Week 2: Types of analysis of algorithms, Asymptotic notations

Agenda:

- Worst/Best/Avg case analysis
- InsertionSort example
- Loop invariant
- Asymptotic Notations

Textbook pages: 15-27, 41-57

Insertion sort pseudocode (recall)

InsertionSort(A) **sort $A[1..n]$ in place

for $j \leftarrow 2$ to n do
  $key \leftarrow A[j]$ **insert $A[j]$ into sorted sublist $A[1..j-1]$
  $i \leftarrow j-1$
  while ($i > 0$ and $A[i] > key$) do
    $A[i+1] \leftarrow A[i]$
    $i \leftarrow i-1$
  $A[i+1] \leftarrow key$

Analysis of insertion sort

InsertionSort(A)	*cost*	*times*
for $j \leftarrow 2$ to n do	c_1	n
$key \leftarrow A[j]$	c_2	$n-1$
$i \leftarrow j-1$	c_3	$n-1$
while ($i > 0$ and $A[i] > key$) do	c_4	$\sum_{j=2}^{n} t_j$
$A[i+1] \leftarrow A[i]$	c_5	$\sum_{j=2}^{n} (t_j - 1)$
$i \leftarrow i-1$	c_6	$\sum_{j=2}^{n} (t_j - 1)$
$A[i+1] \leftarrow key$	c_7	$n-1$

t_j — number of times the `while` loop test is executed for j.

$$T(n) = c_1 n + (c_2 + c_3 - c_5 - c_6 + c_7)(n-1) + (c_4 + c_5 + c_6) \sum_{j=2}^{n} t_j$$

Quick analysis of insertion sort

- Assumptions:
  - (Uniform cost) RAM
  - Key comparison (KC) happens: $i > 0$ and $A[i] > key$
    (minors: loop counter increment operation, copy)
  - RAM running time proportional to the number of KC
- Best case (BC)
  - What is the best case?
    already sorted
  - One KC for each j, and so $\sum_{j=2}^{n} 1 = n - 1$
- Worst case (WC)
  - What is the worst case?
    reverse sorted
  - j KC for fixed j, and so $\sum_{j=2}^{n} j = \frac{n(n+1)}{2} - 1$
- Average case (AC)

Quick analysis of insertion sort (AC)

- Average case: always ask "average over what input distribution?"

- Unless stated otherwise, assume each possible input equiprobable

  Uniform distribution

- Here, each of the __ possible inputs equiprobable (why?)

- Key observation: equiprobable inputs imply for each key, rank among keys so far is equiprobable

- *e.g.*, when $j = 4$, expected number of KC is $\frac{1+2+3+4}{4} = 2.5$

- Conclusion: expected # KC to insert key j is $\frac{\sum_{i=1}^{j} i}{j} = \frac{j+1}{2}$

- Conclusion: total expected number of KC is

$$\sum_{j=2}^{n} \frac{j+1}{2} = \frac{1}{2} \sum_{j=3}^{n+1} = \frac{1}{2} \left(\frac{(n+1)(n+2)}{2} - 3 \right) = \frac{n^2 + 3n - 4}{4}$$

- We will ignore the constant factors; we also only care about the dominant term, here it is n^2.

Correctness of insertion sort

Claim:

- At the start of each iteration of the `for` loop, the subarray $A[1..j-1]$ consists of the elements originally in $A[1..j-1]$ and in sorted order.

Proof of claim.

- initialization: $j = 2$
- maintenance: $j \to j + 1$
- termination: $j = n + 1$

Loop invariant vs. Mathematical induction

- Common points
  - initialization vs. base step
  - maintenance vs. inductive step
- Difference
  - termination vs. infinite

Correctness & Loop invariant

- Why correctness?
  - Always a good idea to verify correctness
  - Becoming more common in industry
  - This course: a simple introduction to correctness proofs
  - When loop is involved, use loop invariant (and induction)
  - When recursion is involved, use induction
- Loop invariant (LI)
  - Initialization: does LI hold 1^{st} time through?
  - Maintenance: if LI holds one time, does LI hold the next?
  - Termination #1: upon completion, LI implies correctness?
  - Termination #2: does loop terminate?
- Insert sort LI:

  At start of `for` loop, keys initially in $A[1..j-1]$ are in $A[1..j-1]$ and sorted.
  - Initialization: $A[1..1]$ is trivially sorted
  - Maintenance: none from $A[1..j]$ moves beyond j; sorted
  - Termination #1: upon completion, $j = n + 1$ and by LI $A[1..n]$ is sorted
  - Termination #2: `for` loop counter j increases by 1 at a time, and no change inside the loop

Sketch of more formal proof of Maintenance

- Assume LI holds when $j = k$ and so $A[1] \leq A[2] \leq \ldots \leq A[k-1]$

- The `for` loop body contains another `while` loop. Use another LI.

