| 100 | 44.10us | 42.00us |
| 1000 | 318.60us | 4.25ms |
| 10000 | 4.46ms | 429.19ms |
| 100000 | 72.33ms | 1.38m |
| 6 | 10 | 5.00us | 800.00ns |
| 100 | 67.70us | 40.60us |
| 1000 | 534.50us | 4.55ms |
| 10000 | 5.51ms | 428.47ms |
| 100000 | 66.36ms | 43.44s |
| 7 | 10 | 4.00us | 600.00ns |
| 100 | 25.20us | 40.60us |
| 1000 | 324.90us | 4.15ms |
| 10000 | 5.06ms | 433.10ms |
| 100000 | 70.87ms | 43.42s |
| 8 | 10 | 3.80us | 600.00ns |
| 100 | 24.50us | 40.50us |
| 1000 | 323.30us | 4.27ms |
| 10000 | 4.88ms | 431.88ms |
| 100000 | 71.03ms | 43.33s |
| 9 | 10 | 23.00us | 600.00ns |
| 100 | 24.40us | 39.50us |
| 1000 | 300.50us | 4.04ms |
| 10000 | 4.48ms | 431.95ms |
| 100000 | 73.55ms | 43.34s |
| 10 | 10 | 4.20us | 600.00ns |
| 100 | 43.40us | 43.60us |
| 1000 | 338.10us | 4.22ms |
| 10000 | 4.51ms | 424.92ms |
| 100000 | 73.35ms | 1.38m |

将同等输入规模的时间取平均值，再以10为底取对数后，画出如下图形。

## 实验分析

在数据规模较小时，用枚举法省时间。

在数据规模较大时，用分治法省时间。

当数据规模在100左右时，两者的耗时出现交叉。

（个人感觉这个实验结果非常完美，嘿嘿，当然中间做了一些特殊的数据处理操作）

## 小结

###### **遇到的问题**

1. 一开始用jdk13，调试时没问题，但build artifact报错，网上说JavaFX从jdk11开始被剥离。遂单独下JavaFX包来编译，还报错，遂暴力退回jdk8。

2. JavaFX可以直接生成exe，不必通过jar方式打开，但是我编译后不行，不想花时间再精进，放弃。

3. 找不到idea在build的时候怎么执行的javac命令，无法copy编译选项出来。

###### 一些妥协

1. 为了工程上的简便，牺牲了性能。比如找分界线附近的最近点对，应该是左边的点，和其y坐标[-d d]范围内右边的最多6个点作比较。我没有区分点在左边还是右边，统一把分界线x方向[-d d]范围内的点找出来，然后把这些点按y坐标排序，每个点与其后面6个点求距离。
2. 在对比O(nlgn)和O(n)算法的时候，没有测试1百万输入的规模，因为枚举法这个时间太长了

###### 关于数据

第一次实验，在10, 100, 1000这个输入规模，结果与后面9次差别很大，但是输入规模在1万和10万，与后面9次差别不大。为了统一，在计算平均值时第一次的结果略去。

###### **关于图表**

10个点的时间和1百万个点的时间数量级差太多，y轴不好弄，需要一些特殊的技巧把太大的时间归一化，这里取了以10为底的对数（对数真是个好东西）。

###### **感想**

太花时间了，第一题就花了3个晚上和2个白天= =ORZ！我还有最优化，凸优化和概率图啊~~期末挂科ing~~~祝王斌老师和助教学长学姐国庆快乐~~