Algorithm Design and Analysis

吴长亮

Assignment 1

1 Question 1

1.1 Algorithm Description

1.1.1 Description

- 分别找到两个数组的 n/2 位置的中位数 M1，M2
- 比较 M1、M2 大小，去掉每个数组一半的数据
- 递归执行比较

1.1.2 Pseudo-code

1: find_mid(l_1,r_1,l_2,r_2)

2: if l_1==r_1 then

3: return (l_1+l_2)/2

4: end if

5: M1=array1_mid

6: M2=array2_mid

7: if $M1 < M2$ then

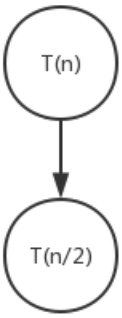
8: find_mid(array1_mid,r_1,l_2,array2_mid)

9: else

10: find_mid(l_1,array1_mid,array2_mid,r_2)

11: end if

1.2 Subproblem Reduction Graph



1.3 Correctness

中点划分中位数，包含所有的数据

1.4 Complexity

$$T(n) = T(n/2) + c$$

所以复杂度为 $O(log_n)$

2 Question 2

2.1 Algorithm Description

2.1.1 Description

- 二叉树的最长路径：1. 左子树的最长路径 2. 右子树的最长路径 3. 左子树的高度加右子树的高度
- 分别递归左右子树

- 计算节点高度
- 左子树的最大长度, 右子树的最大长度，左子树的最大深度 + 右子树的最大深度, 取三者的最大值就是当前节点的最长路径

2.1.2 Pseudo-code

1: max_dis(root)

2: if 叶子节点 than

3: return

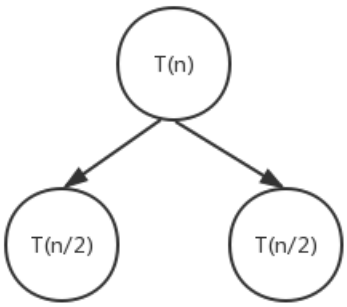
4: left=max_dis(左节点)

5: right=max_dis(右节点)

6: max=max(左子树高度，右子树高度)

7: return max(左子树的最大长度, 右子树的最大长度，左子树的最大深度 + 右子树的最大深度 +2)

2.2 Subproblem Reduction Graph



2.3 Correctness

子树递归调用遍历所有的节点

2.4 Complexity

$$T(n) = 2T(n/2) + c$$

所以复杂度为 $O(n)$

3 Question 3

3.1 Algorithm Description

3.1.1 Description

- 根节点符合
- 取左右子树较小的树，必有极小值

3.1.2 Pseudo-code

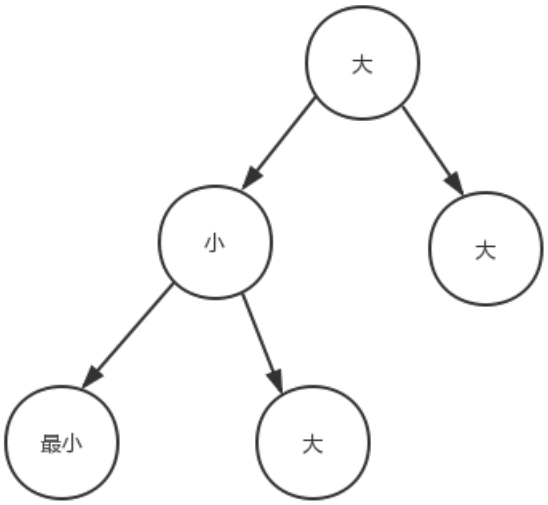
1: local_min(root)

2: if 根节点最小

3: return

4: local_min(min(root− >left,root− >right))

3.2 Subproblem Reduction Graph



3.3 Correctness

取较小值之后，如果左右子树大，那么该节点就最小，否则走到叶子节点就是最小

3.4 Complexity

最坏情况走到叶子节点，复杂度为树高，即 $O(log_n)$