- LI2: let $A^*[1..n]$ denote the list at start of `while` loop. Then each time execution reaches start of `while` loop:
  - $A[1..i+1] = A^*[1..i+1]$
  - $A[i+2..j] = A^*[i+1..j-1]$

- Prove LI2 (exercise)

- Using LI2, prove LI

  Hint: when LI2 terminates, either $i = 0$ or $A[i] \leq key$ (the latter implies either $i = j-1$ or $A[i+1] > key$).

Asymptotic notation for Growth of Functions: Motivations

- Analysis of algorithms becomes analysis of functions:
  - *e.g.*,
    $f(n)$ denotes the WC running time of insertion sort
    $g(n)$ denotes the WC running time of merge sort
  - $f(n) = c_1 n^2 + c_2 n + c_3$
    $g(n) = c_4 n \log n$
  - Which algorithm is preferred (runs faster)?
- To simplify algorithm analysis, want function notation which indicates *rate of growth* (a.k.a., *order* of complexity)

$O(f(n))$ — read as "big O of $f(n)$"

roughly, The set of functions which, as n gets large, grow no faster than a constant times $f(n)$.

precisely, (or mathematically) The set of functions $\{h(n) : N \to R\}$ such that for each $h(n)$, there are constants $c_0 \in R^+$ and $n_0 \in N$ such that $h(n) \leq c_0 f(n)$ for all $n > n_0$.

examples: $h(n) = 3n^3 + 10n + 1000 \log n \in O(n^3)$

$h(n) = 3n^3 + 10n + 1000 \log n \in O(n^4)$

$$h(n) = \begin{cases} 5^n, & n \leq 10^{120} \\ n^2, & n > 10^{120} \end{cases} \in O(n^2)$$

Definitions:

- $O(f(n))$ is the set of functions $h(n)$ that
  - roughly, grow <u>no faster</u> than $f(n)$, namely
  - $\exists c_0, n_0$, such that $h(n) \leq c_0 f(n)$ for all $n \geq n_0$
- $\Omega(f(n))$ is the set of functions $h(n)$ that
  - roughly, grow <u>at least as fast as</u> $f(n)$, namely
  - $\exists c_0, n_0$, such that $h(n) \geq c_0 f(n)$ for all $n \geq n_0$
- $\Theta(f(n))$ is the set of functions $h(n)$ that
  - roughly, grow <u>at the same rate</u> as $f(n)$, namely
  - $\exists c_0, c_1, n_0$, such that $c_0 f(n) \leq h(n) \leq c_1 f(n)$ for all $n \geq n_0$
  - $\Theta(f(n)) = O(f(n)) \cap \Omega(f(n))$
- $o(f(n))$ is the set of functions $h(n)$ that
  - roughly, grow <u>slower</u> than $f(n)$, namely
  - $\lim_{n\to\infty} \frac{h(n)}{f(n)} = 0$
- $\omega(f(n))$ is the set of functions $h(n)$ that
  - roughly, grow <u>faster</u> than $f(n)$, namely
  - $\lim_{n\to\infty} \frac{h(n)}{f(n)} = \infty$
  - $h(n) \in \omega(f(n))$ if and only if $f(n) \in o(h(n))$

Warning:

- the textbook overloads "="
  - Textbook uses $g(n) = O(f(n))$
  - *Incorrect* !!!
    Because $O(f(n))$ is a *set* of functions.
  - Correct: $g(n) \in O(f(n))$
  - You should use the correct notations.

Examples: which of the following belongs to $O(n^3)$, $\Omega(n^3)$, $\Theta(n^3)$, $o(n^3)$, $\omega(n^3)$?

1. $f_1(n) = 19n$

2. $f_2(n) = 77n^2$

3. $f_3(n) = 6n^3 + n^2 \log n$

4. $f_4(n) = 11n^4$

Answers:

1. $f_1(n) = 19n$
2. $f_2(n) = 77n^2$
3. $f_3(n) = 6n^3 + n^2 \log n$
4. $f_4(n) = 11n^4$

- $f_1, f_2, f_3 \in O(n^3)$
  $f_1(n) \leq 19n^3$, for all $n \geq 0$ — $c_0 = 19$, $n_0 = 0$
  $f_2(n) \leq 77n^3$, for all $n \geq 0$ — $c_0 = 77$, $n_0 = 0$
  $f_3(n) \leq 6n^3 + n^2 \cdot n$, for all $n \geq 1$, since $\log n \leq n$
  if $f_4(n) \leq c_0 n^3$, then $n \leq \frac{c_0}{11}$ — no such n_0 exists

- $f_3, f_4 \in \Omega(n^3)$
  $f_3(n) \geq 6n^3$, for all $n \geq 1$, since $n^2 \log n \geq 0$
  $f_4(n) \geq 11n^3$, for all $n \geq 0$

- $f_3 \in \Theta(n^3)$
  why?

