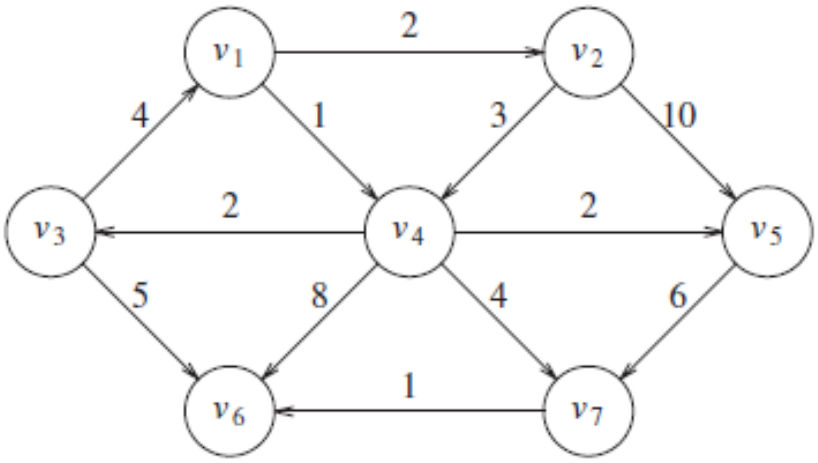
Lab 6: Graph shortest path (Dijkestra) implementation

For this lab, copy the graph implementation code from the book (chapter 8.6, both 'Vertex' and 'Graph' classes)

Write a function 'shortest_path(vertex)', which takes a vertex as input and returns a set of path lists (a path is a list of vertices, starting from the input vertex and ending with the target vertex), and cost of each path. Each path must be the shortest path between the input vertex and the target. For example: from the graph shown in the lecture for this algorithm, the input vertex is v1, the output should be: {(v1,[],0), (v2,[v1,v2], 2), (v3,[v1,v4,v3],3), (v4,[v1,v4],1), (v5,[v1,v4,v5],3), (v6,[v1,v4,v7,v6],6), v7:[v1,v4,v7], 5}



Traverse the graph in 'breadth-first' order. To do this, you'll need to use a queue (you can use a python list, but make sure you take out the nodes in the order they were inserted). Mark every node as 'visited' when you examine it and add its adjacencies to the queue (add a variable to the Vertex class to do that). Update the cost only if it is lower than the existing cost.