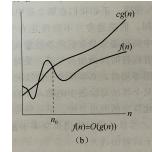


## Asymptotic notation

### $\mathcal{O}$ notation

$$f(n) = \mathcal{O}(g(n)) \quad (\text{大} O, \text{渐近符号}) \quad \text{表示上界}$$

$$\forall C > 0, n_0 > 0 \quad \text{s.t.} \quad 0 \leq f(n) \leq c \cdot g(n) \quad \forall n \geq n_0$$



$$\text{Ex: } 2n^2 = \mathcal{O}(n^3) / \quad 2n^2 \in \mathcal{O}(n^3)$$

$$\mathcal{O}(g(n)) = \left\{ f(n) : \exists C > 0, n_0 > 0, \forall n \geq n_0, \text{s.t. } 0 \leq f(n) \leq c \cdot g(n) \right\}$$

$$\mathcal{O}(n^3) = \left\{ f(n) : \exists C > 0, n_0 > 0, \forall n \geq n_0 \text{s.t. } 0 \leq f(n) \leq c \cdot n^3 \right\}$$

### Macro Convention

$$\text{Ex: } f(n) = n^3 + \mathcal{O}(n^2)$$

$$n^3 + \mathcal{O}(n^2) = \mathcal{O}(n^3)$$

但是  $\forall f(n) \in \mathcal{O}(n^2), \exists h(n) \in \mathcal{O}(n^2) \text{ s.t. } n^3 + f(n) = h(n)$

判断题:  $n^3 + \mathcal{O}(n^2)$  是  $\mathcal{O}(n^3)$  (✓)  
 $\mathcal{O}(n^3)$  是  $n^3 + \mathcal{O}(n^2)$  (✗)

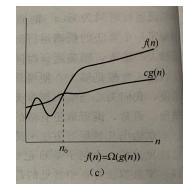
$$\exists h(n) \in \mathcal{O}(n^3)$$

$$\text{s.t. } f(n) = n^3 + h(n)$$

### $\Omega$ notation

$$\Omega(g(n)) = \left\{ f(n) : \exists C > 0, n_0 > 0, \forall n \geq n_0, 0 \leq c \cdot g(n) \leq f(n) \right\}$$

$$\text{Ex: } \sqrt{n} = \Omega(\lg n)$$

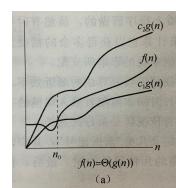


$$\Theta \text{ notation} \quad \Theta(g(n)) = \mathcal{O}(g(n)) \cap \Omega(g(n))$$

对于  $g(n)$

$$\Theta(g(n)) = \left\{ f(n) : \exists C_1, C_2, n_0, \forall n \geq n_0, 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \right\}$$

存在  $C_1, C_2$ , 对于足够大的  $n$ ,  
 使  $f(n)$  处于  $c_1 g(n)$  和  $c_2 g(n)$  之间



记作  $f(n) \in \Theta(g(n))$ ,  $g(n)$  是一个  $f(n)$  的 渐近紧确界

$\Theta(g(n))$  表示每个  $f(n) \in \Theta(g(n))$  的 渐近凸性质, 即  $n$  足够大时,  $f(n) \geq 0$   
因此  $g(n)$  非负

$\Theta$  记号相当于 去掉低阶项, 忽略高阶项系数

Ex

① 证:  $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

$\exists C_1, C_2, n_0$ ,  $\forall n \geq n_0$

s.t.  $C_1 n^2 \leq \frac{1}{2}n^2 - 3n \leq C_2 n^2$

$\Rightarrow C_1 \leq \frac{1}{2} - \frac{3}{n} \leq C_2$

{ 该  $C_2 \geq \frac{1}{2}$   
n > 1 且  $\frac{1}{2} - \frac{3}{n} \leq C_2$  成立  
该  $C_1 \leq \frac{1}{2}$   
 $n > 7$  且,  $C_1 \leq \frac{1}{2} - \frac{3}{n}$  成立

$\Rightarrow \begin{cases} C_2 = \frac{1}{2} \\ C_1 = \frac{1}{2} \\ n_0 = 7 \end{cases}$  说明  $\frac{1}{2}n^2 - 3n = \Theta(n^2)$

② 证:  $n^2 \neq \Theta(n^2)$  反证法

假设  $C_1, n_0$  对  $\forall n \geq n_0$  有  $C_1 n^2 \leq n^2$

$\Rightarrow n \leq \frac{C_1}{2}$   $C_1$  是常量, 对于很大的  $n$ , 不等式不成立

③ 可忽略 1<sup>o</sup> 低阶项 2<sup>o</sup> 最高阶项系数

Ex:  $f(n) = an^3 + bn^2 + cn$ ,  $a, b, c$  为常量, 且  $a > 0$   $\frac{\text{最高阶项}}{\text{总阶数}} \Rightarrow f(n) = \Theta(n^3)$

[证明] 取  $C_1 = \frac{a}{4}$   $C_2 = \frac{7}{4}a$  且  $n_0 = 2 \cdot \max(\frac{16}{a}, \sqrt{\frac{16}{a}})$  ( $\frac{16}{a} < \sqrt{\frac{16}{a}}$ )

假设  $\forall n \geq n_0$  有  $0 \leq an^3 + bn^2 + cn \leq C_1 n^3$

$$\frac{a}{4} \cdot \frac{16a^3}{a} \leq (1 \cdot \frac{16a^3}{a} + b \cdot \frac{16a^2}{a}) \cdot 2 + c \leq \frac{7}{4}a \cdot \frac{16a^3}{a} \quad 4 \cdot \frac{a}{4} \cdot \frac{16a^3}{a} = 0 \cdot \frac{16a^3}{a} + 2 \cdot \frac{16a^3}{a} + c \leq \frac{7}{4}a \cdot \frac{16a^3}{a} + c$$

$$\frac{16a^3}{a} \leq \frac{16a^3}{a} + c \leq \frac{7a^3}{a}$$

$$16 \leq 7c + \frac{2a^3}{a} + c \leq 7(c)$$

Analogies: (类比)

$\Theta \approx \Omega \approx \Theta$

$\Theta$  &  $\omega$  notation

$<$   $>$

$\Theta(g(n)) = \{f(n): \forall c > 0, \exists n_0 > 0,$   
 $\exists \exists \forall n \geq n_0$  s.t.  $0 \leq f(n) \leq c g(n)\}$

Ex:  $\exists n = o(n^2)$ ,  $2n^2 \neq \Theta(n^2)$

在  $f(n) = o(g(n))$  中  $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = 0$

$\omega(g(n)) = \{f(n): \forall c > 0, \exists n_0 > 0$   
 $\exists \exists \forall n \geq n_0$ , s.t.  $0 \leq c g(n) \leq f(n)\}$

Ex:  $\frac{n^3}{2} = \omega(n^2)$ ,  $\frac{n^2}{2} + \omega(n)$

在  $f(n) = \omega(g(n)) \neq \lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} = \infty$

Ex:  $\frac{1}{2}n^2 = \Theta(n^2)$

$\neq o(n^2)$

Recurrences (递归式)

① 代换法

Ex:  $T(n) = 4T(\frac{n}{2}) + n$  [ $T(1) = \Theta(1)$ ]

