**实验1、《分治算法实验》**

**一、实验目的**

1. 了解分治策略算法思想

2. 掌握快速排序、归并排序算法

3. 了解其他分治问题典型算法

**二、实验内容**

1．编写一个简单的程序，实现归并排序。

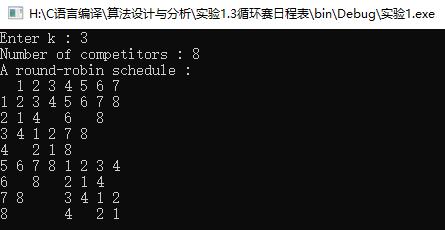
2. 编写一段程序，实现快速排序。

3. 编写程序实现循环赛日程表。设有n=2k个运动员要进行网球循环赛。现要设计一个满足以下要求的比赛日程表：（1）每个选手必须与其它n-1个选手各赛一次（2）每个选手一天只能赛一场（3）循环赛进行n-1天

1. **算法思想分析**
2. 归并排序先是将待排序集合分成两个大小大致相同的集合，分别对每个集合进行排序，递归调用归并排序函数，再是调用合并函数，将两个集合归并为一个排好序的集合。
3. 快速排序先是选择关键数据作为比较量，然后将数组中比它小的数都放到它的左边，比它大的数放大右边，再对左右区间重复上一步，直至各个区间只有一个数。
4. 循环赛日程表先将选手分为两部分，分别排序，再将两部分合并，合并时由于循环赛的规律得知直接将左上角的排序表复制到右下角，左下角的排序表复制到右上角即可。分成两部分时需要利用递归不断分下去直至只剩下一位选手。

**四、实验过程分析**

1. 通过归并算法我对分治算法有了初步的实际操作经验，快速排序与归并算法有很大的相似点，但是在合并时的方法不一样，而循环赛日程表则是思路问题，这个题目编程难点应该在于合并时数组调用的for循环的次数以及起始位置问题。
2. 对于分治算法一般是将大规模问题分解为小问题，通过递归不断分下去，然后对每个小规模用一个函数去求解。适用于小规模独立且易解，可以合并到大问题具有最优子结构的问题。
3. 归并排序和快速排序熟悉书本及PPT基本没有问题，循环赛日程表则是纠结了很久，一开始算法思路并不是十分清晰所以错了很多次，后来想了很久再观察PPT的循环赛日程表得知最终算法，编写代码中遇到了一个小问题，有一部分选手未排序，如图所示：



图中有部分选手未排序，即左下角排序出现了问题，后来直接静态调试，自己按照代码用实际数据去试了一遍，发现是排序时的for循环的次数不对。一开始用最小规模写的代码，即只考虑了两位选手，次数用的是实际数据，但是当规模扩大时该数据是会变化的，所以导致后来的错误。

**五、算法源代码及用户屏幕**

1.（1）算法源码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

归并排序（从小到大）。

codeblock C++

2018/10/16

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std;

//函数声明

void merge\_sort(int a[],int left,int right);

void merge(int b[],int left ,int mid, int right);

int main() {

int q;

cout<<"To invoke the default array, enter 1, and the user-defined array input 2 :"<<endl;

cin>>q;

if(1 == q){

//固定数组

int a[9] = {12, 45, 5, 58, 48, 85, 27, 82, 2};

cout<<"The array before the sort ： "<<endl;

for(int i=0; i<9; i++)

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

cout<<endl;

//对数组进行归并排序

merge\_sort(a, 0, 8); //调用排序函数

cout<<"Sorted array ： "<<endl; //输出排序后的数组

for(int i=0; i<9; i++)

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

cout<<endl;

}

else{

//用户自由输入数组

int n; //数组大小

cout<<"Please enter the size of the array : "; //用户定义数组大小

cin>>n;

int a[n];

cout<<"Please enter the array data : "<<endl;

for(int i=0;i<n;i++){ //用户输入数组数据

cin>>a[i];

}

//对数组进行归并排序

merge\_sort(a, 0, n-1); //调用排序函数

cout<<"Sorted array ： "<<endl; //输出排序后的数组

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

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

cout<<endl;

