项目说明文档

数据结构课程设计

**——8**种排序算法的比较案例

​ 作 者 姓 名： 罗吉皓

​ 学 号： 1652792

​ 指 导 教 师： 张颖

​ 学院、 专业： 软件学院 软件工程

​ 同济大学

​ **Tongji University**

目录

1. 分析 4

1.1 项目名称 ： 8种排序算法的比较案例 4

1.2 项目背景 4

1.3 项目功能分析 4

II. 设计 5

2.1 系统设计 5

2.1.1 系统初始化实现 5

III 实现 6

3.1 冒泡排序的实现 6

3.1.1 思路分析 6

3.1.2 具体实现如下： 7

3.2 选择排序实现 8

3.2.1 思路分析 8

3.2.2 具体实现如下： 8

3.3 直接插入排序实现 9

3.3.1 思路分析 9

3.3.2 具体实现如下： 10

3.4 希尔排序实现 10

3.4.1 思路分析 10

3.4.2 具体实现如下： 11

3.5 快速排序实现 12

3.5.1 思路分析 12

3.5.2 具体实现如下： 13

3.6 堆排序实现 14

3.6.1 思路分析 14

3.6.2 具体实现如下： 15

3.7 归并排序实现 15

3.7.1 思路分析 15

3.7.2 具体实现如下： 16

3.8 基数排序实现 18

3.8.1 思路分析 18

3.8.2 具体实现如下： 18

3.9 时间记录实现 20

IV 测试 20

V 总结 22

VI 参考文献 23

1. 分析

*1.1* 项目名称 ： 8种排序算法的比较案例

1.2 项目背景

随机函数产生10000个随机数，用快速排序，直接插入排序，冒泡排序，选择排序的排序方法排序，并统计每种排序所花费的排序时间和交换次数。其中，随机数的个数由用户定义，系统产生随机数。并且显示他们的比较次数。

1.3 项目功能分析

我们的系统首先要先生成N个随机数（N由用户输入），这10000个随机数分别由8种不同的排序算法进行排序，统计其时间及交换次数。为了公平起见，生成的随机数将被保存，8个排序算法排序的都是同一个随机数数组。

