Model of Computation and Runtime Analysis
Model of Computation
Model of Computation

Specifies

- Set of operations
- Cost of operations (not necessarily time)

Examples

- Turing Machine
- Random Access Machine (RAM)
- PRAM
- Map Reduce(?)
Random Access Machine

Word
- Group of constant number of bits (e.g. byte)
- $\geq \log(\text{input size})$
- Usually integers or floats

Memory
- Big array of words
- Access by address

Operations
- Read and write a word from or into memory
- Arithmetic $+, -, *, /, \mod, [ ]$
- Logic (can be bitwise) $\land, \lor, \xor, \neg$
- Comparison based decisions

Each operation cost 1 unit of time.
Asymptotic Complexity
Asymptotic Complexity

Goal

▶ Determine runtime of an algorithm.

Depends on

▶ Input
▶ Hardware
▶ Programming language, compiler, and runtime environment

Solution

▶ Asymptotic Complexity
▶ How does the runtime behave based on the input size $n$?
Asymptotic Complexity

Hardware
- Raspberry Pi 2B 0.9 GHz
- Nexus 5 2.3 GHz
- Intel i7 4.0 GHz
- Same for other components (e.g. memory)

Runtime Environment
- Machine code (e.g. C++)
- Managed code (e.g. C# / Java)
- Interpreted code (e.g. Python)
- Virtual Machines (e.g. VirtualBox)

Conclusion
- Ignore constant factors.
Consider two algorithms

- \( T_1(n) = n^2 + 5n + 5 \)
- \( T_2(n) = n^2 \)

<table>
<thead>
<tr>
<th>( n )</th>
<th>4</th>
<th>16</th>
<th>64</th>
<th>256</th>
<th>1024</th>
<th>4096</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_1(n) )</td>
<td>41</td>
<td>341</td>
<td>4,421</td>
<td>66,821</td>
<td>1,053,701</td>
<td>16,797,701</td>
</tr>
<tr>
<td>( T_2(n) )</td>
<td>16</td>
<td>256</td>
<td>4,096</td>
<td>65,536</td>
<td>1,048,576</td>
<td>16,777,216</td>
</tr>
<tr>
<td>( T_1/T_2 )</td>
<td>2.5625</td>
<td>1.332</td>
<td>1.0793</td>
<td>1.0196</td>
<td>1.0049</td>
<td>1.0012</td>
</tr>
</tbody>
</table>

Conclusion

- Only keep strongest part. \( (n^2 \text{ in this case}) \)
Consider two algorithms and two computers

- Fast computer and slow algorithm
  10^7 operations per second  \( T_1(n) = n^2 \)

- Slow computer and fast algorithm
  10^4 operations per second  \( T_2(n) = n \lceil \log_2 n \rceil \)

- Input size: 10^6

**Runtime**

- \( T_1 = \frac{(10^6)^2}{10^7} \) s = 10^5 s \approx 27.8 \text{ h} \)

- \( T_2 = \frac{10^6 \lceil \log_2 10^6 \rceil}{10^4} \) s = 2,000 s \approx 33.3 \text{ min} \)

**Conclusion**

- First lower complexity, then constant factors.
Big-O Notation

Based on complexity, $3n^2 - \log_2 n$, and $n^2 + 5n + 5$ are the same as $n^2$.

How do we write this?

Big-O Notation

- $\mathcal{O}(g) = \{ f: \mathbb{N} \to \mathbb{N} | \exists c > 0 \exists n_0 > 0 \forall n \geq n_0: f(n) \leq c \cdot g(n) \}$
- $f \in \mathcal{O}(g)$ means $g$ is an upper bound for $f$.
- $\mathcal{O}(3n^2 - \log n) = \mathcal{O}(n^2 + 5n + 5) = \mathcal{O}(n^2)$

If an algorithm has runtime $n^2 + 5n + 5$, we say it runs in $\mathcal{O}(n^2)$ time.

Note that $\mathcal{O}(n) \subset \mathcal{O}(n \log n) \subset \mathcal{O}(n^2)$
### Complexity Examples

<table>
<thead>
<tr>
<th>Complexity</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>$O(1)$</td>
<td>constant</td>
</tr>
<tr>
<td>$O(\log n)$</td>
<td>logarithmic</td>
</tr>
<tr>
<td>$O(n)$</td>
<td>linear</td>
</tr>
<tr>
<td>$O(n \log n)$</td>
<td>counting, linear search, DFS/BFS</td>
</tr>
<tr>
<td>$O(n^2)$</td>
<td>quadratic</td>
</tr>
<tr>
<td>$O(2^{\log^c n})$</td>
<td>quasi polynomial</td>
</tr>
<tr>
<td>$O(2^n)$</td>
<td>exponential</td>
</tr>
<tr>
<td>$O(n!)$</td>
<td>checking all permutations</td>
</tr>
</tbody>
</table>

† Preliminary result, not peer reviewed yet.
$f \in \mathcal{O}(g)$: $g$ is an upper bound for $f$.

- $\exists c > 0 \exists n_0 > 0 \forall n \geq n_0 : f(n) \leq c \cdot g(n)$
Big-O Notation — $f \in \Omega(g)$

$f \in \Omega(g)$: $g$ is a lower bound for $f$.

