**高级算法课程**

**实验报告**

**实验二：比较三种中位数选取算法的效率**

姓名：朱宸慷

学号：2021110908

班级：2103103

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**

1. 通过比较三种中位数选取算法的效率进一步理解随机算法的概念、简洁性和高效性
2. 理解随机算法参数设置的效率和复杂性分析结果的有效性
3. 规范撰写实验报告

**二、对实验步骤的详细阐述**

整体步骤

1. 实现先排序后直接抽取算法
2. 实现线性时间选取算法
3. lazySelect随机算法
4. 随机产生服从均匀分布、正态分布、Zipff分布的数据和均匀选取的k，开展实验，比较三种算法的性能和扩展性
5. 变换随机算法中的关键参数3/4，重复实验，观察算法性能变化
6. 撰写实验报告

数据生成器

1. 从键盘获取输入n，进行合法性检查，仅接受正整数
2. 根据输入数字n，在[0,10\*n]范围内生成大小为n的随机数组nums []int

先排序后抽取算法

1. 获取随机数组，将其复制到类中
2. 自行实现快速排序算法
3. 数组排序
4. 在排序后的数组中直接给出中位数

线性时间选取算法

1. 获取随机数组，将其复制到类中
2. 基于快速排序的思想，实现寻找数组中第k小的算法
   1. 从给定数组中随机抽取一项元素来划分数组，该元素记为pivot
   2. 将数组中元素划分为小于等于pivot和大于pivot的两部分，比较前者数组元素数量与k，若相等，则返回这一部分；否则就在左侧排除pivot的部分中递归调用该算法。
3. 基于该算法可以在线性时间找到任意第k小的元素，调用并返回中位数

lazySelect算法

1. 写在前面：笔者实现的算法不能对项数少于3的数组使用，否则递归调用会导致栈溢出
2. 首先实现线性时间Rank()函数，用于获取该元素在数组中的递增序
   1. 遍历数组下所有元素，若该元素比给定元素小，则rank++
   2. 遍历完成后，返回结果
3. 基于lazySelect思想实现一个寻找第k小的算法
   1. 原数组大小为n，从原数组中随机选取n^{0.75}项元素构成selNum数组
   2. 调用快速排序，将selNum数组排序，此时原数组中第k小的元素理论上在排序后的数组中的位置为k\*n^{-0.25}，考察该位置大小为n^{0.5}的左右侧邻域（不能超过selNum边界），找出边界值L和H
   3. 计算L和H在原数组中的rank，得到Lp和Hp。
   4. 收集原数组中介于L和H之间的元素，保存为数组P
   5. 判断是否满足Lp<=k && Hp>=k && len(P) <= 4\*n^(0.75) +1 ，若是，则认为选择成功，将数组P快速排序后返回第k-Lp-1项元素；否则递归调用该算法，直到成功为止

**三、实验数据**

**1. 实验设置**

**实验环境**：

Golang 1.22.1

Windows 11

**数据**：

首先随机生成10k，100k，1000k,1e9+1数据，比较运行时间，此处数据均为均匀分布生成，随机算法参数为0.75。之后调整随机算法参数，考察递归次数。

**2. 实验结果**

文本

描述已自动生成

文本

描述已自动生成

文本

描述已自动生成

文本

描述已自动生成

生成长度为100的数列，调整随机算法参数为0.5，结果如下

文本

描述已自动生成

长度不变，调整随机算法参数为0.9时

图形用户界面, 文本, 应用程序

描述已自动生成

**四、对实验结果的理解和分析**

1. 总体来看，线性时间选择算法总是耗时最小
2. 当数据项较少时，快速排序算法耗时比lazySelect耗时少，猜测可能此时抽取数据项以及递归的耗时较大，影响最终总运行时间
3. 当数据项较大时， lazySelect耗时比快速排序算法耗时少，这就意味着此时仅考察部分元素来确定中位数的算法是高效的
4. 当数据量继续增大时，一次递归就选中的概率也随之增大，同时，lazySelect的时间消耗和线性时间选择的消耗进一步靠近。
5. Zipf分布的数组有概率无法使用lazySelect，猜测可能是由于相同的项数太多导致lazySelect难以找到正确解
6. 当调整随机算法参数为0.5时，递归的次数显著增加，调整为0.9时，递归次数也有增加。可见之前所使用的0.75是一个较好的数字。

**五、实验过程中最值得说起的几个方面**

1. 笔者实现的lazySelect对于几个特殊长度的数组会产生不同的错误，例如2，3会导致栈溢出，而4在递归超过100次的情况下勉强可以成功运行

图形用户界面, 文本, 应用程序

描述已自动生成

1. 在对照课件实现lazySelect算法时，会涉及到部分元素的下标错位问题，例如在最后返回结果时，下标实际上和计算结果相差了1

屏幕上有字

描述已自动生成

1. 本实验原本是计划使用python实现的，但是笔者在很长一段时间内都认为没有指针不能完成三个算法的部分操作