Determining precise formula for running-time $f(n)$ is difficult

- Simplify by using a big $O$ expression
  $O(g(n))$ is an upper bound on $f(n)$

**Properties of big-$O$**

1. **Simplicity**
   $$f(n) = O(g(n))$$
   $g(n)$ is a big $O$ bound on $f(n)$
   $g(n)$ is **simple** if
   - It is a single term **AND**
   - Coefficient is $1$

   **Example**
   $$\sum_{i=0}^{n-1} i = \frac{n^2}{2}$$

   Let $g(n) = c_1n^2 + c_2n + c_3$, where $c_1 = c_2 + c_3$, $c_2 = 2c_2$ and $c_3 = c_1 + c_2$

   Let $g(n) = c_1n^2 + c_2n + c_3$, $g_2(n) = c_1n^2$, $g_3(n) = n^2$

   Then $f(n) = O(a_i(n))$ for each $i = 1, 2, 3$
Note: \( g_1 \) & \( g_2 \) are not simple
\( g_2 \) is simple

So the running time is \( O(g_3(n)) \)
which is \( O(n^2) \)

2) Tightness

We want the "tightest" big-O bound we can prove. If \( f(n) = O(g(n)) \)
then we cannot find a func.
\( h(n) \) that grows at least as fast as \( f(n) \) but grows slower
than \( g(n) \)

\( g(n) \) is tight bound on \( f(n) \) if

1. \( f(n) = O(g(n)) \) AND
2. IF \( f(n) = O(h(n)) \) then it is also true that \( g(n) = O(h(n)) \)

ex 2) Running time for alg. in ex(1)
\( f(n) = C^3 n^2 + C'' n + C''' \)

We had \( g_3(n) = n^2 \)
Let \( g_4(n) = n^3 \)
CLAIM #1  \( g_3(n) \) is tight bounded on \( f(n) \)

CLAIM #2  \( g_4(n) \) is NOT a tight bound

Proof of Claim #1

\[ f(n) = O(g_3(n)) \]
Suppose \( f(n) = O(h(n)) \)

Then there are \( c \) and \( n_0 \) such that

\[ f(n) = c'n^2 + c''n + c''\leq c h(n) \]
for all \( n \geq n_0 \)

Then \( h(n) \geq \left( \frac{c'}{c} \right) n^2 \)

for all \( n \geq n_0 \)

But \( g_3(n) = n^2 \) then \( g_3(n) \leq \left( \frac{c}{c} \right) h(n) \)

for all \( n \geq n_0 \)

Thus \( g_3(n) = O(h(n)) \)

Proof of Claim #2

Pick \( h(n) = n^2 \)

we have \( f(n) \) is \( O(h(n)) \)

but \( n^2 \) is not \( O(h(n)) \)

**FACT:** Show \( f(n) = \Theta g(n) \) to prove \( g(n) \) is a tight bound on \( f(n) \)
Ex. In previous example we showed
\[ f(n) = n^8 + 7n^7 - 10n^5 - 2n^4 + 3n^2 - 14 \]
is \(\Theta(n^8)\)

This means \(n^8\) is a tight bound on \(f(n)\)

**Analyzing Running Time of a Program**

<table>
<thead>
<tr>
<th>Big-O</th>
<th>Informal Name</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>(n \log n)</td>
</tr>
<tr>
<td>(O(n^2))</td>
<td>Quadratic</td>
</tr>
<tr>
<td>(O(n^3))</td>
<td>Cubic</td>
</tr>
<tr>
<td>(O(n^k))</td>
<td>Polynomial</td>
</tr>
<tr>
<td>(O(2^n))</td>
<td>Exponential</td>
</tr>
</tbody>
</table>

**Algorithms with running times...**

\[ \{ \Theta(n) \text{ have linear complexity}, \quad \Theta(n^2) \text{ have quadratic}, \quad \Theta(n^k) \text{ have polynomial complexity} \]
Efficient usually means polynomial complexity.

1. Simple Statements $O(1)$
2. If - Statement
3. For - Statement
4. While - Statement
5. Do-While Statement