**随机算法课程**

**实验报告**

**实验二：**比较3种中位数选择算法的性能

姓名：王艺达

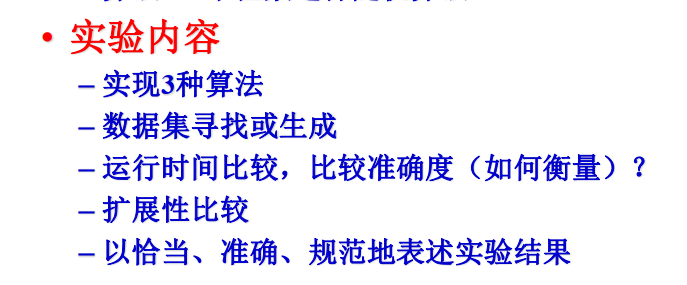
学号：1180301005

班级：1803101

评分表：（由老师填写）

|  |  |
| --- | --- |
| 最终得分： | |
| 对实验题目的理解是否透彻： | |
| 实验步骤是否完整、可信 ： | |
| 代码质量 ： | |
| 实验报告是否规范 ： | |
| 趣味性、难度加分 ： | |
| 特 色： | 1 |
| 2 |
| 3 |

**一、实验题目概述**



**二、对实验步骤的详细阐述**

1.实现了quickSort的递归版本,并且因为python递归层数的限制,进行了递归算法转化为迭代的优化.

2.实现了advancedMedian算法的递归版本,并对其进行了递归算法转化为迭代的优化.

3.实现了rndMedian算法,并且将其优化,使其能够最大程度地使用数组空间(只需要消耗小常数的额外空间),以此减小其运行代价(不需要内存分配)

**三、实验数据**

**1. 实验设置**

**实验环境**：

Python3.9 numpy sympy 函数包

**数据**：

1.数据量:1,500,000

2.数据类型:

Test0:完全随机(由numpy.random.randint生成)

Test1:已排序好(0到1,500,000)

Test2:已倒序好(1,500,000到0)

Test3:同一元素(所有元素都一样)

**2. 实验结果**

**显示说明:**

**naïve: naïve方法进行的数组操作次数**

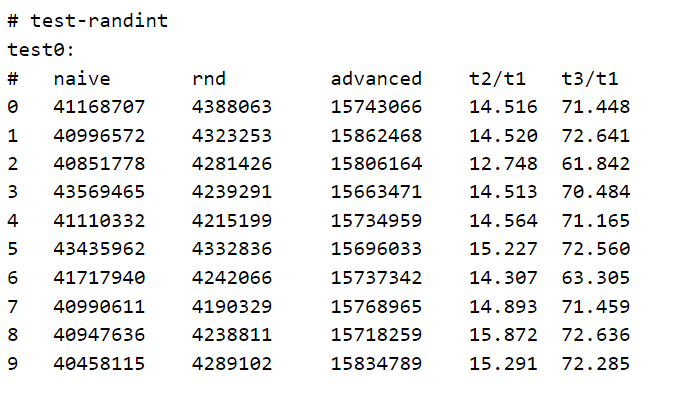
**rnd: 随机方法进行的数组操作次数**

**advanced: 五数取中优化后的方法的数组操作次数**

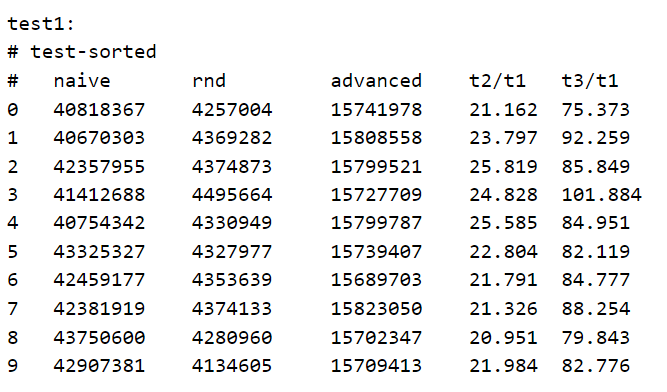
t2/t1:

t3/t1:

