Algorithm Design and Analysis - Assignment 1

2020K8009907032

唐嘉良

2022 年 10 月 13 日

目录

1 Problem One **2**

1.1 Algorithm Description 2

1.2 subproblem reduction graph 3

1.3 the correctness of the algorithm 3

1.4 the complexity of the algorithm 3

2 Problem Two **4**

2.1 Algorithm Description 4

2.2 subproblem reduction graph 5

2.3 the correctness of the algorithm 5

2.4 the complexity of the algorithm 5

3 Problem Three **6**

3.1 Algorithm Description 6

3.2 subproblem reduction graph 7

3.3 the correctness of the algorithm 7

3.4 the complexity of the algorithm 7

1 Problem One

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 define here to be the nth 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 kth 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(logn) queries.

1.1 Algorithm Description

我们采用分治的思想搜寻两个数据集的中位数，每次查询各数据集的中位数并进行比较，以它们的大小关系划分下一次需要进行中位数比较的两个子数据集，重复比较、划分的操作，直到两个子数据集均只剩下一个元素，此时较小者便是两个原始数据集的全局中位数（正确性在1.3中将会详细说明）

Algorithm 1 Find the median value in two databases

```
function MedianSearch($baseA, baseB, n$)
    //The query we use is orgnized as query for array: baseA[k] is the kth element in A
    lowA = 0, lowB = 0;
    highA = n-1, highB = n-1;
    while highA != lowA do
        medianA = baseA[(highA + lowA)/2];
        medianB = baseB[(highB + lowB)/2];
        if medianA>midianB then
            highA = (highA + lowA)/2;
            lowB = (highB + lowB)/2;
        else
            highB = (highB + lowB)/2;
            lowA = (highA + lowA)/2;
        end if
    end while
    return min(baseA[highA], baseB[highB]);
end function
```

1.2 subproblem reduction graph

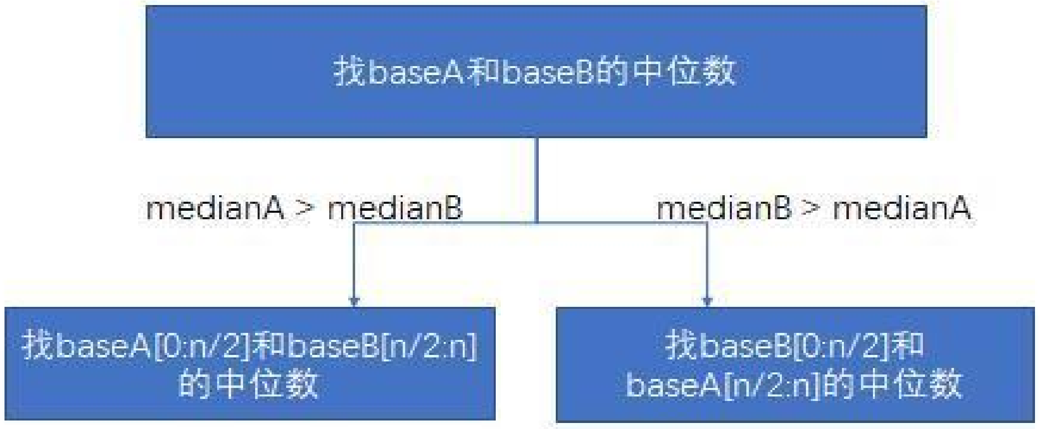
找baseA和baseB的中位数
medianA > medianB
medianB > medianA
找baseA[0:n/2]和baseB[n/2:n]的中位数
找baseB[0:n/2]和baseA[n/2:n]的中位数

图 1: Subproblem reduction graph in problem one

1.3 the correctness of the algorithm

n=1时正确性显然。n¿1时，考虑在Algorithm 1中，我们每次都对两个数据集的中位数进行比较，初始时查询到的是两个原始数据集的中位数，若baseA中位数大于baseB,则说明两个初始数据集的全局中位数必然大于medianB且小于medianA，否则，若全局中位数小于baseB，则两个数据集中小于全局中位数的数值个数小于n，与中位数定义矛盾，若全局中位数大于baseA同理可知矛盾。于是，我们仅需在分割出的baseA和baseB的两个子数据集找寻中位数。事实上本算法的正确性是由下面的断言保证的。

