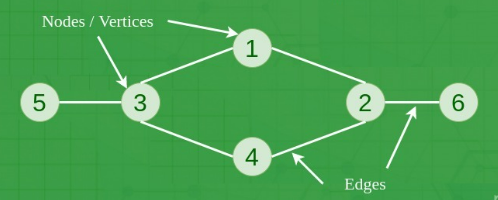
**14. Graph**

**14.1 Giới thiệu về Graph**

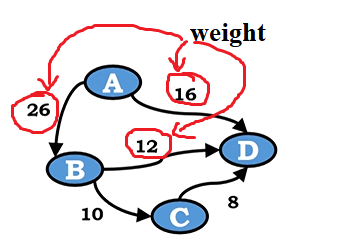
**14.1.1 Khái Niệm Graph**

Graph là một cấu trúc dữ liệu phi tuyến tính bao gồm các đỉnh và cạnh. Các đỉnh đôi khi còn được gọi là các nút (node) và các cạnh (edges) là các đường hoặc cung kết nối bất kỳ hai nút nào trong đồ thị. Một cách chính thức hơn, Đồ thị được tạo thành từ một tập hợp các đỉnh (V) và một tập hợp các cạnh (E). Đồ thị được biểu thị bằng G(E, V).

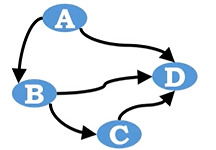


**14.1.2 Một số khái niệm cần chú ý:**

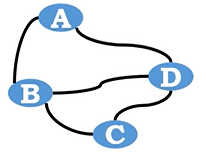
* Vertice: Đỉnh trong Graph, ví dụ như trên hình là: 5, 3, 1, 4, 2, 6.
* Edge: Cạnh của Graph, ví dụ như trên hình: 1-2, 2-1, 2-6, …
* Weight: Số biểu thị cho Edge của Graph



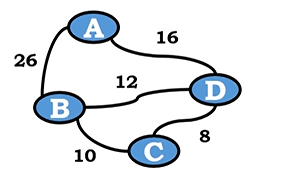
* Directed Edge: 1 cạnh (u, v) bất kì được định hướng, u-v có một thứ tự sẵn trong Graph, kí hiệu là u 🡪 v



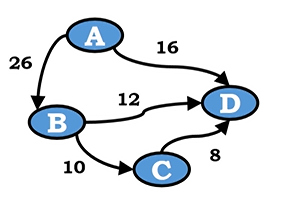
* Undirected Edge: 1 cạnh (u, v) bất kì chưa được định hướng, u-v không có một thứ tự sẵn, không có kí hiệu 🡪



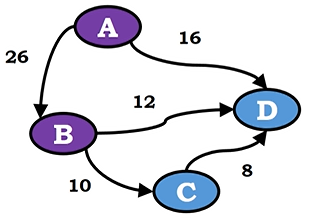
* Weighted Undirected Edge:



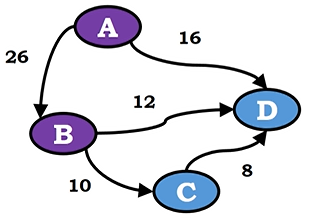
* Weighted Directed Edge:



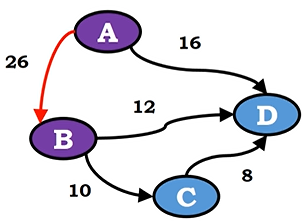
* End Vertice: 2 Vertice được kết nối với nhau thông qua 1 edge



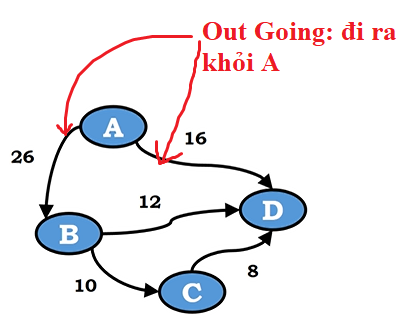
* Adjacent Vertice: 2 Vertice được xem là Adjacent với nhau nếu có 1 Edge kết nối giữa chúng, trong đó 1 vertice đóng vai trò là nguồn, còn 1 vertice đóng vai trò là đích.