—猜:  $T(n) = O(n^k)$

—设:  $T(k) \leq ck^3$   $k < n$

— $T(n) = 4T(\frac{n}{2}) + n$

$\leq 4c(\frac{n}{2})^3 + n$

$= \frac{c}{2}n^3 + n$

$= cn^3 - (\frac{1}{2}cn^3 - n) \geq 0$

↑  
跳跃  
余项

$\leq cn^3$  ( $\frac{1}{2}cn^3 - n \geq 0$ )

e.g.  $c \geq 1$   $n \geq 1$

—猜:  $T(n) = O(n^k)$

—设:  $T(n) \leq ck^3 - ck$  (常数)

— $T(n) = 4(\frac{n}{2})^3 + n$

$\leq 4((\frac{n}{2})^3 - \frac{n}{2}) + n$

$= cn^3 + (1 - 4c)n$

$= cn^3 - (1 - 4c)n$

$\leq cn^3 - cn$  ( $c \geq 1$ )

基推:  $T(1) = \Theta(1) \leq c_1 - c_1$  如果  $c_1$  相当  $c_2$  是最大

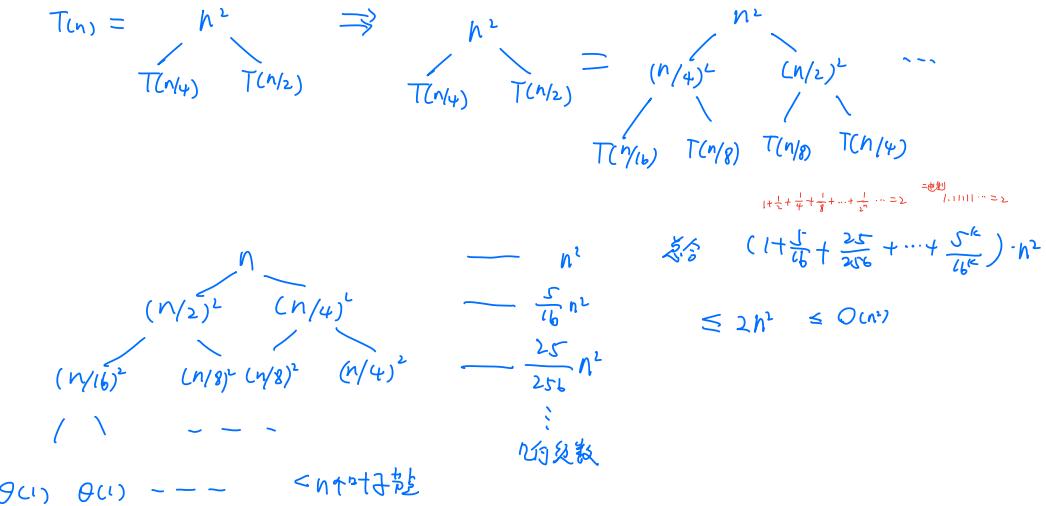
1° 猜数

2° 验证递归式

3° 算出系数

## ② 递归树法

$$Ex: T(n) = T(n/4) + T(n/2) + n^2$$



## ③ 主方法 (中包含递归树上有用)

$$T(n) = a \cdot T(n/b) + f(n)$$

$$a \geq 1 \quad b > 1$$

$f(n)$  渐近函数 (对于存在  $n_0, \exists n \geq n_0$  s.t.  $f(n) > 0$ )

$$Ex: \textcircled{1} \quad T(n) = \frac{4}{a=4} T(n/2) + \frac{n}{b=2} f(n) = n$$

$$n^{\log_b a} = n^2$$

$$\text{Case 1} \quad T(n) = \Theta(n^2)$$

$$\textcircled{2} \quad T(n) = 4T(n/2) + n^2$$

$$n^{\log_b a} = n^2 \quad f(n) = n^2 = \Theta(n^2) \quad (\lg n)^k = 1 \rightarrow k=0$$

$$\text{Case 2} \quad T(n) = \Theta(n^2 \lg n)$$

$$\textcircled{3} \quad T(n) = 4T(n/2) + n^3 = \Theta(n^3)$$

$$\textcircled{4} \quad T(n) = 4T(n/2) + n^2 \lg n = \Theta(n^2 \lg n)$$

主方法适用

比较  $f(n)$  与  $n^{\log_b a}$

$$\text{Case 1: } f(n) = O(n^{\log_b a} - \varepsilon) \quad \varepsilon > 0$$

$$T(n) = \Theta(n^{\log_b a})$$

$$\text{Case 2: } f(n) = \Theta(n^{\log_b a} (\lg n)^k) \quad k \geq 0$$

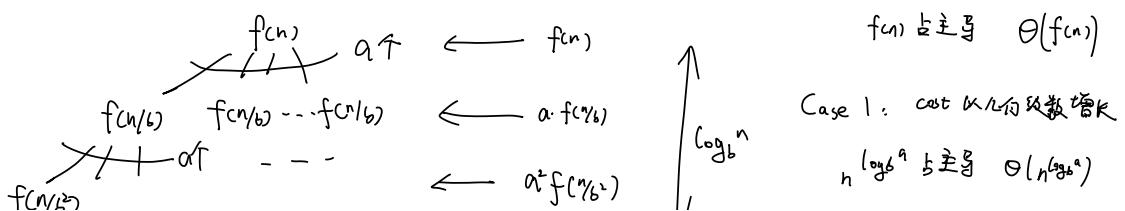
$$T(n) = \Theta(n^{\log_b a} (\lg n)^{k+1})$$

$$\text{Case 3: } f(n) = \Omega(n^{\log_b a} + \varepsilon) \quad \varepsilon > 0$$

$$\& \quad a \cdot f(n/b) \leq (1 - \xi) f(n) \quad \varepsilon' > 0$$

$$\Rightarrow T(n) = \Theta(f(n))$$

## 主方法证明 (课件)



$$\begin{array}{l} \text{Case 1. 叶子层常数: } a = \underbrace{a^{\log_b n}}_{\text{叶子层常数}} = n^{\log_b a} \leftarrow \Theta(n^{\log_b a}) \\ \text{cost} = f(n) \cdot \log_b^n = \Theta(\lg n) \\ T(n) = \Theta(n^{\log_b a} (\lg n)^k \cdot \lg n) \end{array}$$

1 分 2 合 3 合并

运行时间

$$1) \text{合并排序: } T(n) = \underbrace{2T(n/2)}_{\substack{\text{问题规模} \\ \text{子问题递归数}}} + \underbrace{\theta(n)}_{\substack{\text{分治法运行时间} \\ (\text{不包括返回内容})}}$$

$$\begin{aligned} \text{主方法 } T(n) &= b \cdot T(n/a) + f(n) \quad \text{假设 } f(n) \text{ 和 } n^{\log_b a} \\ a=2, b=2, f(n) &= n^{\log_2 2} = n \quad \begin{cases} f(n) = \Theta(n^{\log_2 2} (\lg n)^k) & k=0 \\ T(n) = \Theta(n^{\log_2 2} (\lg n)^{k+1}) & k+1 \end{cases} \\ T(n) &= n \lg n \end{aligned}$$

$$\Rightarrow \text{case 2: } = \Theta(n \lg n)$$

二分查找 在有序数组中查找数  $x$

1. 分: 比较  $x$  与数组元素

2. 合: 在子数组中递归查找

3. 合: 无

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

$$a=1, b=2, f(n) = n^{\log_2 1} = 1 = \Theta(1)$$

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

乘方问题 设  $x, n \in \mathbb{Z}^+, n \geq 0$  计算  $x^n$

$$\begin{aligned} 1^{\circ} \quad & x \cdot x \cdots x = x^n \quad 2^{\circ} \text{ 分治法} \quad x^n = \begin{cases} x^{n/2} \cdot x^{n/2} & n=2k \\ x^{\frac{n-1}{2}} \cdot x^{\frac{n-1}{2}} \cdot x & n=2k+1 \end{cases} \end{aligned}$$

$$\begin{aligned} T(n) &= T(n/2) + \Theta(1) \\ &= \Theta(\lg n) \end{aligned}$$

$$\begin{aligned} 3^{\circ} \text{ 斐波那契数} \quad F_n &= \begin{cases} 0 & n=0 \\ 1 & n=1 \\ F_{n-1} + F_{n-2} & n \geq 2 \end{cases} \\ F_{n-1} + F_{n-2} &= F_n \end{aligned}$$