断言：我们找到的两个子数据集的局部中位数，必然是两个原始数据集的全局中位数。考虑在两个子数据集中增添本已经被划分走的数，使得子数据集能够还原到上一次迭代时的子数据集。显然上一次迭代划分走了两个数据集各一半的数据，而且这些数据中大于等于和小于等于局部中位数的数据个数相等，这就保证该局部中位数在上一次迭代时的子数据集中仍是中位数。重复上述操作，易见算法最终得到的中位数就是原始数据集的全局中位数。

1.4 the complexity of the algorithm

由于我们采用二分的查询方式，查询次数的时间复杂度为 $T(n) = 2T(n/2) = O(\log n)$.

2 Problem Two

Consider an n-node complete binary tree T, where n = $2^d - 1$ for some d. Each node v of T is labeled with a real number x_v. 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 x_v is less than the label x_w 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 specifed in the following implicit way: for each node v, you can determine the value x_v by probing the node v. Show how to find a local minimum of T using only O(logn) probes to the nodes of T.

2.1 Algorithm Description

本题仅仅需要找到一个T的局部最小值即可，自然的思路是先看根节点，如果根节点不是局部最小值，那么从左右子树的根节点开始找局部最小值，即将T逐层拆分成左右子树，对根节点进行局部最小值的判断

Algorithm 2 Find a local minimum in complete binary tree

```
function FindLocalMin($root$)
    if $root == NULL$ then
        return EMPTY;
    end if
    if $root \to left == NULL$ and $root \to right == NULL$ then
        return $root \to value$;
    end if
    if $root \to value < root \to left \to value$ and $root \to value < root \to right \to value$ then
        return $root \to value$;
    end if
    if $root \to value > root \to left \to value$ then
        $FindLocalMin(root \to left)$;
    end if
    if $root \to value > root \to right \to value$ then
        $FindLocalMin(root \to right)$;
    end if
end function
```

2.2 subproblem reduction graph

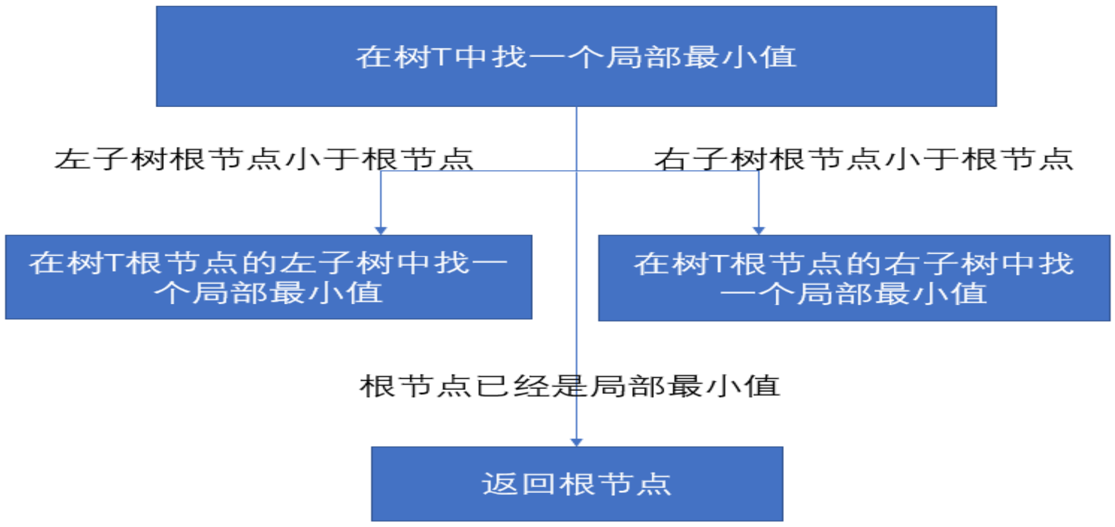

图 2: Subproblem reduction graph in problem two

2.3 the correctness of the algorithm

如果根节点就是局部最小值，那么直接返回根节点value即可；如果根节点不是局部最小值，事实上我们在左右子树选择比根节点小的子树进行submerge即可，此时已满足它比上一级节点要小，只要再与下一级节点进行同样的大小判断即可。