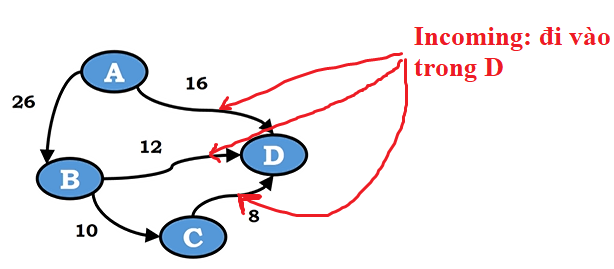
* Incident Edge: Nếu vetex(số cạnh) là 1 trong số end point



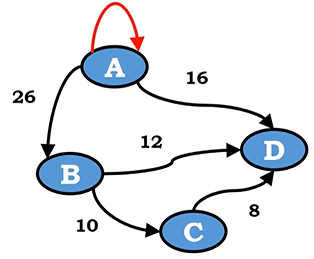
* Outgoing Edge: Các Edge đi ra khỏi 1 Vertice



* Incoming Edge: Các Edge đi vào trong 1 Vertice



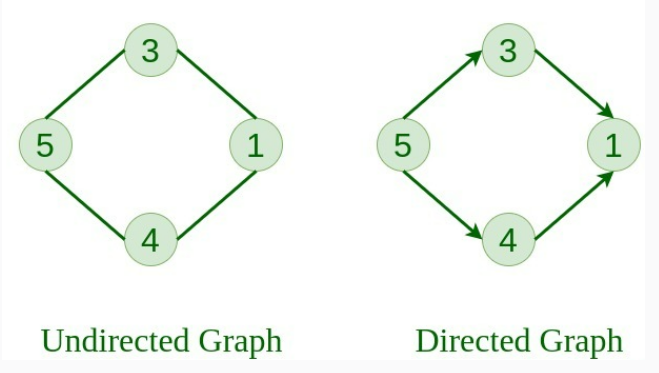
* Self loop: Vertice tự có 1 Edge với chính nó.



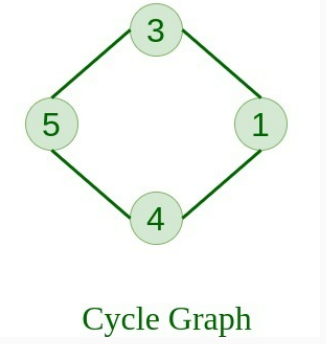
* Vertex: Số lượng Edge(cạnh)

**14.1.3 Đường Đi – Path**

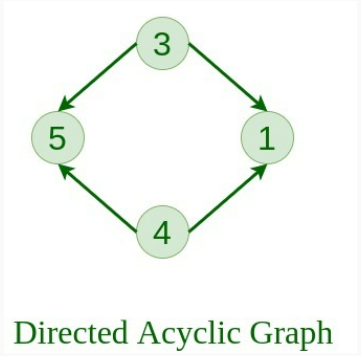
* Khái Niệm: Là 1 tập hợp các cạnh đi từ 1 vetex này đến 1 vetex khác
* Một số Path quan trọng:
* Directed Graph: Directed Graph là đồ thị trong đó các cạnh có hướng. Tức là các đỉnh là các cặp có thứ tự trong định nghĩa của mỗi cạnh.
* Undirected Graph: Undirected Graph là đồ thị trong đó các cạnh không có hướng. Tức là các đỉnh là các cặp không có thứ tự trong định nghĩa của mỗi cạnh.



* Cycle: Đường đi bắt đầu và kết thúc ở cùng 1 vetex



* Directed Cyclic Graph: Là Graph mà bên trong không tồn tại Cycle

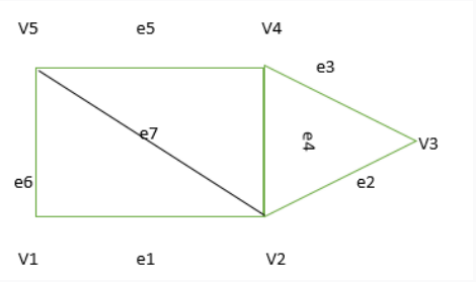


**14.1.4 SubGraph và Các thành phần kết nối:**

* SubGraph: Là Một graph bên trong 1 Graph nào đó lớn hơn.

Ví dụ:

Đồ thị G1 = (V1, E1) được gọi là đồ thị con của đồ thị G(V, E) nếu V1(G) là tập con của V(G) và E1(G) là tập con của E(G) sao cho mỗi cạnh của G1 có cùng đỉnh đầu và đỉnh cuối như trong G.



