算法设计与分析
Algorithms Design & Analysis

第三讲：递归关系

递归关系（Recurrence Relations）

- Recurrence relations describe functions over the natural numbers.(递归关系描述的是自然数上的函数关系)
- They express the value of a function for a given integer $n>0$ as functions of the values of the function for arguments less than n.(对于某个$n>0$ 的函数值，通过小于n的函数值表示出来)

递归实例(Recurrence Examples)

$$s(n)=\begin{cases}0 & n=0\\ c+s(n-1) & n>0\end{cases}$$

$$s(n)=\begin{cases}0 & n=0\\ n+s(n-1) & n>0\end{cases}$$

$$T(n)=\begin{cases}c & n=1\\ 2T\left(\frac{n}{2}\right)+c & n>1\end{cases}$$

$$T(n)=\begin{cases}c & n=1\\ aT\left(\frac{n}{b}\right)+cn & n>1\end{cases}$$

为什么要分析递归关系？

- **The running times of many algorithms (especially recursive algorithms) are easily expressed using recurrence relations.（许多算法，特别是递归算法，时间开销函数都可以用递归关系来描述）**
- **We need to solve them, that is, derive closed expressions of the functions expressed by recurrences using O-, Ω-, and Θ-notation.（分析开销函数并表示出来）**
- **Example:** (Merge Sort)

$$T(n)=\begin{cases}O(1) & \text{if } n=1\\ T(\lceil n/2\rceil)+T(\lfloor n/2\rfloor)+O(n) & \text{if } n>1\end{cases}$$

求解方法

- 置换法（**Substitution**）
  - Make a guess and verify it(假设-论证).
- 递归树（**Recursion Tree**）
  - Allows us to arrive at a guess(帮助猜想).
  - The guess can then be verified using the substitution method(置换法论证).
- 迭代法（**Iteration**）
- 主方式（**Master Theorem**）
  - Provides solutions to recurrences of a quite restricted, but very common, nature(定理化).

置换法

- **The three steps of the substitution method:** (置换法的三步骤)

1. Make a good guess(猜想)
2. Verify the guess, assuming that it can be verified for some base case $n = n_0$
   (验证猜想对于$n = n_0$的正确性)
3. Choose an n_0 for which the guess works and such that Step 2 for any $n > n_0$ does not depend on the claim for some $n' < n_0$
   (验证猜想对于$n > n_0$的正确性)

如何获得好的猜想?(How To Make a Good Guess)

- Experience helps(经验)
- Variable substitution（更换变元）
- Prove loose upper and lower bounds and tighten them step by step(先松后紧)

例

Determine an upper bound on the recurrence
(求以下迭代关系的上界)：
$T(n) = 2T(\lfloor n/2 \rfloor) + n$

- Making a good guess(猜想)
  - $T(n) = 2T(n/2) + \Theta(n) \rightarrow T(n) = \Theta(n \lg n)$

Guess：$T(n) = 2T(\lfloor n/2 \rfloor) + n \rightarrow T(n) = O(n \lg n)$

- Use induction to find the constants and show that solution works： $\forall$ c, s.t (such that)(归纳证明$\forall$ c, n>N满足)

$T(n) \le cn \lg n$?

例（续）

假设在$n \ge 2$时对于$\lfloor n/2 \rfloor$成立,
即$T(\lfloor n/2 \rfloor) \le c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)$

$$
\begin{aligned}
T(n) &= 2T(\lfloor n/2 \rfloor) + n \\
&\le 2(c\lfloor n/2 \rfloor \lg(\lfloor n/2 \rfloor)) + n \\
&\le cn \lg(n/2) + n \\
&= cn \lg n - cn \lg 2 + n \\
&= cn \lg n - cn + n \\
&\le cn \lg n \qquad \text{对于} c>1
\end{aligned}
$$

例（续）

- Examples:
  - $T(n) = 2T(n/2) + \Theta(n) \rightarrow T(n) = \Theta(n \lg n)$
  - $T(n) = 2T(\lfloor n/2 \rfloor) + n \rightarrow T(n) = \Theta(n \lg n)$
  - $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n \rightarrow$???

