091M4041H - Assignment 1

胡鹏飞 计控学院 201828007329009

1 Divide and Conquer

You are interested in analyzing some hard-to-obtain data from two separate databases. Each database contains n numerical values, so there are 2n values total and you may assume that no two values are the same. You'd like to determine the median of this set of 2n values, which we will determine here to be the n^{th} smallest value.

However, the only way you can access these values is through queries to the databases. In a single query, you can specify a value k to one of the two databases, and the chosen database will return the k^{th} smallest value that it contains. Since queries are expensive, you would like to compute the median using as few queries as possible.

Give an algorithm that finds the median value using at most O(log n) queries.

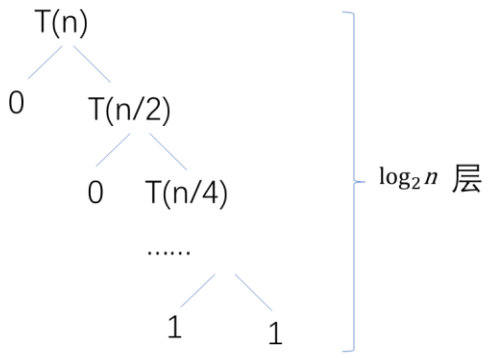
1.1 Algorithm Description

题目要求是在两个元素个数均为 n 且没有相同数据的数据库中，即 2n 个数据中找到中位数，这里定义的中位数是第 n 小的元素。按照分治的思想，首先，如果只有两个元素，显而易见，中位数是很好找的，然后将这个大问题分解，假设两个数据库中的数据为有序且为递增，采用二分法，在 n 个数据中寻找中位数，即在数据库 D1 中找到中位数 med1，在数据库 D2 中找到中位数 med2,然后比较 med1 与 med2，若 med1 小于 med2，那么中位数在 D1 库的右侧与 D2 库的左侧之间，将 D1 库的右侧作为新的 D1，将 D2 库的左侧作为新的 D2；相反，若 med1 大于 med2，那么中位数在 D1 库的左侧与 D2 库的右侧之间，将 D1 库的左侧作为新的 D1，将 D2 库的右侧作为新的 D2，直到最后只剩两个数，比较返回小的值即可。

1.2 Pseudo-code

```
FindMed(D1; D1_START; D2; D2_START; n)
    if n == 1 then
        return min(D1[D1_START];D2[D2_START])
    end if
        med1 = D1[D1_START + n/2 -1]
        med2 = D1[D2_START + n/2 -1]
        if med1< med2 then
            D1NEW=D1[med1 to n]
            D2NEW=D2[0 to med2]
            FindMed(D1NEW; D1NEW _START ; D2NEW; D2NEW _START; n/2)
        else
            D1NEW=D1[0 to med1]
            D2NEW=D2[med2 to n]
            FindMed(D1NEW; D1NEW _START ; D2NEW; D2NEW _START; n/2)
        end if
return min(med1,med2)
```

1.3 Subproblem reduction graph



1.4 Prove the correctness

因为题目是要在数据中寻找中位数，在给出的解决方法中，每次都删减掉 k 个比中位数小的数据和 k 个比中位数大的数据，所以每次删减之后数据的中位数和删减之前数据的中位数相比并没有改变，算法无误，因为共有$log_2 n$层，所以查询了$log_2 n$次，满足题目限制要求。

1.5 Analyse the complexity

因为每一步都将数据删减为原来一半，所以时间复杂度为：

$$T(\mathrm{n}) = \mathrm{cn} + \frac{cn}{2} + \frac{cn}{4} + \cdots\cdots + \frac{cn}{2^{\log n - 1}} = 2\mathrm{cn} - 2c = O(\mathrm{n})$$

2 Divide and Conquer

Given a binary tree, suppose that the distance between two adjacent nodes is 1, please give a solution to find the maximum distance of any two node in the binary tree.

