Trước hết, tôi có chương trình để tạo ngẫu nhiên một ma trận kề A của đồ thị như sau:

```
from itertools import product
from random import choice
import numpy as np
#ma trận trả về này có kiểu np.array 2 chiều
def random_graph(n, directed = False):
  # n là số cạnh
  # directed là biến để chọn loại đồ thị trả về:
  #   True: đồ thị có hướng
  #   False: đồ thị vô hướng (mặc định)
  A = np.zeros((n, n), dtype = int)
  for i, j in product(range(n), repeat=2):
    A[i, j] = choice((0,1)) if (directed and i != j) or i > j else 0
  if directed:
    return A
  else:
    return A + A.T
```

Ví dụ:

```
random_graph(5)
#có thể trả về
#array([[0, 1, 1, 0, 0],
#       [1, 0, 0, 1, 0],
#       [1, 0, 0, 0, 1],
#       [0, 1, 0, 0, 0],
#       [0, 0, 1, 0, 0]])
```

Cho đồ thị G(V, E) có ma trận kề A.

Bài 1. Viết chương trình nhận tham số là ma trận A (kiểu dữ liệu np.array 2 chiều) và trả về kết quả đây là đồ thị vô hướng hay hữu hướng. Để kiểm tra một ma trận có đối xứng hay không, dùng (A == A.T).all()
Ví dụ:

```
A = np.array([[0, 1], [1,0]])
is_directed_graph(A)
#trả về False, do A đối xứng nên đồ thị tương ứng là vô hướng
```

Bài 2. Viết chương trình nhận tham số là ma trận A, trả về số cạnh của đồ thị.

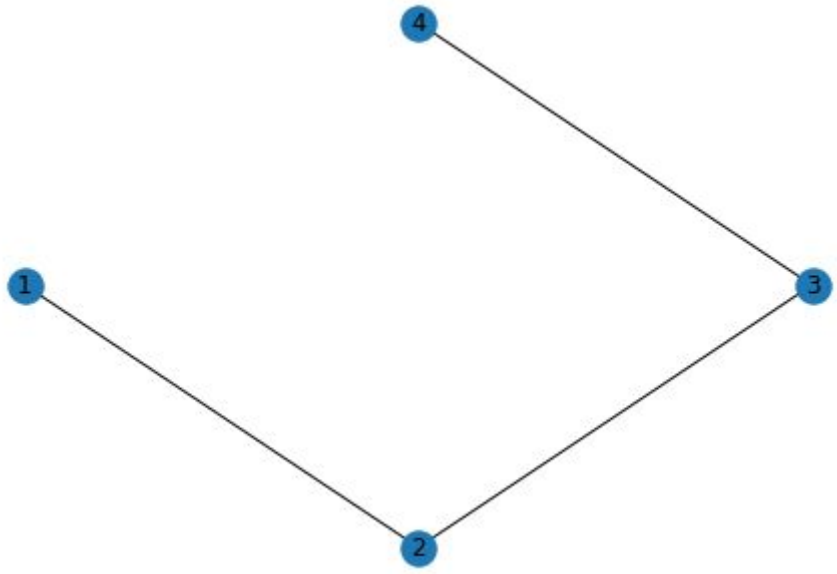
Bài 3. Viết chương trình nhận tham số là ma trận A, trả về danh sách các cạnh của đồ thị. Một cạnh được biểu diễn dưới dạng (u, v)

```
A = np.array([[0, 1], [1, 0]])
get_edges(A)
# trả về [(1, 2)]
```

Bài 4. Chúng ta có thể biểu diễn đồ thị G(V, E) bằng hình ảnh như sau:

```
import networkx as nx
# khởi tạo đồ thị vô hướng trống
# nếu muốn khởi tạo đồ thị có hướng, dùng nx.DiGraph()
g = nx.Graph()
# thêm cạnh
g.add_edges(1, 2)
g.add_edges(2, 3)
g.add_edges(3, 4)
# vẽ đồ thị, with_labels để vẽ ra nhãn của đỉnh
nx.draw_shell(g, with_labels=True)
```

Nếu chạy đoạn code trên, chúng ta được



Viết chương trình với tham số là ma trận A, vẽ ra đồ thị.