原创 【排序算法】基数排序详解

2019-09-26 22:58:22 我是一只计算鸡 阅读数 21 更多

笔者今天学习后缀数组，发现需要用到基数排序，所以赶紧学习一波。

基数排序是基于关键字排序，先比较第一个关键字，再比较第二个关键字。。。

举个栗子：假如我们有一堆数，17，8，25，78，123，512，250

实际操作时，是将依次将个位、十位、百位的相同的数放入同一个桶中，高位不足补零。然后再从桶中收集元素。

一、先按个位排序，将个位相同的数放入同一个桶中

0：250

1：

2：512

3：

4：

5：25

6：

7：17

8：8，78

9：

然后我们从桶中收集元素依次得到：250，512，25，17，8，78

二、按十位排序，将十位相同的数放入同一个桶中

0：8

1：512，17

2：25

3：

4：

5：250

6：

7：78

8：

9：

然后我们从桶中收集元素依次得到：8，512，17，25，250，78

三、最后按百位排序，将百位相同的数放入一个桶中

0：8，17，25，78

1：

2：250

3:

4:

5: 512

6:

7:

8:

9:

然后我们从桶中依次收集元素得到：8，17，25，78，250，512

经过如上操作后，数组变得有序了。

我们来考虑一下代码的实现。首先我们要进行多少个关键字次排序，也就是最多有多少位数。然后我们要有10个桶分别代表0~9。然后考虑一下怎桶中的元素。我们可以模拟一个大桶，将所有小桶的元素放进去，记录一个相对位置，即对应位相同的放在一起，我们可以处理桶中元素个数的和元素的对应位相同的数的最后一个数所在的位置。这样我们直接将对应位相同的数倒着放进大桶中就完成收集了。

要是你看不同上述过程也没有关系，我们假设来收集第二次的元素。

0: 8

1: 512，17

2: 25

3:

4:

5: 250

6:

7: 78

8:

9:

首先我们得到十位为0的数有1个，十位为1的有2个，十位为2的有1个，十位为5的数有1个，十位为7的数有1个。

我们处理一波前缀和，分别得到0：1，1：3，2：4，5：5，7：6

我们开始收集，现在元素的位置应该是上一次的位置关系：250，512，25，17，8，78

我们倒着收集：

先收集到250 ，由于十位为5，所以放在十位为5的最后一个位置即5，位置-1，为4，但不会被访问，因为十位为5的只有一个

收集512，由于十位为1，所以放在十位为1的最后一个位置3，位置-1，为2，

收集25，由于十位为2，所以放在十位为2的最后一个位置4，位置-1，为3

收集17，由于十位为1，所以放在十位为1的上一个位置的前一个，即2，位置-1，为1

收集8：由于十位为0，所以放在十位为0的最后一个位置1，位置-1，为0

收集78：由于十位为7，所以放在十位为7的最后一个位置6，位置-1，为5

位置-1：下一次相同位相同的数应该放在的位置，由于我们提前处理了前缀和，所以不会位置越界。

按位置收集以后得到：8，17，512，25，250，78

```
#include<bits/stdc++.h>
using namespace std;
const int maxn = 10000 + 7;
int a[maxn]; // 待排序数组
int n;  // 个数
```

```
int c[10];  // 小桶
int b[maxn]; // 大桶
int countbit(int *a, int n)  // 计算最大位数
{
        int digit = 1;
        int num = 10;
        for(int i = 0; i < n; i++) {
                while(a[i] > num) {
                        digit++;
                        num *= 10;
                }
        }
        return digit;
}
int radixsort(int *a, int n)
{
        int digit = countbit(a, n);
        int radix = 1; // 1 个位 10 十位 100 百位
        for(int i = 1; i <= digit; i++) {  // 对每一位排序
                for(int j = 0; j < 10; j++) c[j] = 0; // 初始化桶
                for(int j = 0; j < n; j++) { //对对应位相同的数计数
                        int k = (a[j] / radix) % 10;
                        c[k] ++;
                }
                for(int j = 1; j < 10; j++) c[j] += c[j-1]; // 每一个相同数字的第一个数的位序
                for(int j = n - 1; j >= 0; j--) { //倒着将每一个数放入大桶中
                        int k = (a[j] / radix) % 10;
                        b[c[k] - 1] = a[j]; // 将a[j] 装入对应的c[k]桶中, b[] 相当于是一个大桶
                        c[k]--;
                }
                for(int j = 0; j < n; j++) a[j] = b[j]; // 将大桶中的元素复制回去
                radix *= 10;  // 比较下一位
        }
        return 0;
}
int main()
{
        while(cin >> n) {
                for(int i = 0; i < n; i++) cin >> a[i];
                radixsort(a, n);
                for(int i = 0; i < n; i++) cout << a[i] << " ";
                cout << endl;
        }
        return 0;
}
```


文章最后发布于: 201