算法作业

2018E8018661007 张振国

2018 年 10 月 10 日

1 问题1

从2个数据库中寻找第n小的数。

1.1 算法简述

由问题分析可知，向数据库输入一个k，数据库会返回第k小的数；2个数据库中共2n个数据都不相同；算法复杂度要求为O(log n)
该问题可以看成是在2个等长有序数组a,b中寻找2个数组整体的中位数。由于总共2n个数，其中位数必定是中间2个数的和的一半，而原问题是求第n小的数，因此只需要取中间2个数的前1个。
分析：可以将问题divide为分别寻找2个数组的中位数，然后使用中位数将1个数组分为2个数组，再通过比较4个小数组头尾的值的大小判断整体所求的中位数位于哪两个小数组内，由于位于2侧的2个小数组等长，因此舍弃掉的话对所求中位数无影响。
随后采用递归思想，继续求小数组中位数对数组进行分割，将可能存在所求中位数的两个数组保留，另外2个数组舍弃。直到最后能够通过常数步比较得到中位数为止。

1.2 算法伪代码

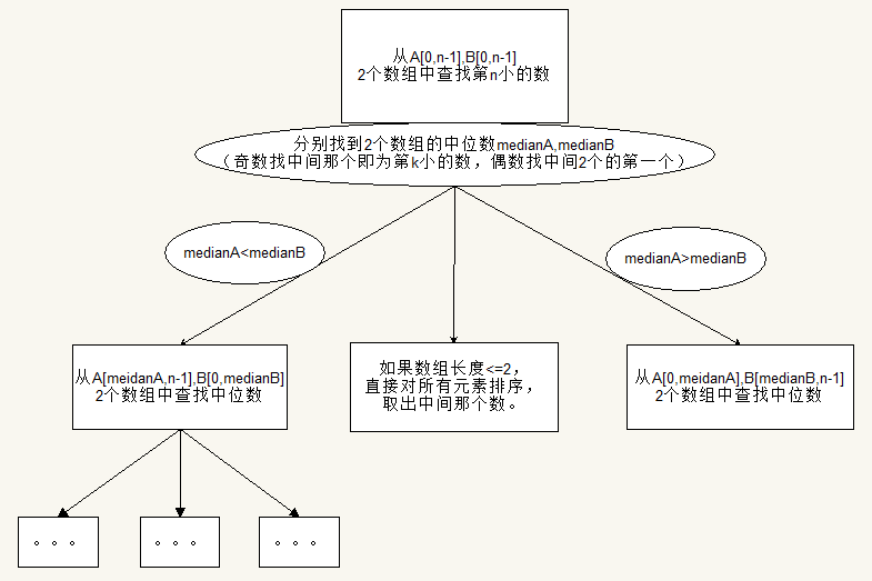
Algorithm 1 递归求中位数

Require: 2个数组的左右边界下标

Ensure: 中位数

```
1: function FindMedian(leftA, rightA, leftB, rightB)
2:     if rightA − leftA <= 1 then
3:         sort all elements in two arrays
4:         get the median or the first number of two medians.
5:     end if
6:     return median
7:     determine two arrays' relationship,互斥还是相交
8:     if leftA > rightB||leftB > rightA then
9:         if leftA > rightB then
10:            return rightB
11:        end if
12:        if leftB > rightA then
13:            return rightA
14:        end if
15:    else
16:        medianA ← (leftA + rightA)/2
17:        medianB ← (leftB + rightB)/2
18:        if medianA < medianB then
19:            return FindMedian(medianA, rightA, leftB, medianB)
20:        else
21:            return FindMedian(leftA, medianA, medianB, rightB)
22:        end if
23:    end if
24: end function
25:
```

1.3 子问题简化图



1.4 算法正确性证明

略

1.5 算法复杂性分析

略

2 问题2

求二叉树中任意两节点的最远距离。

2.1 算法简述

任意2个节点的最远距离，可以理解为2个节点之间的路径穿过的祖先节点的左子树和右子树的最深距离的和。
因此求任意2个节点的最远距离可以转化为求每个节点的左子树和右子树的最深深度，然后再遍历一遍所有节点，将每个节点的左右子树的最深深度相加的和取最大值即为所求的最远距离。

