---------------------------------------------------

计划课程安排：

FUNDAMENTAL(7天)         该阶段已结束

OOP(11天)                 该阶段已结束

JAVASE01(10天)        该阶段已结束

JAVASE02(12天)        该阶段已结束

WEBBASIC(11天)       该阶段已结束

JQUERY(2天)           该阶段已结束

DATABASE(6天)     ←当前为该阶段第2天

JDBC(3天)

SERVLETJSP(12天)

SPRINGMYBATIS01(8天)

AJAX(2天)

SPRINGMYBATIS02(12天)

LINUX(2天)

NGINX(2天)

REDIS(2天)

T-STORE(5天)

预计合计：107天

预计剩余：51天

注：上述信息随时可能发生更改，仅供参考。

---------------------------------------------------

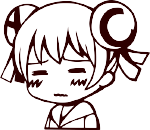
**2019年5月17日**

**Java就业班-第66天**

**9:30实训室2-自习**

到了教室才发现人数不到20人，一半以上的人都直接翘了……虽然只是个自习就是了。

不过留在这里的人几乎都是在扣手机，没有几个真正在学习的。



今天的自习对我来说也就上午几个小时，要学点什么才好呢……

今天就研究一下java的几个基础排序算法吧。

**10:47实训室2-自习**

插入排序：

package Sort;

import java.util.Arrays;

/\*\*

 \* 插入排序（InsertSort）

 \* 插入排序是一种简单直观的排序算法。他的工作原理是通过构件有序序列，

 \* 对于未排序数据，在已排序序列中从后向前扫描，找到相应位置并插入。

 \*

 \* @author Kamui

 \*

 \*/

public class InsertSort {

    private static final int num [] =

       {12,23,53,216,56,234,77,46,99,4,76,48,96,65,74,24,51,19,200,100};

    public static void sort(int arr[]) {

       for (int i = 0; i < arr.length; i++) {

           int temp = arr[i];//由temp取得下标i的数进行比较

           int j = i-1;//使循环的遍历的小标不会超过i

           for ( ; j >= 0 && arr[j] >temp ; j--) {

               //如果前一个下标的数比temp大，则将大于temp的值在数组里往后移动一个下标单位

              arr[j + 1] = arr[j];

           }

           arr[j + 1] = temp ;

       }

       System.out.println(Arrays.toString(arr) + "插入排序");

    }

    public static void main(String[] args) {

       sort(num);

    }

}

算法研究起来还真是耗时间……也可能是我有点蠢…

**11:19实训室2-自习**

希尔排序：

package Sort;

import java.util.Arrays;

/\*\*

 \* 希尔排序（Shell sort）

 \* 也称递增减量排序算法，是插入排序的一种更高效的改进版本。

 \* 希尔排序是非稳定排序算法。

 \* 工作原理是每一轮按照事先决定的间隔进行插入排序，间隔会依次缩小，最后一次一定要是1。

 \*

 \* ※希尔排序属于不稳定排序。

 \* @author Kamui

 \*

 \*/

public class ShellSort {

    private static final int num [] =

       {12,23,53,216,56,234,77,46,99,4,76,48,96,65,74,24,51,19,200,100};

    public static void sort(int arr[]) {

       int i ;

       int j ;

       int temp ;

       int gap = 1 ;

       int len = arr.length;

       while(gap < len / 3) {

           gap = gap \* 3 + 1 ;

       }

       for( ; gap > 0 ; gap /= 3 ) {

           for( i = gap ; i < len ; i++ ) {

              temp = arr[i] ;

              for(j = i - gap ; j >= 0 && arr[j] > temp ; j -=gap) {

                  arr[j + gap] = arr[j] ;

              }

              arr[j + gap] = temp;

           }

       }

       System.out.println(Arrays.toString(arr) + "希尔排序");

    }

    public static void main(String[] args) {

       sort(num);