}

return 0;

}

void merge\_sort(int a[],int left,int right){

if(left<right){

int mid = (left + right)/2; //求数组的中间元素标号

merge\_sort(a,left,mid); //对数组左半部分调用归并排序函数

merge\_sort(a,mid+1,right); //对数组右半部分调用归并排序函数

merge(a,left,mid,right); //将左右部分合并到同时排好序到原数组上

}

}

void merge(int a[],int left ,int mid, int right){

//创建动态数组，暂时存储对左右部分排序后的数组

int \*temp = new int[right-left+1];

int p,q,r; //设置三个变量，一次代替left，mid，right

p=left;

q=mid+1;

r=right;

int i=0;

//比较左右部分数据大小，小的先放入动态数组

while(p<=mid && q<=right){

if(a[p]<=a[q]){ //左半部分数据较小，先入temp数组

temp[i] = a[p];

i++;

p++;

}

else{

temp[i] = a[q]; //右半部分数据较小，放入temp数组

i++;

q++;

}

}

//左半部分有剩余，

while(p<=mid){

temp[i] = a[p];

i++;

p++;

}

//右半部分有剩余

while (q<=right){

temp[i] = a[q];

i++;

q++;

}

//将重新排好序的数组归并到原数组上

for(int j=right; j>=left; j--){

i--;

a[r] = temp[i];

r--;

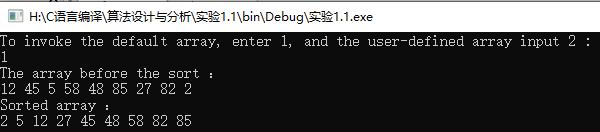
}

delete[] temp; //释放动态数组内存

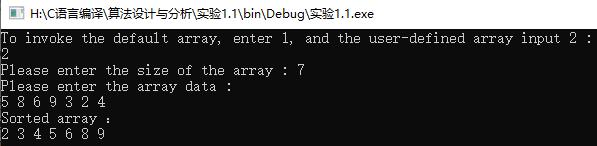
}

1. 用户屏幕

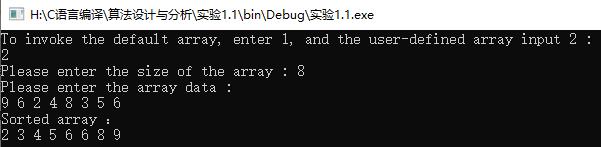
程序采用默认数组数据：12 45 5 58 48 85 17 82 2



用户输入数组大小为7，数据：5 8 6 9 3 2 4



用户输入数组大小8，数据：9 6 2 4 8 3 5 6



2.（1）算法源码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

快速排序。（从小到大）

codeblock C++

2018.10.20

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

using namespace std;

//函数声明

void quicksort(int a[],int left,int right);

int \_partition(int a[],int left,int right);

int main(){

int q;

cout<<"quicksort"<<endl;

cout<<"To invoke the default array, enter 1, and the user-defined array input 2 :"<<endl;

cin>>q;

if(1 == q){

//固定数组

int a[9] = {12, 45, 5, 58, 48, 85, 27, 82, 2};

cout<<"The array before the sort ： "<<endl;

for(int i=0; i<9; i++)

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

cout<<endl;

//对数组进行快速排序

quicksort(a, 0, 8); //调用排序函数

cout<<"Sorted array ： "<<endl; //输出排序后的数组

for(int i=0; i<9; i++)

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

cout<<endl;

}

else{

//用户自由输入数组

int n; //数组大小

cout<<"Please enter the size of the array : "; //用户定义数组大小

cin>>n;

int a[n];

cout<<"Please enter the array data : "<<endl;

for(int i=0;i<n;i++){ //用户输入数组数据

cin>>a[i];

}

//对数组进行快速排序

quicksort(a, 0, n-1); //调用排序函数

cout<<"Sorted array ： "<<endl; //输出排序后的数组

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

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

cout<<endl;

}

return 0;

}

void quicksort(int a[],int left,int right){

int q;

if(left<right){

q=\_partition(a,left,right); //分解

quicksort(a,left,q); //排序

quicksort(a,q+1,right);

