# 实验一 分治与递归

一、问题描述：设有n个互不相同的元素x1,x2,…,xn,每个元素xi带有一个权值wi，。若元素xk满足，且，则称元素xk为x1,x2,…,xn的带权中位数。编写算法使能够在最坏情况下用O(n)时间找出n个元素的带权中位数。

二、问题分析：不做任何限制要求的情况下，这个问题很容易解决，最容易想到的是对所有元素进行排序，然后从小到大遍历，一定能找到带权中位数。但由于算法要求最坏情况下用O(n)时间找出n个元素的带权中位数，故不能对数组进行完全排序，因为快速排序等排序算法无法保证再最坏情况下用O(n)时间找出带权中位数。本题可以使用分治与递归的思想设计算法。

三、算法设计：

设计一个函数WeightMedian，递归调用该函数，每次选中一个元素为基准进行划分，合并已经确定不可能是带权中位数的元素，从而每次减少需要搜索的元素个数，从而保证时间复杂度达到要求。

四、算法实现：每次从所有元素中选取一个元素（本算法选择了中间元素），以该元素分界，对所有元素进行一个划分，使小于它的所有元素排在该元素左侧，使所有大于它的元素排在该元素右侧。对左右所有元素的权值分别求和，由定义知可能出现一下三种情况：

1：左侧所有元素权值之和≤1/2，右侧所有元素权值权值之和≤1/2，此时该元素即为带权中位数。

2：左侧所有元素权值之和≤1/2，右侧所有元素权值之和＞1/2，此时带权中位数一定在该元素右侧的所有元素中。

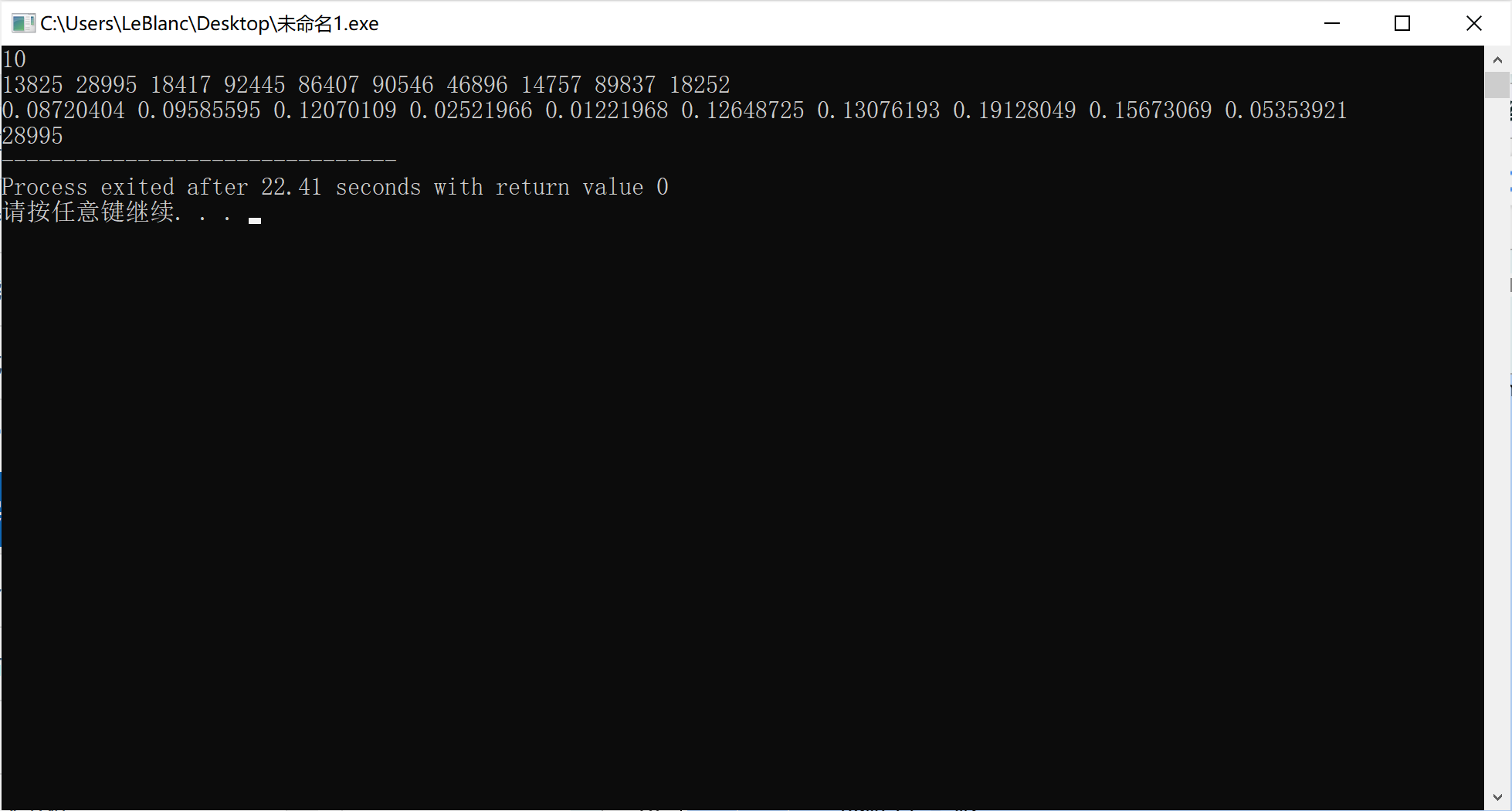
3：右侧所有元素权值之和≤1/2，左侧所有元素权值之和＞1/2，此时带权中位数一定在该元素左侧的所有元素中。

