**数据结构课程设计**

**班级：1615403**

**学号：161540205**

**姓名：张佳钰**

**指导老师：孙涵**

6、排序算法比较 （必做）（排序）

[问题描述]

利用随机函数产生8个样本的20000个随机整数（其中之一已经是正序，之一是逆序），利用直接插入排序、折半插入排序，起泡排序、快速排序、选择排序、堆排序，基数排序七种排序方法进行排序（结果为由小到大的顺序），并统计每一种排序算法对不同样本所耗费的时间（即比较次数）。

[基本要求]

（1） 原始数据存在文件中，用相同样本对不同算法进行测试；

（2） 屏幕显示每种排序算法对不同样本所花的比较次数和比较时间；

采用的数据结构：

静态数组

算法思想：

对快排进行了改进算法 将递归改为栈 这样之后不会报错

源程序：

#include<iostream>

#include<fstream>

#include<stack>

using namespace std;

int count[9][9];

int file\_num=8;

#define Space\_Size 20000

#define RD 10

#define Key\_Size 8

typedef int InfoType;

int maxn=Space\_Size-5;

int f[10];

int e[10];

typedef struct SLNode

{

int keys[Key\_Size];

InfoType otherInfo;

int next;

int key;

SLNode()

{

key=0;

//memset(keys,0,sizeof(keys));

for(int i=0;i<Key\_Size;i++)

{

keys[i]=0;

}

next=0;

}

}LNode;

typedef struct

{

SLNode r[Space\_Size+1];

int keynum;

int length;

}SLList;

typedef int ArrType[RD];

void init\_L(SLList \*b,int i,int tmp)

{

int j=0;

b->r[i].key=tmp;

b->r[i-1].next=i;

while(tmp>0)

{

b->r[i].keys[j]=tmp%10;

tmp/=10;

j++;

}

}

typedef struct q\_sort

{

int low;

int high;

q\_sort()

{

low=0;

high=0;

}

q\_sort(int l,int h)

{

low=l;

high=h;

}

}q\_sort;

stack<q\_sort>q\_st;

typedef struct heap\_sort

{

int low;

int high;

heap\_sort()

{

low=0;

high=0;

}

heap\_sort(int l,int h)

{

low=l;

high=h;

}

}heap\_sort;

stack<heap\_sort>h\_st;

void read\_numbers(int\* b,int type)

{

fstream f1;

char file\_name[28];

char base\_name[]="random\_number";

char \*p;

p=file\_name;

char num[]="0";

char base\_prefix[]=".txt";

int j=0;

num[0]='0'+type;

strcpy(file\_name,base\_name);

strcat(file\_name,num);

strcat(file\_name,base\_prefix);

// cout<<file\_name<<endl;

f1.open(file\_name,ios::in);

int i=1;

while(i<=maxn)

{

f1>>b[i++];

}

f1.close();

}

void read\_radix\_numbers(SLList \*b,int type)

{

fstream f1;

char file\_name[28];

char base\_name[]="random\_number";

char \*p;

p=file\_name;

char num[]="0";

char base\_prefix[]=".txt";

int j=0;

num[0]='0'+type;

strcpy(file\_name,base\_name);

strcat(file\_name,num);

strcat(file\_name,base\_prefix);

// cout<<file\_name<<endl;

f1.open(file\_name,ios::in);

int i=1;

int tmp;

while(i<maxn)

{

f1>>tmp;

init\_L(b,i++,tmp);

}

f1.close();

}

void display\_number(int a[])

{

int i=1;

while(i<=maxn)

{

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

if(i%10==0)

cout<<endl;

}

}

/\*

堆排序、直接插入排序、折半插入排序，起泡排序、快速排序、选择排序，基数排序

\*/

void HeapAdjust(int \*a,int low,int high,int col)

{

a[0]=a[low];

for(int j=2\*low;j<=high;j\*=2)

{

if((j<high)&&(a[j]<a[j+1]))

{

j++;

count[0][col]++;

}

if(a[0]>=a[j])

{

count[0][col]++;

break;

}

a[low]=a[j];

low=j;

}

a[low]=a[0];

}

void HeapSort(int \*a,int col)

