算法分析第二周作业

计科卓越班 杨小川 20174179

1. 2.1-2 重写INSERTION-SORT，使之按非升序排序（而不是非降序）

伪代码：

for j = 2 to A.length

key = A[j]

i = j - 1

while i > 0 and A[i] < key

A[i + 1] = A[i]

i = i - 1

A[i + 1] = key

1. 2.2-2 选择算法：首先找出A中的最小元素并将其与A[1]中的元素交换，再找出A中次小元素与A[2]交换。对A中前n-1项元素进行交换。 写出其伪代码。该算法维持的循环不变式是什么？为什么只需要对前n-1个元素运行而不是全部元素？用θ记号给出选择排序的最好情况和最坏情况运行时间。

伪代码：

for i = 1 to A.length - 1

min = i

for j = i + 1 to A.length

if A[j] < A[min]

min = j

temp = A[i]

A[i] = A[min]

A[min] = temp

维持循环的不变式：

每次循环的前i个元素是整个数组中最小的元素且以非递减方式排序。

只需要对前n-1个元素运行而不是全部元素：

因为在执行到最后一步（n-1）时只剩下了最大元素和次大元素，n-1步将次大元素放置在A[n-1]，则最大元素自然就在A[n]。

用θ记号给出选择排序的最好情况和最坏情况运行时间：（假设每次赋值或比较花费时间均为1）

最好情况：原数组本身有序，t = 4(n-1)+n(n-1)/2

最坏情况：原数组逆序，t = 4(n-1)+2n(n-1)/2

所以最好与最坏情况时间复杂度均为θ(n2)

1. 2.3-4 把插入排序表示为如下递归过程。为了排序A[1...n]，我们递归地排序A[1...n-1]，然后A[n]插入已排序的数组A[1...n-1]。为插入排序这个递归版本的最坏情况运行时间写一个递归式。

伪代码：

for i = 2 to A.length

Temp = A[i]

for j = i - 1 to 1

if A[j] > temp

A[j + 1] = A[j] // 元素移位

else

A[j + 1] = temp // 元素插入

Break

最坏情况下即为原数组逆序，每次if语句比较之后都执行移位。

所以T(n):

N = 1时，T(n) = 1；

N > 1时，T(n) = T(n - 1) + 2n - 1 （即每次插入新元素时都进行 n - 1次比较，n - 1次移位 和 1次比较）

1. 编写程序，在数组中同时查找最大值和最小值，用递归程序实现

编程环境：Qt5.X + Windows10 + MinGW7.3.0

#include <QCoreApplication>

#include <iostream>

#include <list>

using namespace std;

struct MAX\_MIN\_PAIR {

int max;

int min;

};

void find(int A[], int l, int r, MAX\_MIN\_PAIR\* p)

{

int temp = r - l;

if (temp > 2)

{

find(A, l, int(temp/2), p);

find(A, l+int(temp/2)+1, r, p);

}

else

{

if (A[l] > A[r])

{

if (A[l] > p->max)

p->max = A[l];

if (A[r] < p->min)

p->min = A[r];

}

else

{

if (A[r] > p->max)

p->max = A[r];

if (A[l] < p->min)

p->min = A[l];

}

}

}

// 返回的结构体即为最大值与最小值。

int main(int argc, char \*argv[])

{

QCoreApplication a(argc, argv);

//以下为测试代码

struct MAX\_MIN\_PAIR mmp;

int A[11] = {1, 7, 8, 5, 6, 13, 0, 11, 34, 4, 2};

mmp.max = mmp.min = A[0];

find(A, 0, 10, &mmp);

cout << mmp.min << " " << mmp.max << endl;

return a.exec();

}

运行结果：

