算法设计与分析 测试二

直接插入排序、合并排序、快速排序时间复杂度测试

班级：2015211312

学号：2015211484

姓名：刘佳鑫

- 数据集生成方法

```
void UpSorted(int A[],int n)//升序
{     int i;
      for(i=0;i<=n;i++)
      {     A[i]=i*2;
      }

 }

 void DownSorted(int A[],int n)//降序
 {    int i;
      for(i=0;i<=n;i++)
      {     A[n-i+1]=i*2;
       }
 }

 void Rands(int A[],int n)//随机
 {    int i;
      for(i=0;i<=n;i++)
      {     A[i]=rand()%1000;
       }
 }
```

● 测试手段

通过生成升序、降序、随机数据来测试快速排序、合并排序、直接插入排序三种算法的时间复杂度以及腾挪次数和比较次数。

1.快速排序：

平均情况：随机数列，取第一个数为划分点，进行排序。

最好情况：随机数列，每次随机取划分点，进行排序。

最坏情况：已排好序的序列（此程序采用升序），取第一个数为划分点，进行排序。

同时，上课时要求的特殊序列，一次最好，一次最差。因为想不出来如何生成，所以未能实现。（之后会参考同学的算法进行测试）

2.合并排序：

平均情况：随机数列，进行排序。

最好情况：因为理论上时间复杂度和平均情况相同，借鉴网络上的方法，取几次实验时间最短值取近似最好情况。

最坏情况：因为理论上时间复杂度和平均情况相同，所以只能用取几次实验时间最大值取近似最好情况。

3.直接插入排序：

平均情况：随机数列，进行排序。

最好情况：已经排好序的升序序列。

最坏情况：已经排好序的降序序列。

比较次数：只要存在两个数比较的情况，则算一次比较。

腾挪次数：合并排序中，合并子段时算腾挪次数，但最后把整个排好序的数组挪到原数组中不算腾挪。直接插入排序中，temp 的来回赋值的那两次移动未算入腾挪次数。

● 测试结果

▸ 小数据测试正确性

```
平均情况
快速排序
腾挪次数：19
比较次数25
第1次时间为0秒
原数据：
467 334 500 169 724 478 358 962 464 705
排序之后的：
169 334 358 464 467 478 500 705 724 962
```

```
合并排序
腾挪次数：40
比较次数30
第1次时间为0秒
原数据：
467 334 500 169 724 478 358 962 464 705
排序之后的：
169 334 358 464 467 478 500 705 724 962
```

```
插入排序
腾挪次数：27
比较次数10
第1次时间为0秒
原数据：
467 334 500 169 724 478 358 962 464 705
排序之后的：
169 334 358 464 467 478 500 705 724 962
```

- 数据较多时比较时间和腾挪及比较次数

表格如下

具体每次数据查看请见文件“Result.txt”

	N	100			1000			10000			100000		
	情况	平均情况	最好情况	最坏情况	平均情况	最好情况	最坏情况	平均情况	最好情况	最坏情况	平均情况	最好情况	最坏情况
快速排序	Iterations(K)	1	1	1	1	1	1	1	1	1	1	1	1
	腾挪次数	389	503	99	5669	6713	999	115832	123367	9999	5733368	5851296	停止运行
	比较次数	648	719	4950	9921	10939	49950	170618	169015	49995000	6238549	6359893	
	Duration(sec)	0	0	0	0	0	0.016	0.0079	0	0.5	0.1909	0.172	
合并排序	Iterations(K)	1	1	1	1	1	1	1	1	1	1	1	1
	腾挪次数	695	695	695	9998	9998	9998	136341	136341	136341	1693030	1693030	1693030
	比较次数	564	553	573	8728	8741	8696	123700	123754	123556	1566321	1566339	1566415
	Duration(sec)	0	0	0	0.0015	0	0	0.0047	0	0.016	0.0782	0.062	0.079
插入排序	Iterations(K)	1	1	1									
	腾挪次数	2548	0	4950	253703	0	49950	24913952	0	49995000	溢出	0	704982704
	比较次数	100	101	101	1000	1001	1001	10000	10001	10001	100000	100001	100001
	Duration(sec)	0	0	0	0.0064	0	0.016	0.2965	0	0.641	32.2879	0	64.773

- 测试结论：

在数据量大时，明显可以看出直接插入排序算法相比于其他两个算法时间性能很差，理论平均值应为 O（n^2)。同时，直接插入排序的最坏情况和最好情况也差异甚大。

快速排序和合并排序在我测的几组数据中时间差异不大，可能由于快速排序表现不稳定。

同时发现以下问题：

1. 在数据量较大时，发现合并排序的平均运行时间小于快速排序，和理论值不符。从网上找了答案，是因为快速排序不能保证每一次都是平均分为两部分，所以在时间性能上不稳定，时间也会因此受到影响。
2. 数据量为 100000 时，在最坏情况下，快速排序会停止运行。网上给的解释是，递归调用次数过多，程序内存不够。
3. 同时，数据量为 100000 时，插入排序的腾挪次数过多，因为使用的是 int 型，所以造成溢出。
4. 仍有几个未解决的问题：
   1） 快速排序的最优情况时，腾挪次数和比较次数并不少于平均情况，但数据量大时时间的确比平均情况短。
   2） 快速排序最优情况选择的随机取划分点的方法似乎也不能完全保证最优，因为在最优情况下，其有时花费的时间仍大于归并排序。
   2）插入排序在数据量为 100000 时，腾挪次数在平均情况时溢出，但是最坏情况时却能正常显示。