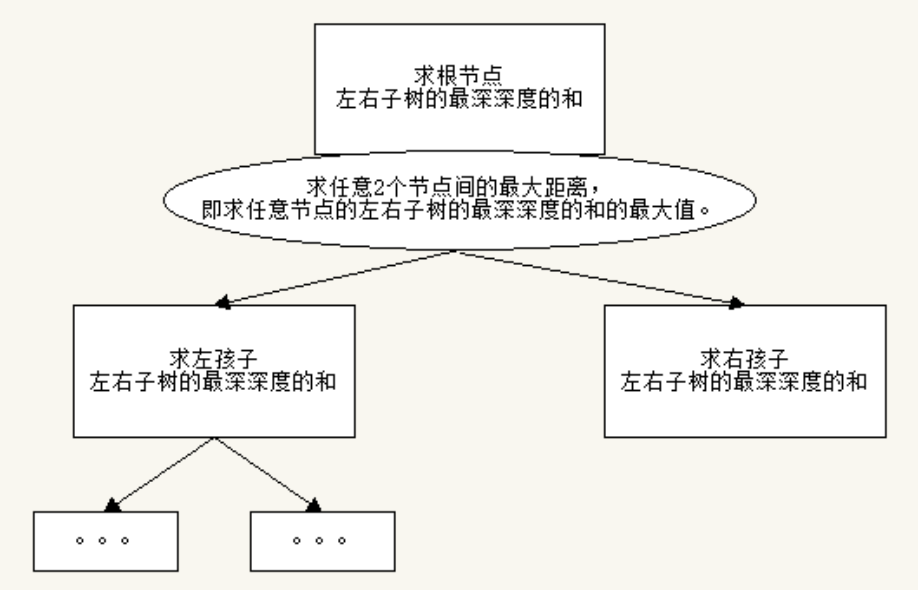
2.2 算法伪代码

Algorithm 2 求二叉树中任意两节点的最远距离

Require: $currentNode$当前节点,$maxDistance$用于存储最大距离的参数
Ensure: $currentMaxDistance$ 当前节点的树的最大深度

```
function FindMaxDistance(currentNode, maxDistance)
    if leftNode exists then
        leftMax ← FindMaxDistance(leftNode, maxDistance) +1
    else
        leftMax ← 0
    end if
    if rightNode exists then
        rightMax ← FindMaxDistance(rightNode, maxDistance) +1
    else
        rightMax ← 0
    end ifcurrentMaxDistance ← leftMax + rightMax
    if currentMaxDistance > maxDistance then
        maxDistance ← currentMaxDistance
    end if
    return Max(leftMax, rightMax)
end function
```

2.3 子问题简化图



2.4 算法正确性证明

略

2.5 算法复杂性分析

略

3 问题3

找出完全二叉树的一个局部极小值

3.1 算法简述

找出完全二叉树的一个局部极小值,即探测左右孩子节点的值是否比自己的大，则此节点的值为局部极小值。如果有孩子节点的值比自己的小，则说明此节点不是局部极小值，也说明孩子节点可能为局部极小值，从而去看孩子节点的孩子节点的值的大小。
那么该问题可divide为判断该节点是否为局部极小值，若不是则继续递归判断值比自己小的孩子节点是否为局部极小值。

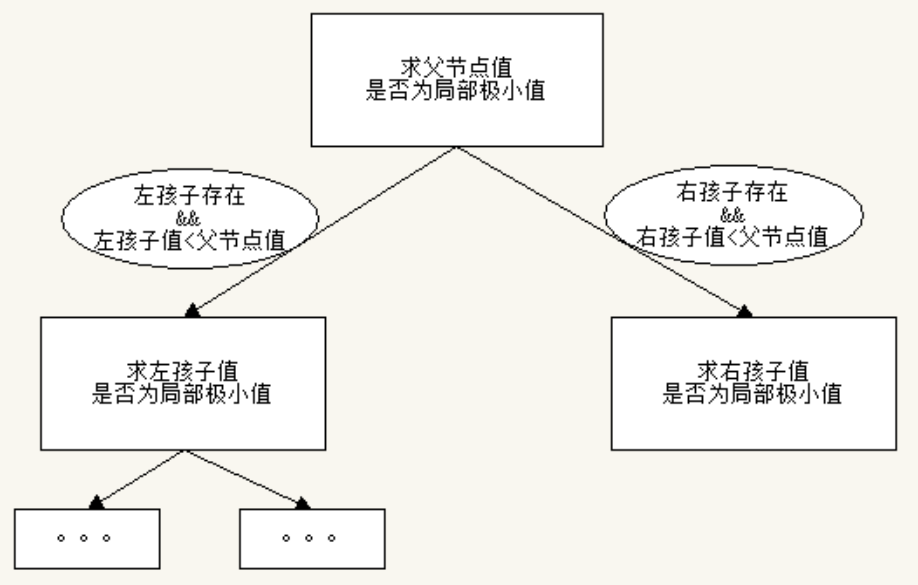
3.2 算法伪代码

Algorithm 3 找出完全二叉树的一个局部极小值

Require: $currentNode$当前节点
Ensure: 局部极小值点

```
function findLocalMin(currentNode)
    if leftNode, rightNode don't exist then
        return currentNode.X
    else
        if leftNode.X > currentNode.X and rightNode.X > currentNode.X then
            return currentNode.X
        end if
    end if
    if leftNode exists and leftNode.X < currentNode.X then
        return findLocalMin(leftNode)
    end if
    if leftNode exists and leftNode.X < currentNode.X then
        return findLocalMin(leftNode)
    end if
end function
```

3.3 子问题简化图



3.4 算法正确性证明

略

3.5 算法复杂性分析

略