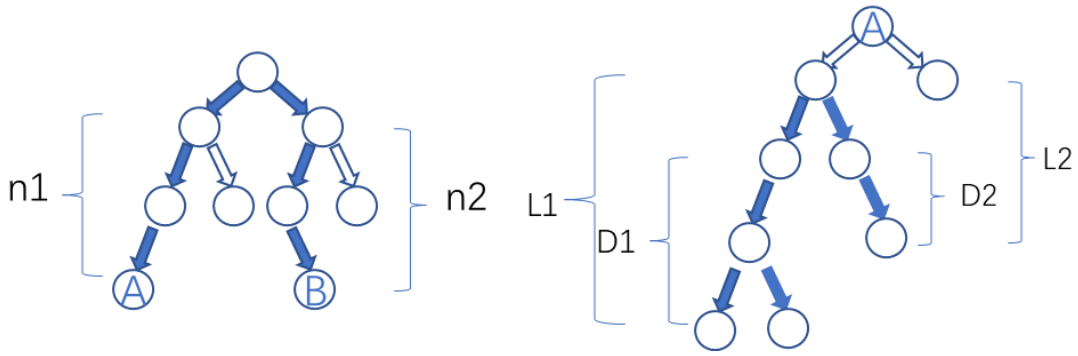
2.1 Algorithm Description

题目要求找出二叉树中的最大距离。根据这两个点的分布情况，可将题目分为两类，一类为两个点分别在根节点的两侧，一类为两个点均在根节点的一侧。对于第一类情况，分别求出两个节点到根节点的距离，相加即可；对于第二类情况，需计算两个节点到公共节点的距离，之后相加。其实第一类情况可看为第二类情况特殊情况，求二叉树中的最大距离其实就是求某个节点左子树的高度加上右子树的高度加 2，所以遍历每个节点，取其左右子树高度之和加 2 的最大值即可。

2.2 Pseudo-code

```
MaxDistance(*pNode,X)
    if (pNode == 3) then
        return 2
    end if
    LeftMaxDistance = MaxDistance(pNode->pLeft, MaxDistance) + 1
    RightMaxDistance= MaxDistance(pNode->pRight, MaxDistance) + 1
    Distance = LeftMaxDistance + RightMaxDistance
    X = X > Distance ? X : Distance
return X
```

2.3 Subproblem reduction graph



2.4 Prove the correctness

无论什么情况，两个节点的最大距离即他们到最近的公共节点的距离之和：

假设只有一个根节点和两个子节点，那么公共节点即为根节点，两个子节点的距离即1+1=2，算法计算 0+0+2=2，无误。

假设要计算的两个节点在根节点的两侧，同样那么公共节点即为根节点，两节点的最大距离很显然为 n_1+n_2，与$(n_1-1)+(n_2-1)+2$ 相同，算法无误。

假设要计算的两个节点在根节点的同一侧，假设他们到公共节点的距离分别为 L_1、L_2，公共节点的子树高度分别为为 D_1、D_2，那么这两个节点的最大距离为 L_1+L_2，算法计算结果为 D_1+D_2+2，又有 $L_1= D_1+1$、$n_2= D_2+1$，所以算法无误。

2.5 Analyse the complexity

算法要遍历每一个节点，复杂度为 O(n),在每个节点要求左右子树的高度，复杂度为 O(log n),所以算法复杂度即

$$T(\mathrm{n}) = O(\mathrm{n}\log\ \mathrm{n})$$

3 Divide and Conquer

Consider an n-node complete binary tree T, where n = 2d -1 for some d. Each node v of T is labeled with a real number xv. You may assume that the real numbers labeling the nodes are all distinct. A node v of T is a local minimum if the label xv is less than the label xw for all nodes w that are joined to v by an edge.

You are given such a complete binary tree T, but the labeling is only specified in the following implicit way: for each node v, you can determine the value xv by probing the node v. Show how to find a local minimum of T using only O(log n) probes to the nodes of T.

3.1 Algorithm Description

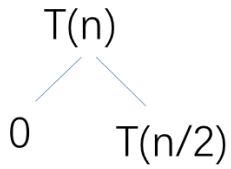
题目要求寻找一个完全二叉树的局部最小点，从根节点出发即可，若根节点小于两个子节点，根节点即为局部最小点，返回即可；若左节点小于根节点，继续走左节点即可；若左节点大于根节点，则走右节点。

3.2 Pseudo-code

```
FindLocalMin(X)
    if X not leave then
        if X < XL and X < XR then
            return X
```

```
        elseif X > XL then
            return FindLocalMin(XL)
        elseif X > XR then
            return FindLocalMin(XR)
        end if
    else
return X
```

3.3 Subproblem reduction graph



3.4 Prove the correctness

算法往下执行的条件为该节点小于其父节点，所以查看该节点的子节点即可，若两个子节点均比其小，则已找到局部最小点，若遍历到叶子节点，则其再没有子节点，而且小于父节点，所以此时叶子节点即局部最小点。

3.5 Analyse the complexity

因为每一步都走其中一支，最坏的情况即从根节点走到了叶子节点，所以时间复杂度为：

$$T(\mathrm{n}) = O(\log\ \mathrm{n})$$