4<sup>°</sup> 平方递归

$$\text{原理: } \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n$$

$$\Rightarrow \text{Time: } \Theta(\lg n)$$

$$\text{证: Base: } \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^0 = \begin{pmatrix} F_1 & F_0 \\ F_0 & F_{-1} \end{pmatrix}$$

$$\begin{aligned} \text{step: } \begin{pmatrix} F_{n+1} & F_n \\ F_n & F_{n-1} \end{pmatrix} &= \underbrace{\begin{pmatrix} F_n & F_{n-1} \\ F_{n-1} & F_{n-2} \end{pmatrix}}_{\text{递推}} \cdot \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^{n-1} \\ &= \begin{pmatrix} F_n + F_{n-1} & F_{n-1} + F_{n-2} \\ F_n & F_{n-1} \end{pmatrix} \end{aligned}$$

Time:  $\Sigma(\psi^n)$   $\psi = \frac{1+\sqrt{5}}{2}$  指数级

2<sup>°</sup> 自下而上递归, 计算前两项相加获得后一项

计算  $F_0, F_1, F_2, \dots, F_n$

时间:  $\Theta(n)$

3<sup>°</sup> 朴素平方递归式

$$F_n = \varphi^n / \sqrt{5} \quad \text{取整至最近整数 } p \text{ 为 } F_n$$

$$= \begin{pmatrix} 1 & 1 \\ 1 & 0 \end{pmatrix}^n$$

现实中不可行

矩阵乘法  $A = [a_{ij}]$   $B = [b_{ij}]$   $i, j = 1, 2, \dots, n$  P43

$$\Rightarrow C = [c_{ij}] = AB$$

$$C_{ij} = \sum_{k=1}^n a_{ik} b_{kj} \quad \Theta(n^3)$$

把  $n \times n$  的矩阵分成  $2 \times 2$  的块矩阵  
大小为  $n/2 \times n/2$  的矩阵

$$\begin{bmatrix} r & s \\ t & u \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \cdot \begin{bmatrix} e & f \\ g & h \end{bmatrix}$$

$$\begin{aligned} r &= ae + bg && 8 \text{ 次 } n/2 \times n/2 \text{ 矩阵递归乘积} \\ s &= af + bh && 4 \text{ 次 矩阵求和 } \Theta(n^2) \quad a=8, b=2, f=g=1, h=1 \\ t &= ce + dg && \text{case 1} \\ u &= cf + dh \end{aligned}$$

$$T(n) = 8T(n/2) + \Theta(n^2)$$

$$= \Theta(n^3)$$

Sorassen 算法 减少矩阵乘法  $\rightarrow 7$

$$P_1 = a(f+h)$$

$$r = p_5 + p_4 - p_2 + p_6$$

$$P_2 = (a+g) \cdot h$$

$$s = p_1 + p_2$$

$$P_3 = (c+d) \cdot e$$

$$t = p_3 + p_4$$

$$P_4 = d(g-e)$$

$$u = p_5 + p_1 - p_3 - p_7$$

$$P_5 = (a+d)(e+h)$$

$$U = \frac{(a+g+h+d)}{2} + \frac{(a-g)}{2} - \frac{(a+d)}{2} - \frac{(d+g)}{2}$$

$$P_6 = (b-d)(g+h)$$

$$= dh + cf$$

$$P_7 = (a-c)(e+f)$$

1. 分  $A, B$  算  $P_1, P_2 \dots P_7$   $\Theta(n^2)$

2. 治 (递归处理两个子问题  $P_i$ )

3. 合  $r, s, t, u$

$$T(n) = 7T(n/2) + \Theta(n^2)$$

最坏情况  $T(n) = \Theta(n^{2.376})$

$$= \Theta(n^{4.7}) = \Theta(n^{2.81})$$

## QuickSort 快排

- 一种分治算法
- 不需要额外内存

### 分治

1) 分: 把原数组分为两个子数组



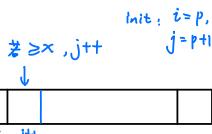
2) 治: 递归处理两个子数组的划分

3) 合并

第一个子数组  
内容大小  $\leq X \leq$  第二个子数组内容大小  
 $\uparrow$   
pivot

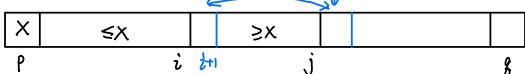
$\Theta(n)$  个分治的子程序

### PARTITION

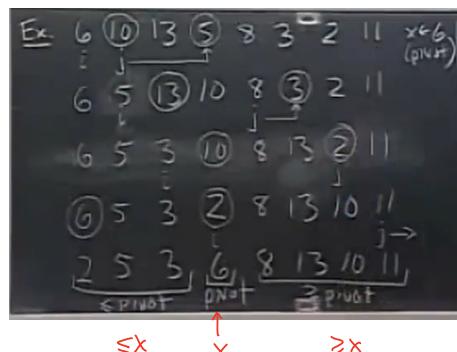


swap  $i \leftrightarrow j$

$\geq x \leftrightarrow \leq x$



运行一趟 Time =  $\Theta(n)$



分析: 设: 元素均不同 (最差情况)

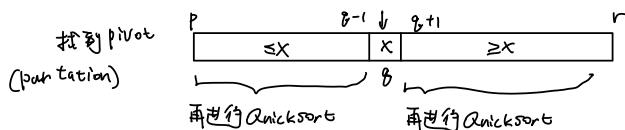
$T(n) =$  最坏情况下排  $n$  个元素

• 数组已被逆序排列

• 划分后有一个区间没有元素 (pivot 取到最大/最小)

Quick Sort ( $A, p, r$ )

