

算法分析与设计第一次实验

问题描述

如下：

设有 n 个互不相同的元素 x_1, x_2, \dots, x_n ，每个元素 x_i 带有一个权值 w_i ，且 $\sum_{i=1}^n w_i = 1$ 。若元素 x_k 满足 $\sum_{x_i < x_k} w_i \leq \frac{1}{2}$ 且 $\sum_{x_i > x_k} w_i \leq \frac{1}{2}$ ，则称元素 x_k 为 x_1, x_2, \dots, x_n 的带权中位数。请编写一个算法，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。↵

问题分析

我们每次从数组中随机选取一个作为 pivot，将原数组划分为小于 pivot 和大于 pivot 两部分。分别计算这两部分数组的权重之和，**则我们要找的带权中位数必然位于权重之和大于 1/2 的那半边**。理想情况下，每次都可以排除大约一半的元素。可以证明，这个算法的平均时间复杂度是 $O(n)$ 。我们在此基础上加以改进，使得该算法的最坏复杂度也是 $O(n)$ 。我们可以通过选择“中点”作为 pivot，以此为基准对数组进行划分，从而保证在后续搜索过程中复杂度不会退化。每一次都可以把数组缩小到原来的 $1/f$ ($0 < f < 1$)，因此算法复杂度就是 $T(n) = O(n)$ 。

算法设计

- input: 输入所需要的数据。为方便起见我们初始将数组每五个划分一组，初始权值 $k=0.5$ ，初始值 $p=0$, $r=n-1$ 。
- step 1: 如果数组的尺寸小于5，那么我们就没有必要划分数组。直接进行如下步骤：

```
sort(arr, arr+n);
```

然后即可找到带权中位数。

如果尺寸大于5，那么进入第二步。

- step 2: 对每个划分好的数组，使用 sort 找到每一组的中位数，记录之后找出这些数字中的中位数作为 pivot。

- step 3:使用新的pivot对以上数组做新的划分分成两部分, $arr[0, l]$ 和 $arr[l + 1, n - 1]$,
- step 4:对两个数组分别计算权重总和, 得到 w_1 和 w_2 。如果 $w_1 > 1/2$ 则对前半部分数组回到step 2进行操作, 如果是 $w_2 > 1/2$ 则对后者进入step 2 进行操作, 直到找到中位数为止。

最后输出即可。

代码实现

```
#include <iostream>
#include <vector>
#include <algorithm>
using namespace std;
struct Node
{
    int value;
    double weight;
};
int partition(vector<Node>&A, int p, int r)
{
    int less = p - 1, i;
    int pivot = p + rand() % (r - p + 1);
    for (i = p; i <= r; i++)
    {
        if (A[i].value < A[pivot].value)
        {
            less++;
            swap(A[less], A[i]);
        }
    }
    swap(A[less + 1], A[pivot]);
    return less + 1;
}
int WeightedMedian(vector<Node>&A, int p, int r)
{
    if (p == r)
        return A[p].value;
    if (r - p == 1)
    {
        if (A[p].weight == A[r].weight)
            return (A[p].value + A[r].value) / 2;
```

```

        if (A[p].weight > A[r].weight)
            return A[p].value;
        else
            return A[r].value;
    }
    int q = partition(A, p, r);
    double wl = 0, wr = 0;
    for (int i = p; i <= q - 1; i++)
    {
        wl += A[i].weight;
    }
    for (int i = q + 1; i <= r; i++)
    {
        wr += A[i].weight;
    }
    if (wr < 0.5 && wl < 0.5)
        return A[q].value;
    else
    {
        if (wl > wr)
        {
            A[q].weight += wr;
            WeightedMedian(A, p, q);
        }
        else
        {
            A[q].weight += wl;
            WeightedMedian(A, q, r);
        }
    }
}

void Print(vector<Node>A)
{
    for (int i = 0; i < A.size(); i++)
        cout << A[i].value << " ";
    cout << endl;
    for (int i = 0; i < A.size(); i++)
        cout << /*setprecision(2)<< */A[i].weight<<" ";
    cout << endl;
}

```

```
void Initial(vector<int>&B,int n)
{
    for (int i = 0; i < n; i++)
    {
        B.push_back(0);
    }
}
```

// main 函数由题目提供

测试结果

2

已结束

已提交 2023年10月14日 星期六 10:47

10.00

[回顾](#)

通过所有测试! ✓

正确

此次提交得分: 10.00/10.00。

为了检验程序的鲁棒性，我们再随机生成数据进行检测：

数组大小n: 20

生成的随机数组和权值分别是： -275290598 -1549867294 -420857853 -109248054 -59889593
1643365975 1868612198 -1794809413 -2134120633 -1773750043 -2020051294 -129433642 -
779483109 -1162285790 1554013829 -1785991055 1284287495 -1343857651 601409563
-1737117466

对应的权重是： 0.09656773144589867 0.046950984199436954 0.07856479691969345
0.061517083696472616 0.09657596561987604 0.05006905851403734
0.019385880956985953 0.06949921230701882 0.012433377060254656
0.011400760770300828 0.015413426947396303 0.01608756351168227
0.06389094485653934 0.009702421455657029 0.09095470236769912
0.04405999293998829 0.0784899415262453 0.08014506119093096 0.04432717466324082
0.013963919050644962

带权中位数是:-275290598。符合条件。