* Các thành phần kết nối (Connected Component):

Là các subgraph được kết nối với nhau.

* Articulation Point: Là các cạnh bị loại bỏ bên trong 1 connected components.
* Bi-Connected Component: Các Connected Component được kết nối với nhau bởi 2 edge.
* Strongly Connected Graph: Là Graph mà khi bất kì đỉnh nào cũng được in-degree-indeg bởi 1 đỉnh khác trong connected component.

**14.2 Các Phương Thức của Graph**

Nguyên tắt: Graph được cấu thành bởi Vertices (đỉnh) và Edge (cạnh)

Các phương thức:

* create(n): Tạo Graph chỉ bao gồm các đỉnh và không có cạnh.
* insert\_edge(u, v, w=1): Tạo Edge từ u đến v, lưu trữ weight là w.
* remove\_edge(u, v): xóa edge từ u đến v.
* exist\_edge(u, v): trả về true nếu tồn tại 1 edge ở giữa u và v, ngược lại trả về false.
* vertex\_count(): trả về số lượng vertices(đỉnh) bên trong 1 graph.
* edge\_count(): trả về số lượng edges bên trong 1 graph.
* vertices(): trả về toàn bộ vertices(các đỉnh) bên trong graph.
* edge(): trả về toàn bộ edges(các cạnh) bên trong graph.

