计算机算法设计与分析

第 1 次作业

刘炼
202128013229021

Problem 1

Assignment Problem2

Solution

根据题目的描述，对三种不同的节点进行进行分析：对于根节点而言，如果它比两个子节点小，那么其本身就是一个 local minimum 的节点。而对于中间节点而言，其需要和两个子节点与一个父节点进行比较。对于叶子节点，其只需要和对应的父节点进行比较。

算法思路：可以利用根节点向下进行划分，这样就可以在子树上进行考虑。具体而言，从根节点开始，将其与两个子节点进行比较。如果左子节点的 label 小于根节点的 label，那么对以左子节点为根节点的子树进行相同的处理；对右节点同理。如果根节点比两个子节点的 label 都小，那么返回根节点，作为一个 local minimum。否则一直递归到叶子节点，如果一个节点没有子节点，那么返回该节点作为 local minimum。

Pseudo-Code shown as follows.

Algorithm 1 Local Minimum Node

Input: T, X ▷ T is a n-node complete binary tree, X is the set of value for all node in Tree T.

function LOCALMINIMUM(T, X, id)
    $left \leftarrow 2 * id + 1$
    $right \leftarrow 2 * id + 2$
    if $x[id] > x[left]$ **then**
        return LOCALMINIMUM(T,X,left) ▷ call function for left subtree.
    else if $x[id] > x[right]$ **then**
        return LOCALMINIMUM(T,X,right) ▷ call function for right subtree.
    else
        return $T[id], X[id]$
    end if
end function
Initialization: $id \leftarrow 0$
$v, x_v \leftarrow$ LOCALMINIMUM(T, X, id)

Output: Local Minimum node and its value v, x_v ▷ v is the local minimum node, x_v is its label value.

Subproblem Reduced Graph

具体分解效果如图1所示。

Proof of the correctness

使用结构归纳法证明如下：

- Initialization：对于高度为 2 的完全二叉树，其只包含有三个节点，其中必存在一个节点是 local minimum 的节点
- Generalization: 假设对于高度为 d 的完全二叉树，其存在一个 local minimum 的节点为 v，其值为 x_v。对于一个高度为 $d+1$ 的完全二叉树，存在两种情况：1. 如果节点 v 所在的高度不为 d，那么 v 节点仍然是一个 local minimum 节点，而只需要 $T(2^d - 1)$ 的时间就可以完成计算 2. 如果节点 v 所在的高度为 d，那么就需要判断该节点与其两个子节点的大小，如果该节点比两个子节点的 label 数值小，那么 v 仍然是 local minimum 的节点，否则，比它小的子节点是一个 local minimum 节点。在这种情况下，需要多两次比较，整体的时间为 $T(2^{d+1} - 1) = T(2^d - 1) + O(n)$

Problem 1 continued on next page...

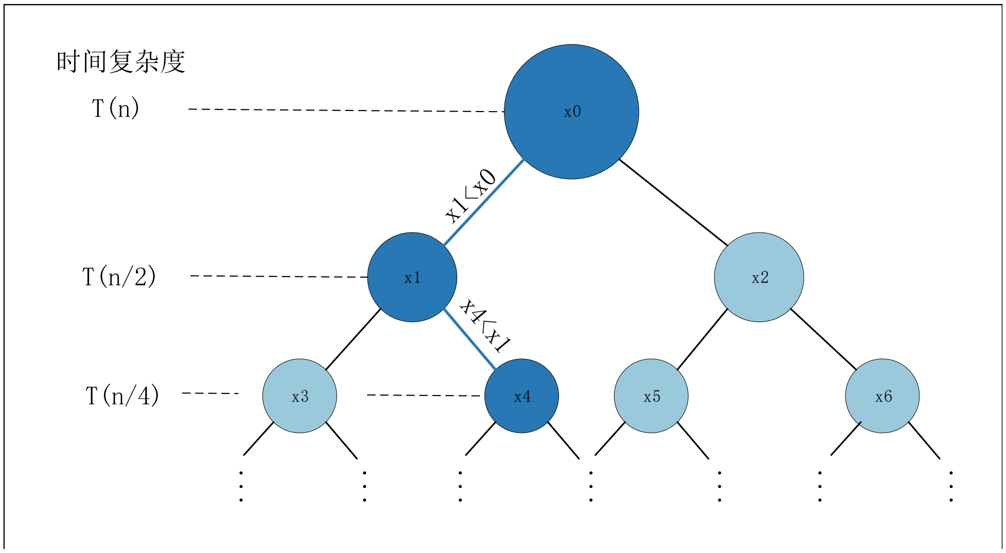

Figure 1: 寻找 Local Minimum 的递归分解图

Analysis of Complexity

时间复杂度: 根据分析，只需要将根节点与两个子节点相比较，然后就可以从其中一个子树中递归执行该算法，故有：

$$T(n) = T(\frac{n}{2}) + O(1) \tag{1}$$

故时间复杂度应该为：$O(\log n)$

空间复杂度: 由于所有操作都是在本地执行的，空间复杂度应该为：$O(1)$

Problem 2

Assignment Problem4

Solution

根据题目的描述，实际上需要找到这个有序数组的最左端和最右端，那么将其分成两个问题，也就是找到数组中第一个等于 target 和最后一个等于 target 的数，就满足了题意。