Pivot  $x$

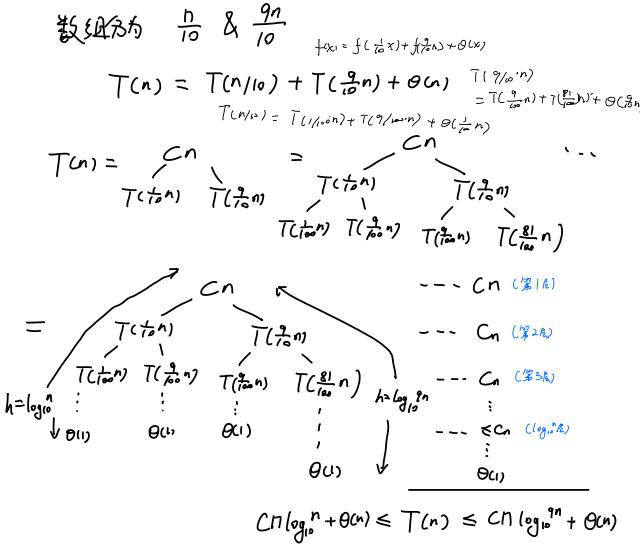


分析：最佳情况（直观理解即可）

- 数组分为  $n/2$  &  $n/2$

$$T(n) = 2T(n/2) + \Theta(n)$$

$$= \Theta(n \lg n)$$



用代换法可证明

$$T(n) = T(n/10) + T(9n/10) + \Theta(n)$$

$$\text{Guess } T(n) = O(n \lg_{10}^{9n})$$

$$\begin{aligned} \text{Assume } T(n) &\leq C(\lg_{10}^{9n}) \\ T(n) &\leq C\left(\frac{n}{10} \lg_{10}^{9n}\right) + C\left(\frac{9n}{10} \lg_{10}^{9n}\right) + \Theta(n) \\ &= \frac{1}{10} C(n \lg_{10}^{9n} - \lg_{10}^{9n}) + \frac{9}{10} C(n \lg_{10}^{9n} - n \lg_{10}^{9n}) + \Theta(n) \\ &= \frac{1}{10} \left[ C(n \lg_{10}^{9n}) - C(n \lg_{10}^{9n}) \right] \\ &+ \frac{9}{10} \left[ C(n \lg_{10}^{9n}) + C(n \lg_{10}^{9n}) - C(n \lg_{10}^{9n}) \right] + \Theta(n) \\ &= C(n \lg_{10}^{9n}) + C(n \lg_{10}^{9n}) + \Theta(n) \end{aligned}$$

$$\begin{aligned} \text{Assume } T(n) &= T(n/10) + \Theta(n) \\ \text{Given } T(n) &= O(n \lg_{10}^{9n}) \\ \text{Assume } T(n) &\leq C(\lg_{10}^{9n}) + k \\ T(n) &\leq C\left(\frac{n}{10} \lg_{10}^{9n}\right) + \frac{k}{10} + C\left(\frac{9n}{10} \lg_{10}^{9n}\right) + \frac{9k}{10} \\ &= \frac{1}{10} C(n \lg_{10}^{9n}) + \frac{9}{10} C(n \lg_{10}^{9n}) + \frac{k}{10} + \frac{9k}{10} \\ &= C(n \lg_{10}^{9n}) - \frac{1}{10} Cn - \frac{9}{10} C(n \lg_{10}^{9n}) + \frac{1}{10} Cn + \frac{9}{10} Cn + \frac{10k}{10} \\ &= C(n \lg_{10}^{9n}) + n + C(n \lg_{10}^{9n}) \\ &\leq C(n \lg_{10}^{9n}) + n \end{aligned}$$

$$\text{③ } = \sum_{k=0}^{n-1} X_k (T(k) + T(n-k-1) + \Theta(n))$$

$$E[T(n)] = E\left[\sum_{k=0}^{n-1} X_k (T(k) + T(n-k-1) + \Theta(n))\right]$$

$$\begin{aligned} &= \sum_{k=0}^{n-1} E[X_k (T(k) + T(n-k-1) + \Theta(n))] \quad \text{X}_k \text{ 相互独立} \\ &= \sum_{k=0}^{n-1} E[X_k] \cdot E[T(k) + T(n-k-1) + \Theta(n)] \quad \text{由 } T(n) = X_0 (T(0) + T(n-1) + \Theta(n)) + X_1 (T(1) + T(n-2) + \Theta(n)) \dots \\ &= \frac{1}{n} \sum_{k=0}^{n-1} E[T(k)] + \frac{1}{n} \sum_{k=0}^{n-1} E[T(n-k-1)] + \frac{1}{n} \sum_{k=0}^{n-1} \Theta(n) \quad = \frac{1}{n} \cdot \Theta(n^2) = \Theta(n) \end{aligned}$$

$$= \frac{2}{n} \sum_{k=0}^{n-1} E[T(k)] + \Theta(n)$$

把  $k=0, 1$  的项吸收入  $\Theta(n)$  中（让余数简洁）

（此时为最坏情况）

$$T(n) = T(0) + T(n-1) + \Theta(n)$$

$$= \Theta(1) + T(n-1) + \Theta(n)$$

$$= T(n-1) + \Theta(n)$$

$$= \Theta(n^2) \quad (\text{等差数列})$$

最坏情况

$$T(n) = \frac{Cn}{T(0)} = \frac{Cn}{\frac{C(n-1)}{T(0)}} = \frac{Cn}{\frac{C(n-1)}{\frac{C(n-2)}{T(0)}}} = \dots = \frac{Cn}{\frac{C(n-1)}{\frac{C(n-2)}{\frac{C(n-3)}{T(0)}}}} = \dots = \frac{Cn}{\frac{C(n-1)}{\frac{C(n-2)}{\frac{C(n-3)}{\frac{C(n-4)}{T(0)}}}}} = \dots$$

$$\Rightarrow \Theta\left(\sum_{k=1}^n C_k\right) = \Theta(n^2)$$

$$T(n) = \Theta(n) + \Theta(n^2) = \Theta(n^2)$$

最坏情况与最好情况轮流出现

$$L(n) = 2L(n/2) + \Theta(n) \quad \text{Lucky}$$

$$U(n) = L(n-1) + \Theta(n) \quad \text{Unlucky}$$

$$\begin{aligned} L(n) &= 2L(n/2-1) + \Theta(n/2) + \Theta(n) \\ &= 2L(n/2-1) + \Theta(n) \quad (\text{主定理}) \\ &= \Theta(n \lg n) \end{aligned}$$

随机化快排 (随机选择主元)

· 运行时间与输入顺序无关

· 不易引起最差情况

分析：①

选择一个随机主元 (pivot)

随机变量  $T(n)$  表示运行时间

设这些随机数相互独立

For  $k = 0, 1, 2, \dots, n-1$ , let

$$X_k = \begin{cases} 1 & \text{产生 } (k) : (n-k-1) \text{ 的划分} \\ 0 & \text{else} \end{cases}$$

$$\text{② } E[X_k] = 0 \cdot P\{X_k=0\} + 1 \cdot P\{X_k=1\}$$

$$= P\{X_k=1\} = \frac{1}{n}$$

$$\Rightarrow E[T(n)] = \frac{2}{n} \sum_{k=2}^{n-1} E[T(k)] + \Theta(n)$$

(4) 欲证:  $E[T(n)] \leq an \lg n$   $a > 0$

代换法: 令  $a$  足够大

s.t.  $\underline{an \lg n} \geq E[T(n)]$  (对于  $n$  足够大)