例（续）

- Examples:
  - $T(n) = 2T(n/2) + \Theta(n) \rightarrow T(n) = \Theta(n \lg n)$
  - $T(n) = 2T(\lfloor n/2 \rfloor) + n \rightarrow T(n) = \Theta(n \lg n)$
  - $T(n) = 2T(\lfloor n/2 \rfloor + 17) + n \rightarrow T(n) = \Theta(n \lg n)$

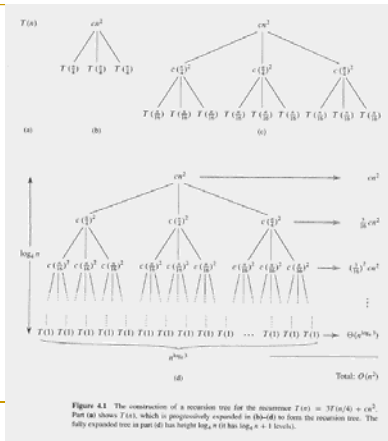
Changing Variables（更换变元）

- Use algebraic manipulation to turn an unknown recurrence into one similar to what you have seen before. (通过数学变化,将陌生的迭代关系转变为熟悉的形式)
  - Example: $T(n) = 2T(n^{1/2}) + \lg n$
  - Rename **$m = \lg n$** and we have $T(2^m) = 2T(2^{m/2}) + m$
  - Set $S(m) = T(2^m)$ and we have $S(m) = 2S(m/2) + m \Rightarrow S(m) = O(m \lg m)$
  - Changing back from $S(m)$ to $T(n)$, we have $T(n) = T(2^m) = S(m) = O(m \lg m) = O(\lg n \lg \lg n)$

递归树（Recursion Tree）

- The recursion tree method expands the recurrence and visualizes this expansion.
- (展开使之可视化)
- **Example:**

$$
\begin{aligned}
T(n) &= 3\, T(n/4) + n^2 \\
&= 3\,\left(T(n/16) + n^2/16 \right) + n^2 \\
&= 3\,\left((T(n/64) + n^2/256) + n^2/16 \right) + n^2 \\
&= \ldots
\end{aligned}
$$



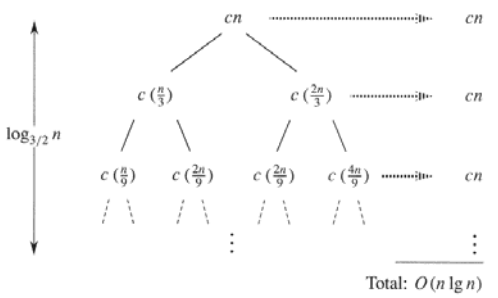


Figure 4.2 A recursion tree for the recurrence $T(n) = T(n/3) + T(2n/3) + cn$.

迭代法（Iteration）

1. Expand the recurrence(展开)
2. Work some algebra to express as a summation(代数运算)
3. Evaluate the summation(求和)

$$
s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases}
$$

- s(n) =
  c + s(n-1)
  c + c + s(n-2)
  2c + s(n-2)
  2c + c + s(n-3)
  3c + s(n-3)
  …
  kc + s(n-k) = ck + s(n-k)

$$
s(n) = \begin{cases} 0 & n = 0 \\ c + s(n-1) & n > 0 \end{cases}
$$

- So far for n >= k we have
  - s(n) = ck + s(n-k)
- What if k = n?
  - s(n) = cn + s(0) = cn

$$s(n)=\begin{cases}0 & n=0\\ c+s(n-1) & n>0\end{cases}$$

- So far for n >= k we have
  - s(n) = ck + s(n-k)
- What if k = n?
  - s(n) = cn + s(0) = cn
- So

$$s(n)=\begin{cases}0 & n=0\\ c+s(n-1) & n>0\end{cases}$$

- Thus in general
  - s(n) = cn

$$s(n)=\begin{cases}0 & n=0\\ n+s(n-1) & n>0\end{cases}$$

- s(n)

= n + s(n-1)
= n + n-1 + s(n-2)
= n + n-1 + n-2 + s(n-3)
= n + n-1 + n-2 + n-3 + s(n-4)
= …
= n + n-1 + n-2 + n-3 + … + n-(k-1) + s(n-k)

$$s(n)=\begin{cases}0 & n=0\\ n+s(n-1) & n>0\end{cases}$$

- s(n)

= n + s(n-1)
= n + n-1 + s(n-2)
= n + n-1 + n-2 + s(n-3)
= n + n-1 + n-2 + n-3 + s(n-4)
= …
= n + n-1 + n-2 + n-3 + … + n-(k-1) + s(n-k)
$= \sum_{i=n-k+1}^{n} i \; + \; s(n-k)$

$$s(n)=\begin{cases}0 & n=0\\ n+s(n-1) & n>0\end{cases}$$

- So far for n >= k we have

$$\sum_{i=n-k+1}^{n} i \; + \; s(n-k)$$

$$s(n)=\begin{cases}0 & n=0\\ n+s(n-1) & n>0\end{cases}$$

- So far for n >= k we have

$$\sum_{i=n-k+1}^{n} i \; + \; s(n-k)$$

- What if k = n?

