With group, review notes on functions from last week, come up with at least one question about functions.

To describe a function (formally):
- $f: A \rightarrow B$ (define domain and codomain)
- Describe mapping. For example
  - $f(x) = \ldots$
  - $A \rightarrow B$
  - $B$

If clear from context, we will sometimes not explicitly state domain & codomain.
Intro to Algorithm Complexity

Important function: worst case time complexity of an algorithm in C.S.

\[ T : D \rightarrow \mathbb{N}, \text{ for } D \subseteq \mathbb{N} \]

\( \uparrow \) \# of operations performed by algorithm in worst case.

Input size

Unless parallel computing, this tells you the time the computer will take to run the algorithm. Just multiply by time to do 1 operation.

Linear Search

- **Input**: \( (a_1, a_2, \ldots, a_n) \), \( x \) \( \Delta \) Input size is \( n \)
- **Output**: \( j \) if \( a_j = x \), 0 otherwise

1. \( i = 1 \)
2. while \( (i \leq n \text{ and } x \neq a_i) \)
3. \( i = i + 1 \)
4. if \( i \leq n \):
   - return \( i \)
5. else:
   - return 0

Q: What is \( T(n) \) for linear search? (Hint: \( n \) is not correct)

Report by:
By group:
Issues:
- too fine-grained/detailed
  - different computers might do operations differently
  - when \( n \) gets large, don't care about \( 100000 \) vs \( 100001 \)
- too difficult to count every operation

**Big-O to the Rescue!**

A special notation to describe how functions grow.

**Def:** Let \( f, g : \mathbb{Z} \rightarrow \mathbb{R}^+ \) or \( f, g : \mathbb{Z} \rightarrow \mathbb{N} \),

Then \( f(x) \) is \( O(g(x)) \) if \( \exists \ k, c \in \mathbb{R} : \forall x \geq k, \)

\[
  f(x) \leq c g(x).
\]

"\( f \) of \( x \) is big-oh of \( g \) of \( x \)"
A function $f(n)$ is said to be $O(g(n))$ if there exists a positive constant $c$ and a non-negative integer $k$ such that $f(n) \leq cg(n)$ for all $n \geq k$. This is denoted as $f(n) = O(g(n))$.

**Example:**

$2n + 4$ is $O(n)$ if there exist $c > 0$ and non-negative integer $k$ such that $2n + 4 \leq cn \quad \forall n \geq k$.

- **Proof:**
  
  For $n \geq 1$, we have $4n \geq 2n + 4$. (Multiply both sides by 4). Thus for $n \geq 1$, $2n + 4 \leq 2n + 4n = 6n$, so $2n + 4 = O(n)$ with $k = 1$, $c = 6$.

  For $n \geq 4$, we have $2n + 4 \leq 2n + n = 3n$, so $2n + 4 = O(n)$ with $k = 4$, $c = 3$.

Infinitely many combos of $K, C$ work. To prove, you need to find **one** combo.

**General trick:** try to get $f(x)$ to look like $g(x)$ by turning bad terms into good terms. Above $4 \rightarrow 4n$ or $4 \rightarrow n$
Q: Prove other functions for linear search # of operations is \( O(n) \)

Starting to see why big-O is good for algorithm time complexity:

- Small differences in how you calculate don't matter
- Not too fine grained

However big-O is only upper bound:

ex: \( 7x + 1 \) is \( O(x^2) \)

pf: We have \( 7x + 1 \leq 7x + x \) for all \( x \geq 1 \)

Then \( 7x + x = 8x \leq x^2 \) for all \( x \geq 8 \)

Thus with \( K=8 \), \( C=1 \), \( 7x + 1 = O(x^2) \)
Q: Prove $10x^2$ is not $O(x)$. This means there do not exist constants $k, C$, such that $10x^2 < Cx$ for all $x \geq k$.

Pf: For contradiction, assume $k, C$ exist. Then for all $x \geq k$, we have

$$10x^2 < Cx$$

When $x > 0$, we have $x \leq \frac{C}{10}$. Thus, this inequality holds only when $0 < x \leq \frac{C}{10}$, which contradicts that it should hold for all $x \geq k$. 