递归方程的求解

分治递归问题的时间复杂度分析

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

YouTube频道: hwdong

递归方程的求解

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

- 迭代展开： 迭代展开递归方程
- 递归树表示：迭代展开的可视化表示
- 假设归纳：先假设,数学归纳法
- 高阶方程的简化：转化为一阶方程
- 主定理：特殊递归方程的解

迭代展开

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

$T(1) = 1$

YouTube频道: hwdong

迭代展开

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

$T(1) = 1$

$$
\begin{aligned}
T(n) &= 2T(n-1)+1 \\
&= 2(2T(n-2)+1\ )+1 \\
&= 2(2(2T(n-3)+1\ )+1\ )+1 \\
&= \ldots \\
&= 2^{n-1}T(1)+2^{n-2}+2^{n-3}+\ldots+2+1 \\
&= 2^{n-1}+2^{n-2}+2^{n-3}+\ldots+2+1 = 2^{n}-1
\end{aligned}
$$

数列和

- 等差数列($a_1,a_2,\ldots,a_n$)的和为: $(a_1 + a_n)n/2$

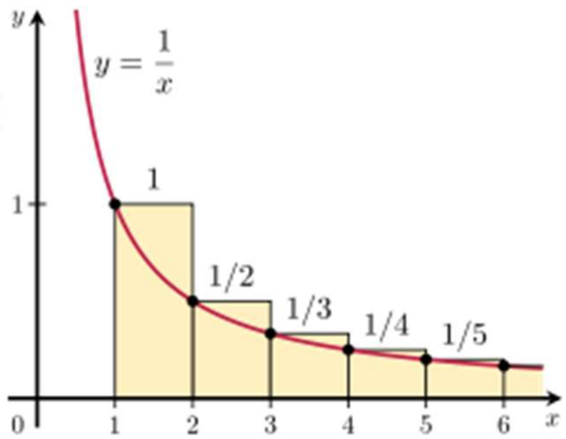
  $1+3+5+\ldots(2n-1) = n^2$

- 等比数列($a,aq,\ldots,aq^{n-1}$)的和为: $a(q^n -1) /(q-1)$

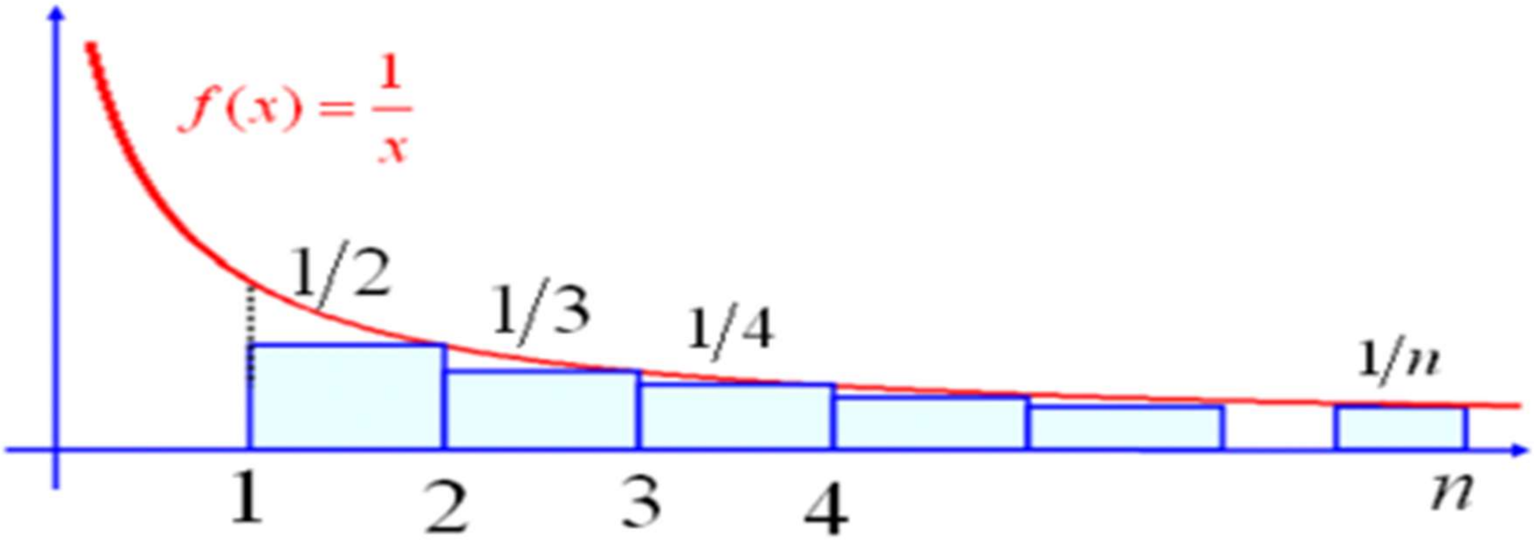
  $1+2+4+\ldots+2^n = 1*(2^{n+1}-1)/(2-1) = 2^{n+1}-1$

- 调和数列之和:

  $\ln(n+1)< 1+1/2+1/3+\ldots+1/n <1+ \ln n$



$$\sum_{n=1}^{k} \frac{1}{n} > \int_1^{k+1} \frac{1}{x}\, dx = \ln(k+1)$$



$$s_n = 1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{n} < 1 + \int_1^n \frac{1}{x}\,dx < 1 + \ln n$$

$$\therefore \quad \ln(1+n) < s_n < 1 + \ln n$$

换元迭代

$T(n) = 2T(n/2)+n-1$

$T(1)=0$

- 可令$n = 2^k$,则：

  $T(2^k) = 2T(2^{k-1})+ 2^k - 1$

$$T(2^k) = 2T(2^{k-1}) + 2^k - 1$$

$$= 2(2T(2^{k-2}) + 2^{k-1} - 1) + 2^k - 1 = 2^2T(2^{k-2}) + 2^k - 2 + 2^k - 1$$

$$= 2(2(2T(2^{k-3}) + 2^{k-2} - 1) + 2^{k-1} - 1) + 2^k - 1$$

$$= 2^3T(2^{k-3}) + 2^k - 2^2 + 2^k - 2 + 2^k - 1$$

$$= \ldots$$

$$= 2^kT(2^0) + k2^k - (2^{k-1} + \ldots 2 + 1)$$

$$= nT(1) + k2^k - (2^k - 1)$$

$$= k2^k - 2^k + 1 = n\log n - n + 1$$

递归树表示

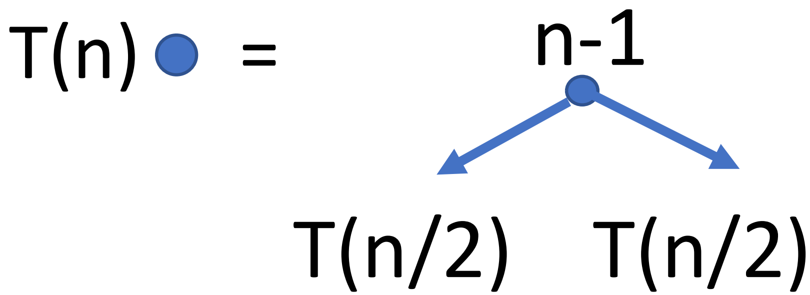