(注:  $\sum_{k=2}^{n-1} k \lg k \leq \frac{1}{2} n^2 (\lg n - \frac{1}{8} n^2)$ )

$$E[T(n)] \leq \frac{2}{n} \sum_{k=2}^{n-1} ak(\lg k + \Theta(n))$$

$$\leq \frac{2a}{n} \cdot \left( \frac{1}{2} n^2 \lg n - \frac{1}{8} n^3 \right) + \Theta(n)$$

$$= \underline{an \lg n - \left( \frac{1}{4} n - \Theta(n) \right)}$$

目标项

余项

$$\leq an \lg n \quad \text{if } a \text{ 足够大 s.t. } \frac{an}{4} > \Theta(n)$$

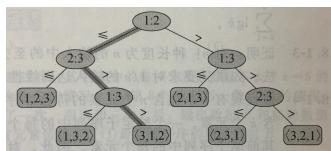
$$T(n) = \begin{cases} T(0) + T(n-1) + \Theta(n) & 0:n-1 \text{ 型划分} \\ T(1) + T(n-2) + \Theta(n) & 1:n-2 \text{ 型划分} \\ \vdots & \vdots \\ T(n-1) + T(0) + \Theta(n) & n-1:0 \text{ 型划分} \end{cases}$$

比较排序模型  
只用比较的方式决定顺序

### 决策树模型

比较排序可抽象为  
此模型

Ex:  $\text{Sort } \langle a_1, a_2, a_3 \rangle$



对  $\langle a_1, a_2, \dots, a_n \rangle$

- 每一节点内有  $i, j$   $i, j \in \{1, 2, \dots, n\}$
- 比较  $a_i, a_j$
- 左子树:  $a_i \leq a_j$       右子树:  $a_i > a_j$
- 每一叶子节点代表1种排序

$\langle \pi(1), \pi(2), \dots, \pi(n) \rangle$

s.t.  $a_{\pi(i)} \leq a_{\pi(j)} \leq \dots \leq a_{\pi(n)}$

用决策树表示决策 (比较排序)

对于随机数等不是一棵树

• 对于每个长  $n$  的数组构一个树

• 将所有结果列出

— running time = 路径长度

— 最差运行时间 = 决策树高度

决策树下界, 高度:  $\Omega(n \lg n)$

证: 令: 叶子节点数  $\geq n!$

$$\begin{aligned} \text{设: 树高 } h \Rightarrow \text{叶子节点数} \leq 2^h \\ \Rightarrow n! \leq 2^h \\ \Rightarrow h \geq \lg n! \geq \lg((n/e)^n) = n \lg(n/e) \\ = n(\lg n - \lg e) \\ = \Omega(n \lg n) \end{aligned}$$

证明, 决策树是渐进最优的  
随机数据在随机情况下  
也是渐进最优的

### 线性时间排序

#### ① 计数排序

Input  $A[1, \dots, n]$ , each  $A[i] \in \{1, 2, \dots, k\}$

Output  $B[1, \dots, n] = \text{有序非空}$

辅助空间  $C[1, \dots, k]$

$$Ex: A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 4 & 1 & 3 & 4 & 3 \end{bmatrix}$$

$$B = \begin{bmatrix} 1 & 3 & 3 & 4 & 4 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 2 & 1 & 4 & 0 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 2 & 2 \\ 1 & 2 & 3 & 4 \end{bmatrix}$$

$$j = A[i] \rightarrow i \\ BC(A[i], j) = AC[j] \\ CC(A[i], j) = CC(A[j]) - 1$$

### 算法的稳定性

#### 基数排序

Ex:	3 2 9	7 2 0	7 2 0	3 2 9
	4 5 7	3 5 5	3 2 9	3 5 5
	6 5 7	4 3 6	4 3 6	4 3 6
	8 3 9	4 5 7	8 3 9	4 5 7
	4 3 6	6 5 7	3 5 5	6 5 7
	7 2 0	3 2 9	4 5 7	7 2 0
	3 5 5	8 3 9	6 5 7	8 3 9

C1111111

Time:  $O(k+n)$  if  $k = O(n)$  then  $O(n)$  time

正确性：设在  $t$  位之前的数已排序

即低  $t-1$  位有序

• 对  $t$  位排序

↳ • 两个数  $t$  位相同  $\Rightarrow$  保持原顺序

$\Rightarrow$  低  $t$  位有序

• 两个数  $t$  位不同  $\Rightarrow$  排序

对每一轮用计数排序

$O(k+n)$

—  $\exists$   $n$  整数 每一个  $b$  bits 长 (取值,  $0 \sim 2^b - 1$ )

— 将整数拆为  $b/r$  位数字, 每个  $r$  bits 是  $(base 2^r)$   
轮数  
计数中的“ $K$ ”

Time:  $O(b/r \cdot (n+k)) = O(b/r \cdot (n+2^b))$

理解:  $r \geq \frac{b}{r} \cdot n \Rightarrow r$  最大值:  $\lceil \lg n \rceil$  ( $n \geq 2$ )  
 $r \geq \frac{b}{r} \cdot 2^b \Rightarrow r$

对  $r$  算出最大值  
 $r = \lceil \lg n \rceil$  得运行时间的上界

$r = \lceil \lg n \rceil \Rightarrow O(\frac{b \cdot n}{\lceil \lg n \rceil})$

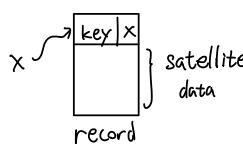
if nums in range  $0 \sim 2^b - 1$

$\Rightarrow$  Time =  $O(d \cdot n)$  ( $0 \sim n^{d-1}$ )

若  $d = O(1) \Rightarrow$  Time =  $O(n)$

## Symbol-table problem

Table S



Operations

- Insert ( $S, x$ )  $S \leftarrow S \cup \{x\}$  } dynamic set
- Delete ( $S, x$ )  $S \leftarrow S - \{x\}$
- Search ( $S, k$ ) ret  $x$  \$  
 $key[x] = k$   
or nil if no such  $x$

发生冲突时:

## 直接寻址表 Direct access table

• keys from  $U = \{0, 1, \dots, m-1\}$

• keys 之间相互独立

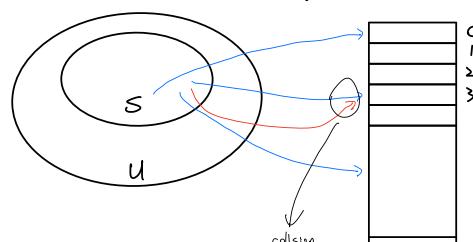
• Array  $T[0, \dots, m-1]$  表示动态集合  $S$

$$T[k] = \begin{cases} x & \text{if } x \in S, key[x] = k \\ \text{nil} & \text{else} \end{cases}$$

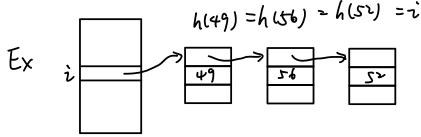
Ops take  $O(1)$  time

## Hashing

Hash function: 随机映射 keys 到 slots of Table T



## ① 用链表存储



Analysis:

最坏情况: 全在 1 个 slot 里

访问时间:  $\Theta(n)$  ( $|S|=n$ )

平均情况:

- 假设为简单均匀 hash
- 每个 key 有相同概率被映射至表 T 中
- key 之间相互独立  $\Rightarrow$

开放寻址法 - 没有链接

- 系统地搜索哈希表，直到 1 个空 slot

$$\cdot h: U \times [0, 1, \dots, m-1] \rightarrow [0, 1, \dots, m-1]$$

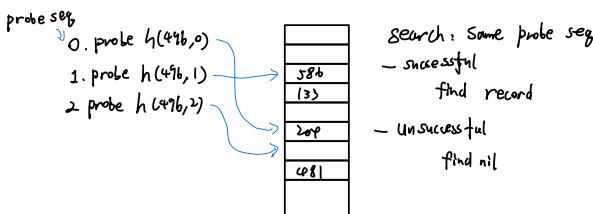
↓            ↓            ↓  
 全域      探查序号      插槽  
 Universal      probe #      slots  
 of keys

probe seq (探查序) 是算术排列的 (随机的  $O(nm)$ )

