**华南农业大学**

**《算法分析与设计》课程实验**

**实验1：排序**

年级专业：

学生姓名： 吴嘉豪

学生学号： 202125810321

指导老师： 赵峰

实验时间： 2024 年 04月03 日

一、实验内容

编写合并排序与快速排序程序，对随机生成的实数进行排序，记录运行时间进行比较。

二、实验步骤

1、分别生成10万、20万、40万、70万、100万、200万、400万、600万、900万个随机实数，用两种算法进行排序，记录18个运行时间；

2、对两个算法进行理论分析，推导时间复杂度；

3、分别以列表、图形的方式对两个算法的运行时间、理论用时进行比较，验证时间复杂度与实际运行时间是否一致，给出结论。

4、记录实验中遇到的问题与错误。

三、实验结果

1.

---------快速排序----------

开始执行

程序结束--  当前测试的数组大小为100000

耗费的时间为29ms

----------归并排序----------

开始执行

程序结束--  当前测试的数组大小为100000

耗费的时间为48ms

----------快速排序----------

开始执行

程序结束--  当前测试的数组大小为200000

耗费的时间为49ms

----------归并排序----------

开始执行

程序结束--  当前测试的数组大小为200000

耗费的时间为107ms

----------快速排序----------

开始执行

程序结束--  当前测试的数组大小为400000

耗费的时间为106ms

----------归并排序----------

开始执行

程序结束--  当前测试的数组大小为400000

耗费的时间为214ms

----------快速排序----------

开始执行

程序结束--  当前测试的数组大小为700000

耗费的时间为215ms

----------归并排序----------

开始执行

程序结束--  当前测试的数组大小为700000

耗费的时间为410ms

----------快速排序----------

开始执行

程序结束--  当前测试的数组大小为1000000

耗费的时间为295ms

----------归并排序----------

开始执行

程序结束--  当前测试的数组大小为1000000

耗费的时间为553ms

----------快速排序----------

开始执行

程序结束--  当前测试的数组大小为2000000

耗费的时间为634ms

----------归并排序----------

开始执行

程序结束--  当前测试的数组大小为2000000

耗费的时间为1217ms

----------快速排序----------

开始执行

程序结束--  当前测试的数组大小为4000000

耗费的时间为1581ms

----------归并排序----------

开始执行

程序结束--  当前测试的数组大小为4000000

耗费的时间为2264ms

----------快速排序----------

开始执行

程序结束--  当前测试的数组大小为6000000

耗费的时间为2899ms

----------归并排序----------

开始执行

程序结束--  当前测试的数组大小为6000000

耗费的时间为3460ms

----------快速排序----------

开始执行

程序结束--  当前测试的数组大小为9000000

耗费的时间为5453ms

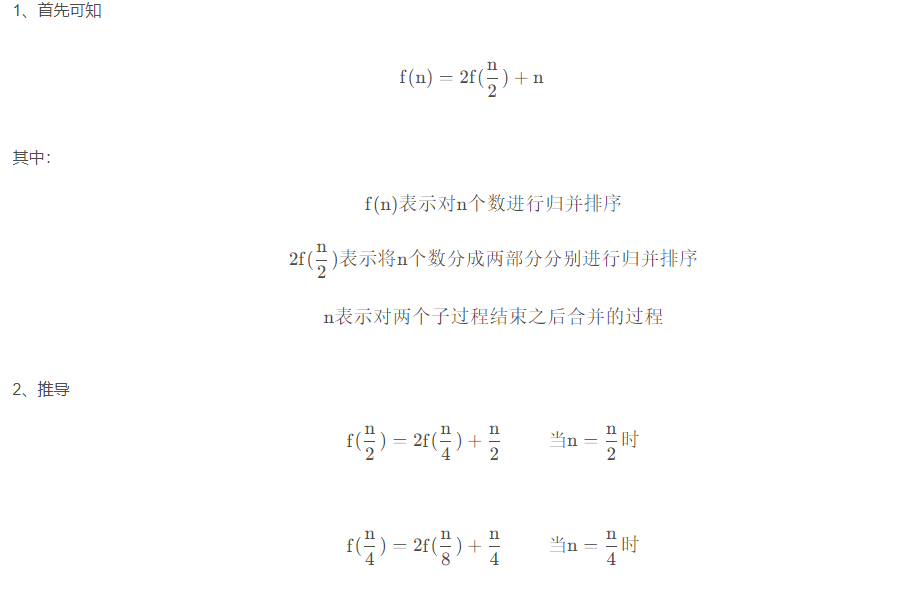
----------归并排序----------

开始执行

程序结束--  当前测试的数组大小为9000000

耗费的时间为5218ms

2.