$$T(n) = 2T(n-1)+1$$

$$T(1) = 1$$

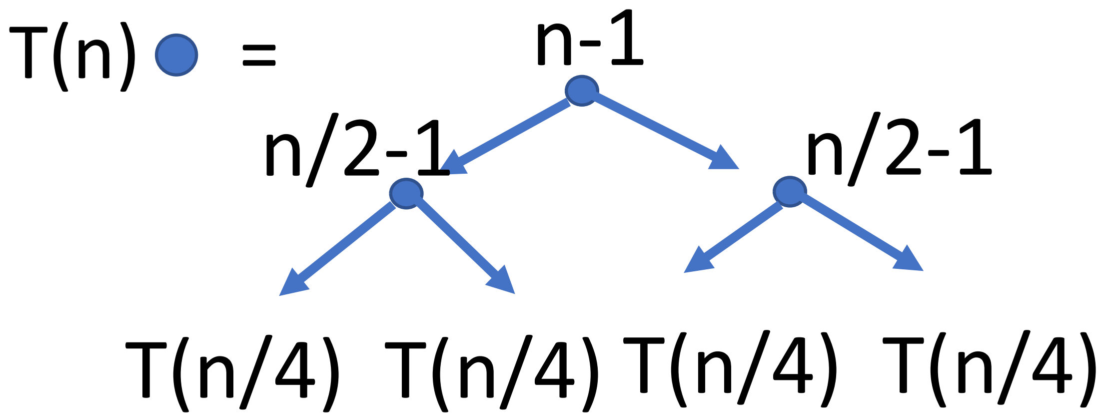
YouTube频道: hwdong

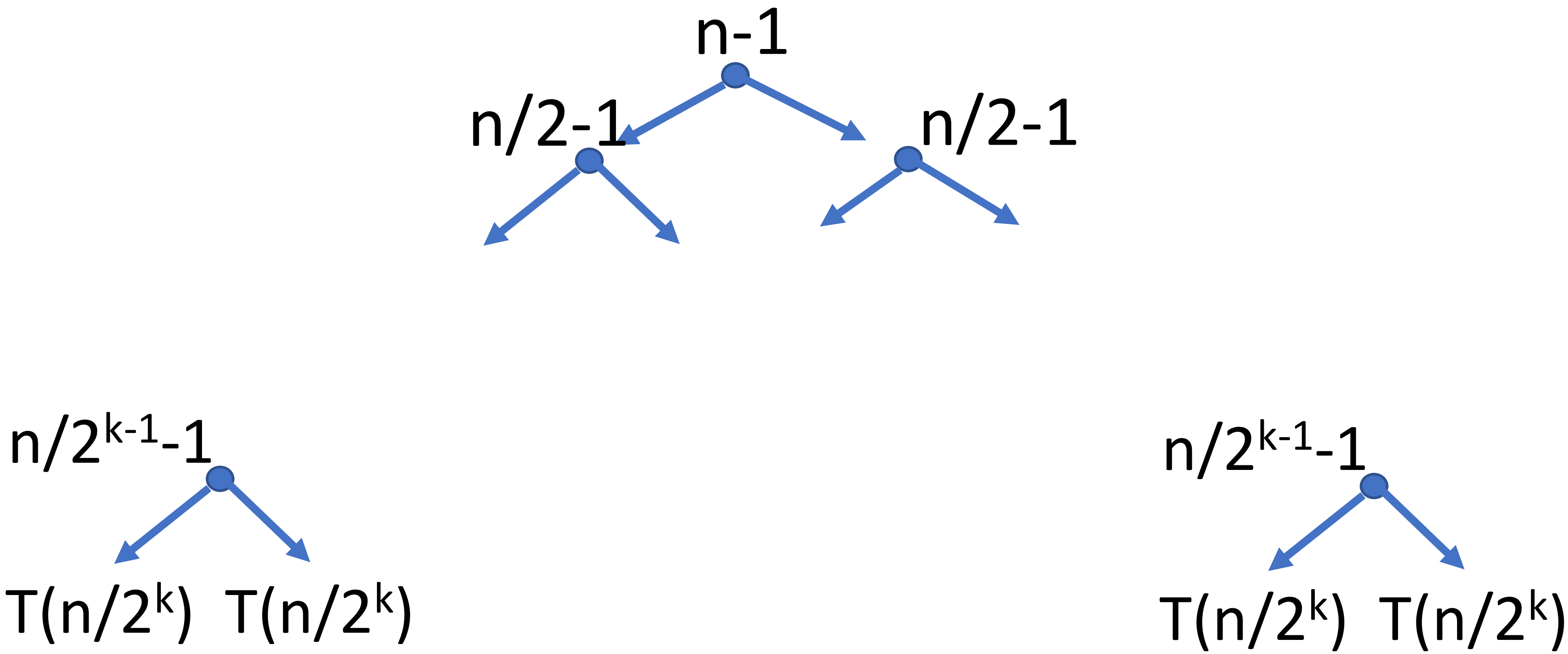
递归树表示

$$T(n) = 2T(n/2)+n-1$$



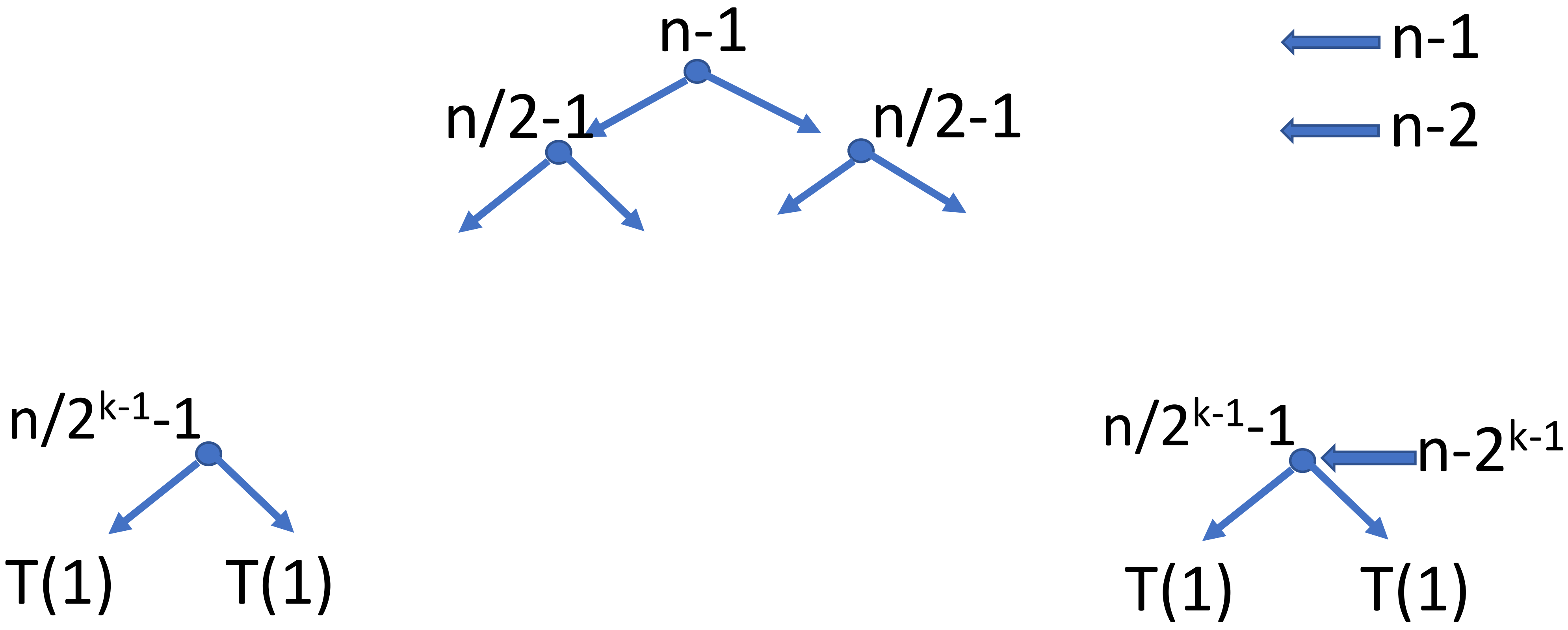
$$T(n) = 2T(n/2)+n-1$$




n-1
n/2-1
n/2-1
n/2^{k-1}-1
T(n/2^k)
T(n/2^k)
n/2^{k-1}-1
T(n/2^k)
T(n/2^k)

$$n-1 + n-2 + \dots + n-2^{k-2} + n-2^{k-1}$$

$$= kn-1-2-\dots-2^{k-1} = kn-2^{k}+1 = n\log n - n + 1$$



假设归纳

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

YouTube频道: hwdong

假设归纳

- 先猜测其数量级的界（迭代展开或递归树），
- 再用数学归纳法证明。

$T(n) = 2T(n/2)+n-1$

$T(1)=0$

- 猜测其紧界为$\Theta(n \log n)$.即 $T(n) = \Theta(n \log n)$.

- 即存在正数c_1,c_2和n_0使得对所有$n \geq n_0$有：

$$c_1 n\log n \leq T(n) \leq c_2 n\log n \}$$

证法1：

1) n=1,显然成立 $T(1)=0 = \Theta(1 \log 1)$

2) 假设<n时，都成立，则:

$$T(n) = 2T(n/2)+n-1 = 2\,\Theta(\,(n/2)\log(n/2)\,)+n-1$$

$$= \Theta(n\,(\log n-\log 2)+n-1) = \Theta(n \log n)$$

证法2：

证明$T(n) = O(n\log n)$且$T(n) = \Omega(n\log n)$

证法2：先 $T(n) = O(n\log n)$

- 分析：要证存在常数 $c>0$ 和 n_0，使 $n> n_0$ 得满足 $T(n) \le cn\log n$

特别对 $n/2$ 也成立，即：$T(n/2) \le cn/2 \log(n/2)$

$T(n) = 2T(n/2)+n-1 \le 2c(n/2)\log(n/2)+n-1$

$\quad = cn\log n - cn + n-1$

只要 $-cn + n-1<0$ 即可，可取 $c>=1$，$n_0=1$

因此，$T(n) = O(n\log n)$

证法2：先 $T(n) = O(n\log n)$

- 证明：取$c=1$, $n_0=1$

1) $T(1)=0<=c *1\log 1 = 0$
2) 假设对小于n的所有n’,

   $T(n')<=c\, n'\log n' = n'\log n'$

特别对n/2也成立，即： $T(n/2) \le cn/2 \log(n/2)$

那么： $T(n) = 2T(n/2)+n-1 \le 2c(n/2)\log(n/2)+n-1$

$= cn\log n - cn + n-1 \le cn\log n$

因此，$T(n) = O(n\log n)$

证法2：再$T(n) = \Omega(n\log n)$

- 证明：证法类似

$T(n) = O(n\log n)$且$T(n) = \Omega(n\log n)$，因此

$T(n) = \Theta(n\log n)$

$$\mathrm{T}(n) \le f(n) + \sum_{i=1}^{k} T(n_i)$$

猜测： $\mathrm{T}(n) \le \mathrm{O}(g(n))$

数学归纳法：

1） 当n= n_0时成立 $\mathrm{T}(n_0) \le \mathrm{O}(g(n_0))$

2）设对小于n的n’， $\mathrm{T}(n') \le \mathrm{O}(g(n'))$

要证 $\mathrm{T}(n) \le \mathrm{O}(g(n))$

$$T(n) \le T(n/5) + T(7n/10) + an$$

猜测: $T(n) = O(n)$

分析: $T(n) \le T(n/5) + T(7n/10) + an$

$$\le c(n/5) + c(7n/10) + an$$

$$\le 9cn/10 \quad + \quad an$$

$$\le \quad c \quad n$$

$9c/10 + a \le c$ $\qquad\qquad$ $10 \quad a \quad \le \quad c$

假如$T(1) = 1$.则$T(1) = 1 \le c \cdot 1 \Rightarrow c \ge 1$

$$T(n) \le T(n/5) + T(7n/10) + an$$

- 证： 取c=max{10a， 1}， $n_0 = 1$
- 1) $T(1) = 1 \le cn = c \cdot 1 = c$ 是成立
- 2) 设对小于n的n’， T(n’)<=cn’

$$
\begin{aligned}
T(n) &\le T(n/5) + T(7n/10) + an \\
&\le c(n/5) + c(7n/10) + an \\
&\le 9cn/10 + an = (9c/10 + a)n \\
&\le (9c/10 + c/10)n = cn
\end{aligned}
$$

高阶方程的化简

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

YouTube频道: hwdong

快速排序算法

快速排序 [**70**, 74,60,76, 83,72,55,65,79]

划分: [65, 55,60] , 70, [83,72,76,74,79]

快速排序 [65, 55,60]

[55, 60,65]

快速排序 [83,72,76,74,79]

[72, 74,76,79,83]

Qsort([70, 74,60,76, 83,72,55,65,79])

划分: [65, 55,60] , 70, [83,72,76,74,79]

Qsort([65, 55,60])

划分: [60, 55], 65, []

Qsort([60,55])

划分: [55], 60, []

Qsort([55])

Qsort([])

Qsort([])

Qsort([83,72,76,74,79])

.

.

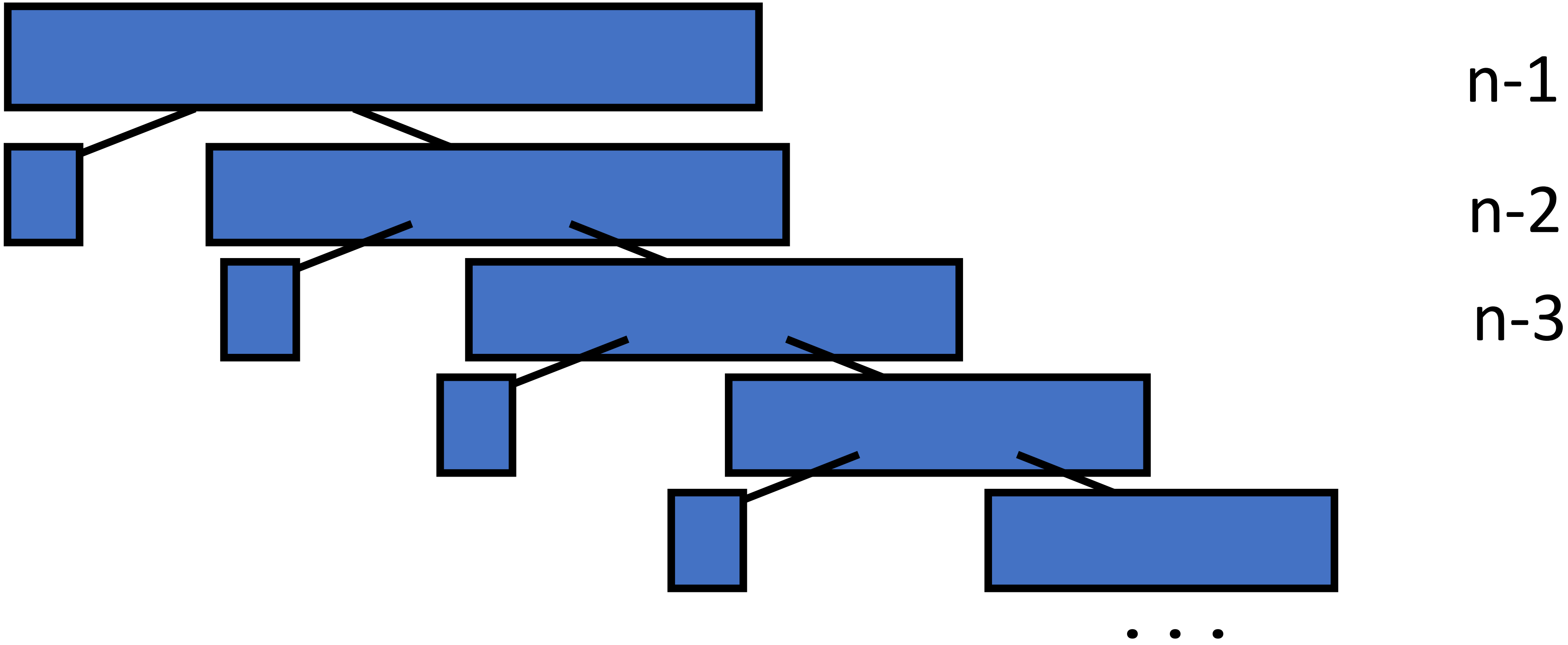
.