II. 设计

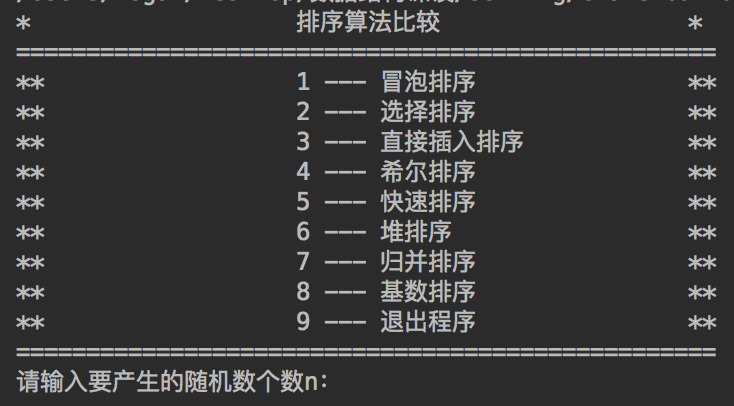
2.1 系统设计

​ 通过初始化界面引导用户进行不同算法的选择，随机函数生成随机数数组。

2.1.1 系统初始化实现



输入功能截屏示例：



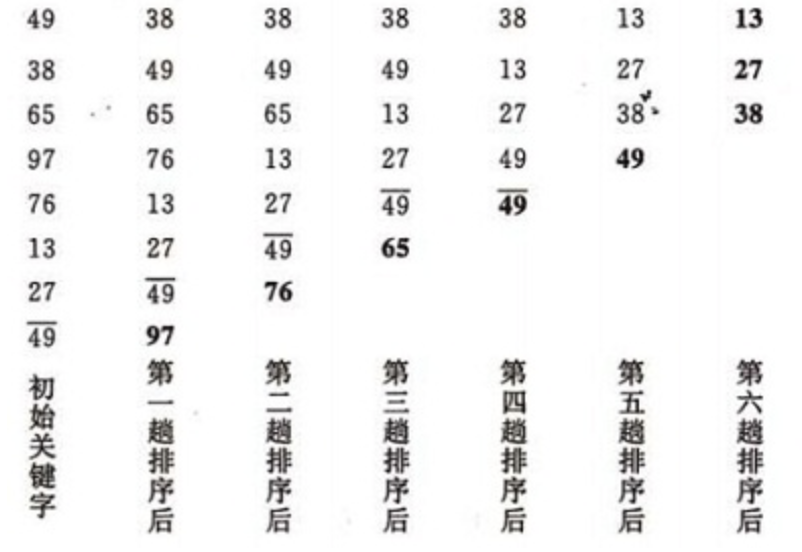
III 实现

3.1 冒泡排序的实现

3.1.1 思路分析

在要排序的一组数中，对当前还未排好序的范围内的全部数，自上而下对相邻的两个数依次进行比较和调整，让较大的数往下沉，较小的往上冒。即：每当两相邻的数比较后发现它们的排序与排序要求相反时，就将它们互换。

冒泡排序的示例：



3.1.2 具体实现如下：

for (int i=0;i<size-1;i++)

      for (int j=i+1;j<size;j++)

      {

          long t=0;

          if (num[i]>num[j])

          //将元素交换 把元素交换到应在的位置

          {

              t=num[i];

              num[i]=num[j];

              num[j]=t;

              changeTimes++;