图形用户界面

中度可信度描述已自动生成

图形用户界面, 应用程序

描述已自动生成

图片包含 图表

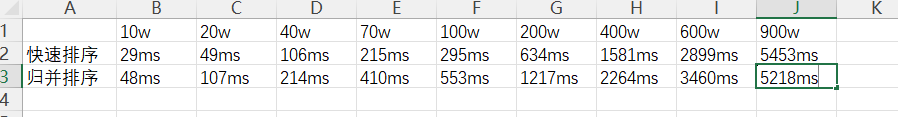
描述已自动生成

对于快速排序：

表格

描述已自动生成

3．



4． 在编写程序的时候， 忘记使用左值引用， 来获取数据， 导致 每次 调用一个函数， 都会重新赋值一遍数组， 耗费时间过多。

快速排序 忘记 先 对 右侧元素进行排序

四、源程序代码

#*include*<iostream>

#*include*<vector>

#*include* <ctime>

#*include* <cstdlib>

#*include*<functional>

using namespace std;

// *int Length = 10000001;*

vector<int> qNums;

vector<int> mergeNums;

void *quickSort*(vector<int> *&* nums){

// *左边的元素比他小， 右边的元素比他大*

    function<int(vector<int> &, int , int)> partition = [&](vector<int>*&* num, int left, int right){

// *选择l 作为 哨兵*

        int l = left,  r = right;

        int tmp = left;

*while*(l < r){

*while*(l < r && num*[*r*]* >= num*[*left*]*) r--;

*while*(l < r && num*[*l*]* <= num*[*left*]*) l++;

*swap*(num*[*l*]*, num*[*r*]*);

        }

// *将 当前 l 和 哨兵的位置进行替换*

*swap*(num*[*l*]*, num*[*tmp*]*);

*return* l;

    };

    function<void(vector<int> &, int, int)> qs = [&](vector<int>*&*  num, int left, int right){

*if*(left >= right) *return* ;

        int mid = *partition(*num, left, right*)*;

*qs(*num, left, mid - 1*)*;

*qs(*num, mid + 1, right*)*;

    };

// *调用函数*

*qs(*nums, 0, nums.*size*() - 1*)*;

}

void *mergeSort*(vector<int> *&* nums){

    function<void(vector<int> &, int , int ,int)> mergeNums = [&](vector<int> *&* num, int left, int mid , int right){

        vector<int> *tmp* (right - left + 1);

        int i = left,  j = mid + 1, k = 0;

*while*(i <= mid && j <= right){

*if*(num*[*i*]* <= num*[*j*]*)

                tmp*[*k++*]* = num*[*i++*]*;

*else* tmp*[*k++*]* = num*[*j++*]*;

        }

//*将剩余的归并*

*while*(i <= mid) tmp*[*k++*]* = num*[*i++*]*;

*while*(j <= right) tmp*[*k++*]* = num*[*j++*]*;

// *归还到原数组*

*for*(k = 0; k < tmp.*size*(); ++k) num*[*k + left*]* = tmp*[*k*]*;

    };

    function<void(vector<int> &, int , int)> spNums = [&](vector<int> *&* num, int left, int right){

*if*(left >= right) *return* ;

        int mid = (left + right) / 2;

*spNums(*num, left, mid*)*;

*spNums(*num, mid + 1, right*)*;

*mergeNums(*num, left, mid , right*)*;

    };

*spNums(*nums, 0, nums.*size*() - 1*)*;

}

/\*\*

*\* 动态代理 的功能*

\*/

void *proxy*(int type,  vector<int> *&* testNumber){

    long startTime = *clock*();

*if*(type == 1){

        cout*<<*"----------快速排序----------"*<<endl*;

        cout*<<*"开始执行"*<<endl*;

*quickSort*(testNumber);

    }

*else*{

        cout*<<*"----------归并排序----------"*<<endl*;

        cout*<<*"开始执行"*<<endl*;

*mergeSort*(testNumber);

    }

    long endTime = *clock*();

// *cout<<<<endl;*

    cout*<<*"程序结束--  当前测试的数组大小为" *+* *to\_string*(testNumber.*size*())*<<endl*;

    cout*<<*"耗费的时间为" *+* *to\_string*(endTime - startTime) *+* "s"*<<endl*;

}

void *solve*(int length) {

*srand*(*time*(nullptr));

// *vector<int> nums(length);*

    qNums.*resize*(length);

    mergeNums.*resize*(length);

*for*(int i = 0; i < length ; ++i){

// *将范围 压缩在 -100000000 ~ 1000000000*

        qNums*[*i*]* = *rand*() % 1000000000;

        mergeNums*[*i*]* = qNums*[*i*]*;

    }

    /\*\*

*\* 快速排序*

*\* 归并排序*

\*/

*proxy*(1, qNums);

*proxy*(2, mergeNums);

}

int *main*(){

    int t = 9;

    int testCases[] = {100000, 200000, 400000, 700000, 1000000, 2000000, 4000000, 6000000, 9000000};

*for*(int i = 0; i < t; ++i){

*solve*(testCases[i]);

    }

*return* 0;

}

五、评语及评分

|  |
| --- |
| 评语及评分  评阅人签名： |