Computational complexity function

- **Important:** If we know that a model is deterministic for a DTM, then it is also deterministic for other deterministic models.

- The computational complexity function of a problem quantifies the amount of computational resources, such as time or space, required by an algorithm to solve the problem as a function of the size of the input.
- It provides a measure of the efficiency of the algorithm, typically expressed in terms of asymptotic notation like $O(n)$, $O(n^2)$, or $O(\log n)$.
- Definition of the asymptotic notation (*Big-Oh* notation):
    - We say that the function $f(k)$ is of order $g(k)$, which is denoted as $f(k) = O(g(k))$, if there exists a constant c such that:

$$|f(k)| \leq |g(k)|$$

    - For almost all values of k (i.e. only for a finite set of values k this can be false).
- The computational complexity function f of an algorithm solving a given problem Π assigns to each instance $I \in D_\Pi$ the maximum number of elementary steps (or units of time) of a digital machine required to solve this instance of the problem.
- In general we will only care about the two most important complexities: polynomial and exponential.
    - Polynomial: an algorithm is considered polynomial-time if its time complexity is of order n^k ($f(n) = O(n^k)$) where k can be any positive integer.
    - Pseudo-polynomial: an algorithm is considered pseudo-polynomial if it's polynomial in the numeric value of the input, rather than the size of the

input. An example of pseudo-polynomial algorithm is primality testing algorithm (algorithm for testing whether a number is prime), or counting sort which has the the complexity $O(n+k)$ where k is the value of the greatest integer in the array.

  - Exponential: an algorithm is considered exponential-time when its time complexity grows exponentially with the input size, i.e. $f(n) = O(2^n)$.

- Exponential-time algorithms are considered to be intractable for large input sizes.