```
class Graph(object):

    def __init__(self, graph_dict=None):

        if graph_dict == None:
            graph_dict = {}
        self._graph_dict = graph_dict

    def edges(self, vertice):
        """ returns a list of all the edges of a vertice"""
        return self._graph_dict[vertice]

    def all_vertices(self):
        """ returns the vertices of a graph as a set """
        return set(self._graph_dict.keys())

    def all_edges(self):
        """ returns the edges of a graph """
        return self.__generate_edges()

    def add_vertex(self, vertex):
        """ If the vertex "vertex" is not in
            self._graph_dict, a key "vertex" with an empty
            list as a value is added to the dictionary.
            Otherwise nothing has to be done.
        """
        if vertex not in self._graph_dict:
            self._graph_dict[vertex] = []

    def add_edge(self, edge):
        """ assumes that edge is of type set, tuple or list;
            between two vertices can be multiple edges!
        """
        edge = set(edge)
        vertex1, vertex2 = tuple(edge)
        for x, y in [(vertex1, vertex2), (vertex2, vertex1)]:
            if x in self._graph_dict:
                self._graph_dict[x].add(y)
            else:
                self._graph_dict[x] = [y]

    def generate_edge(self):
        edges = []
        for vertex in self._graph_dict:
            for neighbour in self._graph_dict[vertex]:
                if {neighbour, vertex} not in edges:
                    edges.append({vertex, neighbour})
        return edges

    def print_graph(self):
        print("--------------------")
        for connect in self._graph_dict:
            connection = self._graph_dict[connect]
            print(connect,"is connected to",connection)

    def __iter__(self):
        self._iter_obj = iter(self._graph_dict)
        return self._iter_obj

    def __next__(self):
        """ allows us to iterate over the vertices """
        return next(self._iter_obj)

    def __str__(self):
        res = "vertices: "
        for k in self._graph_dict:
            res += str(k) + " "
        res += "\nedges: "
        for edge in self.__generate_edges():
            res += str(edge) + " "
        return res

    def find_path(self, start_vertex, end_vertex, path=None):

        if path == None:
            path = []
```

```
            graph = self._graph_dict
            path = path + [start_vertex]
            if start_vertex == end_vertex:
                return path
            if start_vertex not in graph:
                return None
            for vertex in graph[start_vertex]:
                if vertex not in path:
                    extended_path = self.find_path(vertex,
                                                   end_vertex,
                                                   path)
                    if extended_path:
                        return extended_path
            return None

        def check_graph(self):
            list_path = []
            for vertex in self._graph_dict:
                for next_vertex in self._graph_dict:
                    if vertex != next_vertex:
                        path = self.find_path(vertex, next_vertex)
                        if path == None:
                            list_path.append(path)
                        elif path != None:
                            list_path.extend(path)
            if None in list_path:
               print("Yes")
            elif None not in list_path:
                print("No")

    dictionary = { "1st vertex" : {"2nd vertex"},
          "2nd vertex" : {"1st vertex", "3rd vertex", "4th vertex"},
          "3rd vertex" : {"2nd vertex", "4th vertex", "5th vertex"},
          "4th vertex" : {"2nd vertex", "3rd vertes", "6th vertex"},
          "5th vertex" : {"3rd vertex"},
          "6th vertex" : {"4th vertex"},
          "7th vertex" : {"8th vertex"},
          "8th vertex" : {"7th vertex", "9th vertex", "10th vertex"},
          "9th vertex" : {"8th vertex", "10th vertex", "11th vertex"},
          "10th vertex" : {"8th vertex", "9th vertex", "12th vertex"},
          "11th vertex" : {"9th vertex"},
          "12th vertex" : {"10th vertex"}
        }



    graph = Graph(dictionary)
    graph.print_graph()
    print("----------")
    print("Unconnected Graph")
    graph.check_graph()
```

```
     -------------------
     1st vertex is connected to {'2nd vertex'}
     2nd vertex is connected to {'3rd vertex', '1st vertex', '4th vertex'}
     3rd vertex is connected to {'5th vertex', '2nd vertex', '4th vertex'}
     4th vertex is connected to {'3rd vertes', '2nd vertex', '6th vertex'}
     5th vertex is connected to {'3rd vertex'}
     6th vertex is connected to {'4th vertex'}
     7th vertex is connected to {'8th vertex'}
     8th vertex is connected to {'7th vertex', '10th vertex', '9th vertex'}
     9th vertex is connected to {'10th vertex', '8th vertex', '11th vertex'}
     10th vertex is connected to {'8th vertex', '9th vertex', '12th vertex'}
     11th vertex is connected to {'9th vertex'}
     12th vertex is connected to {'10th vertex'}
     ----------
     Unconnected Graph
     Yes
```