递归算法的复杂度分析

- 我们可以用一个式子描述一般递归算法的过程
- $T(n) = a * T(n / b) + f(n)$

主定理（Master theorem）

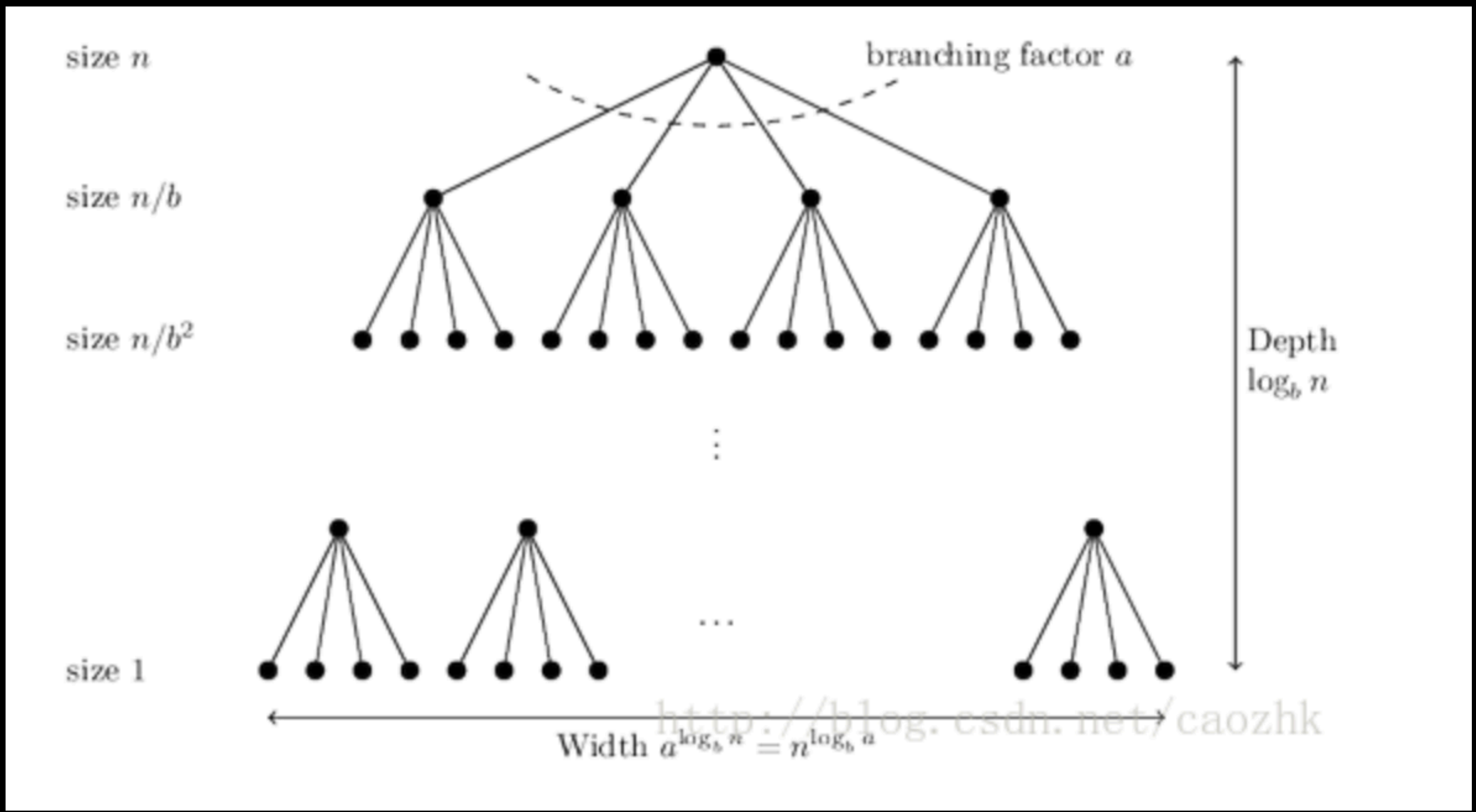
主定理

主定理最早出现在《算法导论》中，提供了分治方法带来的递归表达式的渐近复杂度分析。
规模为n的问题通过分治，得到a个规模为n/b的问题，每次递归带来的额外计算为c(n^d)
T(n) <= aT(n/b)+c(n^d)
那么就可以得到问题的复杂度为：