2.4 the complexity of the algorithm

树的高度至多logn,最多递归logn层，在每一层只需要常数的时间，时间复杂度不超过O(logn)

3 Problem Three

There are n ropes with different length. If m ropes of the same length are cut from them, please find the longest length of each of these m ropes

3.1 Algorithm Description

假设绳子长度均为正整数。本题题意为求多个数的最大公因数，那么需要将多个数的最大公因数转化为更少量数的最大公因数，直至submerge为求两个数的最大公因数。最后从底向上，求各个最大公因数的最大公因数，反复迭代即可得到所有数的最大公因数。算法设计中利用了求两个数最大公因数的辗转相除函数MaxFactor(num1, num2)。

Algorithm 3 Find the max factor of many numbers

```
function MaxFactorofMultiNumbers($num, n$)
    if $n == 2$ then
        return MaxFactor(num[0],num[1]);
    end if
    if $n == 3$ then
        return MaxFactor(MaxFactor(num[0],num[1]),MaxFactor(num[1],num[2]));
    end if
      return MaxFactor(MaxFactorofMultiNumbers(num[0:(n-1)/2],(n+1)/2),MaxFactorofMultiNumbers(num[(n+1)/2
  1],n-(n+1)/2))
end function
```

3.2 subproblem reduction graph

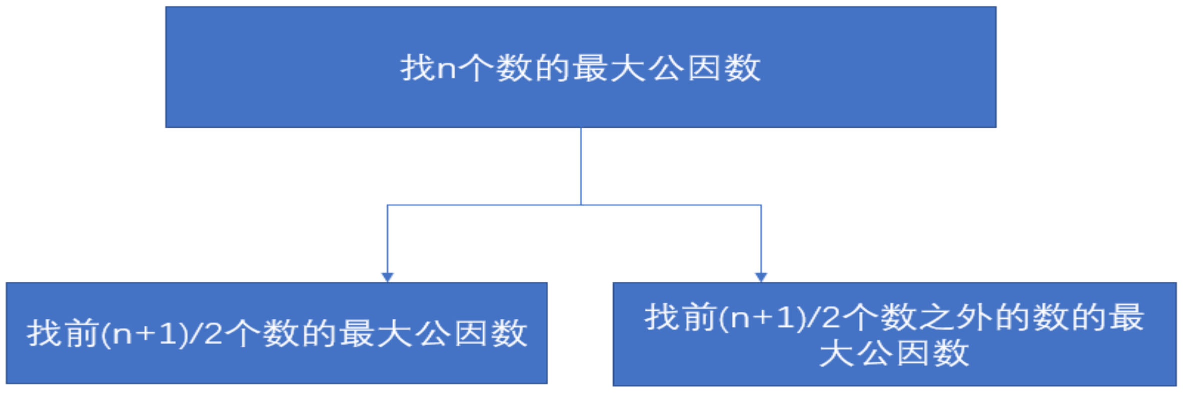
找n个数的最大公因数
找前(n+1)/2个数的最大公因数
找前(n+1)/2个数之外的数的最大公因数

图 3: Subproblem reduction graph in problem three

3.3 the correctness of the algorithm

我们仅需要说明为什么各个子数集的最大公因数的最大公因数是全局数集的最大公因数。首先全局最大公因数一定满足是所有子数集的公因数，因此它一定是每个子数集最大公因数的因数，自然就是各个子数集最大公因数的公因数。另一方面，在各个子数集的最大公因数的公因数中，最大的那个公因数满足是所有子数集的公因数，于是是所有数的公因数，于是是全局公因数，于是就是全局最大公因数。

3.4 the complexity of the algorithm

每次将数组切分一半，需要logn步，总共要调用O(n)次辗转相除函数，辗转相除函数的时间复杂度为O(logm), m为max(num[:]), 所以总的时间复杂度为O(n*logm)