华东师范大学数据科学与工程学院上机实践报告

课程名称：算法设计与分析　　**年级**：22 级　　**上机实践成绩**：

指导教师：金澈清　　**姓名**：石季凡

上机实践名称：数据结构　　**学号**：10225501403　　**上机实践日期**：2023.4.10

上机实践编号：No.6　　**组号**：1-403

一、目的

1．熟悉算法设计的基本思想。
2．掌握栈、链表等基本数据结构。
3．掌握使用非显式指针来表示链表等数据结构的方法。

二、实验内容

1. 用多重数组实现单链表。
2. 实现在该单链表上的“增删查改排”操作，插入并不一定在头部插入，排序可以用任意一种排序方法。注意释放对象。
3. 和用指针实现的链表相比，本次实验中实现的链表有什么使用和性能上的异同呢？（选做）请通过实验来验证你的猜想。

三、使用环境

Windows11，vscode，c 语言

四、实验过程

1. *写出多重数组单链表的源代码。*
2. *验证实现的正确性，给出各种情况的测试数据和运行结果。*
3. *分析并提出假设并验证你的猜想。*

五、总结

【指针型与多重数组型的对比】

由于指针型链表每次增减元素都需要申请新的内存空间，所以这会导致其整个链表在内存上不是连续的，我们可以通过程序来验证这一点，结果如下

首先看到多重数组实现的链表，由于多重数组链表的可用空间是提前申请并排序好的，因此后续的操作都会在固定的一块内存上进行操作，同时由于每个 int 类型为 4 字节，所以构成该链表的数组会更加连续，如下面这个例子，长度为 100 的链表，那么我接下来的操作就是在 400 字节的一个范围内操作。

```
100
address_key= 00000000006B1724
address_key= 00000000006B1694
address_key= 00000000006B1614
address_key= 00000000006B16C4
address_key= 00000000006B16E4
address_key= 00000000006B1688
address_key= 00000000006B168C
address_key= 00000000006B1650
address_key= 00000000006B164C
address_key= 00000000006B1734
address_key= 00000000006B1634
address_key= 00000000006B1660
address_key= 00000000006B16BC
address_key= 00000000006B1664
address_key= 00000000006B171C
address_key= 00000000006B1628
address_key= 00000000006B1604
address_key= 00000000006B1638
address_key= 00000000006B1670
address_key= 00000000006B1690
address_key= 00000000006B16F0
address_key= 00000000006B162C
```

接下来看指针实现的链表

```
100
address_tmp = 00000000006F19A0
address_tmp = 00000000006F6840
address_tmp = 00000000006F1760
address_tmp = 00000000006F6560
address_tmp = 00000000006F15B0
address_tmp = 00000000006F67C0
address_tmp = 00000000006F6800
address_tmp = 00000000006F6940
address_tmp = 00000000006F69E0
address_tmp = 00000000006F6B20
address_tmp = 00000000006F1740
address_tmp = 00000000006F6600
address_tmp = 00000000006F65C0
address_tmp = 00000000006F18C0
address_tmp = 00000000006F6740
address_tmp = 00000000006F1530
address_tmp = 00000000006F1570
address_tmp = 00000000006F6920
address_tmp = 00000000006F16E0
```

如图内存同样是跳跃的，但是由于每次插入都需要申请内存，加上每个结构体指针所占的大小为 20 字节，这就需要在更大的范围上进行操作，这两点加起来会消耗更多时间。

但是指针型链表随用随申请的好处就是更节约空间。接下来可以进行插入实验与删除实验来进行对比。

【首先来进行插入函数】

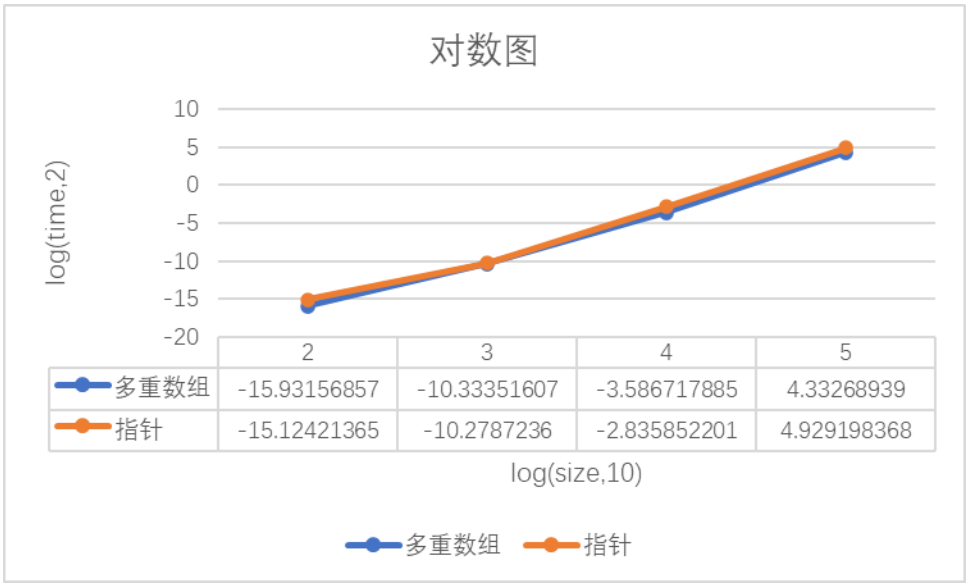
实验步骤：进行 100，1000，10000，100000 次插入操作，每次插入的位置为随机数，范围为（0，count），每次插入的 key 值为随机数，范围为 0，10000

