Running Time Analysis

Program, input size = n (ex: n numbers to sort).

Goals:
1. Express runtime of a program as a function of input size n:
   \[ T(n) = \text{runtime of a program in terms of } n. \]
   \[ f(n) = \text{some function of } n \text{ (ex. } n^2, n^3, \log n) \]

2. Establish relative order between \( T(n) \) and \( f(n) \).

How to compare 2 functions?
Can't just say one is greater than the other.

\[ f(n) = 1000n \quad g(n) = n^2 \]

\[ f(n) > g(n) \text{ for } n < 1000 \]
\[ f(n) < g(n) \text{ for } n > 1000 \]

So instead compare in terms of relative growth rates

growth rate - how fast a function grows asymptotically (i.e. as \( n \to \infty \)).

\[ \frac{f(n)}{g(n)} \]

Fastest growing.
Order of typical growth rates

\[ \begin{align*}
&\text{Constant} \\
&\log(n) \text{ logarithmic} \\
&\log^2(n) \text{ log-squared} \\
&n \text{ linear} \\
&n\log n \\
&n^2 \text{ quadratic} \\
&n^3 \text{ cubic} \\
&n^k \text{ polynomial} \\
&2^n \text{ exponential}
\end{align*} \]

\[ \Rightarrow \text{ "efficient"} \]

1. \( T(n) = O(f(n)) \) - \( T(n) \) grows at a rate \( \leq cf(n) \) for \( c > 0 \)

\( T(n) \) grows no faster than \( f(n) \):

\[ \begin{align*}
\text{ex: } T(n) &= 1 + 5n \\
\text{ex: } T(n) &= 1 + 5n^2
\end{align*} \]

First, which grows faster? \( T(n) = 1 + 5n^2 \)

So this corresponds to the slower program

How to find \( f(n) \):

1. Pick out fastest growing term in \( T(n) \)
2. Drop coefficients

\[ \begin{align*}
\text{ex: } T(n) &= 1 + 5n \\
\text{ex: } T(n) &= 1 + 5n^2 \\
\text{f(n)} &= n \\
\text{f(n)} &= n^2 \\
\text{\therefore } T(n) &= O(n) \\
\text{\therefore } T(n) &= O(n^2)
\end{align*} \]

\[ \text{\textbf{\textit{Notice: } } } T(n) \text{ grows no faster than } f(n) \]
3. \( T(n) = 5 + 20n + 3n^2 \)
\[ T(n) = O(n^2) \]

(2) + (3) grow at same rate.

4. \[ T(n) = \frac{n^2(n^2+1)}{2} = \frac{n^4+n^2}{2} \]
\[ f(n) = n^4 \]
\[ T(n) = O(n^4) \]

5. \( T(n) = 5 \)
\[ f(n) = 1 \]
\[ T(n) = O(1) \]

6. \( T(n) = n(5 + \log n) \)
\[ = 5n + n\log n \]
\[ f(n) = n\log n \]
\[ T(n) = O(n\log n) \]

Tightness: \( f(n) \) expressed in lowest correct order.

ex: \( T(n) = 1 + 100n^2 \) tightest \( f(n) \)?

\[ T(n) = O(n) \] - not correct
\[ T(n) = O(n^3) \] - correct but not tight
\[ T(n) = O(n^2) \] - and tight.
In these examples, we have a formula for $T(n)$. But how to find $T(n)$ of a program?

**General rules**

1. **for loop**:

   for (n iterations)  
   
    body  

   $T(n) = T(\text{body}) \times n$

2. **Nested for loop**:

   for (n iterations)  
   for (m iterations)  
   
    body  

   $T(n) = T(\text{body}) \times n \times m$
3. Consecutive Statements

\[
\text{statement 1} \quad \text{\{} \quad \text{a statement may be a for loop, while loop, print, call to a method, etc.}
\text{statement 2} \quad \text{\}}
\text{statement k}
\]

\[T(n) = \max (T(\text{statement 1}), T(\text{statement 2}), \ldots, T(\text{statement k}))\]

(ex. \(s_1 = n^2\), \(s_2 = n\), \(s_3 = 2^n\))

\[T(n) = 2^n\]

4. If/else:

\[
\text{if (condition)}
\]
\[
\text{<body 1>}
\]
\[
\text{else}
\]
\[
\text{<body 2>}
\]

\[T(n) = \max (T(\text{condition}), T(\text{<body 1>}), T(\text{<body 2>}))\]