中山大学计算机学院本科生实验报告

（2021 学年第 1 学期）

课程名称：Data structures and algorithms　　任课教师：张子臻

年级	20 级	专业（方向）	软件工程
学号	20337270	姓名	钟海财
电话	13397996670	Email	2940599563@qq.com
开始日期	2021/10/14	完成日期	2021/10/19

1. 实验题目

1) 奖学金

2) 明明的随机数

3）Inversion Number

2.实验目的

1）奖学金：通过分数的比较（总分高的排名靠前，总分相同时语文成绩高的排名靠前，总分和语文成绩都相同时学号小的排名靠前），将不同学号的学生进行排序，并输出前 5 名的学号和分数。

2）明明的随机数：输入数字的个数和要排序的数据，对要排序的数据进行排序并除去重复数据，再输出剩余数字的个数及排好序的的数据。

3）Inversion Number：输入数字的总个数和一堆数字，求出这堆数字中逆序对的总个数。

3.程序设计

1）奖学金

设计思路：

首先定义 5 个 int 型数组：id 记录学号，初始时 id[i]=i+1;Ch,Math,Eng，Sum 分别记录语文分数，数学分数，英语分数,总分。

定义两个函数: void swap(int &a,int &b) 和

bool less_(int i,int j,int id[],int Ch[],int Sum[])

swap 函数用于交换学号的位置，less_函数用于判断 id[i]和 id[j]位置的学号所对应的分数的比较结果，如果 id[i]位置学号的分数比 id[j]低时返回 true。

由于每个学号都有唯一对应的分数，要按分数排序，则只需根据分数排序学号 id[]数组，分数低（用_less 函数判断）的学号排在前面，所以我选择使用选择排序，每次只用交换分数的索引（学号为 id[k]时对应分数的索引为 id[k]-1）。

代码：

```
#include<iostream>
using namespace std ;
void swap(int &a ,int &b){
    int t = a ;
    a = b ;
    b = t ;
}
bool less_(int i,int j,int id[],int Ch[],int Sum[]){
        if(Sum[id[i]-1]<Sum[id[j]-1]) return true ;
        else if(Sum[id[i]-1] == Sum[id[j]-1] ){
            if(Ch[id[i]-1]<Ch[id[j]-1]) return true ;
            if(Ch[id[i]-1]==Ch[id[j]-1]&&id[i]>id[j]) return true ;
            return false;
        }
        else return false ;
    }
```

```
int main(){
    int num;
    while(cin>>num){
        int id[num];//记录学号，排序时对学号进行排序
        int Ch[num] ;
        int Math[num];
        int Eng[num];
        int Sum[num] ;
        for(int i = 0 ; i<num ; ++i){ //读入成绩
            int b,c,d;
            cin>>b>>c>>d;
            id[i] =i+1 ;
// id[i]表示第i位置的学号,根据学号可得对应成绩的索引为 id[i]-1
            Ch[i] = b ;
            Math[i] =c ;
            Eng[i] = d ;
            Sum[i] = b+c+d ;
        }
        for(int i = 0; i<num;i++){//对学号id[]进行选择排序,分数低或学号大的排在前面,
            int  k = i;
            for (int j = i + 1; j < num ; j++){
                if (less_(j,k,id,Ch,Sum)) k = j;
            }
            swap(id[i],id[k]); //交换 i，k 位置的学号
        }
        for(int i = 0 ; i<5;++i){ //最高分在最后面
            cout<<id[num-1-i]<<" "<<Sum[id[num-1-i]-1]<<endl ;
        }
    }
}
```

2）明明的随机数

设计思路：

先对数据进行排序，由于每次读入一个数据，故使用插入排序，实现在读入数据的同时进行插入排序。

令 k=数字个数，再对排序好的数组进行遍历，每次循环时：如果相邻两元素相同，令前一个元素为 NULL，并—k;

最后输出 k 和剩余排序好的不为空的数据。

代码：

```
#include<iostream>
using namespace std ;
void swap(int &a,int &b){
    int t = a;
    a = b ;
    b = t ;
}
int main(){
    int num;
    while(cin>>num){
        int data[num] ;
        for(int i=0; i<num ; ++i){//读入数据并进行插入排序
            cin>>data[i] ;
            for(int j = i ; j >0 &&data[j-1] > data[j] ; --j  )
                swap(data[j-1],data[j]) ;
        }
        int k=num;//记录不重复数据的个数
        for(int i = 0 ; i<num-1;++i){//如果有相同数据，令前一个为空
            if(data[i]==data[i+1]){
                data[i]=NULL;
                --k;
            }
        }
        cout<<k<<endl;
        for(int i = 0 ; i<num;++i) //当数据不为空时输出
            if(data[i] != NULL) cout<<data[i]<<" ";
        cout<<endl ;
    }
}
```

3）Inversion Number

设计思路：

用全局变量 count_记录逆序对的个数，一开始我准备暴力破解，每次读入一个数字都判断该数字与前面数字是否组成逆序对，是则++count_。但发现运行时间过长（此时时间复杂度为 $O(n^2)$），所以需要设计一直时间复杂度更小的算法。

故使用分治算法：将这些数据分为前后两段，总逆序对和=前段逆序对和+后段逆序对和+两段间逆序对和，再用相同方法求前段逆序对和，后段逆序对和（最终分为只有 1 个元素，无逆序对；或只有 2 个元素，此时只有段间逆序对和）。综上，总逆序对和=分治算法过程中所有段间逆序对和之和。