}

}

int \_partition(int a[],int left,int right){

//p,q分别代表left，right

int p,q;

p=left;

q=right;

//将关键数据暂时存储到变量中，空出a[p]的位置

int s = a[p];

while(p<q){

//从右端开始找，直到找到一个数小于s,然后赋值给a[p]

while(p<q && a[q]>=s)

q--;

if(p<q)

a[p]=a[q];

//从左端开始找，直到找到一个数大于s，然后将值赋给a[q]

while(p<q && a[p]<=s)

p++;

if(p<q)

a[q]=a[p];

}

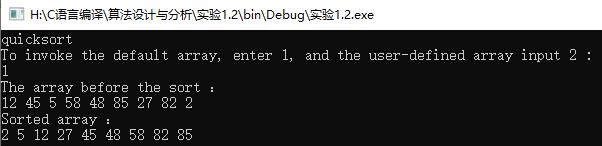
a[p]=s;

return p;

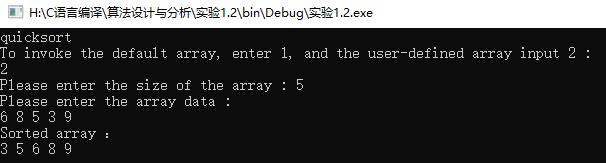
}

（2）用户屏幕

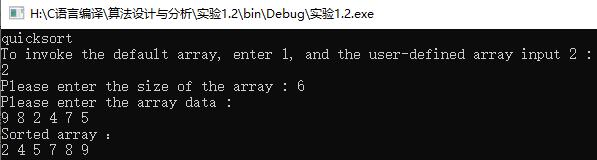
程序采用默认数组数据：12 45 5 58 48 85 17 82 2



自定义输入数组大小：5，数组数据：6 8 5 3 9



自定义输入数组大小：6，数组数据：9 8 2 4 7 5



3.（1）算法源码

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

n=个运动员的循环赛日程表。

codeblocks C++

2018.10.20

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

#include <iostream>

#include<math.h>

#include<iomanip>

using namespace std;

#define a\_size 200

//声明函数

void cycleScheduleSort(int a[a\_size][a\_size],int k,int startRow);

void cs\_merge(int a[a\_size][a\_size],int k,int startRow);

int main()

{

int k;

cout<<"Enter k : ";

cin>>k;

int m = pow(2,k);

cout<<"Number of competitors : "<<m<<endl;

//创建二维数组，存储日程表信息

int a[a\_size][a\_size];

//初始化第一行为天数

for(int i=1;i<m;i++){

a[0][i]=i;

}

//初始化第一列为队伍数

for(int i=1;i<m+1;i++){

a[i][0]=i;

}

//利用循环赛日程表算法排序

cycleScheduleSort(a,k,1);

cout<<"A round-robin schedule :"<<endl;

//输出排序后的日程表

for(int i=0;i<m+1;i++){

for(int j=0;j<m;j++){

if(a[i][j] != 0)

//控制输出格式：左对齐，大小为3

cout<<setiosflags(ios::left)<<setw(3)<<a[i][j]; else

cout<<setiosflags(ios::left)<<setw(3)<<" ";

}

cout<<endl;

}

return 0;

}

void cycleScheduleSort(int a[a\_size][a\_size],int k,int startRow){

if(k>0){

int m = pow(2,k-1); //m记录已排好序的选手数目

//将选手分为两部分，分别排序

cycleScheduleSort(a,k-1,startRow);

cycleScheduleSort(a,k-1,startRow+m);

//将排好序的两部分合并

cs\_merge(a,k,startRow);

}

}

void cs\_merge(int a[a\_size][a\_size],int k,int startRow){

int m = pow(2,k-1);

//将左上角的比赛日程复制到右下角

for(int i=startRow;i<m+startRow;i++){

for(int j=0;j<m;j++){

a[i+m][j+m] = a[i][j];

}

}

//将左下角的比赛日程复制到右上角

for(int i=startRow+m;i<startRow+m\*2;i++){

for(int j=0;j<m;j++){

a[i-m][j+m] = a[i][j];

}

}

}

（2）用户屏幕