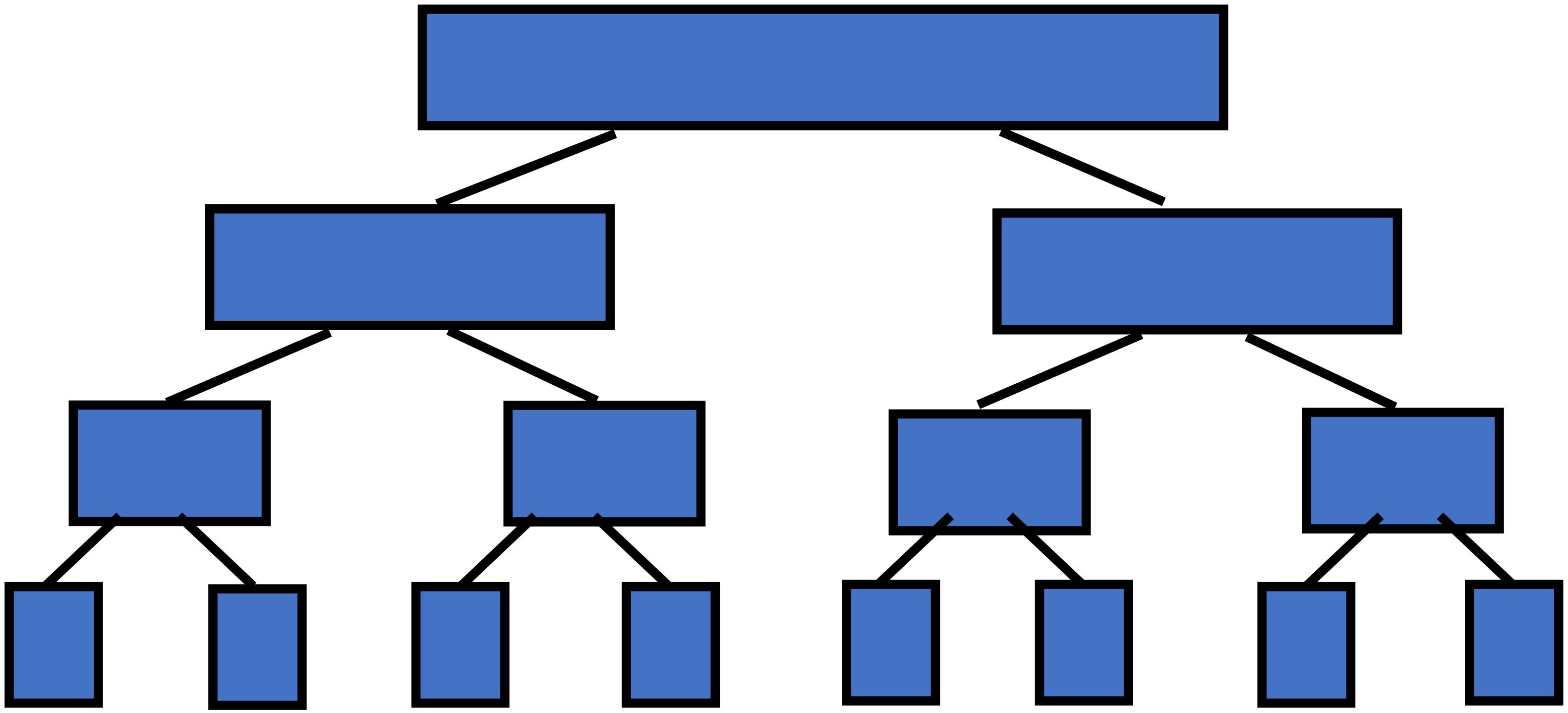
```
void QSort(T a[], int L, int H) {
  if(L < H){   //待排序数列长度大于1

    int pivotloc = Partition(a, L, H);

    //对左子序列进行快速排序
    QSort(a, L, pivotloc - 1);

      //对右子序列进行快速排序
    QSort(a, pivotloc + 1, H);
  }
}
```


n-1
n-2
n-3



$T(n) = 2T(\lfloor n/2 \rfloor)+n-1$ $T(n) = 2T(n/2)+n-1$

$T(n) = \Theta(n\log n)$

平均情况:

$T(0)+T(n-1)+n-1$

$T(1)+T(n-2)+n-1$

$T(2)+T(n-3)+n-1$

...