$$s(n)=\begin{cases}0 & n=0\\ n+s(n-1) & n>0\end{cases}$$

- So far for n >= k we have

$$\sum_{i=n-k+1}^{n} i \; + \; s(n-k)$$

- What if k = n?

$$\sum_{i=1}^{n} i \; + \; s(0) \; = \; \sum_{i=1}^{n} i \; + \; 0 \; = \; n\frac{n+1}{2}$$

$$s(n) = \begin{cases} 0 & n = 0 \\ n + s(n-1) & n > 0 \end{cases}$$

- So far for n >= k we have

$$\sum_{i=n-k+1}^{n} i \quad + \quad s(n-k)$$

- What if k = n?

$$\sum_{i=1}^{n} i \quad + \quad s(0) \quad = \quad \sum_{i=1}^{n} i \quad + \quad 0 \quad = \quad n\frac{n+1}{2}$$

- Thus in general

$$s(n) \quad = \quad n\frac{n+1}{2}$$

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + c & n > 1 \end{cases}$$

- T(n) = 2T(n/2) + c
  =2(2T(n/2/2) + c) + c
  $=2^2T(n/2^2) + 3c$
  $=2^2(2T(n/2^2/2) + c) + 3c$
  $=2^3T(n/2^3) + 4c + 3c$
  $=2^3T(n/2^3) + 7c$
  $=2^3(2T(n/2^3/2) + c) + 7c$
  $=2^4T(n/2^4) + 15c$
  …
  $=2^kT(n/2^k) + (2^k - 1)c$

$$T(n) = \begin{cases} c & n = 1 \\ 2T\left(\frac{n}{2}\right) + c & n > 1 \end{cases}$$

- So far for n > 2k we have
  - $T(n) = 2^kT(n/2^k) + (2^k - 1)c$
- What if k = lg n?
  - $T(n) = 2^{\lg n}\, T(n/2^{\lg n}) + (2^{\lg n} - 1)c$
    = n T(n/n) + (n - 1)c
    = n T(1) + (n-1)c
    = nc + (n-1)c = (2n - 1)c

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- T(n) =
  aT(n/b) + cn
  a(aT(n/b/b) + cn/b) + cn
  $a^2T(n/b^2) + cna/b + cn$
  $a^2T(n/b^2) + cn(a/b + 1)$
  $a^2(aT(n/b^2/b) + cn/b^2) + cn(a/b + 1)$
  $a^3T(n/b^3) + cn(a^2/b^2) + cn(a/b + 1)$
  $a^3T(n/b^3) + cn(a^2/b^2 + a/b + 1)$
  …
  $a^kT(n/b^k) + cn(a^{k-1}/b^{k-1} + a^{k-2}/b^{k-2} + \ldots + a^2/b^2 + a/b + 1)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So we have
  - $T(n) = a^kT(n/b^k) + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$
- For $k = \log_b n$
  - $n = b^k$
  - $T(n) = a^kT(1) + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$
    $= a^kc + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$
    $= ca^k + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$
    $= cna^k/b^k + cn(a^{k-1}/b^{k-1} + \ldots + a^2/b^2 + a/b + 1)$
    $= cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if a = b?
  - T(n) = cn(k + 1)
    $= cn(\log_b n + 1)$
    $= \Theta(n \log_b n)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a < b$?

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a < b$?
  - Recall that $\Sigma(x^k + x^{k-1} + \ldots + x + 1) = (x^{k+1} - 1)/(x-1)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a < b$?
  - Recall that $\Sigma(x^k + x^{k-1} + \ldots + x + 1) = (x^{k+1} - 1)/(x-1)$
  - So:

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \frac{1 - (a/b)^{k+1}}{1 - (a/b)} < \frac{1}{1 - a/b}$$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a < b$?
  - Recall that $\Sigma(x^k + x^{k-1} + \ldots + x + 1) = (x^{k+1} - 1)/(x-1)$
  - So:

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \frac{1 - (a/b)^{k+1}}{1 - (a/b)} < \frac{1}{1 - a/b}$$

  - $T(n) = cn \cdot \Theta(1) = \Theta(n)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \Theta\left((a/b)^k\right)$$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \Theta\left((a/b)^k\right)$$

