Substitution Method

Kuan-Yu Chen (陳冠宇)

2019/03/12 @ TR-310-1, NTUST

Review

$$f(n) = \Theta(g(n)) \text{ and } g(n) = \Theta(h(n)) \text{ imply } f(n) = \Theta(h(n))$$

$$f(n) = O(g(n)) \text{ and } g(n) = O(h(n)) \text{ imply } f(n) = O(h(n))$$

$$f(n) = \Omega(g(n)) \text{ and } g(n) = \Omega(h(n)) \text{ imply } f(n) = \Omega(h(n))$$

$$f(n) = o(g(n)) \text{ and } g(n) = o(h(n)) \text{ imply } f(n) = o(h(n))$$

$$f(n) = \omega(g(n)) \text{ and } g(n) = \omega(h(n)) \text{ imply } f(n) = \omega(h(n))$$

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

$$f(n) = O(f(n))$$

$$f(n) = \Omega(f(n))$$

$$f(n) = \Theta(g(n)) \text{ if and only if } g(n) = \Theta(f(n))$$

$$f(n) = O(g(n)) \text{ if and only if } g(n) = \Omega(f(n))$$

$$f(n) = o(g(n)) \text{ if and only if } g(n) = \omega(f(n))$$

Introduction

- Recall that in divide-and-conquer, we solve a problem recursively, applying three steps at each level of the recursion
  - ***Divide*** the problem into a number of subproblems that are smaller instances of the same problem
    - When the subproblems are large enough to solve recursively, we call that the ***recursive case***
    - Once the subproblems become small enough that we no longer recurse, we say that the recursion "**bottoms out**" and that we have gotten down to the ***base case***
  - ***Conquer*** the subproblems by solving them recursively
    - If the subproblem sizes are small enough, however, just solve the subproblems in a straightforward manner
  - ***Combine*** the solutions to the subproblems into the solution for the original problem

Recurrences

- A ***recurrence*** is an equation or inequality that describes a function in terms of its value on smaller inputs

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

- There are three methods for obtaining asymptotic "Θ" or "O" bounds on the solution
  - Substitution method
  - Recursion-tree method
  - Master method

Substitution Method.

- The ***substitution method*** for solving recurrences comprises two steps:
  1. Guess the form of the solution
  2. Use mathematical induction to find the constants and show that the solution works

- For example
  – Let us determine an upper bound on the recurrence

$$T(n) = 2 \times T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$$

  - We guess that the solution is

$$T(n) = O(n \log_2 n)$$

  - The substitution method requires us to prove that $T(n) \leq cn \log_2 n$ for an appropriate choice of the constant $c > 0$

$T(n) = O(n \log_2 n)$

Substitution Method..

- Substituting into the recurrence

$$
\begin{aligned}
T(n) &= 2 \times T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \\
&\leq 2 \times \left(c \left\lfloor \frac{n}{2} \right\rfloor \log_2\left(\left\lfloor \frac{n}{2} \right\rfloor\right)\right) + n \\
&\leq 2 \times \left(c \frac{n}{2} \log_2\left(\frac{n}{2}\right)\right) + n = cn \log_2\left(\frac{n}{2}\right) + n \\
&= cn \log_2 n - cn \log_2 2 + n \\
&= cn \log_2 n - cn + n
\end{aligned}
$$

- If $c \geq 1$

$$T(n) \leq cn \log_2 n - cn + n \leq cn \log_2 n$$

Making a Good Guess

- Unfortunately, there is no general way to guess the correct solutions to recurrences
  - Guessing a solution takes experience and, occasionally, creativity

- A way to make a good guess is to prove **loose** upper and lower bounds on the recurrence and then reduce the range of uncertainty
  - Take $T(n) = 2 \times T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$ for example
    - Lower bound

$$T(n) = \Omega(n)$$

    - Upper bound

$$T(n) = O(n^2)$$

Subtleties

- Sometimes you might correctly guess an asymptotic bound on the solution of a recurrence, but somehow the math fails to work out in the induction
  - Take $T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 1$ for example
    - Guess the solution is $T(n) = O(n)$
    - We have to show that $T(n) \leq cn$ for a constant c
    - Substituting our guess in the recurrence

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 1 \leq c\left\lfloor \frac{n}{2} \right\rfloor + c\left\lceil \frac{n}{2} \right\rceil + 1 = cn + 1 \nleq cn$$

- Let guess $T(n) \leq cn - d$, where $d \geq 0$

$$T(n) = T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + T\left(\left\lceil \frac{n}{2} \right\rceil\right) + 1 \leq \left(c\left\lfloor \frac{n}{2} \right\rfloor - d\right) + \left(c\left\lceil \frac{n}{2} \right\rceil - d\right) + 1 = cn - 2d + 1 \leq cn - \mathrm{d}$$

- The inequality is true if $d \geq 1$, that is $T(n) = O(n)$
- Consider a **stronger** inductive hypothesis

Pitfalls

- It is easy to err in the use of asymptotic notation
  - Take $T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$ for example
  - Guess the solution is $T(n) = O(n)$
  - We have to show that $T(n) \le cn$

$$T(n) = 2T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n \le 2\left(c\left\lfloor \frac{n}{2} \right\rfloor\right) + n \le cn + n = (c+1)n = O(n)$$

- The **error** is that we have not proved the ***exact form*** of the inductive hypothesis $T(n) \le cn$

Changing Variables

- Sometimes, a little algebraic manipulation can make an unknown recurrence similar to one you have seen before
  - Take $T(n) = 2T\left(\lfloor\sqrt{n}\rfloor\right) + \log_2 n$ for example
    - It looks difficult!
  - Let $m = \log_2 n \Rightarrow 2^m = n,\ 2^{\frac{m}{2}} = (2^m)^{\frac{1}{2}} = \sqrt{2^m} = \sqrt{n}$
  - Changing variables: $T(2^m) = 2T\left(\left\lfloor 2^{\frac{m}{2}} \right\rfloor\right) + m$
  - Moreover, let $S(m) = T(2^m) \Rightarrow S(m) = 2S\left(\left\lfloor \frac{m}{2} \right\rfloor\right) + m$
  - Since we know the solution for $T(n) = 2 \times T\left(\left\lfloor \frac{n}{2} \right\rfloor\right) + n$ is $O(n \log_2 n)$
  - Thus, the solution for $S(m) = 2S\left(\left\lfloor \frac{m}{2} \right\rfloor\right) + m$ is $O(m \log_2 m)$
  - Changing back: $T(n) = T(2^m) = S(m) = O(m \log_2 m) = O(\log_2 n \log_2 \log_2 n)$

Questions?

kychen@mail.ntust.edu.tw