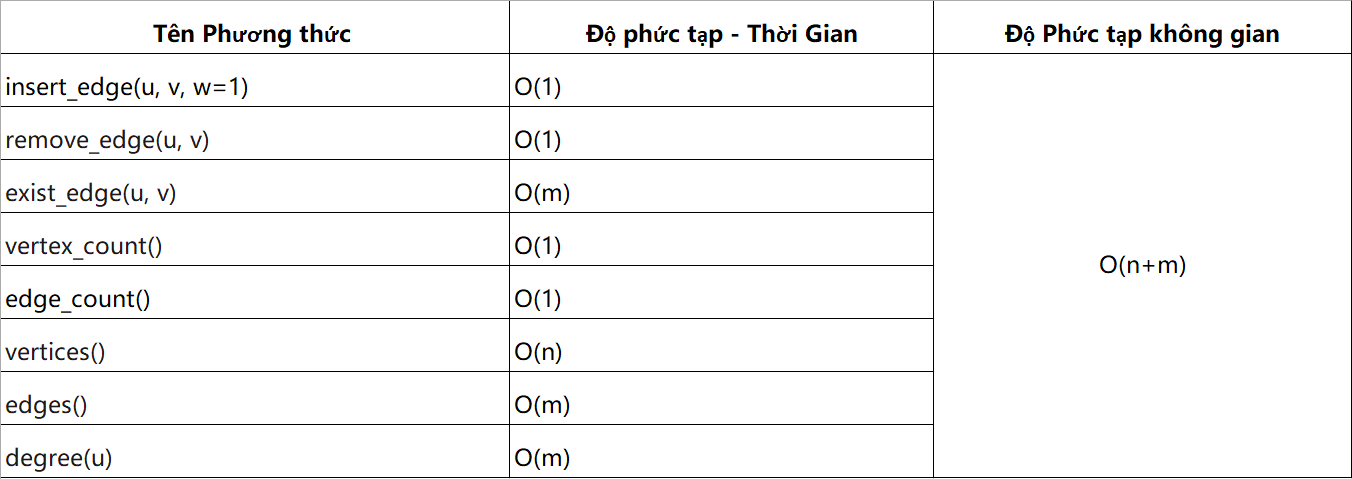
**14.3 Biểu diễn Graph**

Graph có thể được biểu diễn dưới nhiều dạng kiểu cấu trúc dữ liệu khác nhau

**14.3.1 Các kiểu dữ liệu:**

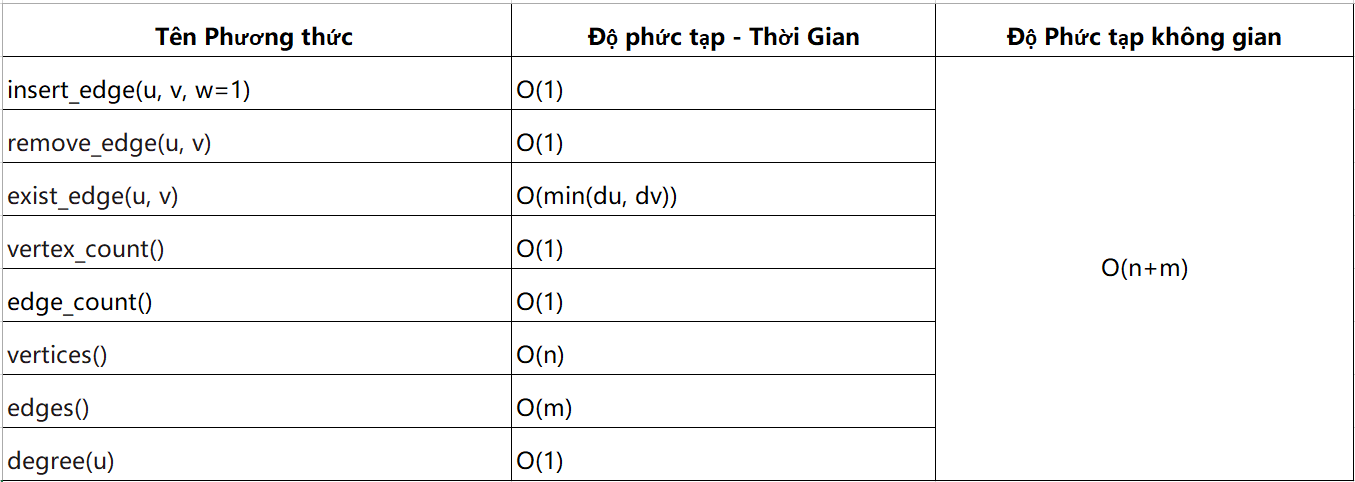
* Edge List: Bao gồm danh sách toàn bộ các edge.

Bảng Phân loại phương thức:



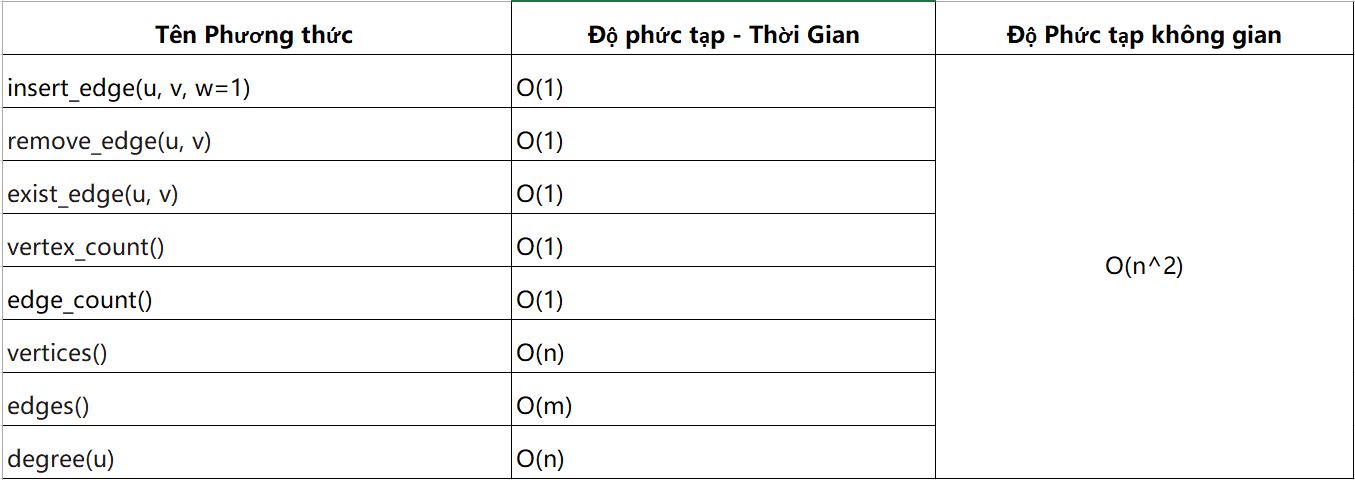
* Adjacency List: Bao gồm các edge được chia ra, đối với từng vertex.