          }

      }

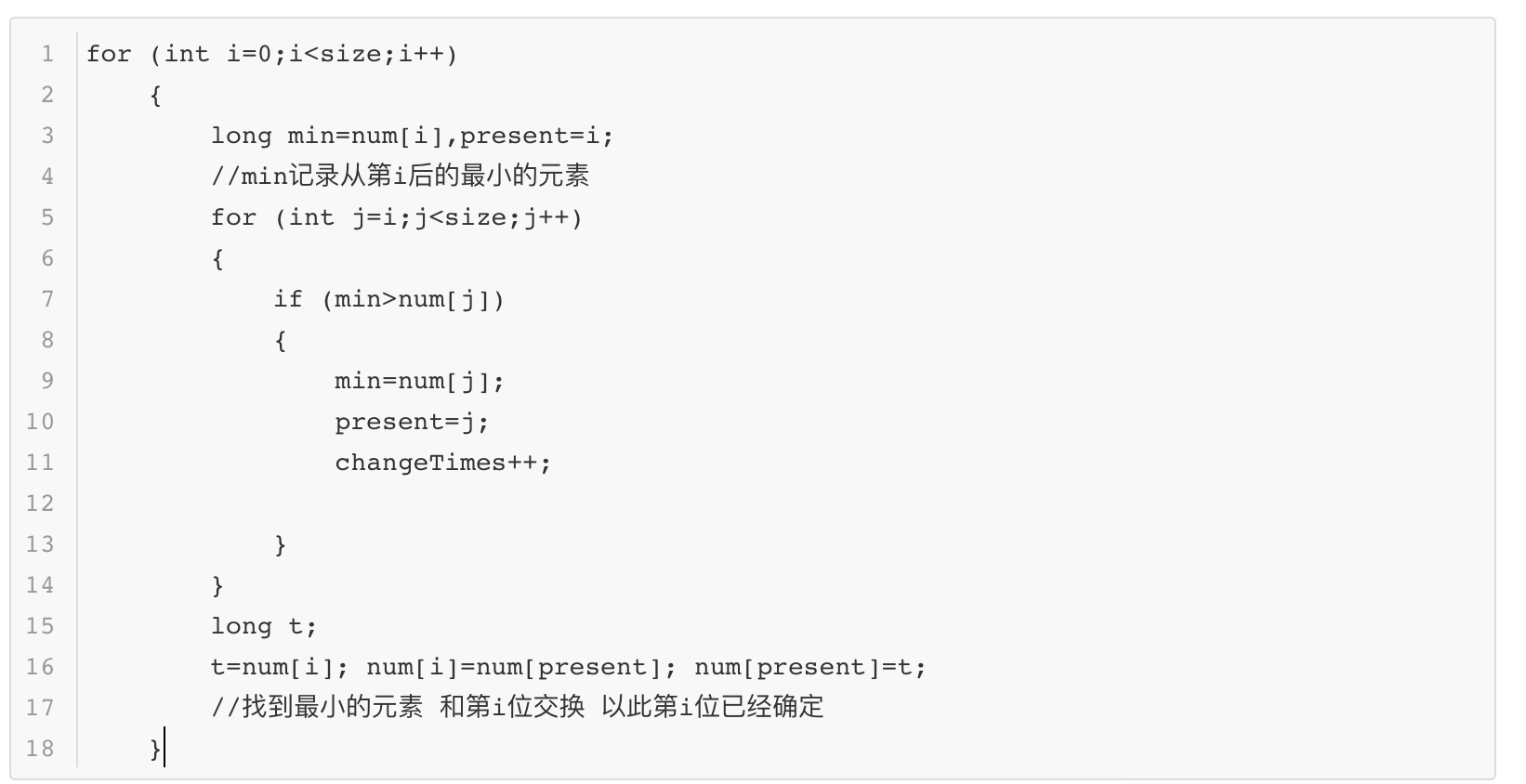
3.2 选择排序实现

3.2.1 思路分析

在要排序的一组数中，选出最小（或者最大）的一个数与第1个位置的数交换；然后在剩下的数当中再找最小（或者最大）的与第2个位置的数交换，依次类推，直到第n-1个元素（倒数第二个数）和第n个元素（最后一个数）比较为止。