{

int high;

count[0][col]++;

for(int i=maxn/2;i>0;--i)

{

count[0][col]++;

HeapAdjust(a,i,maxn,col);

}

count[0][col]++;

count[0][col]++;

for(high=maxn;high>1;high--)

{

count[0][col]++;

a[0]=a[1];

a[1]=a[high];

a[high]=a[0];

HeapAdjust(a,1,high-1,col);

}

count[0][col]++;

}

void Straight\_sort(int \*a,int col)

{

for(int i=2;i<maxn;i++)

{

count[1][col]++;

if(a[i]<a[i-1])

{

a[0]=a[i];

count[1][col]++;

for(int j=i-1;a[0]<a[j];--j)

{

a[j+1]=a[j];

count[1][col]++;

}

count[1][col]++;

a[j+1]=a[0];

}

}

}

int search\_place(int \*a,int i,int col)

{

int left,right,mid;

left=1;

right=i-1;

mid=(i-1)/2;

count[2][col]++;

while(left<=right)

{

count[2][col]++;

// cout<<count[2][col]<<endl;

if(a[i]<a[mid])

{

right=mid-1;

}

else if(a[i]>a[mid])

{

count[2][col]++;

left=mid+1;

}

else

{

count[2][col]++;

return mid;

}

mid=(left+right)/2;

// cout<<left<<" "<<right<<" "<<mid<<endl;

// system("pause");

}

return mid;

}

void Straight\_half\_sort(int \*a,int col)

{

// cout<<"p"<<endl;

for(int i=2;i<maxn;i++)

{

count[2][col]++;

if(a[i]<a[i-1])

{

int n;

a[0]=a[i];

n=search\_place(a,i,col);

for(int j=i-1;j>n;--j)

{

count[2][col]++;

a[j+1]=a[j];

}

// cout<<"po"<<endl;

a[j+1]=a[0];

}

}

}

void bubble\_sort(int \*a,int col)

{

int tag=0;

count[3][col]++;

for(int bound=1;tag==0;bound++)

{

count[3][col]++;

count[3][col]++;

tag=1;

for(int i=maxn;i>bound;--i)

{

count[3][col]++;

if(a[i]<a[i-1])

{

a[0]=a[i];

a[i]=a[i-1];

a[i-1]=a[0];

tag=0;

}

}

count[3][col]++;

}

}

int partition(int \*a,int low,int high,int col)

{

int pivotkey;

a[0]=a[low];

pivotkey=a[low];

while(low<high)

{

while(low<high&&(a[high]>=pivotkey))

{

--high;

}

a[low]=a[high];

// low++;

while(low<high&&(a[low]<=pivotkey))

{

++low;

}

a[high]=a[low];

// high++;

// cout<<low<<" "<<high<<" "<<endl;

}

a[low]=a[0];

return low;

}

void qq\_sort(int \*a,int low,int high,int col)

{

q\_sort q;

q\_st.push(q\_sort(low,high));

int pivotpos;

count[4][col]++;

while(!q\_st.empty()&&low<high)

{

pivotpos=partition(a,low,high,col);

if(pivotpos+1<high)

q\_st.push(q\_sort(pivotpos+1,high));

if(pivotpos-1>low)

q\_st.push(q\_sort(low,pivotpos-1));

q=q\_st.top();

q\_st.pop();

low=q.low;

high=q.high;

// cout<<low<<" "<<high<<" "<<pivotpos<<endl;

// system("pause");

count[4][col]++;

}

}

void Q\_sort(int \*a,int low,int high,int col)

{

count[4][col]++;

cout<<high<<" "<<low<<"pp"<<endl;

if(low<high){

int pivotpos;

pivotpos=partition(a,low,high,col);

Q\_sort(a,low,pivotpos-1,col);

Q\_sort(a,pivotpos+1,high,col);

}

}

void quick\_sort(int \*a,int col)

{

// Q\_sort(a,1,maxn,col);

qq\_sort(a,1,maxn,col);

}

void select\_sort(int \*a,int col)

{

count[5][col]++;

for(int i=1;i<maxn;i++)

{

int k=i;

count[5][col]++;

for(int j=i+1;j<=maxn;j++)

