# Data Structures and Algorithms

This document details how to learn data structures and algorithms

# Determine Data Structure or Algorithm to Learn

In this step, you will simply select a data structure or algorithm you want to learn. A good place to start if the books collection or the data structures and algorithms specialization.

# For Each Data Structure or Algorithm

For example, if you selected LinkedList

1. Watch videos on what a linked list is
2. Determine the operations the linked list supports
3. Create a skeleton class that implements the linked list and all operations. This class should not implement any functionality yet.
4. Create a series of unit tests on the class to test whether all supported operations.
   1. Implement edge and corner cases such as passing null references, passing empty lists, passing objects that the class does not expect (for example pass a number where a string is expected).
   2. Implement stress test on the class and compare your code to code you stole from the internet. For example, a linked list implementation, your implementation should be coming back exactly as a valid implementation.
   3. Show the unit tests failing
5. Create a class-based implementation of the linked list and show the unit tests passing.
6. Look for 1 additional question on the internet. For example, how to reverse a linked list.

# List of Data Structures and Algorithms

Data Structures

1. Queues
   1. Queue
   2. Double ended queue
   3. Priority Queue
2. Trees
3. Binary Tree
4. Binary Search Tree
5. AVL Tree
6. Splay Tree
7. Heap
8. Stack
9. Disjoint Set
10. Hash Table
    1. Hash Function
    2. Distributed Hash Table
11. Graph

Algorithms

1. Rabin-Karm Algorithm
2. Graph algorithms
   1. Graph ordering - pre and post visit order
3. Directed Acyclic Graphs
4. Topological Sort
5. Strongly Connected Components
6. Most direct route in a graph
7. Breadth First Search
8. Shortest path tree
9. Dijkstra's Algorithm

Queue (Completed)

<https://www.geeksforgeeks.org/queue-data-structure/>

Like [Stack](http://quiz.geeksforgeeks.org/stack-set-1/), [Queue](http://en.wikipedia.org/wiki/Queue_%28data_structure%29)is a linear structure which follows a particular order in which the operations are performed. The order is **F**irst **I**n **F**irst **O**ut (FIFO).  A good example of queue is any queue of consumers for a resource where the consumer that came first is served first.  
The difference between stacks and queues is in removing. In a stack we remove the item the most recently added; in a queue, we remove the item the least recently added.

**Operations on Queue:**  
Mainly the following four basic operations are performed on queue:

**Enqueue:**Adds an item to the queue. If the queue is full, then it is said to be an Overflow condition.  
**Dequeue:** Removes an item from the queue. The items are popped in the same order in which they are pushed. If the queue is empty, then it is said to be an Underflow condition.  
**Front:**Get the front item from queue.  
**Rear:** Get the last item from queue.