$T(n-1)+T(0)+n-1$

$T(n) = 2/n\ (T(0)+T(1)+\ldots+T(n-1))+n-1$

$$T(n) = 2/n\ (T(0)+T(1)+\ldots+T(n-1))+n-1$$

$$nT(n) = 2\ (T(0)+T(1)+\ldots+T(n-1))+ n(n-1)$$

$$(n-1)T(n-1) = 2\ (T(0)+T(1)+\ldots+T(n-2))+ (n-1)(n-2)$$

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

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

$$T(n)/(n+1) = T(n-1)/n + 2/(n+1)-2/(n(n+1))$$

$$W(n) \leq W(n-1)+ 2/(n+1) = W(n-2) + 2/n+ 2/(n+1)$$

$$= W(1) + 2/3+\ldots+ 2/n+ 2/(n+1)$$

$$= \Theta\ (\ln n) = \Theta\ (\log n) \Rightarrow T(n) = \Theta\ (n \log n)$$

$1/(n(n+1)) = 1/n-1/(n+1)$

$-2/(1*2)-2/2*3-\ldots-2/(n(n+1)) = -2(1-1/(n+1))$

$$
\begin{aligned}
T(n)/(n+1) &= T(n-1)/n + 2/(n+1)-2/(n(n+1)) \\
&= T(n-2)/(n-1) + 2/n-2/((n-1)n) + 2/(n+1)-2/(n(n+1)) \\
&= T(1)/2 + 2/3+2/4+\ldots 2/(n+1) -2(1-1/(n+1)) \\
&= \tfrac{1}{2}+ 2/3+2/4+\ldots 2/(n+1) -2+2/(n+1) \\
&= 2(1+1/2+1/3+1/(n+1))-c+2/(n+1) \\
&= \Theta (\ln n) = \Theta (\log n)
\end{aligned}
$$

主定理

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

YouTube频道: hwdong

主定理　　多个同规模子问题

Let $T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d)$ *be a recurrence where* $a \geq 1, b > 1$. *Then,*

$$T(n) = \begin{cases} O(n^d \log n) & \textit{if } a = b^d \\ O(n^d) & \textit{if } a < b^d \\ O(n^{\log_b a}) & \textit{if } a > b^d \end{cases}$$

也适用于子问题大小 $\lceil \frac{n}{b} \rceil$, $\lfloor \frac{n}{b} \rfloor$, *or* $\frac{n}{b} + 1$.

- Mult1: $T(n) = 4T\left(\frac{n}{2}\right) + O(n)$.

  The parameters are $a = 4, b = 2, d = 1$, so $a > b^d$, hence $T(n) = O(n^{\log_2 4}) = O(n^2)$.

- Karatsuba: $T(n) = 3T\left(\frac{n}{2}\right) + O(n)$.

  The parameters are $a = 3, b = 2, d = 1$, so $a > b^d$, hence $T(n) = O(n^{\log_2 3}) = O(n^{1.59})$.

- MergeSort: $T(n) = 2T\left(\frac{n}{2}\right) + O(n)$.

  The parameters are $a = 2, b = 2, d = 1$, so $a = b^d$, hence $T(n) = O(n \log n)$.

- Another example: $T(n) = 2T\left(\frac{n}{2}\right) + O(n^2)$.

  The parameters are $a = 2, b = 2, d = 2$, so $a < b^d$, hence $T(n) = O(n^2)$

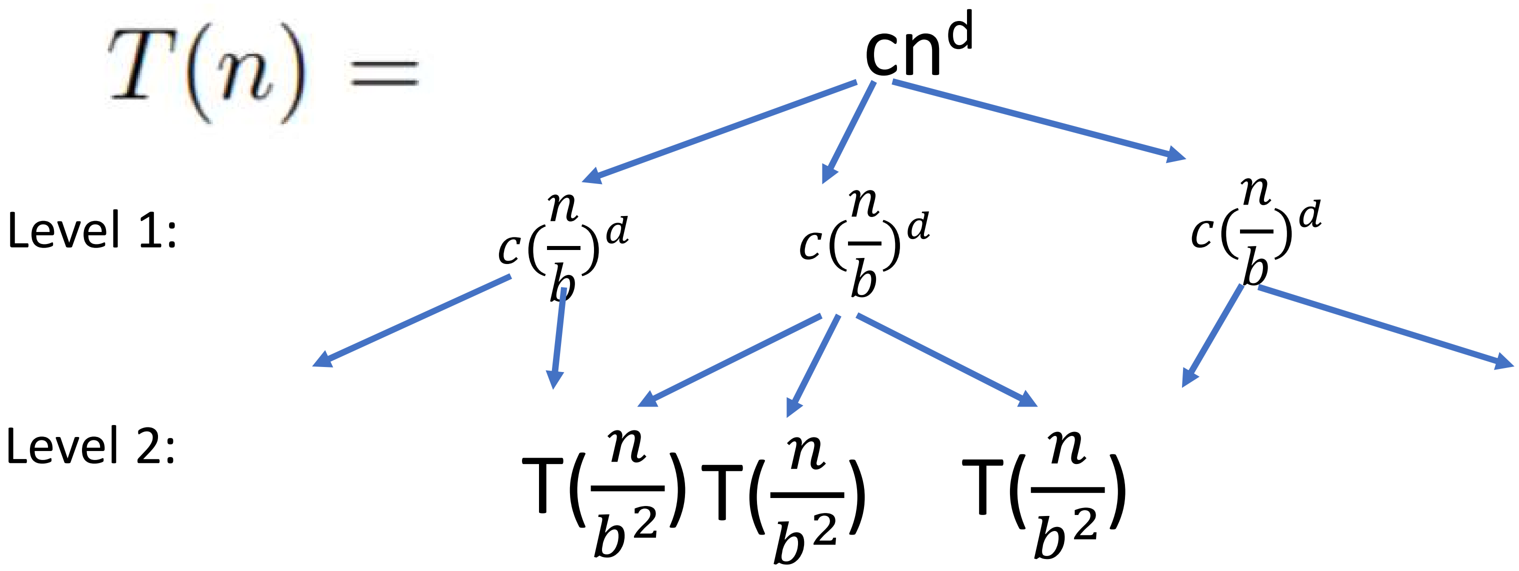
$$T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d) \Longrightarrow T(n) \leq a \cdot T\left(\frac{n}{b}\right) + c \cdot n^d$$