算法思路：利用二分查找的方法，找到数组中第一个满足相等关系的数；然后利用同样的方法，找到数组中最后一个满足相等关系的数，就可以得到结果。

Pseudo-Code shown as follows.

Algorithm 2 BinarySearch

Input: $X, target$ $\triangleright$ X is an array of integers nums sorted in ascending order, $target$ is our searching goal.

Initialization: $left \leftarrow 0, right \leftarrow length(X) - 1$
while left $<$ right **do**
  $middle \leftarrow \lfloor \frac{left+right}{2} \rfloor$
  if $X[middle] < target$ **then**
    $left \leftarrow middle + 1$
  else

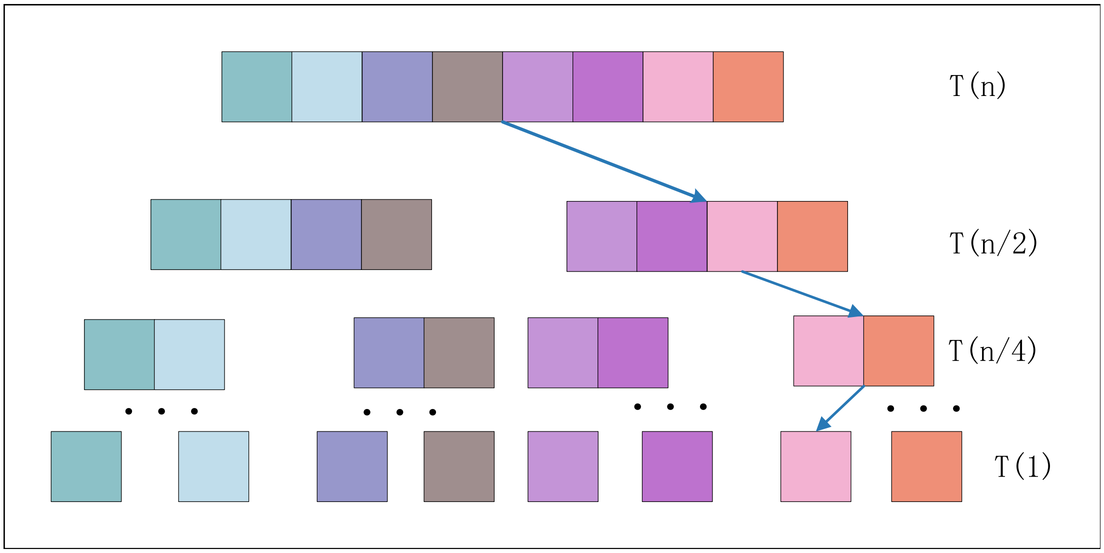

Figure 2: 标准二分查找的递归分解图

        $right \leftarrow middle$
    end if
end while
if $X[middle] \neq target$ **then**
    return $[-1, -1]$
end if
$L \leftarrow middle$
$left \leftarrow 0, right \leftarrow length(X) - 1$
while left $<$ right **do**
    $middle \leftarrow \lceil \frac{left+right}{2} \rceil$
    if $X[middle] > target$ **then**
        $right \leftarrow middle$
    else
        $left \leftarrow middle + 1$
    end if
end while
$R \leftarrow middle$
return $[L, R]$
Output: Local Minimum node and its value v, x_v ▷ v is the local minimum node, x_v is its label value.

Subproblem Reduced Graph

具体分解效果如图2所示。

Proof of the correctness

要证明的循环不变量：当 $target$ 存在于数组中时，在数组 X 的 $left$ 和 $right$ 范围内，至少存在一个数使得 $X[i] == target$

证明：

- Initialization：此时 $left = 0, right = n - 1$, 此时满足循环不变量。

Problem 2 continued on next page...