Table 可能会满 (因为没链接)  $n \leq m$

删除元素会比较难

Ex: Insert  $k=496$



构造探查序列

探查序

$$\text{Linear} - h(k, i) = (h(k, 0) + i) \bmod m \quad i=0, 1, \dots, m-1$$

问题: primary clustering, 某个区域被占满

导致某些区间的 key 集合在整个区间

Double hashing excellent method

$$h(k, i) = (h_1(k) + i \cdot h_2(k)) \bmod m$$

$m=2^r$ ,  $h_2(k)$  为奇数

定义  $\rightarrow$  hash table,  $n$  keys,  $m$  slots

装载因子  $\alpha = n/m$  (每个 slots 中 key 的平均个数)

失败搜索平均时间  $\Rightarrow$  unsuccessful search time

$$\text{Time} = \Theta(1 + \alpha)$$

↑            ↑  
 cost to access slot      cost of searching list

$$\text{Exp search time} = \Theta(1) \text{ if } \alpha = O(1)$$

$$\Rightarrow \text{if } n = O(m)$$

可证 Exp unsuccessful search time  $= \Theta(\log \alpha) \neq \Theta(1)$

选择一个 hash 函数

- 把 keys 均匀分布至 slots 中
- key 本身特性不影响其分布均匀性

## ① Division method (除法哈希法)

$$h(k) = k \bmod m$$

$m$  不宜太小或太大

$m$  应该是一个不接近  $2$  的整数幂的素数

## ② The multiplication method (乘法哈希法)

$$m=2^r \text{ 计算机字长 } W \text{-bit}$$

$$h(k) = (A \cdot k \bmod 2^w) \text{ rsh } (W-r)$$

↓  
 整数且  $2^{W-r} < A < 2^W$

$A$  不宜太接近  $2$  的整数幂

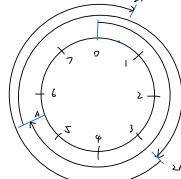
- Fast method: 相乘再右移 ( $\times 2^W$  取余)  
如除法快

Ex:  $m=8=2^3 \quad W=7$

$$\begin{array}{r} .1011001 = A \\ \times 1101011 = k \\ \hline 1001010.0110011 \end{array}$$

mod 6 表示进位      hex 表示结果

车轮模型  
(桶模型)



## Analysis of open addressing

$(0, 1, \dots, m-1)$  的任意一种排列

假设采用均匀哈希，每个 key 都可能地有  $m!$  种探查序列，而

且每个 key 相互独立

$$\text{求证: } E[\# \text{ probes}] \leq \frac{n}{1-\alpha} \quad \alpha < 1 \text{ 且 } n < m$$

↓  
探查次数期望

证: (搜索失败概率) 有  $n$  个名在哪个桶里

1. 至少 1 次探查 碰撞概率:  $n/m$

$\Rightarrow$  第 2 次探查 碰撞概率:  $(n-1)/(m-1)$

$\Rightarrow$  第 3 次探查 碰撞概率:  $(n-2)/(m-2)$

$\vdots$

$\frac{n-i}{m-i} < \frac{n}{m} = \alpha \quad i=1, 2, \dots, m-1$

$$E[\# \text{ probes}] = 1 + \frac{n}{m} \left( 1 + \frac{n-1}{m-1} \cdot \left( 1 + \frac{n-2}{m-2} \cdot \left( \dots \left( 1 + \frac{1}{m-1} \right) \dots \right) \right) \right)$$

## Hashing II

### 哈希的缺点

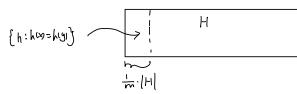
对于任意哈希函数，都有一组不好的keys，全部映射到同一个slot（槽）上  
随机选择一个哈希函数，使之独立于输入的key

### Universal hashing 全域哈希

设U为keys的全集，H是哈希函数的有限集合，H的哈希函数将U的keys映射到哈希表的slot（槽）里

对于  $\forall x, y \in U$  且  $x \neq y$ ，则有  $| \{ h \in H : h(x) = h(y) \} | = \frac{|H|}{m}$

如果哈希函数h是随机从函数集H中选出，x和y发生碰撞的概率是  $\frac{1}{m}$



从哈希函数集里H里随机选择哈希函数h，假设要将n个keys放入T表的m个槽里，

对于key x，x发生碰撞的概率是：

$$E[\# \text{collisions with } x] < \frac{n}{m}$$

证明：设  $C_{xy}$  为随机变量，

表示哈希表T中的keys x与y发生碰撞的总次数

$$\text{设 } C_{xy} = \begin{cases} 1 & (h(x) = h(y)) \\ 0 & \text{else} \end{cases}$$

$$E[C_{xy}] = \frac{1}{m}$$

$$E[C_x] = E\left[\sum_{y \in T - \{x\}} C_{xy}\right]$$

$$= \sum_{y \in T - \{x\}} E[C_{xy}]$$

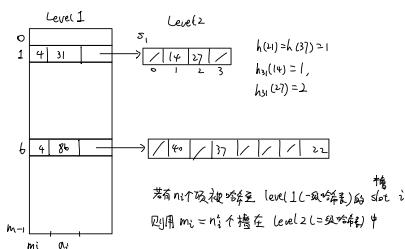
$$= \sum_{y \in T - \{x\}} \frac{1}{m} = \frac{n-1}{m}$$

### 完全哈希 (perfect hashing)

给定n个keys，构建一个静态哈希表（没有插入和删除操作）大小  $O(n)$ 。

使得最坏情况下的查找时间是  $O(1)$

Idea：使用一个双级结构，并在每一级实用全域哈希（哈希表套娃），使得在第二级没有碰撞



$$\leq 1 + \alpha (1 + \alpha (1 + \dots (1 + \alpha))) \\ \leq 1 + \alpha + \alpha^2 + \alpha^3 + \dots \\ = \sum_{i=0}^{\infty} \alpha^i = \frac{1}{1-\alpha} \quad *$$

- $\alpha < 1$   $\alpha$  为常数  $\Rightarrow O(1)$  probes
- Table, 5% fail  $\Rightarrow 2$  probes
- 99%  $\Rightarrow 10$  probes

### 构造一个全域哈希

设m为质数，把全域里的任意key k分解成r+1位：

$$k = \langle k_0, k_1, \dots, k_r \rangle \quad 0 \leq k_i \leq m-1 \quad (\text{把 } k \text{ 变成 } m \text{ 位数})$$

