Bài Tập Thực Hành N°5-6: Đồ thị vô hướng (undirected graph)

**Bài tập 1**

Cho đồ thị với các đỉnh như hình 1(a):

- hãy xây dựng cấu trúc lưu đồ thị (các đỉnh) vào trong máy tính
- liệt kê danh sách tất cả các cạnh liên quan đến đỉnh tương ứng
- liệt kê tất cả các cạnh của đồ thị
- tìm các đỉnh cô lập trong đồ thị


(a) (b)

Hình 1: Biểu diễn đồ thị vào trong máy tính

**Bài tập 2: (Lập trình hướng đối tượng)**

Xây dựng một lớp Graph cơ bản cho đồ thị vô hướng hình 1(b). Thực hiện các yêu cầu sau:

- in ra tất cả các đỉnh của đồ thị
- in tất cả các cạnh tương ứng với các đỉnh
- in tất cả các cạnh của đồ thị
- thêm một đỉnh mới vào đồ thị và in ra lại tất cả các đỉnh của đồ thị
- thêm một cạnh mới vào đồ thị và in ra lại tất cả các cạnh của đồ thị

Hướng dẫn:

```
class Graph(object):

    def __init__(self, graph_dict=None):
        """ initializes a graph object
            If no dictionary or None is given,
            an empty dictionary will be used
        """
        if graph_dict == None:
            graph_dict = {}
        # storing the vertices and their corresponding adjacent vertices.
        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_edges(self):
        """ A static method generating the edges of the
            graph "graph". Edges are represented as sets
            with one (a loop back to the vertex) or two
            vertices
        """
        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 __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
```

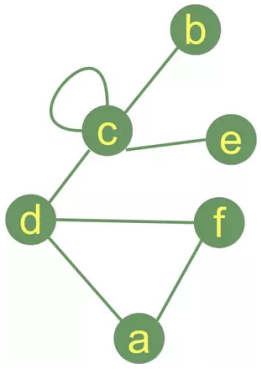
## Bài tập 3

Viết bổ sung các phương thức tìm đường đi từ một đỉnh bắt đầu và kết thúc, liệt kê tất cả đường đi của đồ thị vô hướng trong các hình 1-2 vào lớp Graph ở trên.

```
    def find_path(self, start_vertex, end_vertex, path=None):
        """ find a path from start_vertex to end_vertex
            in graph """
        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 find_all_paths(self, start_vertex, end_vertex, path=[]):
```

```
23          """ find all paths from start_vertex to
24              end_vertex in graph """
25          graph = self._graph_dict
26          path = path + [start_vertex]
27          if start_vertex == end_vertex:
28              return [path]
29          if start_vertex not in graph:
30              return []
31          paths = []
32          for vertex in graph[start_vertex]:
33              if vertex not in path:
34                  extended_paths = self.find_all_paths(vertex,
35                                                       end_vertex,
36                                                       path)
37                  for p in extended_paths:
38                      paths.append(p)
39          return paths
```




Hình 2: Tìm tất cả đường đi cho đồ thị vô hướng

HẾT

Lưu ý: SV nộp bài tập về nhà ở *https://elearning.ntu.edu.vn/*. Nếu SV sao chép bài thì các bài giống nhau sẽ là 0 điểm.