    }

}

这个希尔排序的代码理解起来好困难，我到最后我还是没能绕明白，即便是看文字能知道个大概原理吧，但是看代码和debug还是没能弄懂原理。总之是先把代码抄下来了，以后再找单独的时间研究吧。

**12:13实训室2-自习**

快速排序：

package Sort;

import java.util.Arrays;

/\*\*

 \* 快速排序(Quick Sort)

 \* 又称划分交叉排序，简称快排。基于分治的思想，是冒泡排序的改进型。

 \* 工作原理：首先在数组中选择一个基准点（该基准点的选取可能影响快速排序的效率），

 \* 然后分别从数组的两端扫描数组，设两个指示标志（low指向起始位置，high指向末尾)，

 \* 首先从后半部分开始，如果发现有元素比该基准点的值小，就交换low和high位置的值，

 \* 然后从前半部分开始扫秒，发现有元素大于基准点的值，就交换low和high位置的值，

 \* 如此往复循环，直到low>=high,然后把基准点的值放到high这个位置。

 \* 一次排序就完成了。然后采用递归的方式分别对前半部分和后半部分排序，

 \* 当前半部分和后半部分均有序时该数组就自然有序了。

 \*

 \* ※快速排序是目前已知的排序算法中最快之一。

 \* @author Kamui

 \*

 \*/

public class QuickSort {

    private static final int num [] =

       {12,23,53,216,56,234,77,46,99,4,76,48,96,65,74,24,51,19,200,100};

    public  static void sort(int arr[]) {

       //分别指定数组的第一个下标和最后一个下标为基准low和high

       recursion(arr , 0 , arr.length - 1);

       System.out.println(Arrays.toString(arr) + "快速排序");

    }

    /\*

     \* 第二次排序-递归分组排序

     \*/

    private static void recursion(int arr[] , int low , int high) {

       if(low < high) {

           int middle = getMiddle(arr, low , high);//将数组进行一分为二

           recursion(arr , low , middle - 1);//对低值组进行递归排序

           recursion(arr , middle + 1 , high);//对高值组进行递归排序

       }

    }

    /\*

     \* 第一次排序-位置交换排序

     \*/

    private static int getMiddle(int arr[] , int low , int high) {

       int tmp = arr[low];//指定low下标所指数值为参照值temp

       while(low < high) {

           /\*

            \* 实现low和high的位置交换

            \*/

           while(low < high &&arr[high] >= tmp) {

              high-- ;//比temp指值大时，high基准的下标往前移

           }

           arr[low] = arr[high];//将比当前low更小的数值附过去

           while(low <high && arr[low] <= tmp) {

              low++;//比temp指值小时，low基准的下标往后移

           }

           arr[high] = arr[low];//将比当前high更大的数值附过去

       }

       arr[low] = tmp ;//最后将参照值放入low的位置，第一次排序结束

       return low ;//返回参照值的下标位置参数

    }

    public static void main(String[] args) {

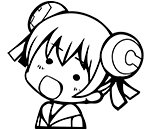
       sort(num);

    }

}

快速排序相对希尔排序要好理解多了，毕竟是基于冒泡算法的升级版。不过好理解归好理解，想要达到掌握的程度，我的理解似乎还是不够透彻。

不过由于时间不多了，下午还要事情要处理，今天就只能先这样了。

感觉每周六的自习都没有什么特别的，也不上课，自习的人也越来越少，要不周六的自习以后就……不更了？

开个玩笑，该学还是要好好学的...真不是我想逃课回家肝奥德赛，不要误会了！

今日感想总结：像我这种蠢的，研究起算法来还真是耗时。既然周末了，也是祝各位周末愉快，我也好好休息一下吧。

今日评分：5/10分



今天就到这里，明天照例休息一天，后天继续投稿。

感谢支持！

================================================================

**Acer程序员群：790482850**

**up主的java学习日记word文件及全部代码（随日记同步更新）：**

Adress：github.com/wt62635