随机选取数  $a = \langle a_0, a_1, \dots, a_r \rangle$  且  $a_i$  是从  $0, \dots, m-1$  随机选取的（一个随机的m进制数）

$$\text{设哈希函数: } h_a(k) = \left( \sum_{i=0}^r a_i k_i \right) \bmod m$$

$$\text{How big is } \#\{h_a\} ? \quad |\{h_a\}| = m^{r+1}$$

求证  $h_a$  是全域的  $H$  满足把任意2个keys映射到同一位置的哈希函数的个数为：  $\frac{|H|}{m}$

证：

$$x = \langle x_0, x_1, \dots, x_r \rangle$$

$$y = \langle y_0, y_1, \dots, y_r \rangle \quad \text{且 } x \neq y$$

即  $x_j$  至少有一个不同（ $x_0, x_1, \dots, x_r$  和  $y_0, y_1, \dots, y_r$  不同）

不妨设  $x_0 \neq y_0$

即  $h_a(x) \neq h_a(y)$  且  $h_a(x) = h_a(y)$  的keys数量

$$P \left[ \sum_{i=0}^r a_i x_i \equiv \sum_{i=0}^r a_i y_i \pmod{m} \right]$$

$$\Rightarrow \sum_{i=0}^r a_i (x_i - y_i) \equiv 0 \pmod{m}$$

$$\Rightarrow a_0 (x_0 - y_0) + \sum_{i=1}^r a_i (x_i - y_i) \equiv 0 \pmod{m}$$

$$\Rightarrow a_0 (x_0 - y_0) \equiv - \sum_{i=1}^r a_i (x_i - y_i) \pmod{m}$$

因  $x_0 \neq y_0$ ，则  $\exists (x_0 - y_0)^{-1}$

$$\Rightarrow a_0 \equiv - \sum_{i=1}^r a_i (x_i - y_i) \cdot (x_0 - y_0)^{-1}$$

假若  $a_0, a_1, \dots, a_r$  因为  $x_0 \neq y_0$  保证了一组  $a_0, a_1, \dots, a_r$

存在  $y_0 - x_0$  一个  $a_0$  的取值导致碰撞

$$\text{导致 } x_0, y_0 \text{ 碰撞的 } h_a \text{ 的数量} = m \cdot m \cdot \dots \cdot m \cdot 1 = \frac{|H|}{m} = m^r \quad *$$

注：  
完美哈希 解说  
令 m 为质数，对  $a_i \in \mathbb{Z}_m$   
整数  
 $\mathbb{Z}_m$  是  $\mathbb{Z}$  mod  $m$ , 即  $0, 1, \dots, m-1$   
 $z = (a_0, a_1, \dots, a_r)$   
存在  $m^r - 1$  个  $z \in \mathbb{Z}_m^r$ , st.  $z^{-1} \equiv 1 \pmod{m}$   
 $E_x, m = 7$   
 $\frac{2}{7}, \frac{1}{7}, \frac{2}{7}, \frac{3}{7}, \frac{4}{7}, \frac{5}{7}, \frac{6}{7}$   
 $2^r 2^r \equiv 1 \pmod{m}$

### level 2 Analysis:

定理：把n个keys哈希映射到  $m = n^2$  个slots中时用到的哈希函数从全域集合H中选择的话，

$$\Rightarrow \text{碰撞次数的期望: } E[\# \text{collisions}] < 1/2$$

证：从全域哈希函数  $h$  的角度看

对2个给定的keys碰撞概率是： $1/m = 1/n^2$

$$E[\# \text{collisions}] = \binom{n}{2} \cdot \frac{1}{n^2} = \frac{n(n-1)}{2} \cdot \frac{1}{n^2} < \frac{1}{2} \quad *$$

马克达不等式，对随机变量  $X \geq 0$   $P\{X \geq t\} \leq \frac{E[X]}{t}$   $\rightarrow$  例  $E[X] = \sum_{x=0}^{\infty} x P[X=x]$

$\Rightarrow \sum_{x=t}^{\infty} x P[X=x] \leq \frac{E[X]}{t}$

没有碰撞的概率： $P[\# \text{collisions}] \geq 1/2$

$$\text{证明: } P[\# \text{collisions}] \leq E[\# \text{collisions}] / 2 < 1/2 \quad *$$

为了查到一个好的level-2哈希函数，只需要随机测试几个，就能很快找到一个满足条件的，因为至少有一半的函数是有效的

证明表的总大小为  $O(n)$ ;

$$m = n$$

对于level-1，把slots（槽）的数量设置成keys的数量，一级存储的大小就是  $O(n)$

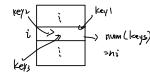
令  $n_i$  为被映射到表T的slot i 中所有keys的数量

则 slot i 中的keys数量

对 level-2 表  $S_i$ ，设槽数为  $m_i = n_i^2$

对 level-2 表  $S_i$ ，设槽数为  $m_i = n_i^2$

$$E[\text{total storage}] = n + E\left[\sum_{i=0}^{n-1} O(n_i^2)\right]$$



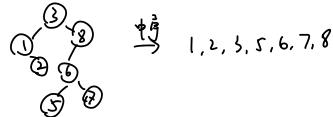
BST 排序 (A):

```

 $T \leftarrow \emptyset$ 
for  $i \leftarrow 1$  to  $n$ 
do Tree-Insert ( $T, A[i]$ )
Inorder-Tree-Walk (root [ $T$ ])

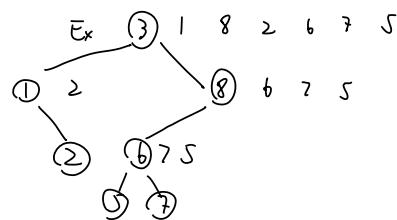
```

Ex:  $A = [3, 1, 8, 2, 6, 7, 5]$



BST Sort & Quick Sort 的关系

BST Sort & Quick Sort 都是在做相同的比较，但比较顺序不同  
是个稳定的排序算法



定理:  $E[\text{height of rand built BST}] = O(\lg n)$

证明大纲:

① 用 Jensen 不等式  $f(E[X]) \leq E[f(X)]$   
 $f(x)$  是凹函数

② 不分析  $X_n$   
 $X_n = r.v. \text{ of } n \text{ 结点构成的 BST 的高度}$   
分析  $Y_n = 2^{X_n}$

③ 证明  $E[Y_n] = O(n^3)$

④ 总结:

$$E[X_n] = E[2^{X_n}] = E[Y_n] = O(n^3)$$

$$\Rightarrow E[X_n] \leq \lg(O(n^3)) = 3\lg n + O(1)$$

BST height Analysis

$X_n = r.v.$  是随机构造的 BST 的高度 (unif dist)

$$Y_n = 2^{X_n}$$

若 root 排第  $k$  位,

$$\Rightarrow X_n \geq \max\{X_{k-1}, X_{n-k}\}$$

$$Y_n = 2 \cdot \max\{Y_{k-1}, Y_{n-k}\}$$

定义 指标 r.v.