- **T(n) = O(n^d log(n)), if a = b^d**
- **T(n) = O(n^d), if a < b^d**
- **T(n) = O(n^logb(a))), if a > b^d**

递归树

- 画递归树是常用的分析递归复杂度的方法
- 把每一层的计算量以及层数用图形表示出来，然后再通过数学式子化简总的计算量，进而得出复杂度的级别



size n
branching factor a
size n/b
size n/b^2
Depth
$\log_b n$
size 1
Width $a^{\log_b n} = n^{\log_b a}$

可见，每次递归把问题分为a个规模为n/b的子问题。从根节点开始，共有logb(n)+1层，叶子节点数为a^(logb(n))。那么，第j层共有a^j个子问题，每个问题规模为n/b^j，每个子问题运算量为c*(n/b^j)^d需要完成的计算量为：

Work per level-j subproblem

$$\le \underbrace{a^j}_{\text{\# of level-j subproblems}} \cdot c \cdot \left(\frac{n}{b^j}\right)^d = cn^d \cdot \left(\frac{a}{b^d}\right)^j$$

Size of each level-j subproblem

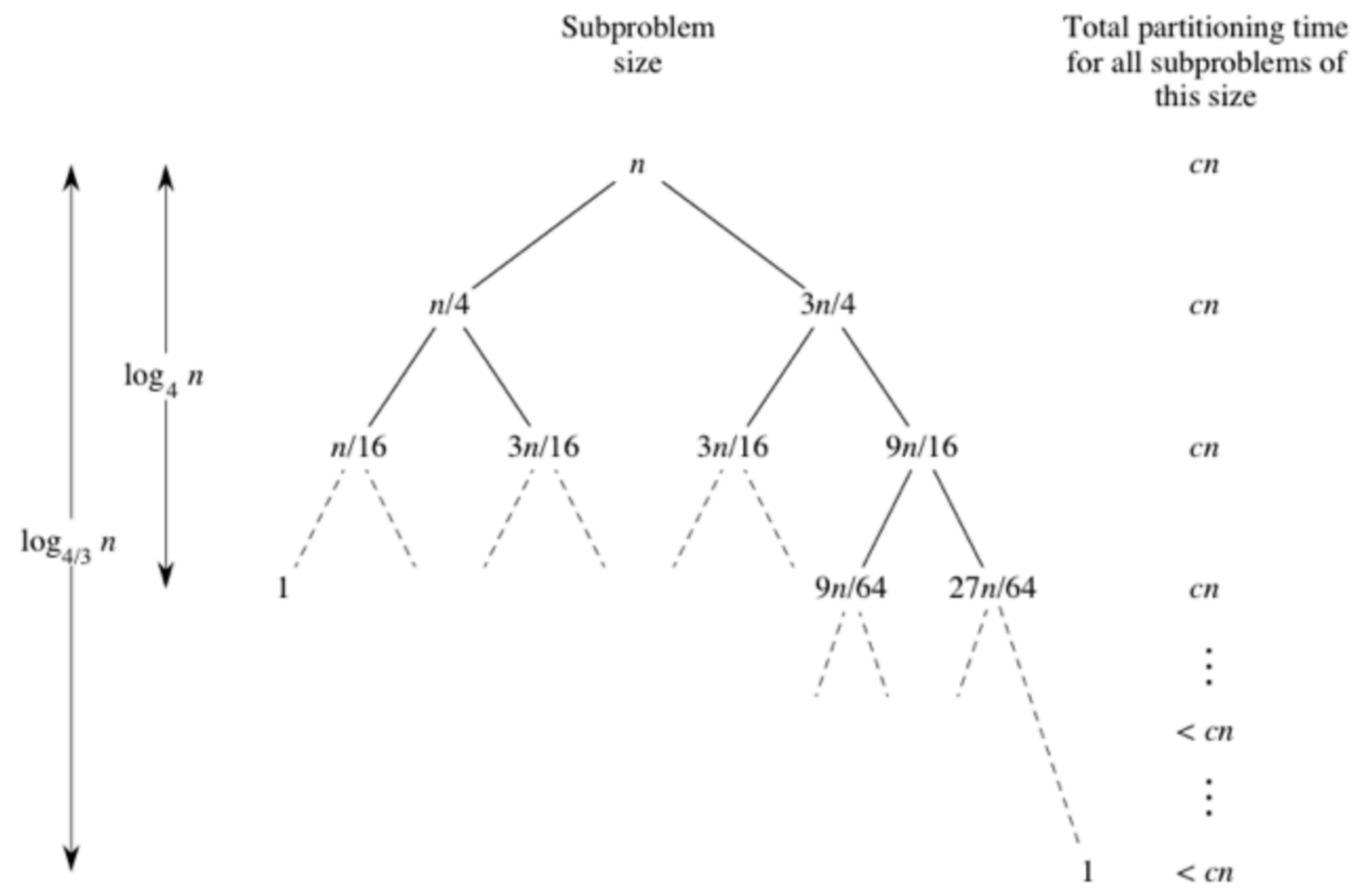
求和得到整个问题的运算量：

$$\text{Total work} \leq cn^d \cdot \sum_{j=0}^{\log_b n} \left(\frac{a}{b^d}\right)^j$$

那么，根据a与b^d的关系，很容易得到主定理。

快速排序复杂度分析

- $T(n) = T(p/q * n) + T((q-p)/q * n) + n$
- $T(n) = n\log n$



$$\log_{4/3} n = \frac{\log_2 n}{\log_2(4/3)} \ ,$$

- 首先，由于每次选取的分割点是随机的，所以我们可以算平均复杂度

- 目的是为了得到T(n)与T(n-1)的递推式子

