Introduction to DS

- A data structure is a specialized format for organizing, processing, retrieving and storing data. There are several basic and advanced types of data structures, all designed to arrange data to suit a specific purpose. Data structures make it easy for users to access and work with the data they need in appropriate ways. Most importantly, data structures frame the organization of information so that machines and humans can better understand it.

- It is not only important to use data structures, but it is also important to choose the proper data structure for each task. Choosing an ill-suited data structure could result in slow runtimes or unresponsive code. Five factors to consider when picking a data structure include the following:

1. What kind of information will be stored?
2. How will that information be used?
3. Where should data persist, or be kept, after it is created?
4. What is the best way to organize the data?
5. What aspects of memory and storage reservation management should be considered?

Some examples of how data structures are used include the following:

- **Storing data.** Data structures are used for efficient data persistence, such as specifying the collection of attributes and corresponding structures used to store records in a database management system.

- **Managing resources and services.** Core operating system (OS) resources and services are enabled through the use of data structures such as linked lists for memory allocation, file directory management and file structure trees, as well as process scheduling queues.

- **Data exchange.** Data structures define the organization of information shared between applications, such as TCP/IP packets.

- **Ordering and sorting.** Data structures such as binary search trees -- also known as an ordered or sorted binary tree -- provide efficient methods of sorting objects, such as character strings used as tags. With data structures such as priority queues, programmers can manage items organized according to a specific priority.

- **Indexing**. Even more sophisticated data structures such as B-trees are used to index objects, such as those stored in a database.

- **Searching.** Indexes created using binary search trees, B-trees or hash tables speed the ability to find a specific sought-after item.
- **Scalability.** Big data applications use data structures for allocating and managing data storage across distributed storage locations, ensuring scalability and performance. Certain big data programming environments -- such as Apache Spark -- provide data structures that mirror the underlying structure of database records to simplify querying.

Characteristics of data structures

1. **Linear or non-linear.** This characteristic describes whether the data items are arranged in sequential order, such as with an array, or in an unordered sequence, such as with a graph.
2. **Homogeneous or heterogeneous.** This characteristic describes whether all data items in a given repository are of the same type. One example is a collection of elements in an array, or of various types, such as an abstract data type defined as a structure in C or a class specification in Java.
3. **Static or dynamic.** This characteristic describes how the data structures are compiled. Static data structures have fixed sizes, structures and memory locations at compile time. Dynamic data structures have sizes, structures and memory locations that can shrink or expand, depending on the use.

Data structure hierarchy