$\rightarrow$  , 1 root 排第  $k$  位

$$= \Theta(n) \quad \text{随机排序} (\text{中大, 1995})$$

Analysis: Time:  $1^{\circ} \notin \Theta, O(n)$

$\therefore n \in \text{Tree-Insert}$ ,  $\begin{cases} \Omega(n \lg n) \\ O(n^2) \end{cases}$

unlucky case: 根结点经常是

BST, 

height:  $O(n)$

Time:  $O(n^2)$

lucky case: 生长于平衡的汉明树

height:  $O(\lg n)$

Time:  $O(n \lg n)$

原理和快速排序一样

Time =  $\sum_{x \in T} \text{depth}(x) = \Theta(n^2)$

随机 BST Sort

① 打乱 A

② BST Sort(A)

Time = 随机快排的时间

$$\begin{aligned} E[\text{BST time}] &= E[\text{rand Quick Sort Time}] \\ &= \Theta(n \lg n) \end{aligned}$$

随机构造 BST: 通过 rand BST Sort 建立树

$$\begin{aligned} \text{Time (BST Sort)} &= \sum_{x \in T} \text{depth}(x) \\ E[\sum_{x \in T} \text{depth}(x)] &= \Theta(n \lg n) \end{aligned}$$

$$E\left[\frac{1}{n} \sum_{x \in T} \text{depth}(x)\right] = \frac{\Theta(n \lg n)}{n} = \Theta(\lg n)$$

$$\begin{aligned} E[x]: \quad &\text{Avg. depth} \\ &\leq \frac{1}{n} (n \lg n + \sqrt{n} \cdot \sqrt{n}) \\ &= O(\lg n) \end{aligned}$$

①  $f$  是  $R \rightarrow R$  凸函数

对  $\forall x, y \in R$

&  $\alpha, \beta \geq 0$  且  $\alpha + \beta = 1$

$$f(\alpha x + \beta y) \leq \alpha f(x) + \beta f(y)$$

引理:  $f$  是  $R \rightarrow R$  凸函数

$x_1, x_2, \dots, x_n \in R$

$\alpha_1, \alpha_2, \dots, \alpha_n \geq 0$

$$\sum_k \alpha_k = 1$$

$$\Rightarrow f\left(\sum_{k=1}^n \alpha_k x_k\right) \leq \sum_{k=1}^n \alpha_k \cdot f(x_k)$$

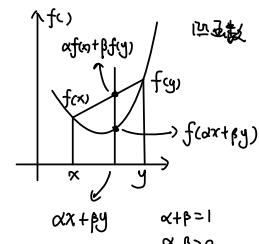
证明:

base:  $n=1 \quad \alpha=1 \quad f(1 \cdot x_1) \leq 1 \cdot f(x_1)$

$$\text{Step: } f\left(\sum_{k=1}^n \alpha_k x_k\right) = f\left(\alpha_n x_n + (1-\alpha_n) \cdot \sum_{k=1}^{n-1} \frac{\alpha_k}{1-\alpha_n} x_k\right)$$

$$\leq \alpha_n \cdot f(x_n) + (1-\alpha_n) f\left(\sum_{k=1}^{n-1} \frac{\alpha_k}{1-\alpha_n} x_k\right)$$

$$\leq \alpha_n f(x_n) + (1-\alpha_n) \sum_{k=1}^{n-1} \frac{\alpha_k}{1-\alpha_n} f(x_k) \quad \leftarrow \text{由归纳假设}$$



$$Z_{nk} = \begin{cases} 1 & \text{else} \end{cases}$$

$$\mathbb{P}[Z_{nk}=1] = \mathbb{E}[Z_{nk}] = 1/2$$

$$Y_n = \sum_{k=1}^n Z_{nk} \cdot (2 \max\{Y_{k-1}, Y_{n-k}\})$$

$$\mathbb{E}[Y_n] = \mathbb{E}\left[\sum_{k=1}^n Z_{nk} (2 \max\{Y_{k-1}, Y_{n-k}\})\right]$$

$$= \sum_{k=1}^n \mathbb{E}[Z_{nk} (2 \max\{Y_{k-1}, Y_{n-k}\})]$$

$$= 2 \sum_{k=1}^n \mathbb{E}[Z_{nk}] \cdot \mathbb{E}[\max\{Y_{k-1}, Y_{n-k}\}]$$

$$\leq \frac{2}{n} \sum_{k=1}^n \mathbb{E}[Y_{k-1} + Y_{n-k}]$$

$$= \frac{2}{n} \sum_{k=1}^n (\mathbb{E}[Y_{k-1}] + \mathbb{E}[Y_{n-k}])$$

$$= \frac{4}{n} \sum_{k=0}^{n-1} \mathbb{E}[Y_k] = \mathbb{E}[Y_n]$$

$$= \sum_{k=1}^n \alpha_k f(x_k) \neq$$

Jensen 不等式证明:

结论:  $f(\mathbb{E}[x]) \leq \mathbb{E}[f(x)]$   $f$  为凸函数

证明,  $x$  是整数 r v

$$f(\mathbb{E}[x]) = f\left(\sum_{x=-\infty}^{\infty} x \cdot P[x=x]\right)$$

$$\leq \sum_{x=-\infty}^{\infty} P[x=x] \cdot f(x)$$

$$= \sum_{y \in \text{range}(f)} y \sum_{x \in \text{for } y} P[x=x]$$

$$= \mathbb{E}[f(x)] \neq$$

代换法: Guess  $\mathbb{E}[Y_n] \leq cn^3$

Proof:

Base:  $n=O(1)$  (若  $n$  充分小时)

$$\text{Step: } \mathbb{E}[Y_n] \leq \frac{4}{n} \sum_{k=0}^{n-1} \mathbb{E}[Y_k]$$

$$\leq \frac{4c}{n} \sum_{k=0}^{n-1} ck^3 \quad (\text{假设假设})$$

$$\leq \frac{4c}{n} \int_0^n x^3 dx = cn^3$$

$$\mathbb{E}[Y_n] = \mathbb{E}[Y_n] \leq cn^3, \text{ 取对数 } \mathbb{E}[X_n] = O(\lg n)$$

## p15 动态规划

$\mathcal{LCS}_x$ , 最长公共子序列问题 (LCS) longest common subsequence  
规定两个序列,  $x 1 \sim m$ ,  $y 1 \sim n$ , 找出一个它们的最长子序列 (LCS)

$$\begin{array}{l} x: A B C B D A B \\ y: B D C A B A \end{array} \left. \begin{array}{l} B D A B \\ B C A B \\ B C B A \end{array} \right\} = \text{LCS}(x, y)$$

① 穷举法

Analysis

• check =  $O(n)$

•  $2^m$  subseq of  $x$

每个 m 维向量对应一个子序列

Time:  $O(n \cdot 2^m)$

② 优化:

1. 查看  $x$  和  $y$  的 LCS 有多长

2. 扩展这个算法找到最长公共子序列 LCS

Notation:  $|S|$  表示  $S$  的长度

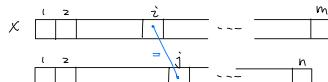
只考虑  $x$  和  $y$  的前缀

Define  $C[i:j] = |\text{LCS}(x[1:i], y[1:j])|$

Then  $C[m, n] = |\text{LCS}(x, y)|$

$$C[i:j] = \begin{cases} C[i-1:j-1] + 1 & \text{if } x[i]=y[j], \\ \max[C[i:j-1], C[i-1:j]] & \text{else} \end{cases}$$

证: case  $x[i] = y[j]$



令  $\mathcal{Z}[1:k] = \text{LCS}(x[1:i], y[1:j])$

即  $C[i:j] = k$

则  $\mathcal{Z}[k] = x[i] = y[j]$  表示是, 第 k 位匹配  $x[i]$  和  $y[j]$

因此  $\mathcal{Z}[1:k-1]$  是  $x[1:i-1]$  和  $y[1:j-1]$  的相同部分

则  $\mathcal{Z}[1:k-1] = \text{LCS}(x[1:i-1], y[1:j-1])$