算法分析与设计第三次作业

黄丛宇 2010212439

October 15, 2010

1 实验环境

- CPU: Intel(R) Core(TM)2 Duo CPU T5870 2.00GHz
- MEM: 1GB
- OS : Debian 5.0 （1GB swap）
- Java: java version "1.6.0_21"

2 排序算法比较

由于Java的Random类之内返回$0 \sim 2^{31}$范围内的随机数，因此，可以使用下面的方法获得$0 \sim 2^{32}$的随机数。

通过Java的Random类的nextInt函数得到两个$0 \sim 2^{31}$范围的随机数，取这两个数的低16位，拼接成一个$0 \sim 2^{32}$范围内的随机数。

```
/**
 * 使用两个随机的数的低位拼接成一个位的随机数。int1632
 * @return
 */
private int getNextUint()
{
    int re = 0;
    int leftPart, rightPart;
    leftPart = random.nextInt();
    rightPart = random.nextInt();
    re = leftPart;
    re <<= 16;
    rightPart &= 0xffff; //取的低位rightPart16
    re += rightPart;
    return re;
}
```

通过上面的函数得到的是$0 \sim 2^{32}$范围内的无符号随机数，Java中没有无符号数，因此只能把这些数当做有符号数来比较。本实验中通过下面的函数实现无符号数的比较。

将一个数看作是十六进制的，从高到低一次比较其十六进制对应的位的值，得到大小关系。代码如下：

```
private static boolean uless(int a, int b)
{
    int aa, bb;
    for(int i = 28; i >= 0; i -= 4){
        /*
         * 将和看作是十六进制的数，和中存储的是abaabb
         * 和的每一位的数值。ab
         * 如，为，那么中存放的就分别是a0x3df32ad3aa
         * 3,d,f,3,2,a,d,3
         */
        aa = (a >> i) & 0xf;
        bb = (b >> i) & 0xf;
        if(aa < bb){
            return true;
        }
        else if(aa > bb){
            return false;
        }
    }
    //a==b
    return false;
}
```

对于32位的数据，在数据量达到10^8时，需要380M的内存。$2 * 10^8$则需要760M的内存。由于我的电脑只有1G的内存和 1G交换区，并且radix sort和merge sort都需要两倍的空间，所以，$2 * 10^8$数据量只测试quicksort。而10^9需要达3.5G的空间，我的电脑无法运行。因此不予测试。

表1是测试结果，取三次均值。表1中，'-'表示没有对这个数据量进行测试，0表示算法的运行时间小于 1ms。

从表中可以看出，在数据量比较小的时候（10,100），理论上算法的运行时间应该都是小于1ms，也就是测出的时间应该是0。但是，表中，Insertion srot和Quicksort对应的时间是1ms。这个时间是程序的加载时间。CPU在运行程序的时候，需要将程序从内存中读到高速缓存中，这个过程需要花费时间。表中的那三个1ms就是由于这个过程造成的。

随着数据量每增加10倍，Insertion sort的运行时间也相应的增加100倍。当数据量大于10^6时，运行时间太长，不予测试。Quicksort和Mergesort的时间复杂度都是$\Theta(nlgn)$，运行时间每次增加12倍左右。Radix sort的时间复杂度是 $\Theta(n)$，运行时间每次增加10倍左右。在排序10^8数据量的时候，由于Mergesort算法申请额外的空间保存中间结果，因此Mergesort的运行时间

表 1 运行结果

数据量	Insertion sort	Quicksort	Mergesort	Radix sort
10	1	1	0	0
100	0	1	0	0
1000	4	2	2	0
10000	302	4	4	1
100000	29247	52	46	10
1000000	3001231	648	593	146
10000000	-	8142	6988	1159
100000000	-	92216	101664	12600
200000000	-	192256	-	-

有较多的增加，这是由于程序需要读写交换区造成的。

在用quicksort排序$2 * 10^8$数据量的时候，程序使用了95%的内存和99%的交换区空间。

3 课后习题

3.1 习题7.3 Stooge sort

a.Ans:

首先，当元素的个数小于等于三个的时候，算法可以正确的对其进行排序。

当元素个数大于三个的时候。在算法中，每次将数组A分成三等分，分别表述为X，Y，Z。其中X为[i, i + k], Y为(i + k, j - k)，Z为[j - k, j]，由于k取的是(j-i)/3的下底，所以，Y的长度会大于等于X和Z的长度。

在第6行的递归调用中，算法将X和Y部分的数据排成有序的。如果其中的某一个元素的最终位置在Z中，那么，这个元素此时只可能在Y中。假如这个元素在X中，那么Y中的所有元素都比这个元素大，那么也就是Y中的所有元素的最终位置都在Z中，但是，这样就造成在最终位置在Z中的元素个数大于Z的长度，因此假设不成立。

此时，最终位置在Z中的元素只存在与Y和Z中。在第7行的递归调用中，可以使Z中的所有元素都在其最终位置上。那么，第8行的递归调用将使X和Y中的元素都在其最终位置上，最终，所有的元素都在其最终位置上。

因此，此算法可以将数组A的元素正确排序。

b.Ans:

由算法可得递归式：

$$T(n) = 3T(2n/3) + \Theta(1)$$

由主定理可得：

$$T(n) = \Theta(n^{log_{1.5}3}) = \Theta n^{2.71}$$

c.Ans:

插入排序的时间复杂度是$\Theta(n^2)$，归并排序，堆排序和快速排序的时间复杂度都是$\Theta(nlgn)$。都要低于这个算法的$\Theta(n^{2.71})$，所以这几个教授浪得虚名。

3.2 习题8.3-4

对所有的元素开根号，然后乘以100，取下底。不同的值得到的结果也不同。这时候元素的范围在0 $\sim$ 100n之间，利用Counting sort可以在O(n)的时间内完成排序。

3.3 习题8.4-4

使用桶排序。对于桶i，存放距离原点的距离在下面的范围内：

$$(\frac{\sqrt{i-1}}{\sqrt{n}}, \frac{\sqrt{i}}{\sqrt{n}}]$$

第一个桶在：

$$[0, \frac{\sqrt{1}}{\sqrt{n}}]$$

这个环的面积是π/n。由于n个点在单位圆上出现的概率是一样的，因此，这个环中点个数的期望值是1。因此这个桶排序的时间复杂度的期望值就是$\Theta(n)$