3.2.2 具体实现如下：



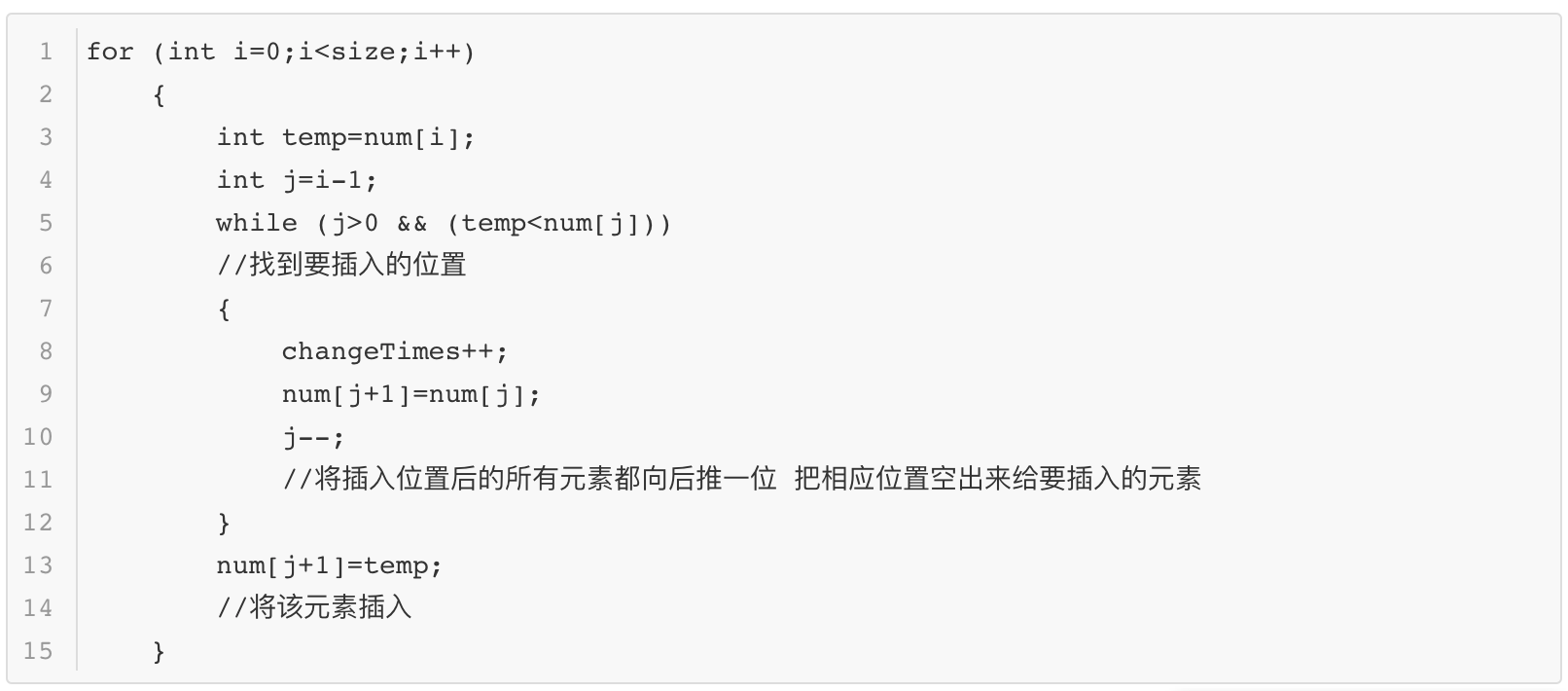
3.3 直接插入排序实现

3.3.1 思路分析

将一个记录插入到已排序好的有序表中，从而得到一个新，记录数增1的有序表。即：先将序列的第1个记录看成是一个有序的子序列，然后从第2个记录逐个进行插入，直至整个序列有序为止。如果碰见一个和插入元素相等的，那么插入元素把想插入的元素放在相等元素的后面。所以，相等元素的前后顺序没有改变，从原无序序列出去的顺序就是排好序后的顺序，所以插入排序是稳定的。



