题目描述

设计一个分治算法来计算二叉树的层数.(空树返回0,单顶点树返回1),并确定它的时间复杂度.

算法思想

divide

将树分为左右子树

conquer

分别求左右子树的层数

merge

空树返回0, 单顶点树返回1

代码展示

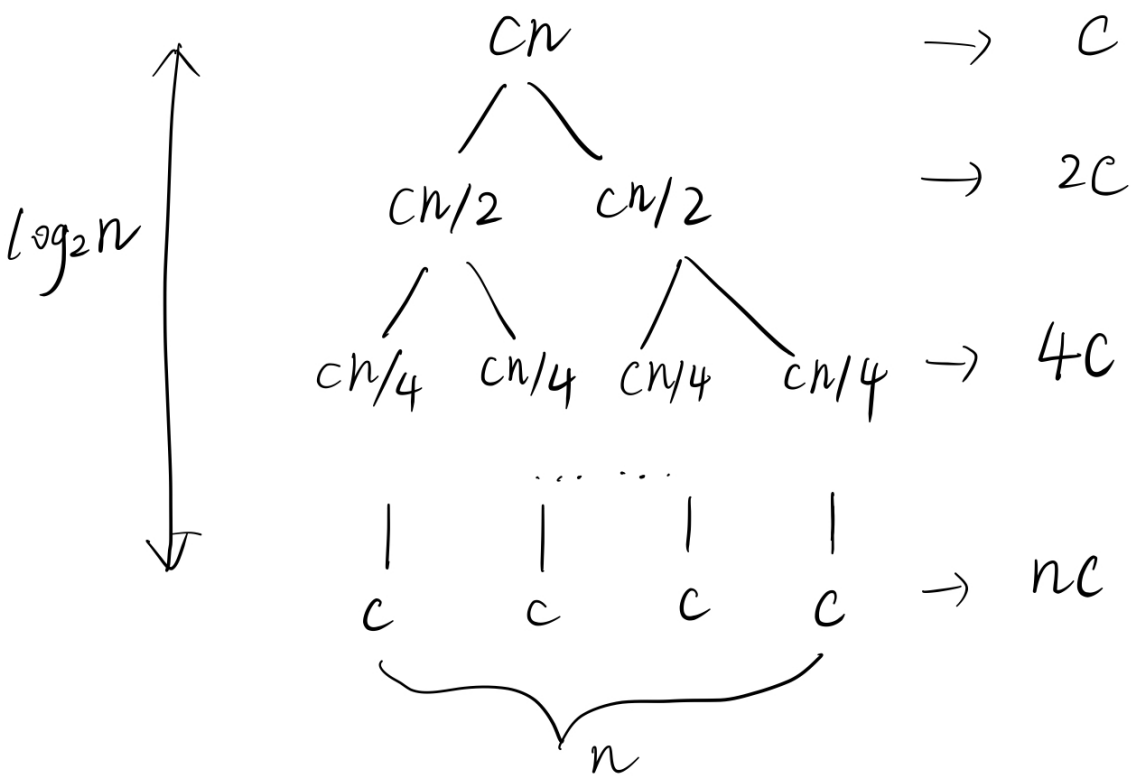
```
typedef struct node{
    int data;
    struct node* left;
    struct node* right;
} Node;

typedef struct{
    Node* root;
} Tree;

int count(Tree* tree){
  if(tree->root == NULL)
    return 0;
  else
    return max(count(left), count(right)) + 1
}
```

复杂度分析

根据代码，可以发现分成了左右子树两个子问题，每个子问题规模为n/2，则表达式为

$$\mathrm{T}(n) = \begin{cases} \Theta(1) & \text{如果 } n = 1 \\ 2T(n/2) + \Theta(n) & \text{如果 } n > 1 \end{cases}$$

同时merge直接返回值，时间复杂度为1，则递归树如下所示：



总代价为：

$$C(1+2+4+\cdots+n) = C\,\frac{(1-2^{\log_2 n})}{1-2} = C(n-1)$$

为 $\Theta(n)$