汇编语言程序设计

实验二 三种排序算法的实现与分析

2005010130 侯杰

一、实验要求

1．分别用 C/C++，汇编语言书写冒泡，选择，快速排序。均要求递归实现。

2．对三种算法的效率进行分析

二、实验分析

由于要求递归实现，与以前的程序书写思路稍有不同。现将基本算法思路阐述如下：

1．冒泡排序的递归算法

目的：实现 N 个元素元素从大到小排序。

算法：如果 N=1 则直接返回，否则开始排序。从第一个元素开始，依次比较相邻的两个元素。若前一个元素小于后一个元素，则将前后两元素交换位置。第一轮扫描交换完毕后，最小的元素必然位于最后。这样对于前 N-1 个元素又是一个相同的子问题，可以递归地再次调用算法。

C++语言实现：用一个大小为 N 的数组存储 N 个要排序的元素，递归调用 Bubble(int* dat,int n)实现排序。其中第一个参数 dat 为数组名称，第二个参数表示此函数对前 n 个元素进行排序。在执行中利用 for 循环语句进行比较和交换工作。最后递归调用 Bubble(dat, n-1)完成剩下操作。

汇编语言实现 :数据段内定义两个变量 dat 和 N。其中 dat 为数组首地址，N 为元素总个数。先将数组首地址和 N 的值进行压栈，然后调用子程序 Bubble

进行排序。在子程序 Bubble 中，先从栈中取到数组首地址和 N，利用 loop 循环实现比较与交换操作。本轮完成后继续将数组首地址和 N-1 进行压栈，再次递归调用 Bubble 完成剩下的操作。

2．选择排序的递归算法

目的：实现 N 个元素从大到小排序

算法：如果 N=1 则直接返回，否则开始排序。从第一个元素扫描到最后一个元素，找到其中最小的元素，若其并不处于最后一个位置，就将其与最后一个元素交换。这样最后一个元素变为最小的元素，而前 N-1 个元素的排序又是一个相同的子问题，递归处理即可。

C++语言实现：用一个大小为 N 的数组存储 N 个要排序的元素，递归调用 Select(int* dat,int n)实现排序。其中第一个参数 dat 为数组名称，第二个参数表示此函数对前 n 个元素进行排序。在执行中利用 for 循环语句寻找到最小元素的下标并保存。然后与最后一个元素的下标(n-1)比较，若不相等则将此元素与最后一个元素交换，再递归调用 Select(dat, n-1)完成剩下操作。

汇编语言实现：数据段内定义两个变量 dat 和 N。其中 dat 为数组首地址，N 为元素总个数。先将数组首地址和 N 的值进行压栈，然后调用子程序 Select 进行排序。在子程序 Select 中，先从栈中取到数组首地址和 N，利用 loop 循环实现最小元素的查找，最后判断最小元素是否是最后一个元素，不是则进行交换。本轮完成后继续将数组首地址和 N-1 进行压栈，再次递归调用 Select 完成剩下的操作。

3．快速排序的递归算法

目的：实现 N 个元素从大到小排序

算法：首先将待排序元素分组。选第一个元素作为基准，把大于第一个元素的元素全放到左边，把小于第一个元素的元素全放到右边。然后将第一个元素插在这两组之间，递归地排序左边组和右边组元素。

C++语言实现：用一个大小为 N 的数组存储 N 个要排序的元素，递归调用 Quick(int* dat, int left, int right)实现排序。其中第一个参数 dat 为数组名称，第二、三个参数表示此次参与排序的元素下标范围。在执行中取下标为 left 的元素作为基准进行左右分组，最终基准元素定位于下标 pivotPos。然后再递归地执行 Quick(dat,left,pivotPos-1)和 Quick (dat, pivotPos+1, right)完成排序。

汇编语言实现：数据段内定义两个变量 dat 和 N。其中 dat 为数组首地址，N 为元素总个数。先将 dat 的地址压栈，然后将 0 及 N-1 压栈，然后调用 Quick 子程序。在子程序中同样先进行左右分组，找到 pivotPos 的地址，然后求出它的下标。然后压 pivotPos 的下标，压 dat 的地址，压 0，压 pivotPos 的下标-1，然后调用 Quick 子程序。返回后弹 pivotPos 的下标，压 dat 的地址，压 pivotPos 的下标+1，压 N-1，再调用 Quick 子程序，完成排序。

