Algorithm Analysis

- The amount of resources used by the algorithm
  - Space
  - Computational time
- Running time:
  - The number of primitive operations (steps) executed before termination
- Order of growth
  - The leading term of a formula
  - Expresses the behavior of a function toward infinity

Asymptotic Notations

- A way to describe behavior of functions in the limit
  - How we indicate running times of algorithms
  - Describe the running time of an algorithm as \( n \) grows to \( \infty \)

- \( O \) notation: asymptotic "less than": \( f(n) \leq g(n) \)
- \( \Omega \) notation: asymptotic "greater than": \( f(n) \geq g(n) \)
- \( \Theta \) notation: asymptotic "equality": \( f(n) = g(n) \)
Logarithms

- In algorithm analysis we often use the notation “log n” without specifying the base.

<table>
<thead>
<tr>
<th>Binary logarithm</th>
<th>Natural logarithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\log_2 n$</td>
<td>$\log_n n$</td>
</tr>
<tr>
<td>$\lg n$</td>
<td>$\ln n$</td>
</tr>
<tr>
<td>$\lg^k n = (\lg n)^k$</td>
<td>$\log x = y \log x$</td>
</tr>
<tr>
<td>$\lg \lg n$ = $\lg(\lg n)$</td>
<td>$\log xy = \log x + \log y$</td>
</tr>
<tr>
<td></td>
<td>$\log x = \log x - \log y$</td>
</tr>
<tr>
<td></td>
<td>$\log_n x = \log_n b \log_b x$</td>
</tr>
<tr>
<td></td>
<td>$a^{\log_b x} = x^{\log_b a}$</td>
</tr>
</tbody>
</table>

Asymptotic notations

- **O-notation**

  $O(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } 0 \leq f(n) \leq cg(n) \text{ for all } n \geq n_0 \}$.

- Intuitively: $O(g(n))$ is the set of functions with a smaller or same order of growth as $g(n)$.

Examples

- $2n^2 = O(n^2)$: $2n^2 \leq cn^2 \Rightarrow 2 \leq c$ and $n_0 = 2$
- $n^2 = O(n^3)$: $n^2 \leq cn^3 \Rightarrow c \geq 1$ and $n_0 = 1$
- $1000n^2 + 1000n = O(n^2)$: $1000n^2 + 1000n \leq 1000n^2 + 1000n^2 = 2000n^2 \Rightarrow c = 2000$ and $n_0 = 1$
- $n = O(n^2)$: $n \leq cn^2 \Rightarrow c \geq 1$ and $n_0 = 1$
Asymptotic notations (cont.)

- **Ω**-notation

\[ \Omega(g(n)) = \{ f(n) : \text{there exist positive constants } c \text{ and } n_0 \text{ such that } \]
\[ 0 \leq cg(n) \leq f(n) \text{ for all } n \geq n_0 \} \]

- Intuitively: \( \Omega(g(n)) \) is the set of functions with a larger or same order of growth as \( g(n) \)

- Examples

  - \( 5n^2 = \Omega(n) \)
    
    \[ \exists c, n_0 \text{ such that: } 0 \leq cn \leq 5n^2 \Rightarrow cn \leq 5n^2 \Rightarrow c = 1 \text{ and } n_0 = 1 \]

  - \( 100n + 5 \neq \Omega(n^2) \)
    
    \[ \exists c, n_0 \text{ such that: } 0 \leq cn^2 \leq 100n + 5 \]
    
    \[ 100n + 5 \leq 100n + 5n (\forall n \geq 1) = 105n \]
    
    \[ cn^2 \leq n(cn - 105) \Rightarrow n(cn - 105) \leq 0 \]
    
    Since \( n \) is positive \( \Rightarrow cn - 105 \leq 0 \Rightarrow n \leq 105/c \)
    
    \[ \Rightarrow \text{contradiction: } n \text{ cannot be smaller than a constant} \]
    
    - \( n = \Omega(2n), n^3 = \Omega(n^2), n = \Omega(\log n) \)

Asymptotic notations (cont.)

- **Θ**-notation

\[ \Theta(g(n)) = \{ f(n) : \text{there exist positive constants } c_1, c_2, \text{ and } n_0 \text{ such that } \]
\[ 0 \leq c_1 g(n) \leq f(n) \leq c_2 g(n) \text{ for all } n \geq n_0 \} \]

- Intuitively: \( \Theta(g(n)) \) is the set of functions with the same order of growth as \( g(n) \)
Examples
- \( n^2/2 - n/2 = \Theta(n^2) \)
  - \( \frac{1}{2} n^2 - \frac{1}{2} n \leq \frac{1}{2} n^2 \forall n \geq 0 \Rightarrow c_2 = \frac{1}{2} \)
  - \( \frac{1}{2} n^2 - \frac{1}{2} n \geq \frac{1}{2} n^2 - \frac{1}{2} n \cdot \frac{1}{2} n (\forall n \geq 2) = \frac{1}{4} n^2 \Rightarrow c_1 = \frac{1}{4} \)
- \( n \neq \Theta(n^2) \): \( c_1 n^2 \leq n \leq c_2 n^2 \) only holds for: \( n \leq 1/c_1 \)
- \( 6n^2 \neq \Theta(n^2) \): \( c_1 n^2 \leq 6n^2 \leq c_2 n^2 \) only holds for: \( n \leq c_2 /6 \)
- \( n \neq \Theta(\log n) \): \( c_1 \log n \leq n \leq c_2 \log n \)
  \[ \Rightarrow c_2 \geq \frac{n}{\log n}, \forall n \Rightarrow n_0 - \text{impossible} \]