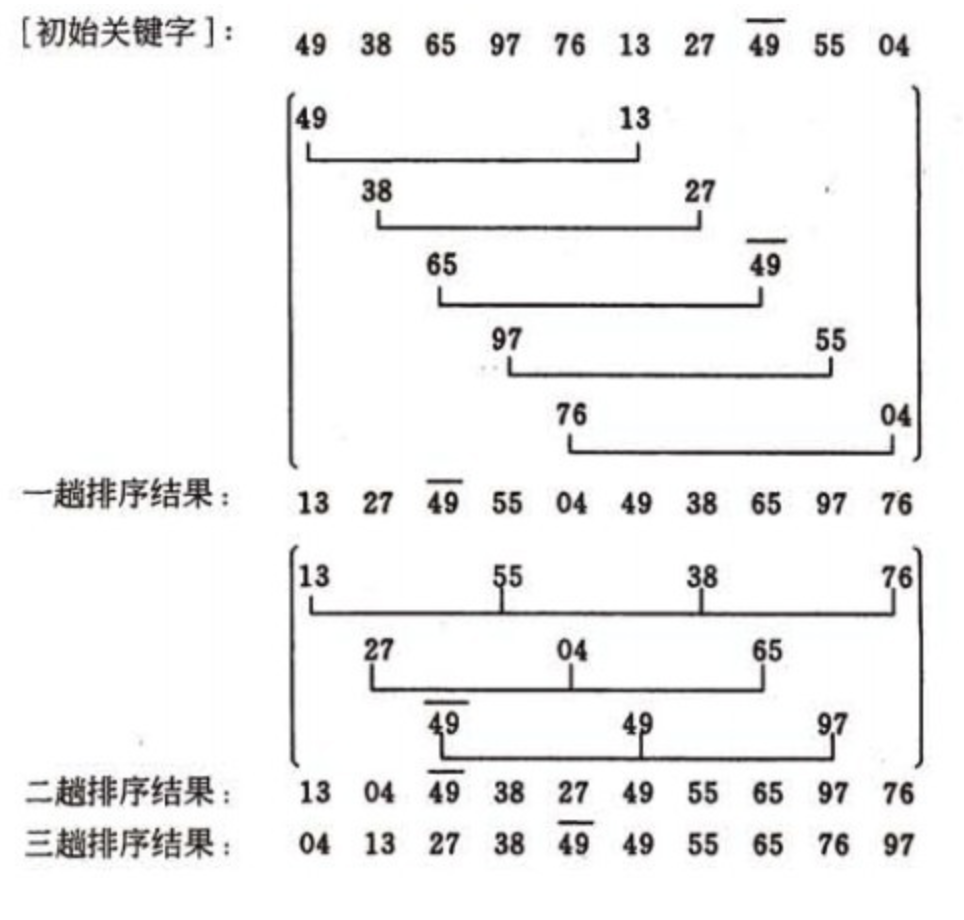
3.3.2 具体实现如下：



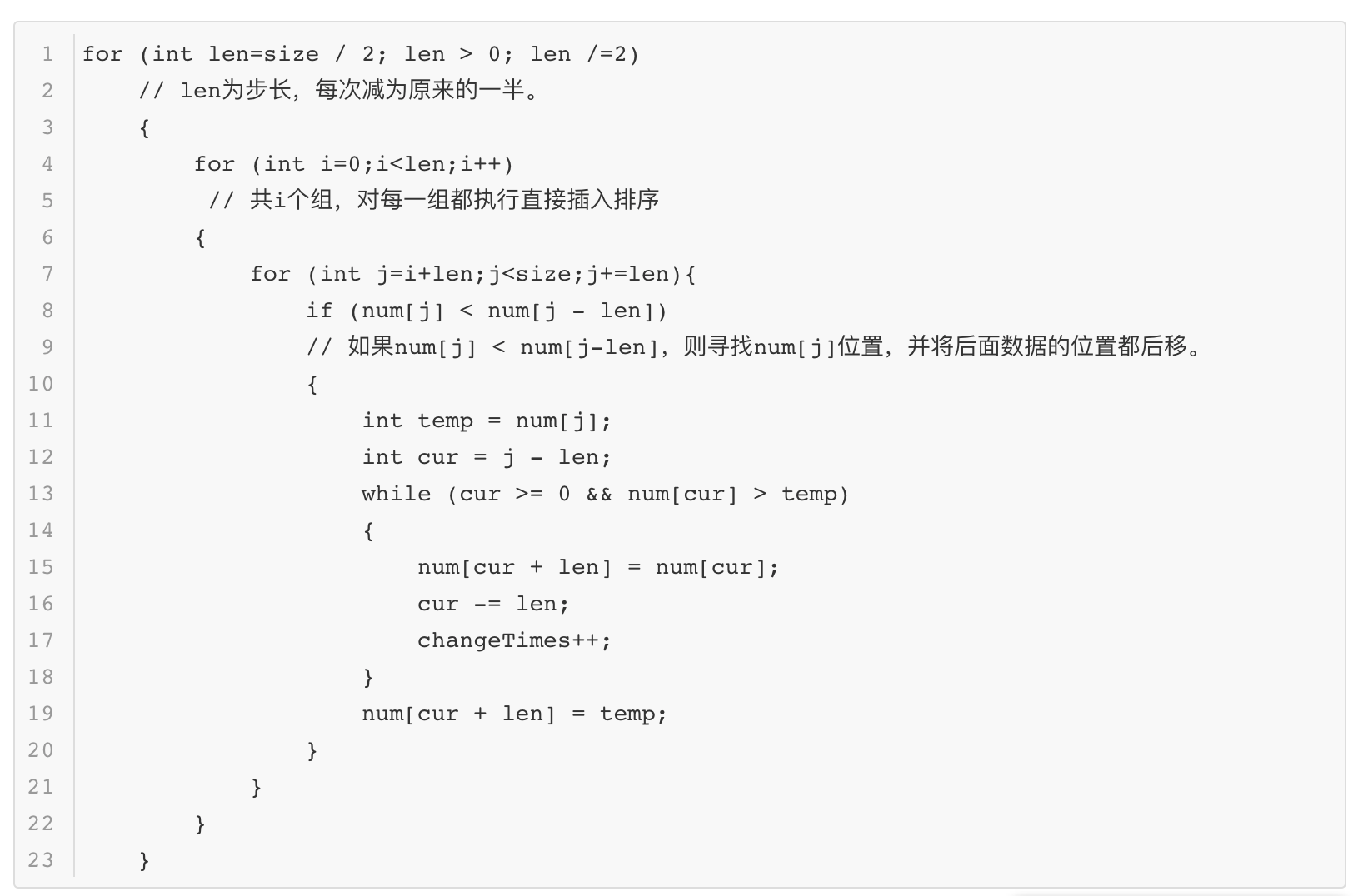
3.4 希尔排序实现

3.4.1 思路分析

先将整个待排序的记录序列分割成为若干子序列分别进行直接插入排序，待整个序列中的记录“基本有序”时，再对全体记录进行依次直接插入排序。每趟排序，根据对应的增量ti，将待排序列分割成若干长度为m 的子序列，分别对各子表进行直接插入排序。仅增量因子为1 时，整个序列作为一个表来处理，表长度即为整个序列的长度。