- $f_1, f_2 \in o(n^3)$
  $f_1(n)$: $\lim_{n\to\infty} \frac{19n}{n^3} = \lim_{n\to\infty} \frac{19}{n^2} = 0$
  $f_2(n)$: $\lim_{n\to\infty} \frac{77n^2}{n^3} = \lim_{n\to\infty} \frac{77}{n} = 0$
  $f_3(n)$: $\lim_{n\to\infty} \frac{6n^3+n^2 \log n}{n^3} = \lim_{n\to\infty} 6 + \frac{\log n}{n} = 6$
  $f_4(n)$: $\lim_{n\to\infty} \frac{11n^4}{n^3} = \lim_{n\to\infty} 11n = \infty$

- $f_4 \in \omega(n^3)$

logarithm review:

- Definition of $\log_b n$ $(b, n > 0)$: $b^{\log_b n} = n$
- $\log_b n$ as a function in n: increasing, one-to-one
- $\log_b 1 = 0$
- $\log_b x^p = p \log_b x$
- $\log_b (xy) = \log_b x + \log_b y$
- $x^{\log_b y} = y^{\log_b x}$
- $\log_b x = (\log_b c)(\log_c x)$

Some notes on logarithm:

- $\ln n = \log_e n$ (natural logarithm)
- $\lg n = \log_2 n$ (base 2, binary)
- $\Theta(\log_b n) = \Theta(\log_{\{\text{whatever positive}\}} n) = \Theta(\log n)$
- $\frac{d}{dx} \ln x = \frac{1}{x}$
- $(\log n)^k \in o(n^\epsilon)$, for any positives k and ϵ

Handy 'big O' tips:

- $h(n) \in O(f(n))$ if and only if $f(n) \in \Omega(h(n))$
- limit rules: $\lim_{n\to\infty} \frac{h(n)}{f(n)} = \ldots$
  - $\ldots\infty$, then $h \in \Omega(f), \omega(f)$
  - $\ldots 0 < k < \infty$, then $h \in \Theta(f)$
  - $\ldots 0$, then $h \in O(f), o(f)$
- L'Hôpital's rules: if $\lim_{n\to\infty} h(n) = \infty$, $\lim_{n\to\infty} f(n) = \infty$, and $h'(n)$, $f'(n)$ exist, then

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

  e.g., $\lim_{n\to\infty} \frac{\ln n}{n} = \lim_{n\to\infty} \frac{1}{n} = 0$
- Cannot always use L'Hôpital's rules. *e.g.*,
  - $h(n) = \begin{cases} 1, & \text{if } n \text{ even} \\ n^2, & \text{if } n \text{ odd} \end{cases}$
  - $\lim_{n\to\infty} \frac{h(n)}{n^2}$ does NOT exist
  - Still, we have $h(n) \in O(n^2)$, $h(n) \in \Omega(1)$, etc.
- $O(\cdot), \Omega(\cdot), \Theta(\cdot), o(\cdot), \omega(\cdot)$

  JUST useful asymptotic notations

Another useful formula: $n! \approx \sqrt{2\phi n}\left(\frac{n}{e}\right)^n$.

Example: The following functions are ordered in increasing order of growth (each is in big-Oh of next one). Those in the same group are in big-Theta of each other.

$$\{n^{1/\log n},\quad 1\},\quad \{\log\log n,\quad \ln\ln n\},\quad \sqrt{\log n},\quad \ln n,\quad \log^2 n,$$

$$2^{\sqrt{\log n}},\quad (\sqrt{2})^{\log n},\quad 2^{\log n},\quad \{n\log n,\quad \log(n!)\},\quad \{n^2,\quad 4^{\log n}\},\quad n^3,$$

$$(\log n)!,\quad \{(\log n)^{\log n},\quad n^{\log\log n}\},\quad \left(\frac{3}{2}\right)^n,\quad n.2^n,\quad e^n,\quad n!,\quad n!^2,\quad 2^{2^n}$$