我们得到了以下实验数据

	多重数组	指针
100	0.000016	0.000028
1000	0.000775	0.000805
10000	0.083232	0.140063
100000	20.14974	30.46748

其中最左列为数据规模，每一个时间的单位为秒。

为了更方便观察，我对数据规模取了以 10 为底的对数，对时间取了以 2 为底的对数。



由结果可见，在插入这一功能上，多数组实现的链表确实更快。

【第二组实验为删除函数】

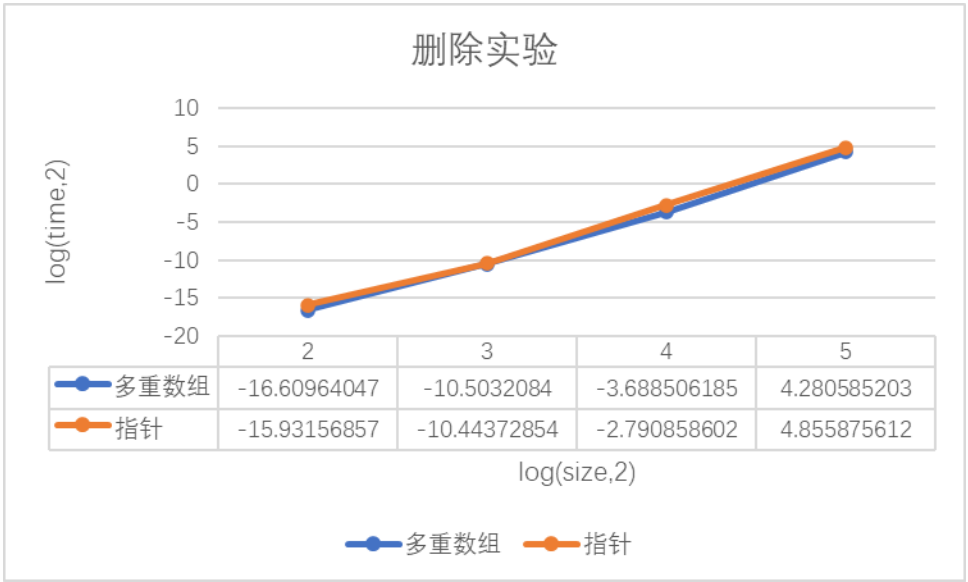
实验步骤：进行 100，1000，10000，100000 次删除操作，每次删除的位置为 0，count-1 的随机数

结果如下

	多重数组	指针
100	0.00001	0.000016
1000	0.000689	0.000718
10000	0.077562	0.1445
100000	19.435	28.95771

其中最左列为数据规模以及删除次数（二者相等），每一个时间的单位为秒。

为了更方便观察，我对数据规模取了以 10 为底的对数，对时间取了以 2 为底的对数。



由于多重数组不需要 free 内存，加上其内存跨度小，这使得其在已有链表上的操作更快。

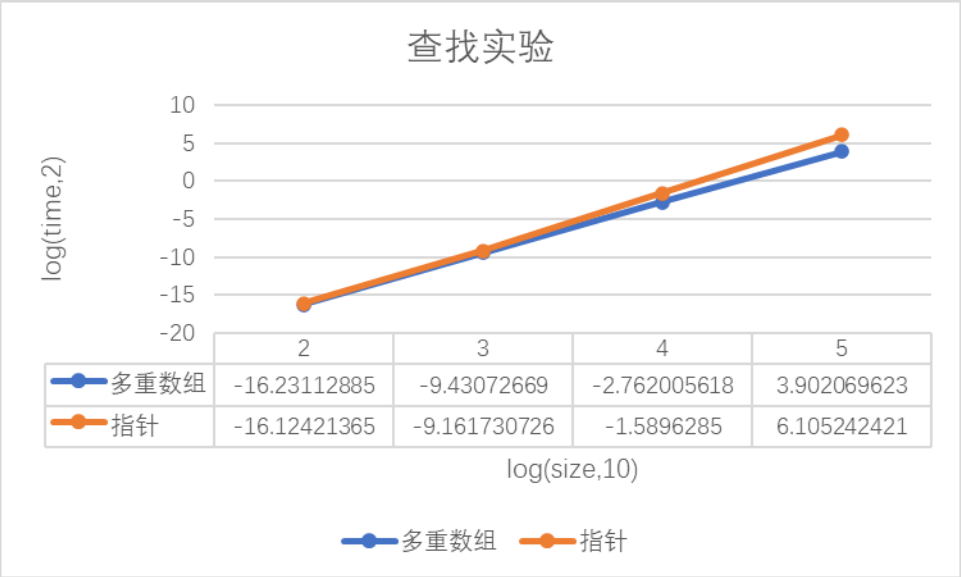
【第三组实验为查找函数】

每次查找随机位置的值，随机位置取值为 0 到 count-1，分别查找 100,1000,10000,100000 次

	多重数组	指针
100	0.000013	0.000014
1000	0.001449	0.001746
10000	0.147419	0.332257
100000	14.94996	68.84321

其中最左列为查找次数，每一个时间的单位为秒。

为了更方便观察，我对数据规模取了以 10 为底的对数，对时间取了以 2 为底的对数。



这一实验的数据更能反映出多重数组型链表内存分布范围较小以及操作更简洁的优势，因为查找操作不需要 malloc 来分配内存。

根据上面的结果我们可以看到，由于 malloc 操作与地址的跨度，指针型链表确实更加耗时。

【对比总结】

总而言之，多重数组链表牺牲了空间来换取了时间的减少，而指针型链表牺牲了时间来换取了空间的减少。