3.4.2 具体实现如下：



3.5 快速排序实现

3.5.1 思路分析

1）选择一个基准元素,通常选择第一个元素，中间的元素或者最后一个元素,

2）通过一趟排序讲待排序的数组分割成独立的两部分，其中一部分记录的元素值均比基准元素值小。另一部分记录的元素值比基准值大。

3）此时基准元素在其排好序后的正确位置

4）然后分别对这两部分记录用同样的方法继续进行排序，直到整个序列有序。



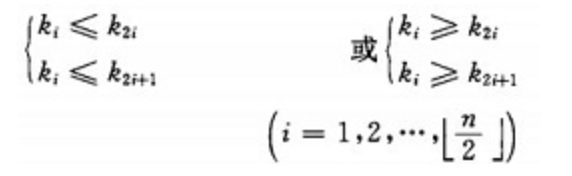
3.5.2 具体实现如下：



3.6 堆排序实现

3.6.1 思路分析

堆排序的主要操作就是建立一个最小堆。堆的定义如下：具有n个元素的序列（k1,k2,...,kn),当且仅当满足

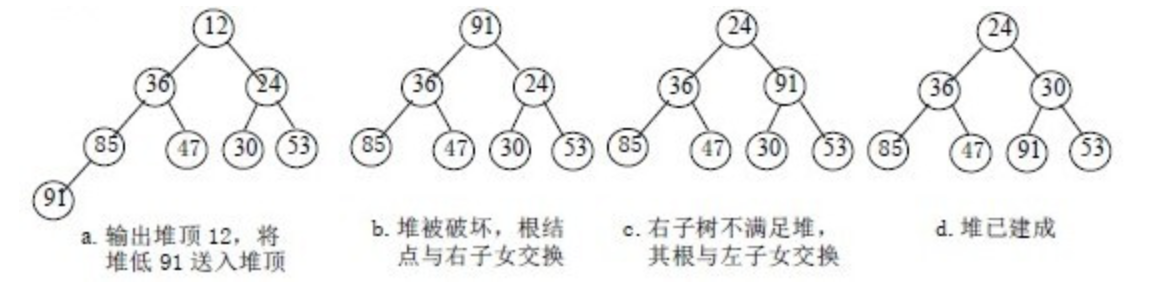


时称之为堆。由堆的定义可以看出，堆顶元素（即第一个元素）必为最小项（小顶堆）。

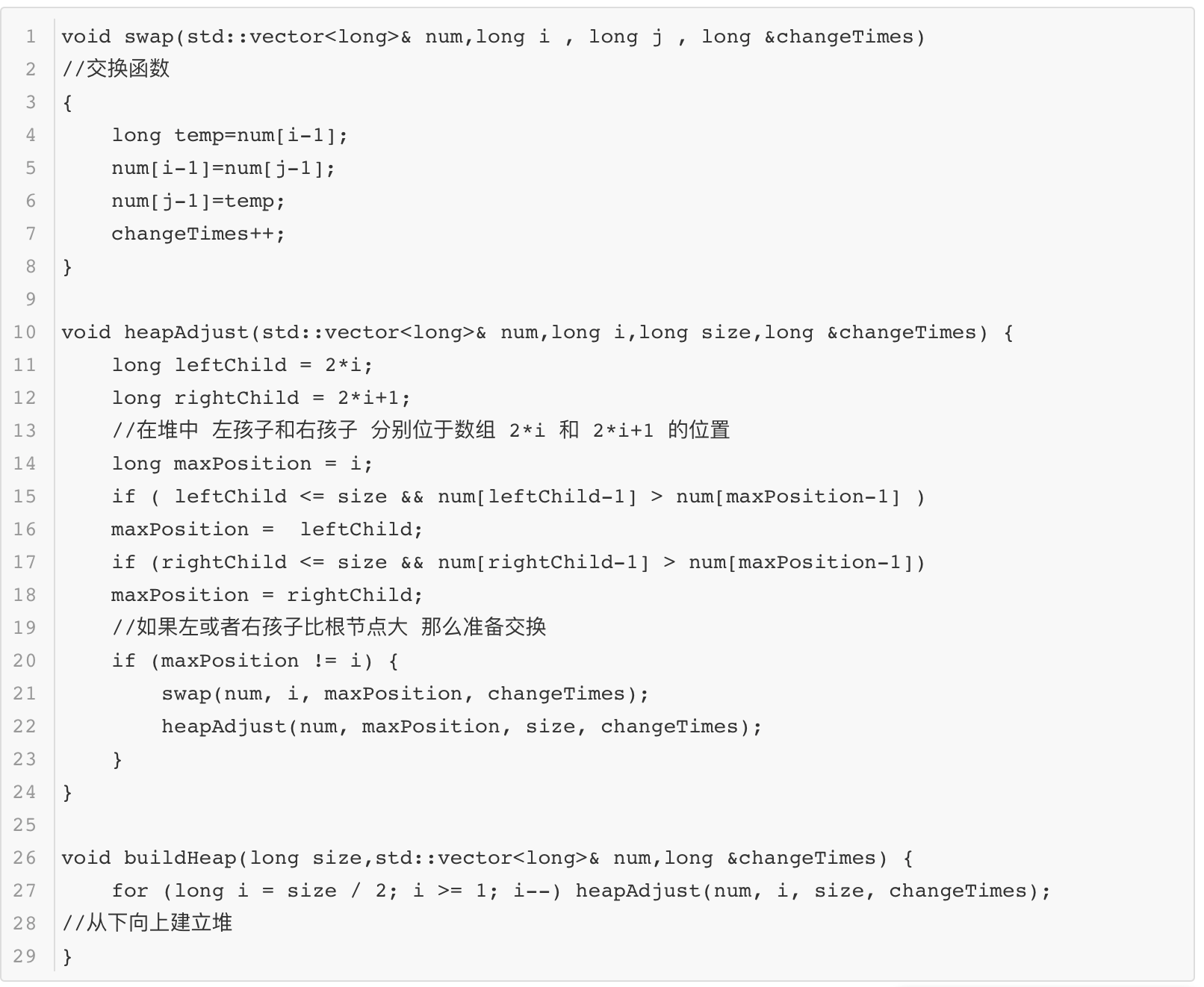
若以一维数组存储一个堆，则堆对应一棵完全二叉树，且所有非叶结点的值均不大于(或不小于)其子女的值，根结点（堆顶元素）的值是最小(或最大)的。

