算法第二次作业

2020年4月2日　20:53

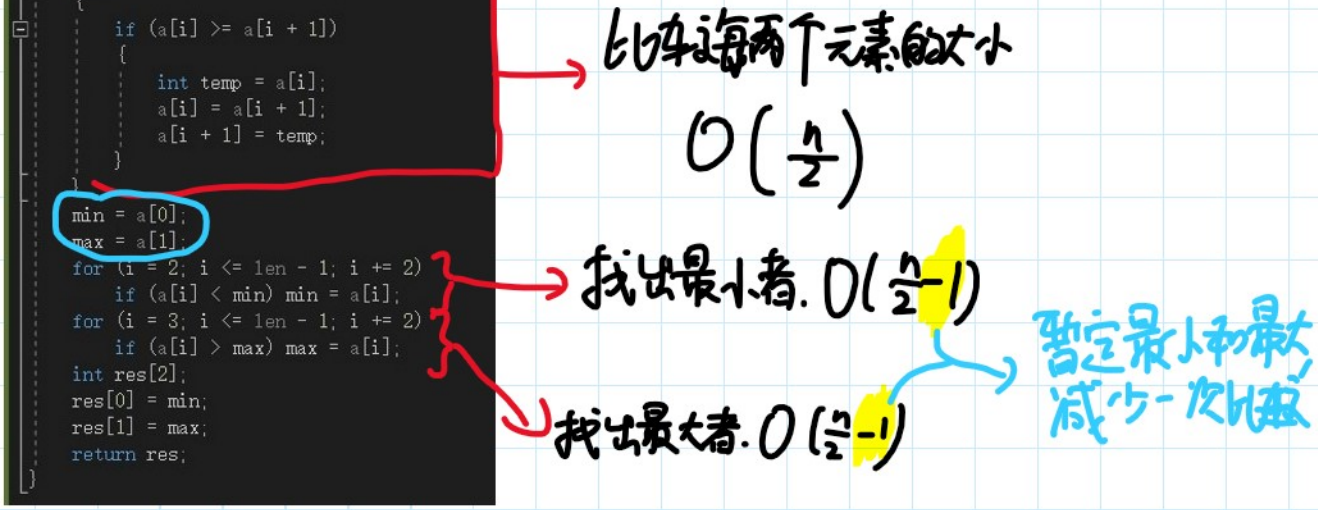
朱一帆-计科智科-058

1. 算法分析题：

给定数组$a[0:n-1]$，试设计一个算法，在最坏情况下用$\left\lceil \frac{3}{2}n-2 \right\rceil$次比较找出$a[0:n-1]$中元素的最大值和最小值。

解：算法主体函数设计如下图所示。设计思路为：先将每两个元素排序，较大元素放前面，较小元素放后面。其次再从较大元素中找出最大者，从较小元素中找到最小者，即求得所需结果。

```
int* twoSort(int a[],int len)
{
    int i;
    int min, max;
    for (i = 0; i <= len - 1; i += 2)
    {
        if (a[i] >= a[i + 1])
        {
            int temp = a[i];
            a[i] = a[i + 1];
            a[i + 1] = temp;
        }
    }
    min = a[0];
    max = a[1];
    for (i = 2; i <= len - 1; i += 2)
        if (a[i] < min) min = a[i];
    for (i = 3; i <= len - 1; i += 2)
        if (a[i] > max) max = a[i];
    int res[2];
    res[0] = min;
    res[1] = max;
    return res;
}
```



总的时间复杂度为$O\left(\frac{3n}{2}-2\right)$测试用例和主函数，以及输出结果如下所示。

```
int main()
{
    int a[8] = { 16, 46, 65, 32, 84, 32, 21, 26 };
    int* res = twoSort(a, 8);
    cout << "min = " << res[0] << "\tmax = " << res[1] << endl;
    std::cout << "Hello World!\n";
}
```

```
C:\WINDOWS\system32\cmd.exe
min = 16        max = 84
Hello World!
请按任意键继续. . .
```

