上海大学计算机工程与科学学院

数据结构 (2) 个人作业报告

作业：实验四 排序 验证性实验

姓名：林艺珺

学号：18120189

日期：2020 年 3 月 13 日

题目 计数排序算法的实现与验证

假设 n 个关键字互不相同的数据元素存放于数组 a 中，可按下述方法实现计数排序。另设一个数组 C，对数组 a 中每个元素 a[i] 统计关键字小于它的元素数并存入 C[i] 中，则 C[i]=0 的数据元素 a[i] 必为关键字最小的记录，C[i]=1 的记录必为关键字为第二小的数据元素..... 从而根据 C 中元素的大小来重新排列 a 中的记录。完成算法实现后，用大数据量进行测试，并讨论此种方法的优缺点。

测试分析

计数排序的基本思想是对于给定的输入序列中的每一个元素 x，确定该序列中值小于 x 的元素的个数（此处并非比较各元素的大小，而是通过对元素值的计数和计数值的累加来确定）。一旦有了这个信息，就可以将 x 直接存放到最终的输出序列的正确位置上。是一种用空间换时间的排序方法。

以下右侧程序的时间复杂度为 $O(n+k)$（其中 k 是整数的范围），快于任何比较排序算法。是通过对元素值的计数和计数值的累加来确定排序，需要得知所有元素中的最大值才可以完成排序，如超出程序设定的上限，则会出错。以下左侧的程序其本质还是使用比较元素的方式得到结果，但无需知道元素的范围.

计数排序算法没有用到元素间的比较，它利用元素的实际值来确定它们在输出数组中的位置。因此，计数排序算法不是一个基于比较的排序算法，从而它的计算时间下界不再是 $O(nlogn)$。另一方面，计数排序算法之所以能取得线性计算时间的上界是因为对元素的取值范围作了一定限制，即 $k=O(n)$。如果 $k=n^2,n^3,\ldots$，就得不到线性时间的上界。

源码程序

计数排序算法程序 $O(n^2)$

```
#ifndef __Count_SORT_H__
#define __Count_SORT_H__
template <class Type>
void CountSort(Type elem[], int n)
{
    int C[1000000] = {0};
    for (int i = 0; i < n; i++)
//计算关键字小于元素数
        for (int j = 0; j < n; j++)
            if (elem[j] < elem[i])
                C[i]++;
    Type temp[1000000];
    for (int m = 0; m < n; m++)
//复制原始数组
        temp[m] = elem[m];
    for (int k = 0; k < n; k++)
//排序
        elem[C[k]] = temp[k];
}
#endif
```

计数排序算法程序 $O(n)$

```
#ifndef __Count_SORT_H__
#define __Count_SORT_H__
template <class Type>
void CountSort(Type elem[], int n)
{
    int max = 1000;
    int C[1000000] = {0};
    Type temp[1000000];
    for (int m = 0; m < n; m++)
        temp[m] = elem[m];
    for (int i = 0; i < n; ++i)
        ++C[temp[i]];
    for (int i = 1; i <= max; ++i)
        C[i] += C[i-1];
    for (int i = n-1; i >= 0; --i)
        elem[--C[temp[i]]] = temp[i];
}
#endif
```

测试程序

```
#include "Assistance.h"
#include "CountSort.h"
int main()
{
    //10个互不相同的无序数
    int a[] = {55, 63, 38, 21, 66, 92, 16, 46, 87, 72};
    int n = 10;
    cout << "排序前：" << endl;
    Display(a, n);
    CountSort(a, n);
    cout << "排序后：" << endl;
    Display(a, n);

    //100个互不相同的无序数
    int b[] = {89, 84, 5,..., 67, 72};
    n = 100;
    cout << "排序前：" << endl;
    Display(b, n);
    CountSort(b, n);
    cout << "排序后：" << endl;
    Display(b, n);

    //10个互不相同的有序数
    int c[] = {16, 21, 38, 46, 55, 63, 66, 72, 87, 92};
    n = 10;
    cout << "排序前：" << endl;
    Display(c, n);
    CountSort(c, n);
    cout << "排序后：" << endl;
    Display(c, n);

    //1000个互不相同的无序数
    int d[] = {566, 551, 877,..., 515, 157};
    n = 1000;
    cout << "排序前：" << endl;
    Display(d, n);
    CountSort(d, n);
    cout << "排序后：" << endl;
    Display(d, n);

    return 0;
}
```