  - $T(n) = cn \cdot \Theta(a^k / b^k)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \Theta\left((a/b)^k\right)$$

  - $T(n) = cn \cdot \Theta(a^k / b^k)$

$= cn \cdot \Theta(a^{\log_b n} / b^{\log_b n}) = cn \cdot \Theta(a^{\log_b n} / n)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \Theta\left((a/b)^k\right)$$

  - $T(n) = cn \cdot \Theta(a^k / b^k)$

$= cn \cdot \Theta(a^{\log_b n} / b^{\log_b n}) = cn \cdot \Theta(a^{\log_b n} / n)$

recall logarithm fact: $a^{\log_b n} = n^{\log_b a}$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \Theta\left((a/b)^k\right)$$

  - $T(n) = cn \cdot \Theta(a^k / b^k)$

$= cn \cdot \Theta(a^{\log_b n} / b^{\log_b n}) = cn \cdot \Theta(a^{\log_b n} / n)$

recall logarithm fact: $a^{\log_b n} = n^{\log_b a}$

$= cn \cdot \Theta(n^{\log_b a} / n) = \Theta(cn \cdot n^{\log_b a} / n)$

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So with $k = \log_b n$
  - $T(n) = cn(a^k/b^k + \ldots + a^2/b^2 + a/b + 1)$
- What if $a > b$?

$$\frac{a^k}{b^k} + \frac{a^{k-1}}{b^{k-1}} + \cdots + \frac{a}{b} + 1 = \frac{(a/b)^{k+1} - 1}{(a/b) - 1} = \Theta\left((a/b)^k\right)$$

  - $T(n) = cn \cdot \Theta(a^k / b^k)$

$= cn \cdot \Theta(a^{\log_b n} / b^{\log_b n}) = cn \cdot \Theta(a^{\log_b n} / n)$

recall logarithm fact: $a^{\log_b n} = n^{\log_b a}$

$= cn \cdot \Theta(n^{\log_b a} / n) = \Theta(cn \cdot n^{\log_b a} / n)$

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

$$T(n) = \begin{cases} c & n = 1 \\ aT\left(\frac{n}{b}\right) + cn & n > 1 \end{cases}$$

- So…

$$T(n) = \begin{cases} \Theta(n) & a < b \\ \Theta(n \log_b n) & a = b \\ \Theta\left(n^{\log_b a}\right) & a > b \end{cases}$$

主方式(The Master Theorem)

- Given: a *divide and conquer* algorithm (分治算法)
  - An algorithm that divides the problem of size *n* into *a* subproblems, each of size *n*/*b*
  (将规模为*n*的问题分解为*a*个规模为*n*/*b*的问题求解)
  - Let the cost of each stage (i.e., the work to divide the problem + combine solved subproblems) be described by the function *f*(n)
  (子问题求解和合并代价)
- Then, the Master Theorem gives us a cookbook for the algorithm's running time:(公式化求解开销函数)

The Master Theorem

- if T(n) = aT(n/b) + f(n) then

$$T(n) = \begin{cases} \Theta\left(n^{\log_b a}\right) & f(n) = O\left(n^{\log_b a - \varepsilon}\right) \\ \Theta\left(n^{\log_b a} \log n\right) & f(n) = \Theta\left(n^{\log_b a}\right) \\ \Theta\left(f(n)\right) & f(n) = \Omega\left(n^{\log_b a + \varepsilon}\right) \text{ AND } \\ & af(n/b) < cf(n) \text{ for large } n \end{cases} \quad \begin{matrix} \varepsilon > 0 \\ c < 1 \end{matrix}$$

Using The Master Method

- T(n) = 9T(n/3) + n
  - a=9, b=3, f(n) = n(对照主方式定理)
  - $n^{\log_b a} = n^{\log_3 9} = \Theta(n^2)$
  - Since $f(n) = O(n^{\log_3 9 - \varepsilon})$, where ε=1, case 1 applies:
  $$T(n) = \Theta\left(n^{\log_b a}\right) \text{when } f(n) = O\left(n^{\log_b a - \varepsilon}\right)$$
  - Thus the solution is $T(n) = \Theta(n^2)$

小结

- ***Substitution method***(置换法)
  - Guess the solution
  - Verify it using induction (substitution)
- ***Recursion tree method***(递归树)
  - Use a tree to visualize how a recurrence unfolds(可视展开)
  - Calculate cost of every level and sum these costs(逐层求和)
- ***Master method***:(主方式)
  - Provides a cook-book recipe for a restricted, but common class of recurrences(公式化)