# Introduction to Data Structures and Algorithms

Devere Anthony Weaver

---

## 1 Data Structures

A **data structure** is a way of organizing, storing, and performing operations on data. These operations include:

- accessing/updating data
- searching
- inserting
- removing

Some basic data structures to be studied are:

- **Record** - stores subitems (fields) with a name associated with each subitem
- **Array** - stores an ordered list of items, accessible by positional index
- **Linked list** - stores and ordered list of items in nodes, each node stores data and a pointer to the next node
- **Binary tree** - each node stores data and has up to two children (left and right)
- **Hash table** - stores unordered items by mapping (hashing) each item to a location in an array
- **Heap** - a max-heap is a tree that maintains the property that a node's key is greater than or equal to the node's children's keys; a min-heap is less than or equal to
- **Graph** - represents connections among items using vertices connected by edges; each vertex represents an item in a graph while an edge represents a connection between two vertices in a graph

---

## 2 Introduction to Algorithms

An **algorithm** describes a sequence of steps to solve a computational problem or perform a calculation.
A **computational problem** specifies (i) input, (ii) a question, and (iii) output.

Algorithm efficiency is usually measured most commonly in algorithm runtime and an efficient algorithm is typically considered one whose runtime increases no more than polynomially with respect to input size.

**NP-complete** problems are a set of problems for which no known *efficient* algorithm exists. All elements within this set have the following characteristics,

1. No efficient algorithm has been found to solve an NP-complete problem
2. No one has proven that an efficient algorithm to solve an NP-complete problem is impossible
3. If an efficient algorithm exists for one NP-complete problem, then all NP-complete problems can be solved efficiently *How? Proof?*

This concept can be important to know because if a given problem is known to be NP-complete, then one shouldn't waste time finding the optimal solution, rather a good solution might be able to work.

## 3 Relation between data structures and algorithms

Data structures define how data is organized and the operations on the data and the algorithms to implement those operations are typically specific to each data structure.

## 4 Abstract Data Types

An **abstract data type** is a data type described by predefined user operations without indicating how each operation is implemented.

Some common ADTs (and their underlying data structures) are,

- **List** - holding ordered data (array, linked list)
- **Dynamic Array** - holding ordered data and allowing indexed access (array) [*Why doesn't linked list work for this one?*]
- **Stack** - items are only inserted on or removed from the top fo the stack (linked list)
- **Queue** - items are inserted at the end of the queue and remove from the front (linked list)
- **Deque** - items can be inserted and removed at both the font and the back (linked list)
- **Bag** - storing items in which the order doesn't matter and duplicate items are allowed (array, linked list)
- **Set** - collection of distinct items (binary search tree, hash table)
- **Priority queue** - queue where each item has a priority and items with higher priority are closer to the front of the queue (heap)
- **Dictionary (map)** - associates (maps) keys with values (hash table, binary search tree)