**算法分析与设计实验报告**

**第 1 次实验**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 姓名 | 杨杰 | 学号 | 201908010705 | | 班级 | 计科1907 |
| 时间 | 3.17 | 地点 | 软件大楼 | | | |
| 实验名称 | 分治法查找最大最小值 | | | | | |
| 实验目的 | 通过上机实验，要求掌握分治法查找最大最小值算法的问题描述、算法设计思想、程序设计。 | | | | | |
| 实验原理 | 利用分治法查找数组元素的最大值与最小值，根据不同数据规模的测试用例，能准确输出测试用例的最大值和最小值，并计算出程序运行所需要的时间。 | | | | | |
| 实验步骤 | 1. 将问题分解，如果数组中只有一个元素，那么只需要比较该元素与当前的最大值和最小值并进行更新即可。 2. 将问题扩大，数组中有多个元素，则可以通过套用二分法进行拆分，递归将整个数组拆分为多个只有一个元素的小数组，即分解为多个1问题。 3. 最后将每一个小数组得到的结果进行比较，选取其中的最大值和最小值对存储最大值和最小值的变量进行更新。 | | | | | |
| 关键代码 |  | | | | | |
| 算法复杂度分析 |  | | | | | |
| 测试结果  （含运行时间） |  | | | | | |
| 实验心得 | 最初调用rand()函数生成随机数时，并没有指定随机数的种子，导致每次生成的随机数都是相同的，后来调用srand((int)time(NULL))函数，将随机数种子设置为系统时间，解决了这个问题。  clock()函数、GetTickCount()函数、timeGetTime()函数、Boost库中的timer、高精度时控函数QueryPerformanceFrequency()和QueryPerformanceCounter()，以上函数都可以记录程序运行时间，但由于数据量较小时，程序运行时间很短，前三个函数的精度不足以计时，这时应该选用高精度时控函数QueryPerformanceFrequency()和QueryPerformanceCounter()。  经过这次实验，我对于分治法查找最大值最小值的相关代码已基本熟悉，算法知识得到了复习与巩固。在写代码与调试的过程中，在解决问题过程中，丰富了个人编程的经历和经验，提高了个人解决问题的能力。 | | | | | |
| 实验得分 |  | 助教签名 | |  | | |

**附录：完整代码**

*//查找最大最小值.cpp*

#include <iostream>

#include <fstream>

#include <windows.h>

#include <cmath>

using namespace std;

int A[1000001];

void BS(int a[],int left,int right,int& max,int& min)

{

    int middle,max1,max2,min1,min2;

    if(left==right)

    {

        max=a[left];

        min=a[left];

    }

    else

    {

        middle=(left+right)/2;

        BS(a,left,middle,max1,min1);

        BS(a,middle+1,right,max2,min2);

        if(min1>min2)

            min=min2;

        else

            min=min1;

        if(max2>max1)

            max=max2;

        else

            max=max1;

    }

}

int main()

{

    string files[6]={"100.txt","10k.txt","1M.txt",

                    "out\_100.txt","out\_10k.txt","out\_1M.txt"};

    fstream file;

    int len=100,k=0;

    for(int i=0;i<3;i++)

    {

        file.open(files[i],ios::in);

        for(int i=0;i<len;i++)

            file>>A[i];

        file.close();

        int max=A[0],min=max;

        LARGE\_INTEGER time,time0,time1;

        QueryPerformanceFrequency(&time);

        QueryPerformanceCounter(&time0);

        BS(A,0,len-1,max,min);

        QueryPerformanceCounter(&time1);

        file.open(files[i+3],ios::out);

        file<<"最大值为："<<max<<" 最小值为："<<min<<" 时间为："<<1000.0\*(time1.QuadPart-time0.QuadPart)/time.QuadPart<<"ms"<<endl;

        file.close();

        len\*=pow(10,k);

        k+=2;

    }

    return 0;

}

*//测试数据生成器.cpp*

#include <iostream>

#include <time.h>

#include <fstream>

#include <cmath>

#define random(x) (int)(((double)rand()/RAND\_MAX)\*pow(10,rand()%x))

using namespace std;

const int maxn=1e6;*//1M*

int array[maxn+5]={0};*//存储产生的随机数*

void print(fstream &outfile,int n,int A[])*//输出*

{

    for(int i=0;i<n;i++)

        outfile<<A[i]<<endl;

}

int main()

{

    srand((int)time(NULL));*//time seed*

    string files[5]={"100.txt","1k.txt","10k.txt","100k.txt","1M.txt"};

    fstream file;

    for(int i=0;i<maxn;i++)*//随机产生1M的数据*

        array[i]=random(7);

    int k=100;

    for(int i=0;i<5;i++)*//装载5个test文件*

    {

        file.open(files[i],ios::out);*//打开文件*

        print(file,k,array);

        file.close();*//关闭文件*

        k\*=10;

    }

    cout<<"随机数生成成功！"<<endl;

    return 0;

}