Algorithms and Functions

Algorithm: a finite set of precise instructions for performing a computation or for solving a problem.

It is the black-box with three properties:

- **Definiteness**: each step is defined precisely
- **Finiteness**: end after a finite many steps
- **Generality**: deal with all problems of the desired form, not just for some particular input values.

Try to do the following job:

1. Find the maximal element in a finite sequence
   - Way 1: Compare two each time, and keep the current “leader”.
     Let \( \max := a_1 \). For \( i = 2 \) to \( n \), if \( \max < a_i \), then \( \max := a_i \)

2. Locate an element in a given list
   - Way 1: linear search, or sequential search – Compare the desired item with elements in the list one by one.
   - Way 2: Binary search: Order the list first (like a dictionary), then check the middle, go to first half, or the second half according to \( x < a_{\text{middle}} \), or \( x > a_{\text{middle}} \).
   **Important idea**: PUSH THINGS INTO A SIMILAR, BUT SMALLER PROBLEM!

3. Sort the given data in order
   - Way 1. Bubble sort: push the largest element to the end, then work with a shorter list.
     Algorithm: For \( i = 1 \) to \( n - 1 \), if \( a_i > a_{i+1} \), interchange \( a_i \) with \( a_{i+1} \). Then repeat for the remaining \( n - 1 \) numbers.
   - Way 2. Insertion Sort: Read the number and write them in order one-by-one.
     Algorithm: Begin with \( a_1 \). For \( j = 2 \) to \( n \), compare \( a_j \) with the current list, and insert it in the right position.
   - Way 3. Merge Sort: Split the list into two, sort each half, then merge.

Which way is better? How to compare two algorithms? — Compute how “expensive” they are, in terms of time or space spent. → **Complexity**.

Count the number of “major operations” used. It depends on how you write the algorithm.

For examples,
- **Find the max**: \( n - 1 \) comparisons.
- **Linear Search**: in worst case, \( n - 1 \) comparisons.
- **Binary search**: \( 1 + f(n/2) \) Later, we will see it is about \( \log_2 n \).
- **Bubble sort**: \( (n - 1) + (n - 2) + \cdots + 1 = n(n - 1)/2 \).
- **Insertion Sort**: Worst case \( 1 + 2 + \cdots + (n - 1) \)
- **Merge Sort**: \( 2f(n/2) + n \). Later, we will see it is about \( n \log(n) \)

Q: with two complexity functions, how to compare them?
Definition 1. We say that $f$ is of $O(g)$ if there are constant $C$ and $K$ such that

$$|f(x)| \leq C|g(x)|$$

whenever $x > K$.

Read: $f$ is big-O of $g$. $g$ dominates $f$, $f$ is dominated by $g$.

Variation of notations:

1. We say that $g$ is of $\Omega(g)$ if $f$ is $O(g)$. That is, if there are constant $C$ and $K$ such that

$$|f(x)| \leq C|g(x)|$$

whenever $x > K$.

2. We say that $f$ is $\Theta(g)$ if $f$ is $O(g)$ and $\Omega(g)$. That is, if there are constant $C_1$, $C_2$, and $K$ such that

$$C_2|g(x)| \leq |f(x)| \leq C_1|g(x)|$$

whenever $x > K$.

Examples:

1. $f(x) = x^2 + 2x + 4$, $g(x) = x^2$.
2. $f(x) = 4x^2 - 8x$, $g(x) = x^2/2$. 