- $\exists c > 0 \exists n_0 > 0 \forall n \geq n_0: f(n) \geq c \cdot g(n)$
- $f \in \Omega(g) \iff g \in \mathcal{O}(f)$
Big-O Notation — $f \in \Theta(g)$

$f \in \Theta(g)$

- $\exists c_1, c_2 > 0 \exists n_0 > 0 \forall n \geq n_0 : c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)$
- $\Theta(g) = \mathcal{O}(g) \cap \Omega(g)$
Big-O Notation — $f \in o(g)$

$f \in o(g)$: $f$ is dominated by $g$.

- $\forall c > 0 \exists n_0 > 0 \ \forall n \geq n_0 : f(n) \leq c \cdot g(n)$
  (This includes $c \leq 1$.)
Big-O Notation

\(f \in \mathcal{O}(g)\): \(g\) is an upper bound for \(f\).
\>
\(\exists c > 0 \ \exists n_0 > 0 \ \forall n \geq n_0 : f(n) \leq c \cdot g(n)\)

\(f \in \Omega(g)\): \(g\) is a lower bound for \(f\).
\>
\(\exists c > 0 \ \exists n_0 > 0 \ \forall n \geq n_0 : f(n) \geq c \cdot g(n)\)

\(f \in \Theta(g)\)
\>
\(\exists c_1, c_2 > 0 \ \exists n_0 > 0 \ \forall n \geq n_0 : c_1 \cdot g(n) \leq f(n) \leq c_2 \cdot g(n)\)

\(\Theta(g) = \mathcal{O}(g) \cap \Omega(g)\)

\(f \in o(g)\): \(f\) is dominated by \(g\).
\>
\(\forall c > 0 \ \exists n_0 > 0 \ \forall n \geq n_0 : f(n) \leq c \cdot g(n)\)

(This includes \(c \leq 1\).)
True or False? Explain your answer.

a) \( f \in \mathcal{O}(g) \) implies \( g \in \mathcal{O}(f) \)

b) \( f + g \in \Theta(\min(f, g)) \)

c) \( f \in \mathcal{O}(g) \) implies \( \log f \in \mathcal{O}(\log g) \)

d) \( f \in \mathcal{O}(g) \) implies \( 2^f \in \mathcal{O}(2^g) \)

e) \( f \in \mathcal{O}(f^2) \)

f) \( f \in \mathcal{O}(g) \) implies \( g \in \Omega(f) \)

g) \( f(n) \in \Theta(f(n/2)) \)

h) \( g \in o(f) \) implies \( f + g \in \Theta(f) \)
Rank the following functions by order of growth. Partition your list into equivalence classes such that functions $f_i$ and $f_j$ are in the same class if and only if $f_i \in \Theta(f_j)$.

<table>
<thead>
<tr>
<th>Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^2n$</td>
</tr>
<tr>
<td>$n^{1/\log n}$</td>
</tr>
<tr>
<td>$\log \log n$</td>
</tr>
<tr>
<td>$n \cdot 2^n$</td>
</tr>
<tr>
<td>$\log n$</td>
</tr>
<tr>
<td>$n^{\log \log n}$</td>
</tr>
<tr>
<td>$n^3$</td>
</tr>
<tr>
<td>$1$</td>
</tr>
<tr>
<td>$2^{\log n}$</td>
</tr>
<tr>
<td>$(\log n)^{\log n}$</td>
</tr>
</tbody>
</table>
Joe claims he can prove that $2^n \in O(1)$. His proof goes by induction on $n$.

**Base case**
- $2^1 = 2$, i.e., $2^1 \in O(1)$.

**Inductive step**
- Assume now that $2^{n-1} \in O(1)$ (Inductive Hypothesis).
- $2^n = 2 \cdot 2^{n-1}$
- Because $2f(n) \in O(f(n))$, $2^n \in O(1)$.

What is wrong with Joe’s “proof”??
Runtime Analysis for Recurrences
Divide and Conquer

Idea

- Split problem into smaller sub-problems.
- Solve sub-problems recursively.
- Combine solutions of sub-problems to solve original problem.

Examples

- Binary Search
- Merge sort, Quicksort
- Matrix multiplication
- Drawing binary trees
Runtime of Divide and Conquer

General Formula

\[ T(n) = \begin{cases} 
O(1) & \text{if } n = 1 \\
 a \cdot T\left(\frac{n}{b}\right) + f(n) & \text{if } n > 1 
\end{cases} \]

For simplicity, we ignore the case \( n = 1 \).

Binary Search

\[ T(n) = T\left(\frac{n}{2}\right) + 1 \]

Merge sort

\[ T(n) = 2T\left(\frac{n}{2}\right) + n \]
Solving Recurrence

Substitution Method
- Guess a (upper or lower) bound
- Prove it using induction

Recursion Tree
- Convert recurrence to tree.
- Each node represents a function call.
- Add cost of each layer and of all layers.