{

if(a[j]<a[k])

k=j;

count[5][col]++;

}

count[5][col]++;

if(i!=k)

{

a[0]=a[i];

a[i]=a[k];

a[k]=a[0];

}

count[5][col]++;

}

}

void radix\_display\_number(SLList \*b)

{

int i=0;

while(i<maxn)

{

cout<<b->r[i++].next<<" ";

if(i%10==0)

cout<<endl;

}

}

void Distribute(SLNode \*r,int i,int \*f,int \*e,int col)

{

int p;

int j;

count[6][col]++;

for(j=0;j<RD;j++)

{

f[j]=0;

count[6][col]++;

}

count[6][col]++;

for(p=r[0].next;p;p=r[p].next)

{

j=r[p].keys[i];

if(!f[j]) f[j]=p;

else r[e[j]].next=p;

e[j]=p;

count[6][col]++;

}

}

void Collect(SLNode \*r,int i,int \*f,int \*e,int col)

{

int j;

int t;

for(j=0;!f[j];j++);

r[0].next=f[j];

t=e[j];

count[6][col]++;

while(j<(RD-1))

{

count[6][col]++;

for(j++;(j<(RD-1))&&(!f[j]);j++)

{

count[6][col]++;

}

count[6][col]++;

if(f[j])

{

r[t].next=f[j];

t=e[j];

}

count[6][col]++;

}

r[t].next=0;

}

void RadixSort(SLList \*L,int col)

{

int i;

int j=0;

for(i=0;i<5;i++)

{

count[6][col]++;

Distribute(L->r,i,f,e,col);

Collect(L->r,i,f,e,col);

}

SLNode \*r;

r=L->r;

i=r[0].next;

cout<<"done"<<endl;

/\* while(i!=0)

{

j++;

if(j%10==0) cout<<endl;

cout<<r[i].key<<" ";

i=r[i].next;

}

\*/

}

/\*

堆排序、直接插入排序、折半插入排序，起泡排序、快速排序、选择排序，基数排序

\*/

int main()

{

int a[30005];

// SLList \*b;

// b=new SLList;

memset(count,0,sizeof(count));

cout<<" heap Straight Straight\_half bubble quick select RadixSort"<<endl;

for(int i=0;i<file\_num;i++)

{

SLList \*b;

b=new SLList;

read\_numbers(a,i);

HeapSort(a,i);

read\_numbers(a,i);

Straight\_sort(a,i);

read\_numbers(a,i);

Straight\_half\_sort(a,i);

read\_numbers(a,i);

bubble\_sort(a,i);

read\_numbers(a,i);

quick\_sort(a,i);

read\_numbers(a,i);

select\_sort(a,i);

read\_radix\_numbers(b,i);

RadixSort(b,i);

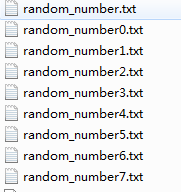
cout<<i+1<<" "<<count[0][i]<<" "<<count[1][i]<<" "<<count[2][i]<<" "<<count[3][i]<<" "<<count[4][i]<<" "<<count[5][i]<<" "<<count[6][i]<<endl;

}

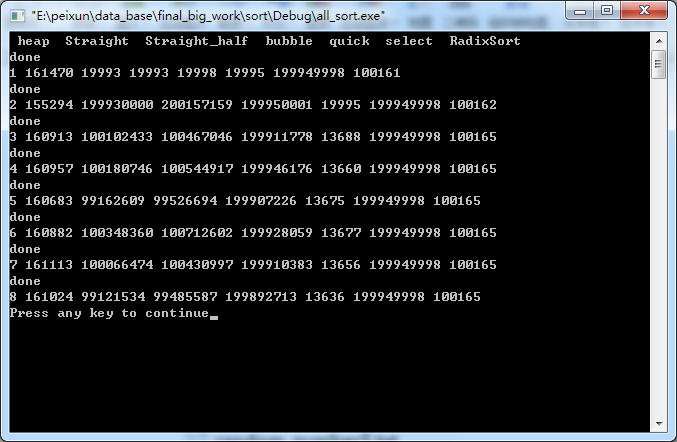
return 0;

}

测试数据：



测试结果：



代码行数：

502行