四、具体实现

1.冒泡排序

C++语言实现：（另附 Bubble.cpp）

汇编语言实现：（另附 Bubble.asm）

2.选择排序

C++语言实现：（另附 Select.cpp）

汇编语言实现：（另附 Select.asm）

3.快速排序

C++语言实现：（另附 Quick.cpp）

汇编语言实现：（另附 Quick.asm）

五、排序速度测试

以上三种排序效率不同，我自己做了一个测试，来测定不同语言下，不同排序算法写的排序程序的运行时间。

测试数据：20000 个随机数，大小为 0~0xffff。

测试结果如下：

	冒泡排序程序	选择排序程序	快速排序程序
C++版本	0.479s	0.213s	0.034s
汇编版本	0.775s	0.450s	0.045s

六、实验结果分析

首先，让我们从理论的角度分析一下这三种算法的特点

（1）n 个元素进行冒泡排序，需要进行(n-1)+(n-2)+…+1=(n-1)*n/2 次比较。其算法时间复杂度为 O(n^2)的。同时需要有一个 temp 元素作为临时交换用，所以其空间复杂度为 O(1)。

（2）n 个元素进行选择排序，需要进行(n-1)+(n-2)+…+1=(n-1)*n/2

次比较。其算法时间复杂度也是 O(n^2)的。同时也需要有一个 temp 元素作为临时交换用，所以其空间复杂度也为 O(1)

（3）n 个元素进行快速排序，经理论证明得知其平均时间复杂度为 O(nlogn)，但在最坏情况下时间复杂度也将退化到 O(n^2)。此算法平均情况下空间复杂度为 O(logn)，但在最坏情况下空间复杂度也将达到 O(n)。

以上理论分析可以看出，快速排序确实是这三种排序里面最快的，同时它的空间需求也较大，在实现的过程中需要较多堆栈空间。而冒泡排序和选择排序时间复杂度就大了一个数量级，所以就会慢的多。但他们的空间需求也较小。

从实验数据可以看出，快速排序用的时间几乎是前两种排序所需时间的十分之一或更小，这与理论相符。但同时也看出，选择排序好像比冒泡排序更快一些。

这一点也可以从理论上找到根据。刚才我们比较的都是进行排序所需要的比较次数，这一点冒泡与选择基本相同。但是冒泡排序每进行一轮，最多可能会发生 O(n)次元素交换操作，而选择排序每进行一轮，只在最后有可能进行一次交换操作，之前进行的仅仅是地址的赋值操作，所以其交换次数应为 O(1)，小于 O(n)。正是这个原因，使得选择排序综合运行时间由于冒泡排序。

另外一个敏感的问题，就是 C++与汇编写出的程序到底哪个更快些。我在这里的 C++版本和汇编版本所用的算法，数据结构是完全相同的，所以运行速度应该取决于指令编译结果的好坏及指令执行的效率。可我发现 C++竟然比汇编编的程序执行速度还快。对其原因，我提出了一下几种猜想：

（1）我用的 C++编译器是 Microsoft Visual C++ 2008 的编译器，比较新，可能对于当今流行的 cpu 指令做过优化，有可能运行了一些高级指令，较好地利用了当今 cpu 的性能。

（2）我的 cpu 是 32 位 cpu，运行为 16 位 cpu 写的汇编程序时，并不能完全发挥出系统的性能优势。

（3）我的运行平台是 windows vista x86。这里没有 dos 实模式，只有一个 cmd 程序，实质上是一个虚拟的 dos 环境，也就是个虚拟机。有可能这里的指令，内存都是经过映射才执行的。就像运行在 java 虚拟机上的 java 程序速度并不佳一样，在建立在虚拟机层次的 dos 环境中运行，必然会影响速度。

六、实验后体会与收获

首先，通过本次实验，我进一步熟悉了汇编语言的语法及格式，对堆栈传参用的更加熟练。同时，通过编写快速排序这样具有一定规模的汇编程序，锻炼了不少思维。尤其是其中对于堆栈的使用，如何取到正确的地址，如何在多次调用后保证栈的平衡，确实是一个需要好好练习、思考的地方。

其次，通过这次实验我感受了同样的算法，同样的数据结构在两种语言平台实现感觉的不同。C++可以有抽象的数组的概念，而且它的指针也是有类型的，会自动根据类型来进行自加自减操作。而汇编中一切都是绝对地址的概念。不同的元素下标，不同的首地址，不同的元素类型，都会使程序完全改写。这样一来编写汇编程序需要注意的事情确实很繁琐，而且辛辛苦苦编出来的程序复用性，可移植性也较差。C++语言使用方便，尤其是递归调用时根本不用自己操心堆栈的问题，由系统自动压栈退栈，但这在某种程度上也造成了一些资源的浪费，影响了效率。所以说 C++更抽象，更接近人的思维。汇编更底层，更接近机器的实际工作。