$$T(n) = T(i) + T(n-i) + c \cdot n$$

$$E(T(i)) = \frac{1}{n}\sum_{j=0}^{n-1} T(j)$$

$$E(T(n-i)) = E(T(i))$$

$$T(n) = \frac{2}{n}\left(\sum_{j=0}^{n-1} T(j)\right) + c \cdot n$$

$$n \cdot T(n) = 2 \cdot \left(\sum_{j=0}^{n-1} T(j) \right) + c \cdot n^2$$

$$(n-1) \cdot T(n-1) = 2 \cdot \left(\sum_{j=0}^{n-2} T(j) \right) + c \cdot (n-1)^2$$

$$nT(n) - (n-1)T(n-1) = 2T(n-1) + 2cn - c$$

$$nT(n) = (n+1)T(n-1) + 2cn$$

$$
\begin{aligned}
\frac{T(n)}{n+1} &= \frac{T(n-1)}{n} + \frac{2c}{n+1} \\
\frac{T(n-1)}{n} &= \frac{T(n-2)}{n-1} + \frac{2c}{n} \\
\frac{T(n-2)}{n-1} &= \frac{T(n-3)}{n-2} + \frac{2c}{n-1} \\
&\vdots
\end{aligned}
$$

$$\frac{T(n)}{n+1} = \frac{T(1)}{2} + 2c\sum_{j=3}^{n+1}\frac{1}{j}$$

As n gets very large, $\sum_{j=3}^{n+1}\frac{1}{j}$ approaches $\ln(n) + \gamma$, where γ is Euler's constant, $0.577\ \ldots$

Hence

$$\frac{T(n)}{n+1} = \frac{T(1)}{2} + 2c\cdot \ln(n) + 2c\gamma = \ln(n) + c_2 = O(\ln(n))$$

and so

$$T(n) = O(n\cdot \log(n))$$