$$T(n) =$$

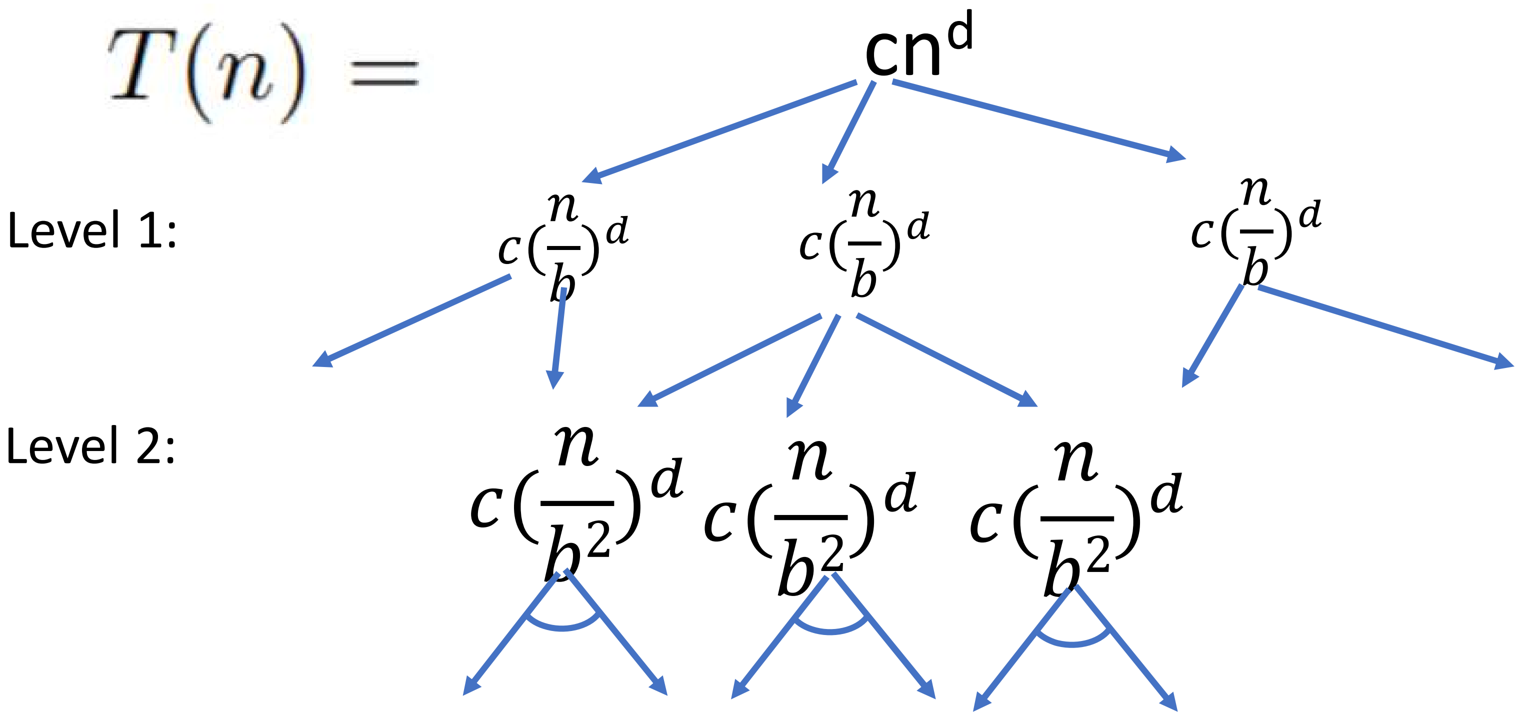
$$\begin{array}{ccc} & cn^d & \\ \swarrow & \downarrow & \searrow \\ T\left(\frac{n}{b}\right) & T\left(\frac{n}{b}\right) & T\left(\frac{n}{b}\right) \end{array}$$

Level 1:

$$T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d) \Longrightarrow T(n) \leq a \cdot T\left(\frac{n}{b}\right) + c \cdot n^d$$

