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：**林子骥

上机实践名称：数据结构　　**学号：**10225501460　　**上机实践日期：**4.6

上机实践编号：No.6　　**组号：**1-460

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。

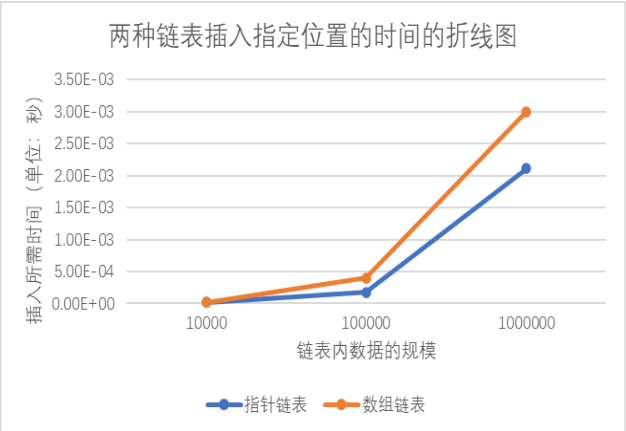
二、实验内容

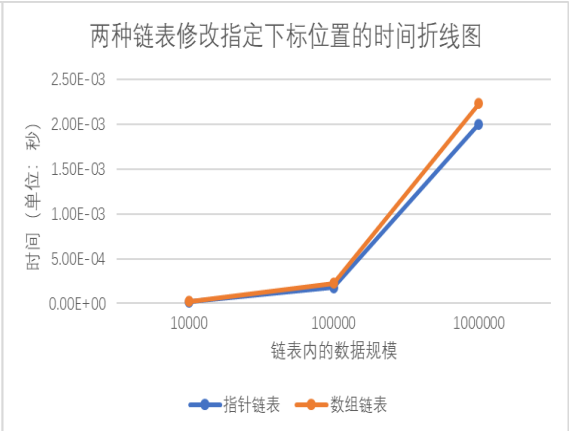
1. 用多重数组实现单链表。
2. 实现在该单链表上的“增删查改排”操作，插入并不一定在头部插入，排序可以用任意一种排序方法。注意释放对象。
3. 和用指针实现的链表相比，本次实验中实现的链表有什么使用和性能上的异同呢？（选做）请通过实验来验证你的猜想。

三、使用环境

可以使用你自己喜欢的编程语言和环境来实现。

四、实验过程





两种链表在不同数据规模将数据插入指定位置所需时间的表格（单位：秒）

	10000	100000	1000000
指针链表	1.80E-05	1.78E-04	2.11E-03
数组链表	1.80E-05	4.00E-04	3.00E-03

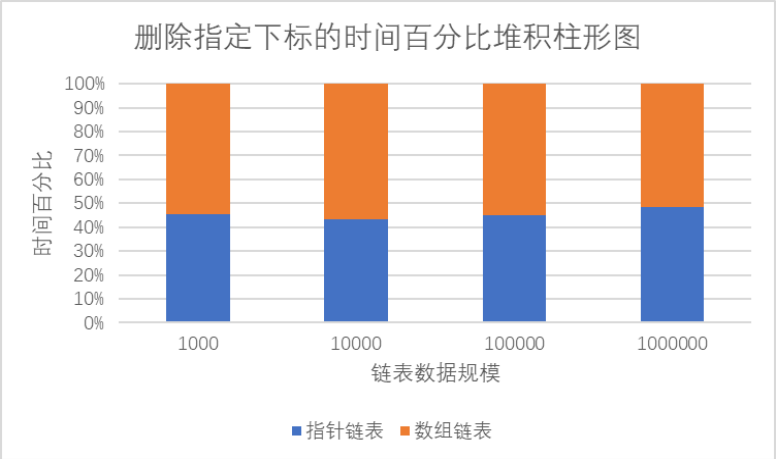
两种链表在不同数据规模修改（查找）指定位置数据的时间表格（单位：秒）

	10000	100000	1000000
指针链表	1.90E-05	0.000179	0.002001
数组链表	2.50E-05	0.000228	0.002234

单个对数组链表的插入与修改与删除功能分析，发现随着数据规模的线性增加，其插入的时间也线性增加，其时间复杂度为 O（N），这是因为其数组需要通过一个 while（index--）循环来查看所要元素在数组的实际位置，其最坏情况将会扫描每一个数组。

数组链表和指针链表插入，删除，修改功能的相同点：其时间复杂度都是线性的。

插入、修改性能的不同点：数组的操作时间会略快。在大多数情况，多重数组实现的链式数据结构的数据存储是连续的，而指针实现的链表的数据存储是不连续的。不连续的数据存储对于内存访问是十分不友好的。尽管多重数组链表通过数组下标来简洁访址，而链表通过指针值直接访址。前者需要进行内存地址的计算， 而后者可以直接访问内存，使得指针链表省去了时间来计算不必要的地址值，但这一因素的影响远小于前者。