堆排序就是一个不断取出堆顶元素的过程。堆顶元素一定是整个堆里最小的（最大的），取出堆顶元素后，对后面(n-1)个元素重新调整使之成为堆。 依此类推，直到只有两个节点的堆，并对它们作交换，最后得到有n个节点的有序序列。

部分过程如图所示：



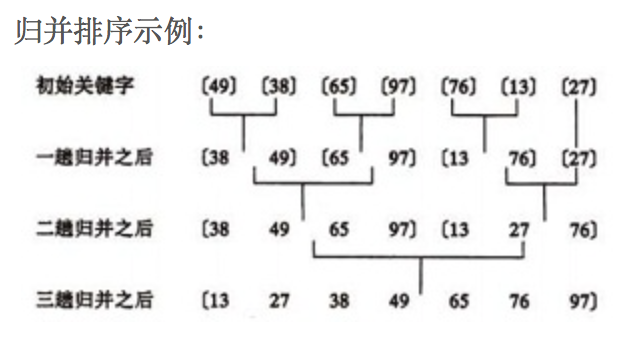
3.6.2 具体实现如下：



3.7 归并排序实现

3.7.1 思路分析

归并（Merge）排序法是将两个（或两个以上）有序表合并成一个新的有序表，即把待排序序列分为若干个子序列，每个子序列是有序的。然后再把有序子序列合并为整体有序序列。



归并排序在最后的合并时，对于两个合并的数组，我们需要一个辅助数组，通过两个数组指针不断遍历，将小的元素放入辅助数组中，并后移，直至两个数组完全并入辅助数组为止。

3.7.2 具体实现如下：



3.8 基数排序实现

3.8.1 思路分析

本项目中，我使用的是最低位优先法来实现基数排序。对数字型或字符型的单关键字，可以看作由多个数位或多个字符构成的多关键字，此时可以采用“分配-收集”的方法进行排序，这一过程称作基数排序法，其中每个数字或字符可能的取值个数称为基数。是按照低位先排序，然后收集；再按照高位排序，然后再收集；依次类推，直到最高位。基数排序基于分配排序，分别收集，所以是稳定的。

