//顺序表基本运算算法

#include <stdio.h>

#define MAXL 100 //最大长度

typedef int KeyType; //定义关键字类型为int

typedef char InfoType;

typedef struct

{

KeyType key; //关键字项

InfoType data; //其他数据项，类型为InfoType

} RecType; //查找元素的类型

void swap(RecType x, RecType y) //x和y交换

{

RecType tmp = x;

x = y; y = tmp;

}

void CreateList(RecType R[], KeyType keys[], int n) //创建顺序表

{

for (int i = 0;i < n;i++) //R[0..n-1]存放排序记录

R[i].key = keys[i];

}

void DispList(RecType R[], int n) //输出顺序表

{

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

printf("%d ", R[i].key);

printf("\n");

}

//----以下运算针对堆排序的程序

void CreateList1(RecType R[], KeyType keys[], int n) //创建顺序表

{

for (int i = 1;i <= n;i++) //R[1..n]存放排序记录

R[i].key = keys[i - 1];

}

void DispList1(RecType R[], int n) //输出顺序表

{

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

printf("%d ", R[i].key);

printf("\n");

}

//快速排序算法

#include "标头.h"

int count = 0;

int partition(RecType R[], int s, int t) //一趟划分

{

int i = s, j = t;

RecType tmp = R[i]; //以R[i]为基准

while (i < j) //从两端交替向中间扫描,直至i=j为止

{

while (j > i && R[j].key >= tmp.key)

j--; //从右向左扫描,找一个小于tmp.key的R[j]

R[i] = R[j]; //找到这样的R[j],放入R[i]处

while (i < j && R[i].key <= tmp.key)

i++; //从左向右扫描,找一个大于tmp.key的R[i]

R[j] = R[i]; //找到这样的R[i],放入R[j]处

}

R[i] = tmp;

return i;

}

void QuickSort(RecType R[], int s, int t) //对R[s..t]的元素进行快速排序

{

int i;

RecType tmp;

if (s < t && count < 5) //区间内至少存在两个元素的情况

{

count++;

i = partition(R, s, t);

DispList(R, 10); //调试用

QuickSort(R, s, i - 1); //对左区间递归排序

QuickSort(R, i + 1, t); //对右区间递归排序

}

}

/\*

int main()

{

int i,n=10;

RecType R[MAXL];

KeyType a[]={15,18,29,12,35,32,27,23,10,20};

CreateList(R,a,n);

printf("排序前:"); DispList(R,n);

QuickSort(R,0,n-1);

printf("排序后:"); DispList(R,n);

return 1;

}

\*/

int main()

{

int i, n = 12;

RecType R[MAXL];

KeyType a[] = { 57,40,38,11,13,34,48,75,6,19,9,7 };

CreateList(R, a, n);

printf("排序前:"); DispList(R, n);

QuickSort(R, 0, n - 1);

printf("排序后:"); DispList(R, n);

printf("count=%d\n", count);

return 1;

}

1. 对于例题所给例子“对其中较小的4个元素排序”只需调用4次“quicksort”递归，因而只需在判断语句中加入“count<5”的条件即可。
2. 在平均情况下，这样的解法大约比全部元素的快速排序节约k/n的计算量，当k远远小于n是，这种方法比较有效。