数据结构实验报告

评分

满分——5分

学号：2015111823 姓名：钟文清 专业：计算机科学与技术

知识范畴：内部排序 完成日期：2017年06月23日

实验题目：三种平均时间复杂度为O(nlogn)的内部排序算法的实现

实验内容及要求：

输入*n*个整数，用快速排序、堆排序与2路归并排序算法实现由小到大排序并输出排序结果。要求排序数据及排序结果用字符文件实现输入与输出。

实验目的：掌握快速排序、堆排序与2路归并排序算法。

数据结构设计简要描述：

针对三个不同的排序算法，主要采用三个不同的排序函数实现，其中快排是对冒泡排序的一种改进算法，依次移动，而归并排序和堆排序也是按照类似，都采用数组实现元素存储，

算法设计简要描述：

快排以第一个元素为枢轴，分别从后往前和从前往后扫描，满足条件的依次交换，堆排序主要实现初始化堆和调整堆，每次从堆顶开始调整；归并排序采用两两归并实现，知道形成一个长度序列为n的有序序列为止；

输入/输出设计简要描述：

输入：从文件中读入元素个数和元素

输出：三种排序的结果和提示数字

编程语言说明：

采用Codeblocks编写代码，文件读写采用c++格式，内存分配采用c和c++的方法，

主要函数说明：

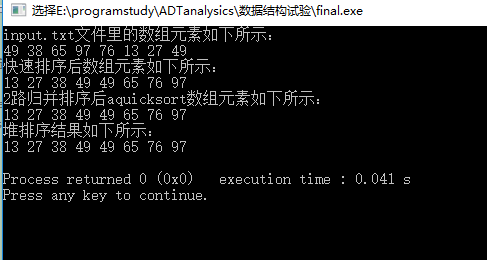
int partition(int arr[],int low,int high)//实现快排

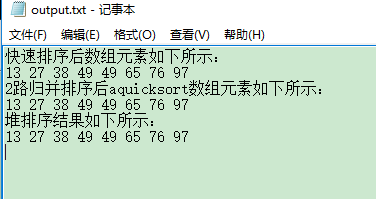
void HeapAdjust(int \*a,int i,int size) //调整堆

void merge(int \*m1, int \*m2, int x, int y, int z)//合并排序

程序测试简要报告：

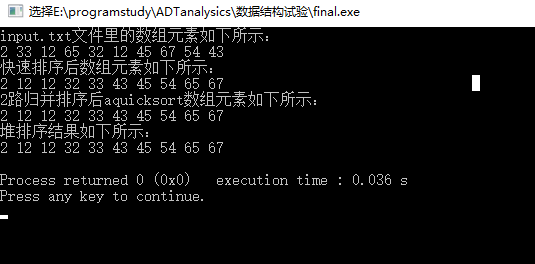
测试用例1：

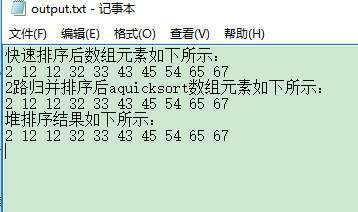




结果符合预期

测试用例2：





结果符合预期

源程序代码：

#include <iostream>

#include <cstdlib>

#include <cstdio>

#include <cmath>

#include <fstream>

#include <ostream>

using namespace std;

//quick sort for array

int partition(int arr[],int low,int high){

int key;

key = arr[low];

while(low < high){

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

high--;

}

if(low < high){

arr[low++] = arr[high];

}

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

low++;

}

if(low < high){

arr[high--] = arr[low];

}

}

arr[low] = key;

return low;

}

void quicksort(int \*arr,int left,int right){

int pos;

if(left < right){

pos = partition(arr,left,right);

quicksort(arr,left,pos-1);

quicksort(arr,pos+ 1,right);

}

}

//堆排序算法实现

void HeapAdjust(int \*a,int i,int size) //调整堆

{

int lchild=2\*i+1;

int rchild=2\*i+2;

int max=i;

if(i<=size/2)

{

if(lchild<=size&&a[lchild]>a[max])

{

max=lchild;

}

if(rchild<=size&&a[rchild]>a[max])

{

max=rchild;

}

if(max!=i)

{

swap(a[i],a[max]);

HeapAdjust(a,max,size);

}

}

}

void BuildHeap(int \*a,int size) //建立堆

{

int i;

for(i=size/2;i>=0;i--)

{

HeapAdjust(a,i,size);

}

}

void HeapSort(int \*a,int size)

{

int i;

BuildHeap(a,size);

for(i=size;i>=0;i--)

{

swap(a[0],a[i]); //交换堆顶和最后一个元素，即每次将剩余元素中的最大者放到最后面

HeapAdjust(a,0,i-1); //重新调整堆顶节点成为大顶堆

}

}

//2路归并实现

void merge(int \*m1, int \*m2, int x, int y, int z)

{

int i,j,k;

i=x;

j=y+1;

for(k=x;k<=z && i<=y && j<=z; k++)

{

if(m2[i]<m2[j])

m1[k]=m2[i++];

else

m1[k]=m2[j++];

}

while(i<=y)

m1[k++]=m2[i++];

while(j<=z)

m1[k++]=m2[j++];

}

void MergePass(int \*m1, int \*m2, int len, int n)

{

int begin=0;

while(begin+2\*len-1<n)

{

merge(m1, m2, begin, begin+len-1, begin+2\*len-1);

begin=begin+2\*len;

}

if(begin+len-1<n)

merge(m1, m2, begin, begin+len-1, n-1);

else

while(begin<n)

{

m1[begin]=m2[begin];

begin++;

}

/\*

for(int t=0;t<10;t++)

printf("%d ",m1[t]);

printf("\n");

\*/

}

void MergeSort(int \*m,int n)

{

int len=1;

int \*tm=(int \*)malloc(sizeof(int)\*n);

while(len<n)

{

MergePass(tm, m, len, n);

len\*=2;

MergePass(m, tm, len, n);

len\*=2;

}

free(tm);

}

int main(){

ifstream fin("input.txt");

if(!fin)

return 0;

int n;

fin>> n;

ofstream fout("output.txt");

if(!fout)

return 0;

int \*a = new int[n]; //测试二路归并排序

int \*aquicksort = new int[n]; //测试快速排序

int \*aheap = new int[n]; //测试堆排序

for(int i = 0; i < n;i++){

fin>>a[i];

}

cout << "input.txt文件里的数组元素如下所示："<<endl;

for(int i = 0; i< n;i++){

cout << a[i]<< " ";

}

cout << endl;

for(int i = 0;i< n;i++){

aheap[i] = aquicksort[i] = a[i];

}

quicksort(aquicksort,0,n-1);

cout << "快速排序后数组元素如下所示："<<endl;

fout <<"快速排序后数组元素如下所示："<<endl;

for(int i = 0; i< n;i++){

cout << aquicksort[i]<< " ";

fout << aquicksort[i]<< " ";

}

cout << endl;

fout <<endl;

MergeSort(a,n);

cout << "2路归并排序后aquicksort数组元素如下所示："<<endl;

fout << "2路归并排序后aquicksort数组元素如下所示："<<endl;

for(int i = 0; i< n;i++){

cout << a[i]<< " ";

fout << a[i]<< " ";

}

cout << endl;

fout <<endl;

cout << "堆排序结果如下所示： "<< endl;

fout << "堆排序结果如下所示： "<< endl;

HeapSort(aheap,n);

for(int i = 0;i < n;i++){

cout << aheap[i] << " ";

fout << aheap[i] << " ";

}

cout << endl;

fout <<endl;

fin.close();

fout.close();

return 0;

}