CS583 Lecture 02

Jyh-Ming Lien

some materials here are based on Prof. Shehu, and Prof. Wang's past lecture notes

Theoretical analysis

- Normally is written as a function, ex:
  $T(n) = an^b + \cdots + cn + d$

- But, there are problems in this representation, namely
  - machine dependent
  -
  -

Order of Growth

- Theoretical analysis focuses on ``order of growth" of an algorithm

Given that $T(n) = \frac{n(n-1)}{2}$, How much time an algorithm will take if the input size n doubled?

- Some common order of growth

$$n, n^2, n^3, n^d, \log n, \log^* n, \log\log n, n\log n, n!, 2^n, 3^n, n^n, \sqrt{n}$$

Asymptotic Notation

- Big $O, \Omega . \Theta$
- upper, lower, tight bound (when input is sufficiently large and remain true when input is infinitely large)
- defines a set of ***similar*** functions

Asymptotic Notation

- Asymptotic notation has been developed to provide a tool for studying order of growth
  - $O(g(n))$: a set of functions with the same or smaller order of growth as $g(n)$
    - $2n^2 - 5n + 1 \in O(n^2)$
    - $2^n + n^{100} - 2 \in O(n!)$
    - $2n + 6 \notin O(\log n)$
  - $\Omega(g(n))$: a set of functions with the same or larger order of growth as $g(n)$
    - $2n^2 - 5n + 1 \in \Omega(n^2)$
    - $2^n + n^{100} - 2 \notin \Omega(n!)$
    - $2n + 6 \in \Omega(\log n)$
  - $\Theta(g(n))$: a set of functions with the same order of growth as $g(n)$
    - $2n^2 - 5n + 1 \in \Theta(n^2)$
    - $2^n + n^{100} - 2 \notin \Theta(n!)$
    - $2n + 6 \notin \Theta(\log n)$

Big O

- **Definition**: $f(n)$ is in $O(g(n))$ if "order of growth of $f(n)$" $\leq$ "order of growth of $g(n)$" (within constant multiple)
  - there exist positive constant c and non-negative integer n_0 such that $f(n) \leq cg(n)$ for every $n \geq n_0$
- **Examples**:
  - $10n \in O(n^2)$
    - why?
  - $5n + 20 \in O(n)$
    - why?
  - $2n + 6 \notin O(\log n)$
    - why?

Big Ω

- **Definition**: $f(n)$ is in $O(g(n))$ if "order of growth of $f(n)$" $\leq$ "order of growth of $g(n)$" (within constant multiple)
  - there exist positive constant c and non-negative integer n_0 such that $f(n) \leq cg(n)$ for every $n \geq n_0$
- **Examples**:
  - $10n \in O(n^2)$
    - why?
  - $5n + 20 \in O(n)$
    - why?
  - $2n + 6 \notin O(\log n)$
    - why?

Big Θ

- **Definition**: $f(n)$ is in $\Theta(g(n))$ if $f(n)$ is bounded above and below by $g(n)$ (within constant multiple)
  - there exist positive constant c_1 and c_2 and non-negative integer n_0 such that $c_1 g(n) \leq f(n) \leq c_2 g(n)$ for every $n \geq n_0$
- **Examples**:
  - $\frac{1}{2}n(n-1) \in \Theta(n^2)$
    - why?
  - $2n - 51 \in \Theta(n)$
    - why?

Comparing OOG

- Verify the notation by compare the order of growth (oog)

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \begin{cases} 0 & t(n) \text{ has a smaller order of growth than } g(n) \\ c > 0 & t(n) \text{ has the same order of growth as } g(n) \\ \infty & t(n) \text{ has a larger order of growth than } g(n) \end{cases}$$

- useful tools for computing limits
  - L'Hôpital's rule

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{f'(n)}{g'(n)}$$

  - Stirling's formula

$$n! \approx \sqrt{2\pi n} \left(\frac{n}{e}\right)^n$$

Bounding Functions

- non-recursive algorithms
  - set up a sum for the number of times the basic operation is executed
  - simplify the sum and determine the order of growth (using asymptotic notation)

1. $\sum_{1=1}^{n} 1 = 1 + 1 + \cdots + 1 = n \in \Theta(n)$
2. $\sum_{1=1}^{n} i = 1 + 2 + \cdots + n = \frac{n(n+1)}{2} \approx \frac{n^2}{2} \in \Theta(n^2)$
3. $\sum_{1=1}^{n} i^2 = 1 + 4 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6} \approx \frac{n^3}{3} \in \Theta(n^3)$
4. $\sum_{1=0}^{n} a^i = 1 + a^1 + \cdots + a^n = \frac{a^{n+1} - 1}{a - 1}, \forall a \neq 1, \in \Theta(a^n)$
5. $\sum a_i + b_i = \sum a_i + \sum b_i$
6. $\sum c a_i = c \sum a_i$
7. $\sum_{1=0}^{n} a_i = \sum_{1=0}^{m} a_i + \sum_{1=m+1}^{n} a_i$

Asymptotic Notation

- why do we need asymptotic notation?
  - to make our life harder and more complicated?

Bounding Recursions

- What is a Recurrence
  - A recurrence is an equation of inequality that describes a function in terms of its value on smaller inputs
  - Recurrences have boundary conditions

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

- Techniques for Bounding Recurrences
  - Expansion
  - Recursion-tree
  - Substitution
  - Master Theorem

Expansion

- Examples

$$T(n) = 2T(n/2) + cn$$

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

Substitution

- make a guess and prove it right
- guess that

  $T(n) = 2T(\frac{n}{2}) + n \in O(n + n \cdot lgn)$, where $T(1) = 1$.

Substitution

- we can also guess that
  $T(n) = 2T(\frac{n}{2}) + n \in O(n)$, where $T(1) = 1$.

Recursion Tree

- Recursion tree is good for make an initial guess of the bound
- Build a recursion tree for
  $T(n) = 2T(n/2) + cn$

Recursion Tree

- Build a recursion tree for
  $T(n) = T(n/4) + T(n/2) + n^2$

Master Theorem

- If $T(n) = aT(n/b) + \Theta(n^d)$

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

- examples

1. $T(n) = 4T(n/2) + n \Rightarrow T(n) =$

2. $T(n) = 4T(n/2) + n^2 \Rightarrow T(n) =$

3. $T(n) = 4T(n/2) + n^3 \Rightarrow T(n) =$

Master Theorem

$$T(n) = aT(n/b) + \Theta(n^d)$$

- Don't use the master theorem when
  -
  -

Probabilistic Analysis

- use of probability theory in the analysis of algorithms
- To perform a probabilistic analysis, we have to **make assumptions on the distribution** of inputs
- After such assumption, we compute an **expected running time** that is computed over the distribution of all possible inputs

Randomized Alg

- some examples of randomized algorithm
  -
  -

Insertion Sort

- Worst case
- Best case
- Average case?
  - not (worst+best)/2
  - assume: every permutation is equally likely (how many permutations in total?)
    - important consequence
    -
  - We show that $T(n) = \Theta(n^2)$

Average Case

- Let random variable k be the number of moves to the right during the intersection sort
- Let random variable k_i be the number of moves to the right when insert $A[1]$ into $A[1], ..., A[i-1]$
- Then, $E[k] = \sum_{i=1}^{i=n} E[k_i]$

what is E[k_i]?

Average Case

Summary

- know what asymptotic notation is
- know how to bound algorithms with and without recursion
- know how to analyze the average case