- Maintenance：每次迭代都比较中间的数 $X[\frac{left+right}{2}]$，并根据此关系动态调整 left 和 right，此时满足循环不变量
- Termination: 终止条件为 $left == right$, 此时判断该数是否与 $target$ 相等，如果相等，则找到相应的位置，不相等则意味着原数组中没有该数，故同样满足循环不变量。

Analysis of Complexity

时间复杂度: 根据分析，只需要进行两次二分查找，而每一次二分查找只需要进行一次比对，故有：

$$T^{'}(n) = T(\frac{n}{2}) + O(1) \tag{2}$$

$$T(n) = 2T^{'}(n) \tag{3}$$

故时间复杂度应该为：$O(\log n)$

空间复杂度: 由于所有操作都是在本地执行的，空间复杂度应该为：$O(1)$

Problem 3

Assignment Problem5

Solution

对于一个 n 边形而言，选择一条边为 $v_1 - v_n$，那么可以从剩下的 n-2 个节点中任意选择一个节点 v_k，连接该两个节点，这样将一个 n 边形划分成了一个 k 边形和一个 $n-k+1$ 边形。**算法思路**：根据上面的思路，可以将一个 n 边形的问题划分成一些子问题，并迭代不同的划分策略，通过求和得到最终的结果。这里有一个特殊情况，也就是 $k = 2$ 的情况下，实际上不进行划分，但应处理为划分方式为 1 种。

Pseudo-Code shown as follows.

Algorithm 3 Triangle Divide

Input: n ▷ n represents the number of nodes for the convex polygon.

Initialization: $k \leftarrow 4, X$ ▷ X is a array with length $n+1$
Initialization: $X[2] \leftarrow 1, X[3] \leftarrow 1$
for $i \leftarrow 4$ **to** n **do**
　$X[i] = 0$
　for $k \leftarrow 2$ **to** $i-1$ **do**
　　$X[i] += X[k] \times X[i-k+1]$
　end for
end for

Output: X[n]

Subproblem Reduced Graph

具体分解效果如图3所示。

Proof of the correctness

使用结构归纳法证明如下：

- Initialization：对于一个三角形，其只存在一个划分为三角形的方式。故此时划分方式为 1

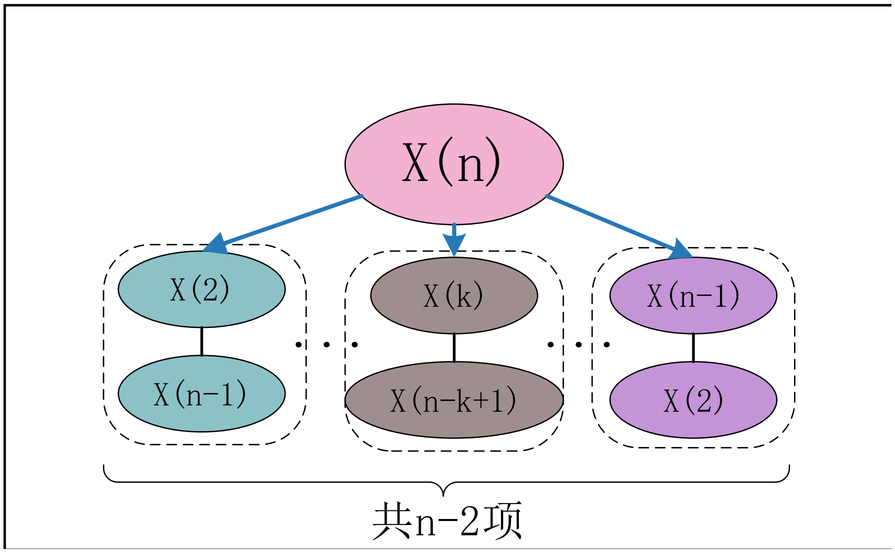

Figure 3: 搜索 n 边形的三角分割方法逻辑图

- Generalization: 假设对于从 3 到 n 边的多边形，其对应划分为三角形的方式分别为 $X[i], i \in 3, 4, ..., n$，那么对于一个 $n+1$ 边形,，选择一条边为 $v_1 - v_{n+1}$，那么可以从剩下的 n-1 个节点中任意选择一个节点 v_k，连接该两个节点，这样将一个 $n+1$ 边形划分成了一个 k 边形和一个 $n-k$ 边形，此时对于 $n+1$ 边形的划分方式的计算应该为：$X[n+1] = X[2][n] + ... + X[k][n-k+2]... + X[n][2]$。故一定能找到 $n+1$ 边形的划分方式，得证。

Analysis of Complexity

时间复杂度根据伪代码，可以看到，有两层循环，每层循环的规模都是 n，所以时间复杂度为：$O(n^2)$
空间复杂度由于需要建立一个长度为 n 的数组，故空间复杂度为：$O(n)$