这时想到同样使用分治算法的归并排序，归并排序的操作含有：等分成两组，排序分的两组，合并的排序好的两组（其中排序调用递归使用归并排序）。在合并过程中，由于前后两段每段都是有序的，当前段的某个元素 data[k1]>后段的某个元素 data[k2]时，前段元素 data[k1]~前段结尾 data[end1]这些元素均与 data[k2]形成逆序对，对于后段元素 data[k2]的逆序对和为 end1-k1+1，即 count_+=end1-k1+1.通过合并过程，便算出了段间逆序对和。

代码：

```
#include <iostream>
using namespace std ;
long long count_;
//归并排序
void  merge_sort(int *data , int start ,int end ,int *data2){
if(end-start==0)  return ;//只有 1 个元素，返回
else{//否则进行排序
   int end1;
   int n=end-start+1 ;
   end1 = (start+end)/2;
   int start2= end1+1 ;
   merge_sort(data,start,end1,data2);
   merge_sort(data,start2,end,data2);
//以下为将排序好的两段合并得到的排序好的 start-end
    int k1= start ;
    int k2 =start2 ;
    for(int i = 0 ; i<n;++i){
        if(k1<=end1&&k2<=end){
            if(data[k1]<=data[k2]){
                data2[start+i]=data[k1] ;
                ++k1;
            }else{
```

```
//此时 data[end1]~data[k1] > data[k2],对于 data[k2]有 end1-k1+1 个逆序对
                data2[start+i]=data[k2] ;
                ++k2;
                count_+=end1-k1+1;//由于左右两端都是有序的
            }
        }else if(k1>end1&&k2<=end){
            data2[start+i]=data[k2];
            ++k2 ;
        }else if(k1<=end1&&k2>end){
            data2[start+i]=data[k1];
            ++k1 ;
        }
    }
    for(int i = 0 ; i < n ;++i)//将排好序的部分拷贝入原数组中
    data[start+i]=data2[start+i];
    }
}

int main(){
    int num;
    while(cin >> num) {
        int data1[num];
        int data2[num];
        for (int i = 0; i < num; ++i)
            cin >> data1[i];
        count_ = 0;
        merge_sort(data1,0,num-1,data2);
        cout << count_ << endl;
    }
}
```

4.程序运行与测试

1）奖学金

序号	输入	输出	是 否 通过
1	8 90 67 80 87 66 91 78 89 91 88 99 77 67 89 64 78 89 98 80 89 89 88 98 78	6 265 4 264 8 264 7 258 3 258	是
2	10 12 32 75 92 72 25 91 71 27 72 23 78 90 90 98 98 90 90 23 76 27 74 69 76 45 58 90 99 88 77	6 278 5 278 10 264 8 219 9 193	是

2）明明的随机数

序号	输入	输出	是 否通过
1	10 1 3 2 3 5 9 12 2 1 3	6 1 2 3 5 9 12	是
2	30 12 32 75 92 72 25 91 71 27 72 23 78 90 90 98 98 90 90 23 76 27 74 69 76 45 58 90 99 88 77	21 12 23 25 27 32 45 58 69 71 72 74 75 76 77 78 88 90 91 92 98 99	是

3）Inversion Number

序号	输入	输出	是否通过
1	10 1 3 2 3 5 9 12 2 1 3	16	是
2	30 12 32 75 92 72 25 91 71 27 72 23 78 90 90 98 98 90 90 23 76 27 74 69 76 45 58 90 99 88 77	178	是

5.实验总结与心得

本次 3 个实验为了加快运行速度，使用了不同的方法：实验 1）使用选择排序只对学号进行排序；实验 2）在读入数据时就进行排序所以选择了插入排序；实验 3）则是使用分治算法，运用了归并排序巧妙计算了前后两段间的逆序对和。

在写完实验报告时，突然想到：对于实验 2）用到的插入排序实现在读入数据时就进行了排序也可以用在实验 1）上：

```cpp
//改进的实验 1）奖学金的代码
#include<iostream>
using namespace std ;
void swap(int &a ,int &b){
    int t = a ;
    a = b ;
    b = t ;
}
bool less_(int i,int j,int id[],int Ch[],int Sum[]){
        if(Sum[id[i]-1]<Sum[id[j]-1]) return true ;
        else if(Sum[id[i]-1] == Sum[id[j]-1] ){
            if(Ch[id[i]-1]<Ch[id[j]-1]) return true ;
            if(Ch[id[i]-1]==Ch[id[j]-1]&&id[i]>id[j]) return true ;
            return false;
        }
        else return false ;
    }
int main(){
    int num;
    while(cin>>num){
        int id[num];//记录学号，排序时对学号进行排序
        int Ch[num] ;
        int Math[num];
        int Eng[num];
        int Sum[num] ;
        for(int i = 0 ; i<num ; ++i){ //读入成绩
            int b,c,d;
            cin>>b>>c>>d;
            id[i] =i+1 ;// id[i]表示第 i 位置的学号,根据学号可得对应成绩的索引为 id[i]-1
            Ch[i] = b ;
            Math[i] =c ;
            Eng[i] = d ;
            Sum[i] = b+c+d ;
            for(int j = i ; j >0 &&less_(j,j-1,id,Ch,Sum); --j  )
                swap(id[j],id[j-1]); ;
        }
        for(int i = 0 ; i<5;++i){ //最高分在最后面
            cout<<id[num-1-i]<<" "<<Sum[id[num-1-i]-1]<<endl ;
        }
    }
}
```

附录、提交文件清单