More on Asymptotic Notations
- There is no unique set of values for \( n_0 \) and \( c \) in proving the asymptotic bounds
- Prove that \( 100n + 5 = O(n^2) \)
  - \( 100n + 5 \leq 100n + n = 101n \leq 101n^2 \)
  for all \( n \geq 5 \)
  \( n_0 = 5 \) and \( c = 101 \) is a solution
  - \( 100n + 5 \leq 100n + 5n = 105n \leq 105n^2 \)
  for all \( n \geq 1 \)
  \( n_0 = 1 \) and \( c = 105 \) is also a solution
Must find SOME constants \( c \) and \( n_0 \) that satisfy the asymptotic notation relation

Comparisons of Functions
- **Theorem:**
  \( f(n) = \Theta(g(n)) \iff f = O(g(n)) \) and \( f = \Omega(g(n)) \)
- **Transitivity:**
  - \( f(n) = \Theta(g(n)) \) and \( g(n) = \Theta(h(n)) \Rightarrow f(n) = \Theta(h(n)) \)
  - Same for \( O \) and \( \Omega \)
- **Reflexivity:**
  - \( f(n) = \Theta(f(n)) \)
  - Same for \( O \) and \( \Omega \)
- **Symmetry:**
  - \( f(n) = \Theta(g(n)) \) if and only if \( g(n) = \Theta(f(n)) \)
- **Transpose symmetry:**
  - \( f(n) = O(g(n)) \) if and only if \( g(n) = \Omega(f(n)) \)
Asymptotic Notations in Equations

- **On the right-hand side**
  - $\Theta(n^2)$ stands for some anonymous function in $\Theta(n^2)$
  
  $2n^2 + 3n + 1 = 2n^2 + \Theta(n)$

  means:
  
  There exists a function $f(n) \in \Theta(n)$ such that
  
  $2n^2 + 3n + 1 = 2n^2 + f(n)$

- **On the left-hand side**

  $2n^2 + \Theta(n) = \Theta(n^2)$

  No matter how the anonymous function is chosen on the left-hand side, there is a way to choose the anonymous function on the right-hand side to make the equation valid.

Practical Complexity
Asymptotic Notations - Examples

- For each of the following pairs of functions, either $f(n)$ is $O(g(n))$, $f(n)$ is $\Omega(g(n))$, or $f(n) = \Theta(g(n))$. Determine which relationship holds.

  - $f(n) = \log n^2$; $g(n) = \log n + 5$  
    
  - $f(n) = n$; $g(n) = \log n^2$  
    
  - $f(n) = \log \log n$; $g(n) = \log n$  
    
  - $f(n) = n$; $g(n) = \log \log n$  
    
  - $f(n) = n \log n + n$; $g(n) = \log n$  
    
  - $f(n) = 10$; $g(n) = \log 10$  
    
  - $f(n) = 2^n$; $g(n) = 10n^2$  
    
  - $f(n) = 2^n$; $g(n) = 3^n$  

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

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

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

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

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

- $f(n) = O(g(n))$
Other Asymptotic Notations

- A function \( f(n) \) is \( o(g(n)) \) if for any positive constant \( c \), there exists \( n_0 \) such that
\[
f(n) < c \cdot g(n) \quad \forall \quad n \geq n_0
\]
or
\[
\lim_{n \to \infty} \frac{f(n)}{g(n)} = 0
\]

- A function \( f(n) \) is \( \omega(g(n)) \) if for any positive constant \( c \), there exists \( n_0 \) such that
\[
c \cdot g(n) < f(n) \quad \forall \quad n \geq n_0
\]
or
\[
\lim_{n \to \infty} \frac{f(n)}{g(n)} = \infty
\]

Intuitions

- Intuitively,
  - \( o() \) is like <
  - \( \omega() \) is like >
  - \( \Theta() \) is like =
  - \( \Omega() \) is like ≥
Typical Running Time Functions

- **Θ(1)** (constant running time):
  - Instructions are executed once or a few times
- **Θ(logN)** (logarithmic):
  - A big problem is solved by cutting the original problem in smaller sizes, by a constant fraction at each step
- **Θ(N)** (linear):
  - A small amount of processing is done on each input element
- **Θ(N logN)**:
  - A problem is solved by dividing it into smaller problems, solving them independently and combining the solution
- **Θ(N^2)** (quadratic):
  - Typical for algorithms that process all pairs of data items (double nested loops)
- **Θ(N^3)** (cubic):
  - Processing of triples of data (triple nested loops)
- **Θ(N^k)** (polynomial):
- **Θ(2^N)** (exponential):
  - Few exponential algorithms are appropriate for practical use

Some Simple Summation Formulas

- **Arithmetic series:**
  \[ \sum_{i=1}^{n} i = \frac{n(n+1)}{2} \]
- **Geometric series:**
  \[ \sum_{i=1}^{n} x^i = \frac{1-x^{n+1}}{1-x} \]
  - Special case: \(x < 1\):
    \[ \sum_{i=1}^{n} x^i = \frac{1}{1-x} \]
- **Harmonic series:**
  \[ \sum_{i=1}^{n} \frac{1}{i} = \ln n + \gamma \]
- **Other important formulas:**
  \[ \sum_{i=1}^{n} \log(k) = n \log n \]
  \[ \sum_{i=1}^{n} i^2 = 1^2 + 2^2 + \ldots + n^2 = \frac{n(n+1)(2n+1)}{6} \]
  \[ \sum_{i=1}^{n} i^3 = 1^3 + 2^3 + \ldots + n^3 = \left(\frac{n(n+1)}{2}\right)^2 \]