Stacks, Queues and Graphs

Data Structures
Lecturer: Caio Fonseca

Introduction

- In this section we will revisit a number of collection types that we (briefly) considered earlier in the module to consider how they might be implemented in light of the techniques we have seen thus far.
- We will specifically consider basic implementations of stacks, queues and graphs.

Stacks

- A stack is a "last in first out" (LIFO) collection of objects whereby only the top (last in) object can be manipulated.
- Possible implementation:
  - A list structure could be used in the first instance (linked list or array-based list).
  - Like for sets, no methods would need to be provided for accessing stack elements by index. Although sequence is important in a stack, only the top (last in) object should be manipulated in general.

Continued…

- The stack could be made Iterable to allow us to access/read/search all elements in sequence.
- Typical stack methods could be provided:
  - push() → add item to top of the stack
  - pop() → remove and return top item
  - top() → return top item without removing it
  - isEmpty() → whether the stack is empty

Queues

- Queues are collections of ordered elements being held prior to processing, usually on a "first in first out" (FIFO) basis.
- A queue's size may be bounded or unbounded.
- Priority queues order items according to some criteria.
- A stack is essentially a "last in first out" (LIFO) queue.
- Possible implementation of a FIFO queue:
  - Once again, a list structure can be used in the first instance (linked list or array based).
    - As we used a linked list for the set and stack example implementations we'll use an array for the (bounded) queue example.
  - Methods to add items to the end and take items from the front are key for FIFO behaviour.

Continued…

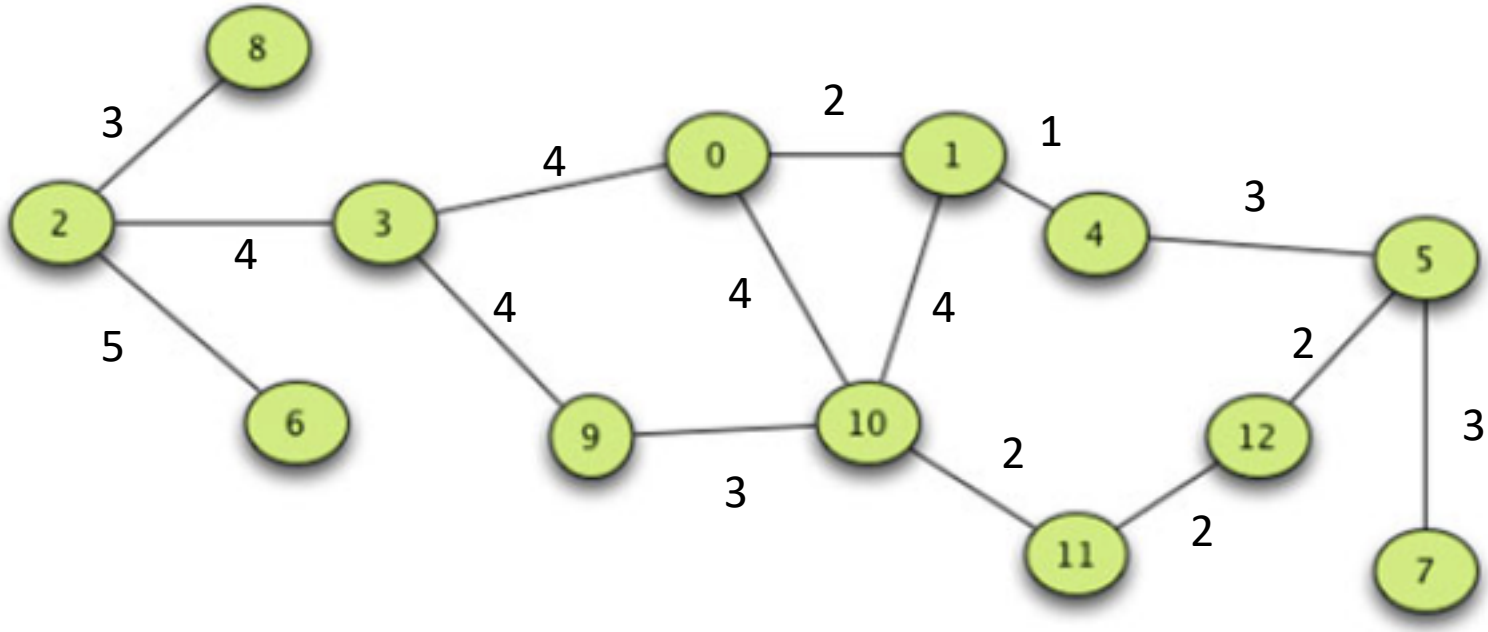
- Typical methods for queues:
  - dequeue() → returns the first item enqueued and removes it from the queue
  - enqueue() → enqueues the item on the queue
  - front() → return the front item without removing it from queue
  - isEmpty() → check to see if queue is empty

