# Lecture 1, Sep 11, 2023

## Runtime Complexity Analysis (Review)

**Definition**

Let $A$ be an algorithm and $t(x)$ be the number of steps taken by $A$ on input $x$; then the *worst-case time complexity* is

$$T(n) = \max_{\text{all input } x \text{ of size } n} t(x) = \max \{\, t(x) \mid x \text{ is of size } n \,\}$$

- "Size" is typically defined as the number of bits used to represent the input; e.g. number of elements in an array, number of nodes/edges in a graph, number of bits of an integer
- To define an upper bound for $T(n)$, we have to prove that for *every* input of size $n$, $A$ takes *at most* some number of steps
- To define a lower bound for $T(n)$, we only have to prove that for *some* input of $n$, $A$ takes *at least* some number of steps

**Definition**

$T(n)$ is $O(g(n))$ iff

$$\exists c > 0, \exists n_0 > 0, \text{ s.t. } \forall n \geq n_0, T(n) \leq c \cdot g(n)$$

In other words, for sufficiently large input and within a constant factor: for *every* input of size $n$, $A$ takes *at most* $c \cdot g(n)$ steps.

**Definition**

$T(n)$ is $\Omega(g(n))$ iff

$$\exists c > 0, \exists n_0 > 0, \text{ s.t. } \forall n \geq n_0, T(n) \geq c \cdot g(n)$$

In other words, for sufficiently large input and within a constant factor: for *some* input of size $n$, $A$ takes *at least* $c \cdot g(n)$ steps.

**Definition**

$T(n)$ is $\Theta(g(n))$ iff it is both $O(g(n))$ and $\Omega(g(n))$.

- The notions of $O, \Omega, \Theta$ allow us to ignore constant factors and restrict the analysis to sufficiently large input sizes