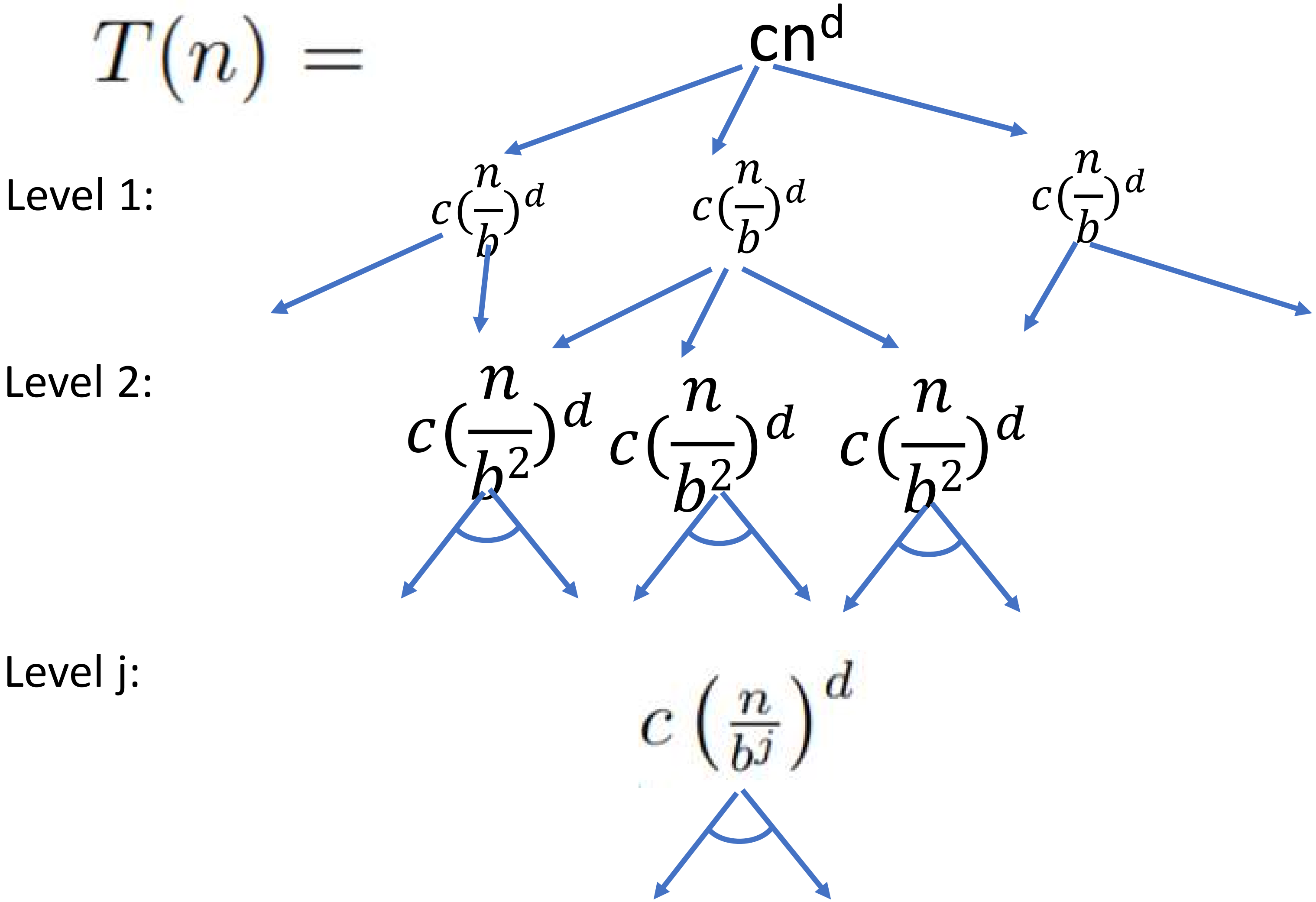


$$T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d) \Rightarrow T(n) \leq a \cdot T\left(\frac{n}{b}\right) + c \cdot n^d$$



$$T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d) \Rightarrow T(n) \le a \cdot T\left(\frac{n}{b}\right) + c \cdot n^d$$

$$T(n) =$$



$$T(n) = a \cdot T\left(\frac{n}{b}\right) + O(n^d) \Rightarrow T(n) \leq a \cdot T\left(\frac{n}{b}\right) + c \cdot n^d$$

$$T(n) = \quad cn^d$$

Level 1: $c(\frac{n}{b})^d \quad c(\frac{n}{b})^d \quad c(\frac{n}{b})^d \qquad a.c(\frac{n}{b})^d$

Level 2: $c(\frac{n}{b^2})^d \quad c(\frac{n}{b^2})^d \quad c(\frac{n}{b^2})^d \qquad a^2.c(\frac{n}{b^2})^d$

Level j: $c\left(\frac{n}{b^j}\right)^d \qquad a^j \cdot c\left(\frac{n}{b^j}\right)^d$

每层 $a^j \cdot c\left(\frac{n}{b^j}\right)^d = cn^d \left(\frac{a}{b^d}\right)^j$

- 总的时间不超过： $cn^d \sum_{j=0}^{log_b n} \left(\frac{a}{b^d}\right)^j$

1. $a = b^d$. $(\log_b n + 1)cn^d = O(n^d \log n)$.

层数

每层

2. $a < b^d$.

$$\sum_{j=0}^{\log_b n} \left(\frac{a}{b^d}\right)^j \le \sum_{j=0}^{\infty} \left(\frac{a}{b^d}\right)^j = \frac{1}{1-\frac{a}{b^d}} = \frac{b^d}{b^d-a}.$$

$$cn^d \cdot \frac{b^d}{b^d-a} = O(n^d).$$

3. $a > b^d$. In this case, $\sum_{j=0}^{\log_b n} \left(\frac{a}{b^d}\right)^j = \frac{\left(\frac{a}{b^d}\right)^{\log_b n+1} - 1}{\frac{a}{b^d} - 1}$.

the total work done is $O\left(n^d \cdot \left(\frac{a}{b^d}\right)^{\log_b n}\right) = O\left(n^d \cdot \frac{a^{\log_b n}}{b^{d \log_b n}}\right)$

$$= O\left(n^d \cdot \frac{n^{\log_b a}}{n^d}\right) = O(n^{\log_b a})$$

主定理2

Let $T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)$ *be a recurrence*
where $a \geq 1$, $b > 1$. *Then,*

- *If* $f(n) = O\left(n^{\log_b(a)-\epsilon}\right)$ *for some constant* $\epsilon > 0$, $T(n) = \Theta\left(n^{\log_b(a)}\right)$.
- *If* $f(n) = \Theta\left(n^{\log_b(a)}\right)$, $T(n) = \Theta\left(n^{\log_b a} \log n\right)$.
- *If* $f(n) = \Omega\left(n^{\log_b(a)+\epsilon}\right)$ *for some constant* $\epsilon > 0$ *and if* $af(n/b) \leq cf(n)$ *for some* $c < 1$ *and all sufficiently large* n, *then* $T(n) = \Theta(f(n))$.

关注我

https://hwdong-net.github.io

Youtube频道:hwdong