Basics of Algorithm Analysis. (Asymptotic Notation)

What are *efficient* algorithms?
- First attempt: When implemented, the algorithm runs fast on real inputs.
  - platform? language? distribution of input.

Stable Matching: \( n = \# \) of men = \# of women.
  - \( N = 2n^2 \) \( \{ \) \( -2n \) preference lists.
  - \( n \) numbers in each list.
- What is the running time\(^*\) as a function of \( N \)?
  - Lucky: \( n \) proposals.
  - Unlucky: \( \approx n^2 \) proposals.

\* Worst-case running time.

Example: Algorithm A
- \# of operations: \( N + 1000 \)
- \( n = 5 \). A is slower. \( N = 10^6 \) then B is slower.

Asymptotic Order of Growth. (Big-O notation).

Definition 1
- Let \( T(n) \) be a function.
- Given a function \( f(n) \), we say \( T(n) \in O(f(n)) \).
- \( T(n) = O(f(n)) \).

Asymptotic upper bound)

iff
- \( \exists \) constants \( C \) and \( n_0 \) such that
- \( \forall n \geq n_0, \ T(n) \leq C \cdot f(n) \)

Example 1: \( 5n^2 + 3n + 1 = O(n^2) \).
Proof: \( n_0 = 1, \ C = 5 + 3 + 1 = 9 \), \( \leq 5n^2 + 3n^2 + n^2 = 9n^2 = Cn^2 \).
Example 2: $5n^2 + 3n - 1000 = \mathcal{O}(n^2)$?

True. $n_0 = 1$, $c = 8$.

Example 3: $n \log_2 n = \mathcal{O}(n)$? False!

We can show no $(n_0, c)$ would work.

Fix any $n_0$ and $c$.

Let $n = \max(n_0, 2^c) + 1 > n_0$.

$n > 2^c \Rightarrow n \cdot \log_2 n > n \cdot c \Rightarrow$ contradiction!

---

Reasons for using big $\mathcal{O}$:

\[
\begin{align*}
&\text{For } i = 1 \text{ to } n-1 \\
&\quad \text{for } j = i+1 \text{ to } n \\
&\quad \text{do something}
\end{align*}
\]

\[
T(n) = (n-1) + (n-2) + \ldots + 1 = \frac{n(n-1)}{2} = \mathcal{O}(n^2).
\]