Graphs

- A Graph is defined by a set of vertices and a set of edges.
- The edges are subsets of the vertices, and are defined by pairs of vertices.
- The graphs can be divided into directed graphs and undirected graphs.
  - A directed graph means that the edges can be traversed in one direction only;
  - An undirected graph means that the edges can be traversed in the two directions.

Graphs

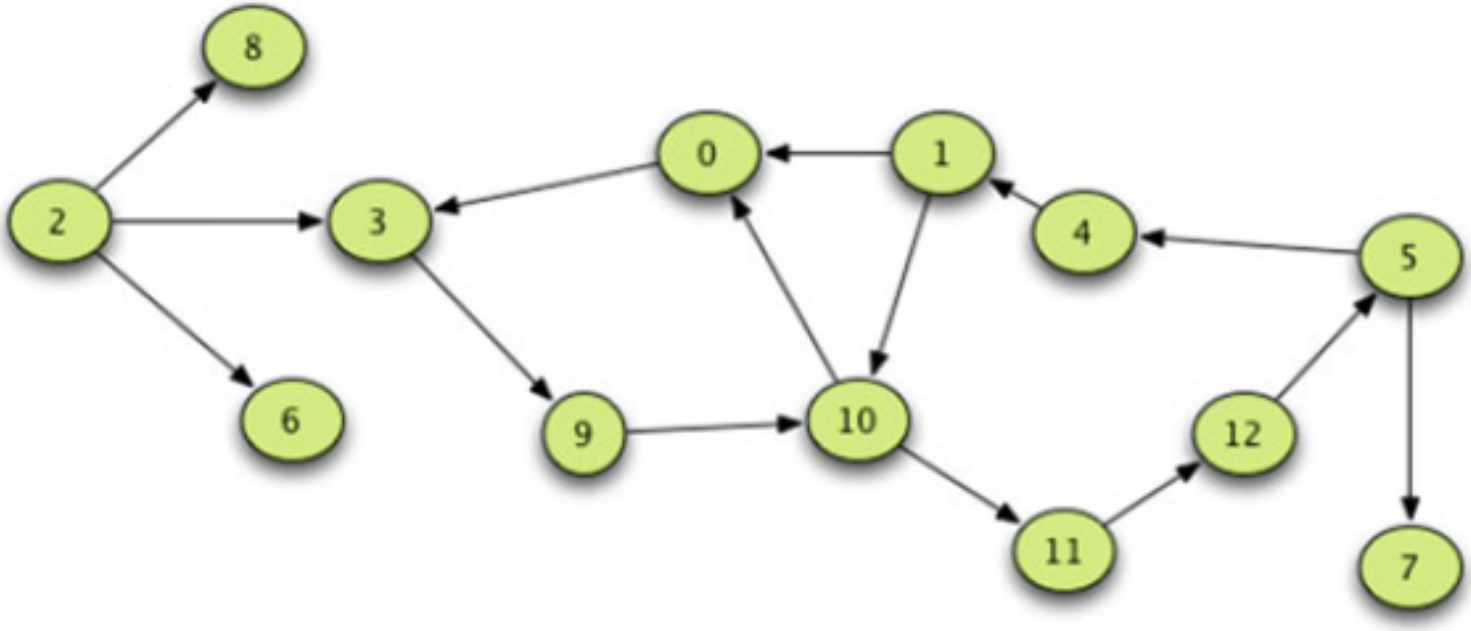
- Undirected Graph:



Ref: Data Structures and Algorithms with Python – Kent Lee & Steve Hubbard

Graphs

- Directed Graph:



Ref: Data Structures and Algorithms with Python – Kent Lee & Steve Hubbard