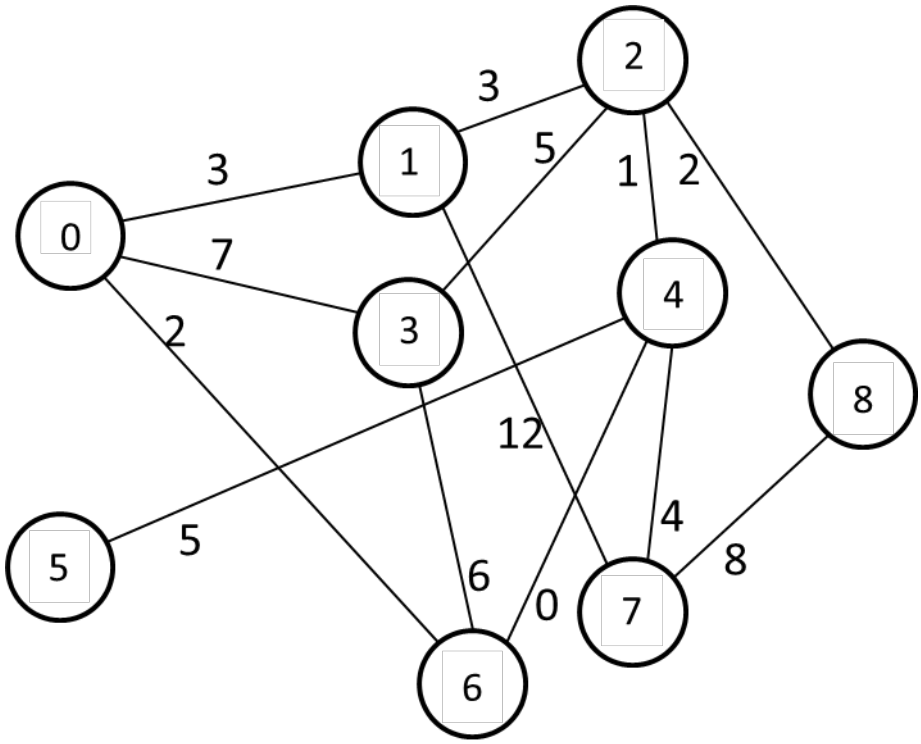
Representing Graphs - Tutorial Exercises



Q1: Write a Python program that

(a) encodes the connectivity and weight information in the network shown above

(b) write functions to:

(i) find the weight of and edge connecting two specified verticies

(ii) lists all pairs of vertices that have a given weight
e.g. for weight 3, it would return [0,1] and [1,2]

(iii) finds the maximum-weight and minimum-weight edges in the network.

(iv) finds the minimum weight edge associated with any given vertex

(v) prints a list of (vertex,weight) pairs, where, for each pair, the weight is the lowest weight of the edges connecting to[1] the vertex , and the list is sorted in order of increasing weight.

That is, if the vertices represent villages in a mountainous region, and the weights represent the time to travel from one village to another, the list shows the villages from the least to the most isolated.

[1] 'connected to the vertex' is sometime written in '***incident on*** the vertex'

Q2 - OPTIONAL

If you had represented the network using **weighted connectivity triples**

e.g. [[1, 2, 3], [1, 4, 7], [1, 7, 2]]

means that **node 1** is connected

to node **2** (with weight 3)
to node **4** (with weight 7)
to node **7** (with weight 2)

(a) If the size of the network were scaled up to tens of thousands or millions of nodes, how would the speed of your implementation (from Q1) compare with an implementation based on **weighted connectivity triples**?

(b) You should consider each of the five functions separately.