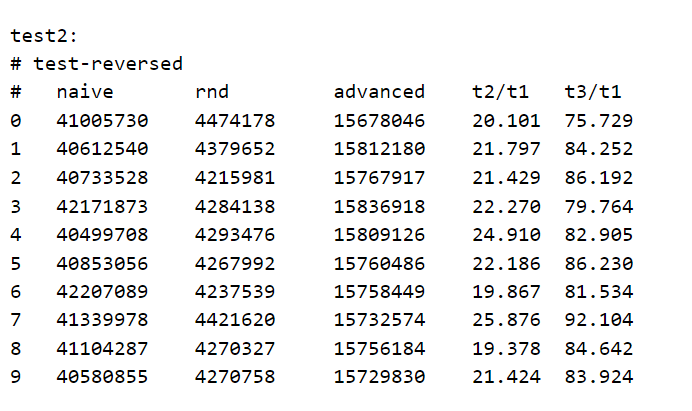
**2.1.Test0随机数:**

****

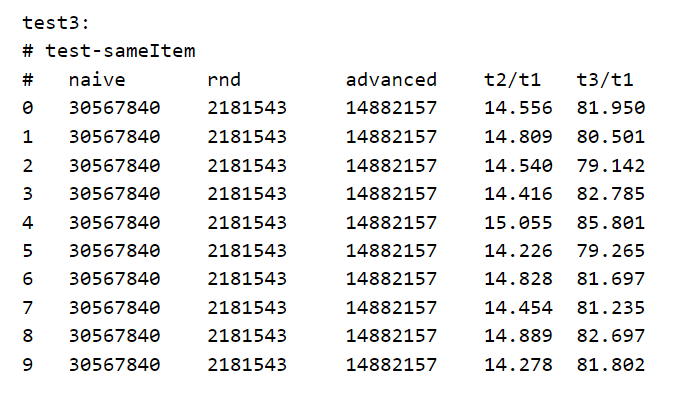
**2.2.Test1已经排序好:**

****

**2.3. Test2 逆序排好:**

****

**2.4. Test3 相同元素:**

****

**四、对实验结果的理解和分析**

**1.advancedMedian原理:**

本质上该算法就是对普通quickSort算法中的partition子过程进行改造后得到的.

根据复杂度分析,quickSort的运行时间长短和partition是否均匀有关:一般来说,partition越均匀,运行效率越接近O(nlogn)的复杂度下限.而partition是否均匀又和pivot元素的选取有关.

据此思路,advancedMedian算法将”优化pivot选取的partition”以及”中位数选取”这两个问题紧密地结合在一起,通过”中位数pivot”促进partition过程的均匀划分,再通过partition将中位数选取问题转化为规模更小的子问题.

**2.rndMedian原理:**

随机选取数组S的一部分作为子数组A,而原数组的中位数很大概率地落在这个子数组的中位数的一个小邻域P内.

判断邻域P内是否真的包含有S的中位数,假如有则进行排序并输出答案,否则重新运行算法

**3.运行时间比较:**

[a].随机方法总体运行时间相对稳定,并且相对于naïve方法的性能至少提高5倍.

[b].五数取中优化算法对于随机数据来说效果尚好,但是对于已经排好的数据来说却因为五数取中的额外开销而效果不佳.

[c].naïve方法在已经排序好的数据上效果较好,其他情况下它的效率远低于随机化方法.

**4.准确性与可扩展性:**

三个算法都能获得准确解,没有误差.三个算法本质上都是求解第k序数的方法,都是可以扩展的.

**五、实验过程中最值得说起的几个方面**

1.将原本的递归算法优化为迭代算法,节省了运行所需要的内存空间以及运行时间.

2.将rndMedian算法进行了优化,使得它不需要消耗任何额外存储空间进行数据存储,节省了运行所需要的内存空间以及运行时间.

3.原始的rndMedian算法的算法实现在**所有元素都相同**的情况下,是不能够正确运行的,经过简单改进可以正确运行.