实验 8 日志

201808010515 黄茂荣

一.实验设计

实验 8 要求我们实现两个查找算法，并对其在 100,1k,10k,100k,1M 的数据规模下对设计出的查找算法进行分析，计算查找运行时间。由于数据较多可以采用随机数生成的方法去随机生成不同规模下的测试数据，进行 100 次成功查找，可以采用随机数生成数组下标存储待查找的数据，100 次失败查找则随机生成测试数据中没有的数据。

实验代码的设计：实验代码中包含一个 set()函数，用作生成测试数据，可以实时根据需要对测试数据进行更新。如果比较不同查找算法的时间消耗，则需注释掉 set(),无需对测试文件进行更新。查找算法分别使用一个函数去实现，在函数内部，每次当开始进行查找时，QueryPerformanceFrequency（获取时间频率）和 QueryPerformanceCounter 进行时间的精确计算。Test()函数通过以文件流对象的为参数的方式，从测试文件中读取测试数据，对随机生成的待查找数据进行查找，每次查找结束后将每一条记录写入输出文件中。主函数利用 for 循环和 switch 语句分别进行 5 次不同规模的数据查找。

二.实验过程记录

2019 年 12 月 18 日

如何记录时间：起初的想法是调用 clock（）函数记录下初始时

间，记录下运行时间差即可，但是由于运行速度很快，时间测量的精度不够。改进后采用如下来测量时间：

```
QueryPerformanceFrequency(&fre);//获得时钟频率
QueryPerformanceCounter(&start);
int low=1,high=n,mid;
while(low<=high)
{
    mid=(low+high)/2;
    count0++;
    if(key<l[mid])
    {
        high=mid-1;
    }
    else if(key>l[mid])
    {
        low=mid+1;
    }
    else
    {
        flag=true;//查找成功
        break;
    }
}
QueryPerformanceCounter(&end);
dft=(double)(end.QuadPart-start.QuadPart)/(double)fre.QuadPart;
```

，dff 为测得的时间，注意需要加相关头文件#include <windows.h>。

文件流的输入与输出：头文件#include <fstream>。文件流读取数据：定义一个 ifsteam 对象，用来进行文件的读取即将文件中的数据读取到控制台中。ifstream in(“文件名.txt”,ios::in)//ios::in 表示写入，然后进行读取,例如将文件中的数字读到数组 a[n]中，即 in>>a[i]。文件流写入数据：定义一个 ofsteam 对象，用来进行文件的写入即将输入或者是处理过的信息存入到文件中去。ofstream out（“文件名.txt”,ios::out）//ios::out 表示读取，然后进行写入,例如将数组 a[n]中的数据写入文件中，即 out<<a[i];out<<"写入数据”。

2019 年 12 月 20 日

Rand（）函数产生的随机数有上限，最大只能产生为 RAND_MAX 的

数据，如果需要产生更大的数，则需要使用其他方法来进行设计。在这里采用一种用 pow()函数产生位数的方法来实现。如下所示

random(x) (int)(((double)rand()/RAND_MAX)*pow(10,rand()%x))

二. 心得和问题

2019 年 11 月 18-22 日

设计实验过程中除了算法本身，还需考虑到其他问题对这次实影响，比如需要考虑如何精确的测量时间，如何生成不同规模的数据文件，怎样去存储测试数据等等，这些都是需要去不断修改，去查资料，才能使自己的代码不断完善。在写代码过程中首先遇到的是时间测量上的精度问题，后来改用更加精确的时间测量。产生随机数由于 rand 函数本身的限制，在数据规模为 1M 的测量的时候，导致其和数据规模为 100k 的时间近似，经过对代码的调试发现了这个问题，并对随机数的产生进行了修改。

三. 实验结果记录及分析

1. 表格中记录的时间单位为 ms

（下表是多次运行程序的平均测试数据，输出文件中是一次测量的结果）

100次成功查找						
数据规模N	顺序查找			二分查找		
	最小查找时间	最大查找时间	平均查找时间	最小查找时间	最大查找时间	平均查找时间
100	0.0001	0.0015	0.000252	0	0.0002	0.000133
1000	0	0.0051	0.001467	0.0001	0.0003	0.000187
10000	0.0004	0.0462	0.012353	0.0001	0.0007	0.000261
100000	0.0003	0.1075	0.034103	0.0001	0.0071	0.000861
1000000	0.0015	0.0978	0.033529	0.0002	0.0029	0.001388

100次失败查找						
数据规模N	顺序查找			二分查找		
	最小查找时间	最大查找时间	平均查找时间	最小查找时间	最大查找时间	平均查找时间
100	0.0003	0.0012	0.000347	0	0.0001	0.000085
1000	0.0026	0.0081	0.002863	0.0001	0.0002	0.000121
10000	0.0262	0.0397	0.026811	0.0001	0.0007	0.000147
100000	0.2619	0.4091	0.273341	0.0001	0.0033	0.000562
1000000	2.6447	3.6397	2.88728	0.0003	0.0043	0.000794

2. 对实验数据进行分析：

纵向对比同一种算法，顺序查找算法，其平均查找时间与数据规模在误差允许的范围内是随着数据规模 N 呈线性增长的，出现某些不是线性增长的 可能是由于数据的偶然性造成的，符合该算法时间复杂度为 O（N）。二分查找算法，其平均查找时间与数据规模在误差允许的范围内是成对数关系增长的，符合时间复杂度为 O(logN).

横向对比两种算法:当数据规模很小时，二分查找算法的时间效率和顺序查找相似，当数据规模很大时，二分查找的时间效率明显高于顺序查找，运行速度快于顺序查找。