3.8.2 具体实现如下：

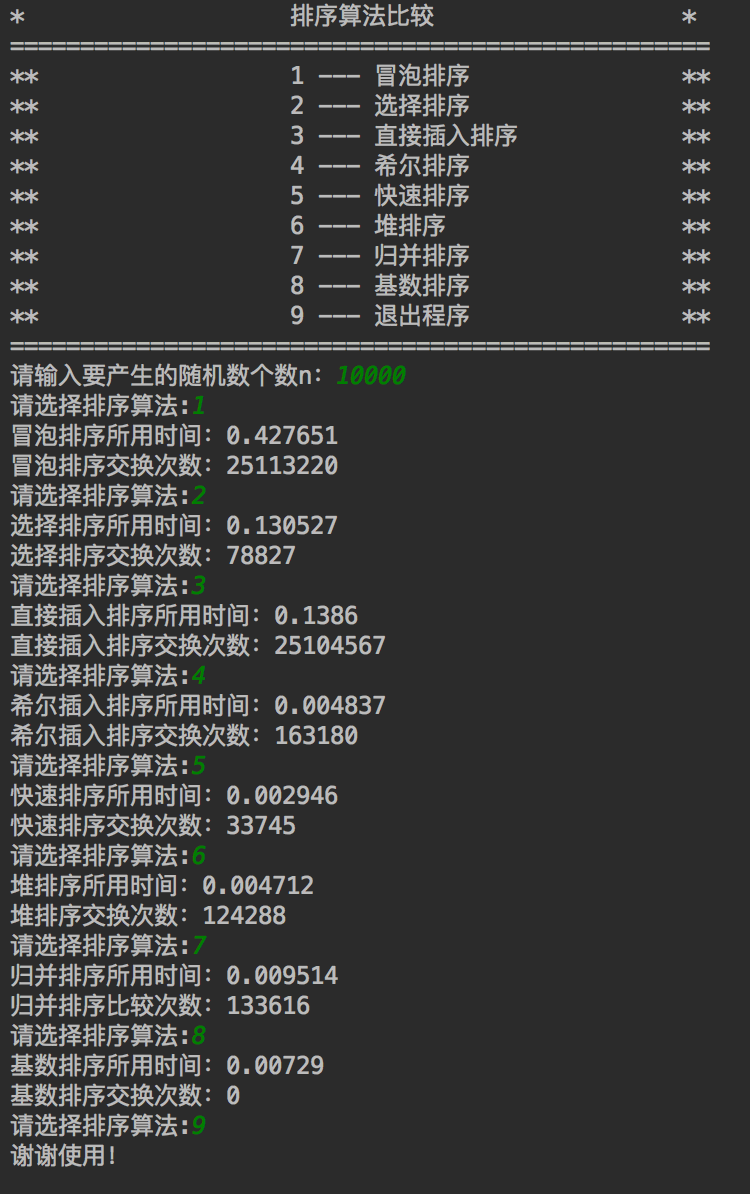


3.9 时间记录实现

为了能够记录不同排序算法所需要的时间，我们在这里引用“time.h“。具体实现如下：



IV 测试



V 总结



由本次课程设计，我们进一步比较了各种不同的排序算法的不同的特性。从时间复杂度来说，直接插入、直接选择和冒泡排序这些比较简单的排序方法大概在O(n^2)的时间复杂度，快速排序、堆排序和归并排序等大概在O(nlog2n)左右，而理想情况下，基数排序可以达到O(n)。从程序的稳定性来说，冒泡排序、插入排序、归并排序和基数排序都比较稳定，相对，选择排序、快速排序、希尔排序、堆排序等就不怎么稳定了。

不同的排序算法各有各的优缺点，在平时使用中，我们要根据实际情况来选择排序算法。

当n较大，则应采用时间复杂度为O(nlog2n)的排序方法：快速排序、堆排序或归并排序序。

当n较大，内存空间允许，且要求稳定性 我们选择使用归并排序。

当n较小，可采用直接插入或直接选择排序。

对于基数排序来说，它是一种稳定的排序算法，但有一定的局限性：1、关键字可分解。2、记录的关键字位数较少，如果密集更好3、如果是数字时，最好是无符号的，否则将增加相应的映射复杂度，可先将其正负分开排序。

VI 参考文献

1.数据结构课本

2.<http://blog.csdn.net/hguisu/article/details/7776068/>