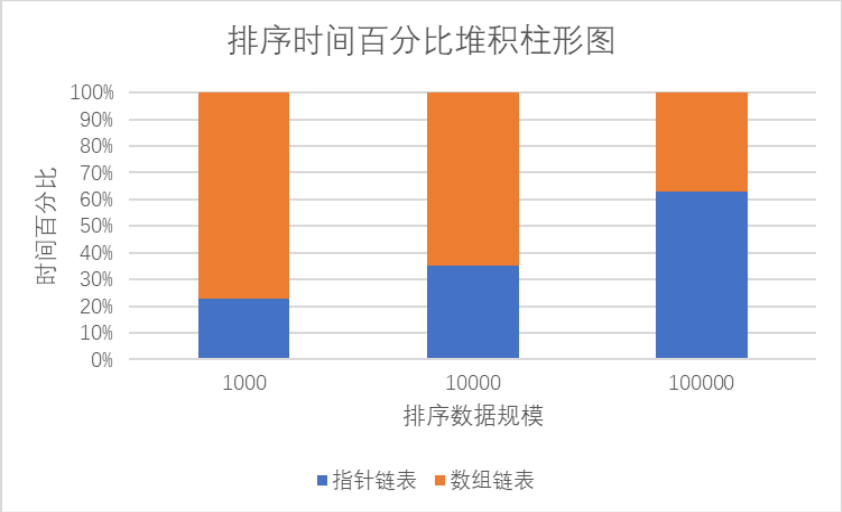


两种链表在不同数据规模删除指定下标数据所用时间（单位：秒）的表格

	1000	10000	100000	1000000
指针链表	5.00E-06	1.90E-05	0.000183	0.002084
数组链表	6.00E-06	2.50E-05	0.000222	0.002226

删除功能下两种链表的相同点：都是线性时间复杂度。

删除性能的不同点：其不同点包含了插入性能的不同点。此外，链表的删除需要进行 delete 操作，这一操作会增加 CPU 的负担，减慢运行时间。而对于多重数组实现的链式结构，其通过在栈上改变 int 变量 free 的值来表示空链表的首位置。



两种链表在不同数据规模下排序所用时间（单位：秒）的表格

	1000	10000	100000
指针链表	0.00055	0.104246	38.5445
数组链表	0.001843	0.190329	22.5377

对多重数组链表进行分析，从表格中可以看出随着数据规模的线性增加，其排序所需的时间成幂数级增加，其时间复杂度为 O（n^2）这是因为这里采用了插入排序。

将两种链表的排序时间进行比对，可以发现在不同数据规模下，其排序时间的相对大小有明显的不同。

先对于较大规模的数据（N=10^6）进行分析。基于指针的链表排序过程需要频繁地进行内存分配和释放，这可能会产生大量的开销。同时，由于链表中的每个元素都有一个指向下一个元素的指针，为了执行排序操作，需要对于每个元素都进行一次指针的操作，这也会带来额外的开销。相比之下，基于多重数组构造的链式结构数据具有更好的局部性和内存对齐性，因此其排序过程通常要比基于指针的链表排序更快。多重数组构造的链式结构数据可以使用数组下标来访问元素，这样可以更有效地利用 CPU 的缓存机制，减少内存访问的延迟。因此，基于多重数组构造的链式结构数据排序时间会比基于指针实现的链式结构数据排序时间更短。

对于数据量较小的情况分析。由于数据量较少，内存分配和释放的开销相对较小，指针实现的链式结构数据排序时间可能会比基于多重数组构造的链式结构数据排序时间更短。同时，当链表的长度相对较短时，指针实现的链式结构数据的指针操作次数也相对较少，对于 CPU 缓存的影响也相对较小，因此指针实现的链式结构数据排序时间可能会更短。

五、总结

本算法实现思路：

本实验建立了三个较大的数组，分别存储 value，pre，和 next。同时将一个 value，pre，next 看作一个节点，这样一来三个大数组可以看成一个链表。并且，将这一链表（三个大数组）不显性的划分为两部分：存储数据的部分和未存储数据的部分。其划分的具体实现

为：

1.数组初始化时，通过 pre 和 next 的值将整个数组连接起来，体现两个数据的相对位置，而不是仅仅通过下标的绝对大小比较判断数据的相对位置

2.引入 int 值 free，记录空链表部分的首地址下标。当引入数据是，将值放在下标 free 处，然后通过 next 数组更新 free 值，作为新的首地址下标。删除数据后，将所删除的数据下标作为新的 free 值，通过 next 与 pre 接入空链表

两种链表的选择：

理论上多重数组实现的链式数据结构的插入指定位置和删除指定位置数据操作的时间复杂度是 O(n)，而指针实现的链表与之相同。但指针链表的 delete 需要占用 cpu 资源。因此，当需要频繁地进行指定位置插入和删除操作时，数组实现的链表更适合。

然而数组链表对 CPU 负担的减轻是通过牺牲部分内存空间实现的，因此对于内存敏感的应用程序，应使用指针实现的链表。

修改：

对于链表的排序可以使用递归排序，可以采用以下步骤：

1. 递归地将链表分成两个子链表，直到每个子链表只有一个节点或为空为止。

2. 对每个子链表递归地进行排序。

3. 将两个有序子链表合并成一个有序链表。

具体实现过程如下：

1. 使用快慢指针法将链表分成两个子链表，快指针每次移动两个节点，慢指针每次移动一个节点，当快指针到达链表末尾时，慢指针指向的节点即为链表中间位置，将链表从中间位置断开，得到两个子链表。

2. 对每个子链表递归地进行排序，直到子链表为空或只有一个节点，此时子链表已经有序。

3. 对于两个有序子链表，使用归并排序的合并过程将它们合并成一个有序链表。具体过程为，从两个子链表的头节点开始，比较两个节点的值，将较小的节点插入到新链表中，并将该节点所在子链表的头节点后移一位，直到某个子链表为空。将另一个子链表剩余的节点直接插入到新链表中。