实验预警

本次实验是英特尔酷睿处理器 i7-8750H(CPU天梯图最上方的存在)。

操作系统是 window10 1803家庭版

测试数据为随机产生的随机数， 测试时间度为 冒泡排序 和 快速排序（快排）

因为c++的 clock（）函数的精度为毫秒级，以下时间数据的复杂度都是 ms

数据规模		快速排序
100	0	0
1000	2	0
10000	208	2
100000	25500	12
1000000	2.56400 * e^6	115

【实验分析】

因为数据比较多，但是排序需要有结果显示，因此本次实验中，exe可执行文件中可以选择是否输出数据到exe，同时两个排序的数据都输出到相应的文件里，不会混淆。

数据生成函数

```
#include <iostream>
#include <cstdlib>
#include <ctime>
#include <fstream>
#include <sstream>

using namespace std;

int main() {
    ofstream output;
    int x;
    cin >> x;
    srand((unsigned)time(NULL));
    output.open("data.txt", ios::trunc);

    for(int i = 0; i < x; ++i) {
        output << rand() << ' ';
    }

    return 0;
```

```
}
```

数据排序

```
#include <iostream>
#include <ctime>
#include <fstream>
#include <sstream>
#include <string.h>
using namespace std;

void maopaosort(int a[],int n)
{
    for(int i=0;i<n;i++)
    {
        for(int j=n-2;j>=i;j--)
        {
            if(a[j]>a[j+1])
            {
                int temp=a[j];
                a[j]=a[j+1];
                a[j+1]=temp;
            }
        }
    }
}
int partition(int arr[], int left, int right)  //找基准数 划分
{
    int i = left + 1 ;
    int j = right;
    int temp = arr[left];

    while(i <= j)
    {
        while (arr[i] < temp)
        {
            i++;
        }
        while (arr[j] > temp )
        {
            j--;
        }
        if (i < j)
            swap(arr[i++], arr[j--]);
        else i++;
    }
    swap(arr[j], arr[left]);
    return j;

}

void quicksort(int arr[], int left, int right)
{
    if (left > right)
        return;
    int j = partition(arr, left, right);
```

```
    quicksort(arr, left, j - 1);
    quicksort(arr, j + 1, right);
}


void trans(char a[],int x)
{
    stringstream ss;
    string str;
    ss << x;
    ss >> str;
    str=str+".txt";
    strcpy(a,str.c_str());
}
void trans_maopao(char a[],int x)
{
    stringstream ss;
    string str;
    ss << x;
    ss >> str;
    str=str+"_maopao.txt";
    strcpy(a,str.c_str());
}
void trans_quick(char a[],int x)
{
    stringstream ss;
    string str;
    ss << x;
    ss >> str;
    str=str+"_quick.txt";
    strcpy(a,str.c_str());
}
int main() {
    int n,target,flag_=0,flag=0;
    cout << "要使用的文件大小为（100,1000,10000,100000,1000000）"<<endl;
    cin >> n;
    cout << "是否要输出排序前的原始数据（1： 是      0：否）"<<endl;
    cin >>flag_;
    cout << "是否要输出排序完成的数据（1： 是      0：否）"<<endl;
    cin >>flag;
    int *tempA = new int[n];
    int *tempB = new int[n];
    char path[n],maopao_out[n],quick_out[n];
    trans(path,n);
    trans_maopao(maopao_out,n);
    trans_quick(quick_out,n);
    ifstream input(path);
    ofstream maopao(maopao_out),quick(quick_out);
    if (!input.is_open())
    {
        cout << "can not open this file" << endl;
        return 0;
    }
    for (int i = 0; i < n; ++i) {
        input >> tempA[i];
        tempB[i] = tempA[i];
    }
    if(flag_!=0)
```

```
    {
        cout <<"原始数据如下：" <<endl;
        for(int i=0;i<n;i++)
        {
            if(i%10==0) cout<<endl;
            cout << tempA[i] <<' ';
        }
    }
    input.close();
    cout <<endl<<"正在开始冒泡排序"<<endl;
    clock_t start_time1 = clock();
    maopaosort(tempA,n);
    clock_t end_time1 = clock();
    cout <<"冒泡排序完成"<<endl;
    if(flag!=0)
    {
        cout << "冒泡排序结果为: "<<endl;
        for(int i=0;i<n;i++)
        {
            if(i%10==0) cout<<endl;
            cout << tempA[i] <<' ';
        }
    }
    cout <<endl<< "冒泡排序时间为："
         << static_cast<double>(end_time1 - start_time1) / CLOCKS_PER_SEC * 1000
         << "ms" << endl;
    cout <<"正在开始快速排序: "<<endl;
    start_time1 = clock();
    quicksort(tempB,0,n-1);
    end_time1 = clock();
    cout <<"快速排序完成"<<endl;
    if(flag!=0)
    {
        cout << "快速排序结果为: "<<endl;
        for(int i=0;i<n;i++)
        {
            if(i%10==0) cout<<endl;
            cout << tempA[i] <<' ';
        }
    }
    cout <<endl<< "快速排序运行时间："
         << static_cast<double>(end_time1 - start_time1) / CLOCKS_PER_SEC * 1000
         << "ms" << endl;
    for (int i = 0; i < n; ++i) {
        maopao << tempA[i]<<' ';
        quick << tempB[i]<<' ';
    }
    return 0;
}
```

冒泡排序为基于书中例子的简单排序。

快排是选定分界点进行的快速排序。

实验总结

在数据量比较少的时候，两种排序方法的时间消耗相差不多，但是当数据量提升，快速排序的时间复杂度远远低于冒泡排序的时间复杂度。

冒泡排序的时间复杂度为 O(n)=n^2 , 而快速排序的时间复杂度为 O(n)=nlogn

由实验测得明显看出来，数据量比较大时，好一些的算法消耗明显低于普通算法，这使得我要更加努力学习算法知识，解决大数据问题。