Master Theorem
- General solution (for some cases)
Substitution Method

Example: \( T(n) = 2T\left(\frac{n}{2}\right) + n \)

Hypothesis: \( T(n) \in \mathcal{O}(n \log n) \), i.e. \( T(n) \leq c \, n \log n \)

\[
T(n) = 2T\left(\frac{n}{2}\right) + n \\
\leq 2c \left(\frac{n}{2}\right) \log\left(\frac{n}{2}\right) + n \\
= c \, n \log n - c \, n \log 2 + n \\
= c \, n \log n - n(c \log 2 - 1) \\
\leq c \, n \log n
\]

\[\square\]
Substitution Method

Example: \( T(n) = 4T\left(\frac{n}{2}\right) + n \)

Hypothesis: \( T(n) \in \mathcal{O}(n^2) \), i.e. \( T(n) \leq c n^2 \)

\[
T(n) = 4T\left(\frac{n}{2}\right) + n \\
\leq 4c \left(\frac{n^2}{4}\right) + n \\
= c n^2 + n
\]

Does not work.

General advise for induction: Make your hypothesis stronger.
Example: \( T(n) = 4T\left(\frac{n}{2}\right) + n \)

Hypothesis: \( T(n) \leq cn^2 - n \)

\[
T(n) = 4T\left(\frac{n}{2}\right) + n \\
\leq 4c\left(\frac{n^2}{4}\right) - 4\left(\frac{n}{2}\right) + n \\
= cn^2 - 2n + n \\
= cn^2 - n
\]
Example: $T(n) = 4T\left(\frac{n}{2}\right) + n^2$
Example: $T(n) = T\left(\frac{n}{2}\right) + T\left(\frac{n}{4}\right) + T\left(\frac{n}{8}\right) + n$

\[
\sum_{i=0}^{\log n} \left(\frac{7}{8}\right)^i n \leq 8n
\]

\[
\sum_{i=0}^{\infty} x^i = \frac{1}{1-x} \text{ for } 0 < x < 1
\]
Recursion Tree

Example: \( T(n) = 3T\left(\frac{n}{2}\right) + n \)

\[
\sum_{i=0}^{\log n} \left(\frac{3}{2}\right)^i n
\]
Example: $T(n) = 3T(n/2) + n$

We know, $\sum_{i=0}^{m} r^i = \frac{r^{m+1} - 1}{r - 1}$

Thus,

$$n \sum_{i=0}^{\log_2 n} \left( \frac{3}{2} \right)^i = n \frac{1.5 \cdot 1.5^{\log n} - 1}{1.5 - 1}$$

$$= 3n \cdot 1.5^{\log n} - 2n$$

$$= 3n \cdot (2^{\log 1.5})^{\log n} - 2n$$

$$= 3n \cdot (2^{\log n})^{\log 1.5} - 2n$$

$$\approx 3n \cdot n^{0.58} - 2n$$

$$= 3n^{1.58} - 2n$$

$$\in \Theta(n^{1.58})$$
Consider a recurrence in the form (with \( a \geq 1, \ b > 1 \))

\[
T(n) = a \cdot T\left(\frac{n}{b}\right) + f(n)
\]

(1) \( f(n) \in \mathcal{O}(n^{\log_b a - \varepsilon}) \Rightarrow T(n) \in \Theta(n^{\log_b a}) \)

(2) \( f(n) \in \Theta(n^{\log_b a}) \Rightarrow T(n) \in \Theta(n^{\log_b a \log n}) \)

(3) \( f(n) \in \Omega(n^{\log_b a + \varepsilon}) \Rightarrow T(n) \in \Theta(f(n)) \)

For (1) and (3), \( \varepsilon > 0 \).

For (3), \( 0 < c < 1 \) and \( af\left(\frac{n}{b}\right) \leq cf(n) \).
Master Theorem

\[ T(n) = 3 \cdot T(\frac{n}{2}) + n \]  
\[ (\text{from recursion tree: } \Theta(n^{1.58})) \]

- \(a = 3, \ b = 2\)
- \(\log_b a = \log_2 3 \approx 1.58\)
- \(f(n) = n, \ f(n) \in \Theta(n^{\log_b a - \varepsilon})\)  
(Case 1)
- \(T(n) \in \Theta(n^{\log_b a}) = \Theta(n^{1.58})\)

\[ T(n) = 4 \cdot T(\frac{n}{2}) + n^2 \]  
\[ (\text{from recursion tree: } \Theta(n^2 \log n)) \]

- \(a = 4, \ b = 2\)
- \(\log_b a = \log_2 4 = 2\)
- \(f(n) = n^2, \ f(n) \in \Theta(n^{\log_b a})\)  
(Case 2)
- \(T(n) \in \Theta(n^{\log_b a \log n}) = \Theta(n^2 \log n)\)
Master Theorem

\[ T(n) = 2T\left(\frac{n}{2}\right) + n^2 \]

- \( a = 2, \ b = 2 \)
- \( \log_b a = \log_2 2 = 1 \)
- \( f(n) = n^2, f(n) \in \Omega(n^{\log_b a + \varepsilon}) \) \hspace{1em} (Case 3)
- \( T(n) \in \Theta(f(n)) = \Theta(n^2) \)