2-10 如果在合并排序算法的分割步骤中，将数组$a[0:n-1]$划分为$\lfloor \sqrt{n} \rfloor$个子数组，每个子数组中有$O(\sqrt{n})$个元素，然后递归地对分割后的子数组进行排序，最后将得到的$\lfloor \sqrt{n} \rfloor$个排好序的子数组合并成所要求排好的数组$a[0:n-1]$。设计一个实现上述策略的合并排序算法，并分析算法的计算复杂性。

解：实现代码如下所示。先使用递归来分割数组，长度为$\sqrt{n}$，直到不能再分割为止。调用Merge函数将分割好的每块再排序合并。思想与二路归并排序类似，只是分割标准不同。

```
void MergeSort(int a[], int left, int right)
{
    if (left < right)
    {
        int i = sqrt(right - left + 1);
        int j;
        for (j = 0; j < i; j++)
            MergeSort(a, left + j * i, left + (j + 1)*i - 1);
        if (left + j * j <= right)
            MergeSort(a, left + j * j, right);

        for (j = 1; j < i; j++)
            Merge(a, left, left + j * i-1, left + (j + 1)*i - 1);
        if (left + j * j - 1 <= right)
            Merge(a, left, left + j * j - 1, right);
    }
}
```

递归调用函数

合并

```
void Merge(int a[], int left, int middle, int right)
{
    int i = left, j = middle + 1, k = 0;
    int* temp = new int[right - left + 1];
    while (i <= middle && j <= right)
    {
        if (a[i] <= a[j])
            temp[k++] = a[i++];
        else
            temp[k++] = a[j++];
    }
    while (i <= middle)
        temp[k++] = a[i++];
    while (j <= right)
        temp[k++] = a[j++];
    int m = 0, n = left;
    for (m, n; n <= right;)
        a[n++] = temp[m++];
}
```

$O(\sqrt{n})$

$O(\sqrt{n})$

$$T(n)=\begin{cases} O(1) & n=1 \\ \sqrt{n}\,T(\sqrt{n})+2\sqrt{n} & n>1 \end{cases}$$

$$\Rightarrow T(n)=O(n\log n)$$

主函数和执行结果如下图所示。

```
int main()
{
    int a[10] = { 15,6,56,9,32,87,32,23,64,48 };
    MergeSort(a, 0, 9);
    for (int i = 0; i < 10; i++)
        cout << a[i] << " ";
    cout << endl;
    return 0;
}
```

```
C:\WINDOWS\system32\cmd.exe
6 9 15 23 32 32 48 56 64 87
请按任意键继续. . .
```

2-1 众数问题（题略，课本P40）

解：算法思想：使用双重循环暴力遍历，并用一辅助数组来记录数字的重复次数。最后读出出现次数最高的数字即为该组数的众数。

```
// zhongshu.cpp : 此文件包含 "main" 函数。程序执行将在此处开始并结束。
//

#include <iostream>
#include <fstream>
using namespace std;

#define MAX_LENGTH 128

int main()
{
        fstream fin("D:\\myPrograms\\zhongshu\\nums.txt");
        int a[MAX_LENGTH];
```

```
    int i = 0;

    while (fin >> a[i++]);                    /*数据录入*/
    a[i] = NULL;
    fin.close();

    int flag[100],j;                          /*统计数字出现的重数*/
    for (j = 0; j < i; j++)
    {
        flag[j] = 1;                          /*初始化每个数字出现一次*/
    }
    flag[j] = NULL;

    int pos =0;
    int* p=a, *q;
    for (p; *p; p++, pos++)
        for (q = p + 1; *q; q++)
            if (*p == *q)
                flag[pos]++;

    int *f = flag;
    int max = 1,posi,count;
    for (count=0,f; *f; f++,count++)
        if (*f > max)
        {
            max = *f;
            posi = count;
        }
    cout << a[posi] << "\n" << max << endl;
}
```

O(n)

O(n)

复杂度$O(n^2)$