Review I: Algorithm Analysis

Viet Tung Hoang

The slides are loosely based on those of Dr Kevin Wayne, Princeton University and Prof. Mary Wootters, Stanford University.
Running Time Matters

- Time:
  - Centuries
  - Years
  - Days
  - Hours
  - Minutes
  - Seconds

- Groves:
  - $n!$
  - $n^3$
  - $n^2$

- Time scale:
  - $10^{10}$ seconds
  - $10^{15}$ minutes
  - $10^5$ hours
  - $10^6$ days
  - $10^7$ centuries

- Practical:
  - $n$
  - $n \log(n)$

- Prohibitive:
  - $n^3$
Asymptotic Behavior

How to analyze running time?

- **Want:** a *simple* analysis, independent of programming language, architecture, etc

**Main idea:** focus on how running time scale with $n$ (input size)

Informally, only focus on the behavior when $n$ is **large enough**
An Illustration: Insertion Sort vs Merge Sort

A slick implementation of Insertion Sort: $n^2$ steps

A poor implementation of Merge Sort: $100n \log_2(n)$ steps

Merge Sort still wins in the long run
Big-Oh Notation

**Upper bounds.** $T(n) \in O(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that $T(n) \leq c \cdot f(n)$ for all $n \geq n_0$

**Example:**

$$T(n) = 32n^2 + 17n + 1$$

$$n_0 = 1 \quad c = 50$$

$$T(n) = O(n^2)$$
A Usage Example: Selection Sort

- Maintain a sorted left half and a right unsorted one

- Loop: Swap the min of right half with its first element, and extend the left half

\[ \begin{align*}
\text{Round 1:} & \quad 36 & 13 & 20 & 25 & 15 & 4 & 51 & 33 & 64 & 22 & 84 & 8 \\
\text{Round 2:} & \quad 4 & 13 & 20 & 25 & 15 & 36 & 51 & 33 & 64 & 22 & 84 & 8 \\
\text{...} & \quad 4 & 8 & 20 & 25 & 15 & 36 & 51 & 33 & 64 & 22 & 84 & 13
\end{align*} \]

**Sorted** | **Unsorted**

\[ O(n) \text{ time} \]

Total \( n \) rounds, and \( O(n^2) \) time
Big-Omega Notation

**Lower bounds.** $T(n) \in \Omega(f(n))$ if there exist constants $c > 0$ and $n_0 \geq 0$ such that $T(n) \geq c \cdot f(n)$ for all $n \geq n_0$

**Example:** $T(n) = 32n^3 + 17n + 1$

$n_0 = 1 \quad c = 32$

$T(n) = \Omega(n^2)$
A Usage Example: Comparison-based Sort

**Want:** Sort these coins based on their weights using a scale

**Rules:**
- Can only weigh two coins $A$ and $B$ at a time
- Can only tell whether $A$ is heavier than $B$, but not how much
A Usage Example: Comparison-based Sort

Comparison-based Sort:
-Determine order based on *pairwise comparison* of elements, not actual value

<table>
<thead>
<tr>
<th>Comparison-based</th>
<th>Not comparison-based</th>
</tr>
</thead>
<tbody>
<tr>
<td>Merge Sort, Quick Sort</td>
<td>Counting Sort</td>
</tr>
<tr>
<td>Heap Sort, Insertion Sort</td>
<td>Radix Sort</td>
</tr>
<tr>
<td>Selection Sort, Bubble Sort</td>
<td>Bucket Sort</td>
</tr>
<tr>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Need extra assumption on the elements

**Theorem:** Any comparison-based sort algorithm must use $\Omega(n \log_2(n))$ comparisons
Big-Theta Notation

**Tight bounds.** \( T(n) \in \Theta(f(n)) \) if there exist constants \( c_1, c_2 > 0 \) and \( n_0 \geq 0 \) such that \( c_1 \cdot f(n) \leq T(n) \leq c_2 \cdot f(n) \) for all \( n \geq n_0 \)

**Example:** \( T(n) = 32n^2 + 17n + 1 \)

\[
\begin{align*}
n_0 &= 1 \\
\quad c_1 &= 32, \quad c_2 = 50
\end{align*}
\]

\( T(n) = \Theta(n^2) \)
A Usage Example: Selection Sort, Again

- Recall: Selection Sort takes $O(n^2)$ time for any input.
- Selection Sort takes $\Omega(n^2)$ time for finding the min elements alone.

Round 1:  

| 36 | 33 | 40 | 15 | 22 | 9 | 7 | 2 | 1 |

$\geq n$ steps

Round 2:  

| 1 | 33 | 40 | 15 | 22 | 9 | 7 | 2 | 36 |

$\geq n - 1$ steps

...  

| 1 | 2 | 40 | 15 | 22 | 9 | 7 | 33 | 36 |

Running time $\geq n + (n - 1) + \cdots + 1 = \frac{n(n + 1)}{2} = \Omega(n^2)$

Conclusion: Selection Sort takes $\Theta(n^2)$ time
Asymptotic Bounds for Common Functions

Polynomials:

\[ f(n) = a_0 + a_1 n + a_2 n^2 + \cdots + a_d n^d \text{ with } a_d > 0 \]

\[ f(n) = \Theta(n^d) \]

Rule: Pick the term of highest order

Log:

\[ \log_a(n) \in \Theta(\log_b(n)) \text{ for any constants } a, b > 0 \]

Write \( O(\log(n)) \) without specifying the base
Asymptotic Bounds for Common Functions

**Exponential**

\[ a^n \text{ with } a > 1 \]

\[ a^n \in \Omega(n^c) \]

**Polynomial**

\[ n^c \text{ with } c > 0 \]

**Log**

\[ \log(n) \]

\[ \log(n) \in O(n^c) \]