Introduction to Algorithms
(Time Complexity)
- Efficiency of an Algorithm
- Constant and Logarithmic Time
- Linear & Quadratic Time
- Quasilinear Time
- Exponential Time
- Determining Complexity

Algorithm - A set of steps a program takes to finish a task.

Guidelines :-
- Clearly defined problem statement, input and output.
- The steps in the algorithm need to be in very specific order.
- The steps also need to be distinct
- The algorithm should produce a result
- The algorithm should complete in a finite amount of time.

Big O Notation - **"O(n)"** - Theoretical definition of the complexity of an algorithm as a function of the size.
- **O(n)** - A function of the size (it measures complexity as the input size grows)
- **O(n)** - upper bound (This means how algo performs in worst case scenario)

- Complexity is relative. (This means it is always compared to other)

Linear search - O(n)
Binary search - O(log n)
Merge sort - O(n log n)

Linear time - **O(n)**
Logarithmic Time - **O(log n)**
Quadratic Time - **O(n^2)**
Quasilinear Time - **O(n log n)**
Polynomial Runtime - **O(n^k)**

Exponential runtimes -
- O(X^n)
- O(n!)

Iterative & Recursive
- **Python hates recursive**
- **Python does not implement tail call optimization.**
- **Swift handles it well.**