出现情况1时，找到带权中位数，返回该元素即可；

出现情况2时，由于带权中位数一定选中元素在右侧的元素中，可以将左侧所有元素的全部元素的权值都赋到选中元素中，下一次递归只对带有新的权值的选中元素和选中元素右侧的所有元素进行搜索，这样可以有效的减少进行比较的元素个数，减少不必要的搜索。重复直到进行到情况1.

出现情况3与情况2基本相同，不再赘述。

五、运行结果



六、总结：

通过分治与递归的方法，避免了对数组排序的繁琐工作，降低了时间复杂度。通过本次实验让我意识到了在设计分治与递归算法时，递归函数中应当设置一个随递归深度变化的参数，本实验设计的这两个参数为 int length、int index，这些参数跟递归的深度有关，这些参数数值的不同使得函数在处理时候的不同。

七、附录与说明：

void WeightMedian(int length, vector<int>num, vector<double>weight, int index)

{

if (length == 1)

cout << num[0];

double sumsmall = 0;

int k = index + (length - 1) / 2;//k表示选中元素在数组中的位置，随递归深入而改变

int mid = num[k];

int i, t = 0;

for (i = index;i < index + length;i++)

{

int m;double n;

if (num[i] < mid)

{

m = num[index + t];

num[index + t] = num[i];

num[i] = m;

n = weight[index + t];

weight[index + t] = weight[i];

weight[i] = n;

if (index + t == k)

k = i;

t++;

}

}

int m = num[k];num[k] = num[index + t];num[index + t] = m;

double n = weight[k];weight[k] = weight[index + t];weight[index + t] = n;

k = index + t;

mid = num[k];

for (int i = index;i < k;i++)

sumsmall += weight[i];

if (sumsmall < 0.5 && (1 - weight[k] - sumsmall) < 0.5)

{

if (sumsmall <= 0.5)

{

cout << num[k];

return;

}

}

else if (sumsmall > 0.5)

{

weight[k] = 1 - sumsmall;

WeightMedian(t + 1, num, weight, index);

}

else if (1 - sumsmall - weight[k] > 0.5)

{

weight[k] = sumsmall + weight[k];

WeightMedian(length - t, num, weight, k);

}

}

设计了WeightMedian函数来完成该功能。

函数中各参数含义如下：

Int length：每次需要搜索的元素个数，随递归次数增加而减小。

vector<int>num：存储元素大小的数组，不发生改变。

vector<double>weight：存储元素权值的数组，特定位置随递归深度不同发生改变，

int index：每次搜索的起始位置，与length一起确定搜索的范围。