Bảng Phân loại phương thức:

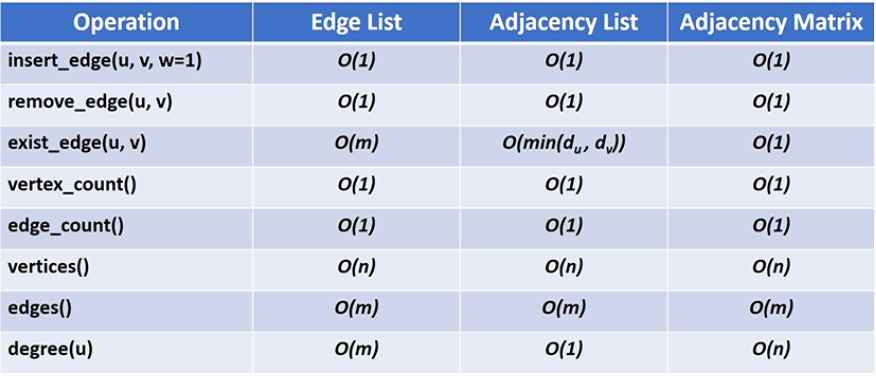


* Adjacency Matrix: Bao gồm 1 ma trận vertices, nơi lưu trữ các thành phần bộ nhớ dùng để tham chiếu đến các edge.

Bảng Phân loại phương thức:



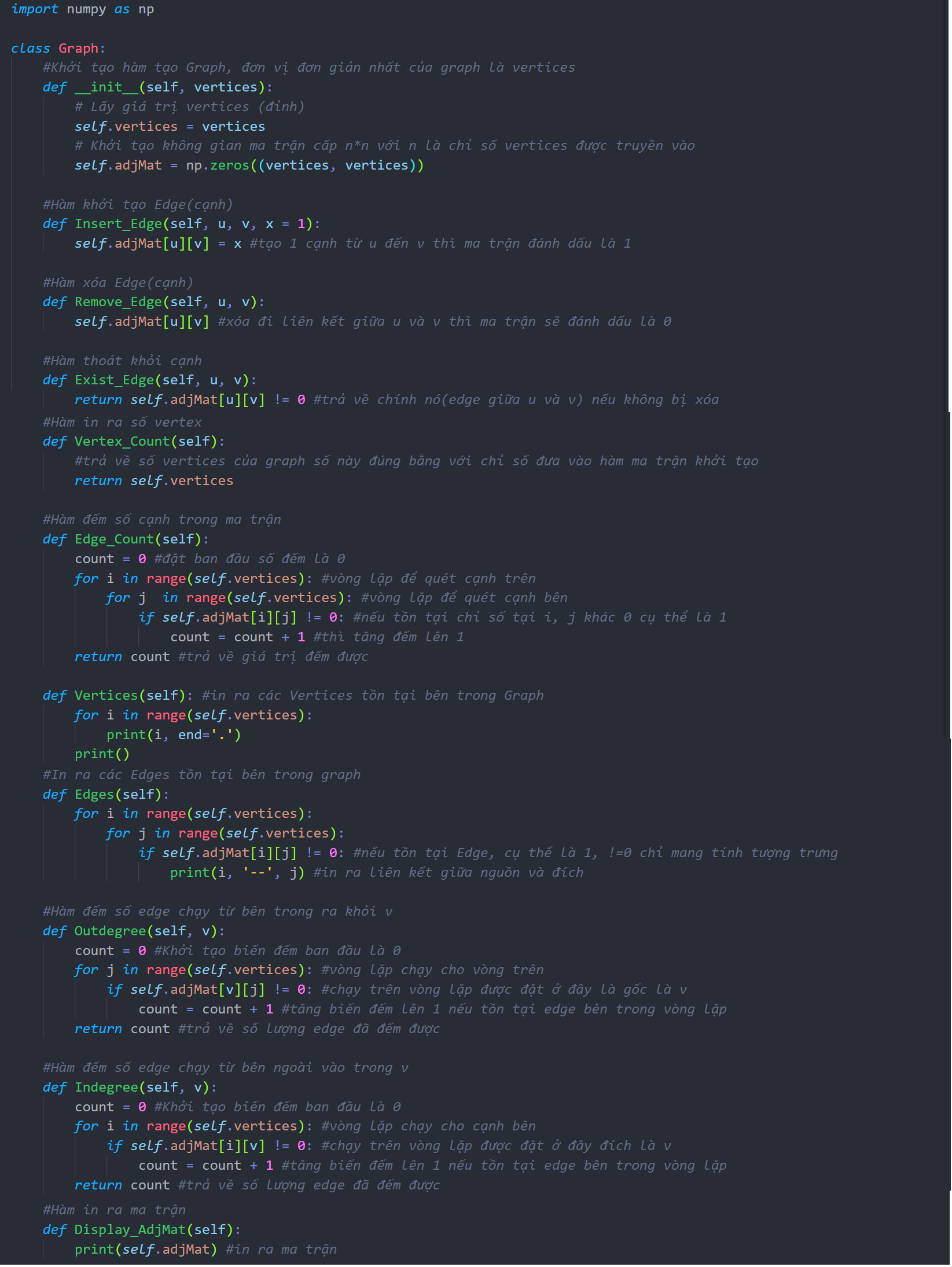
**14.3.2 Tổng Hợp các phương thức:**

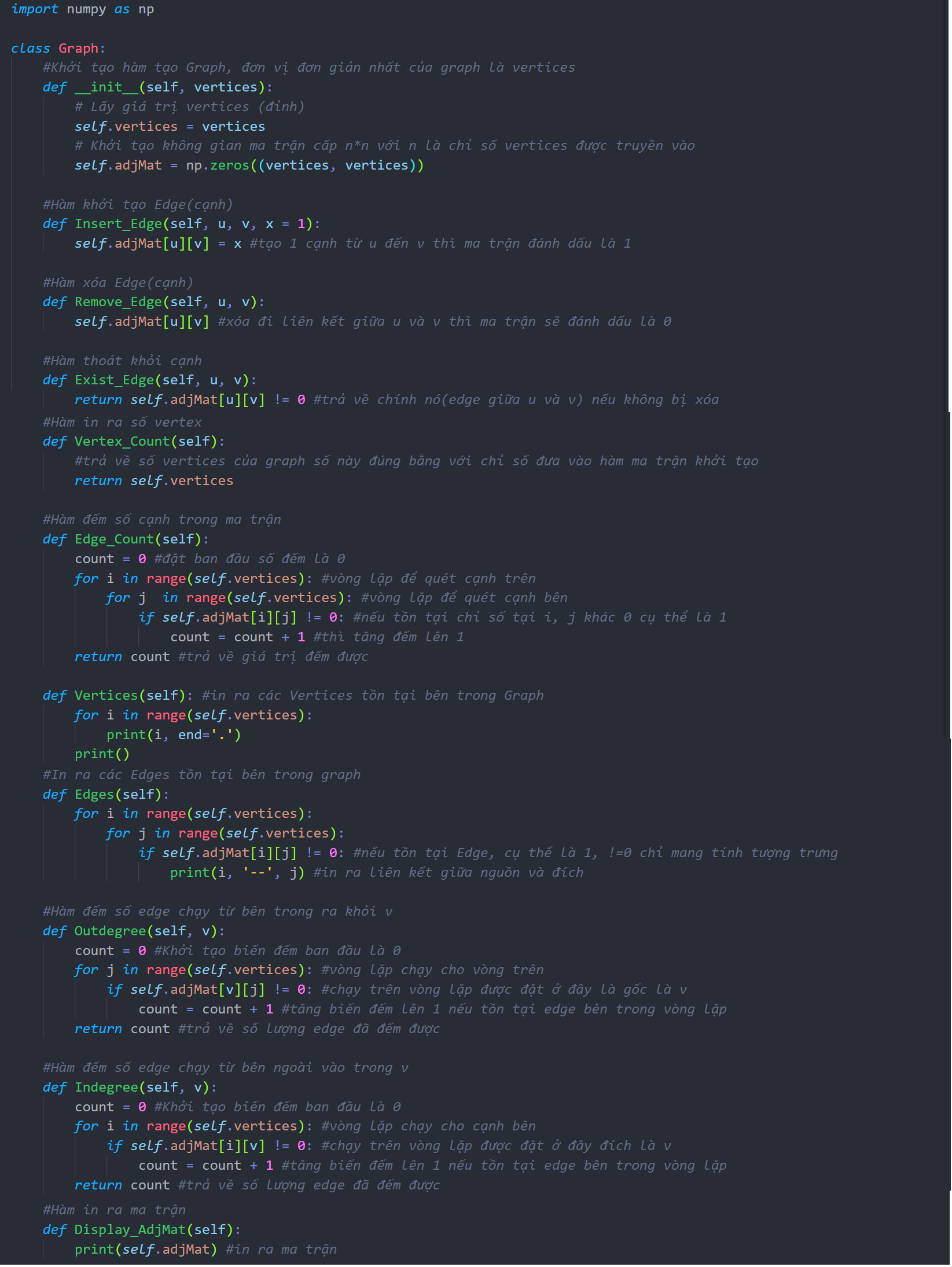




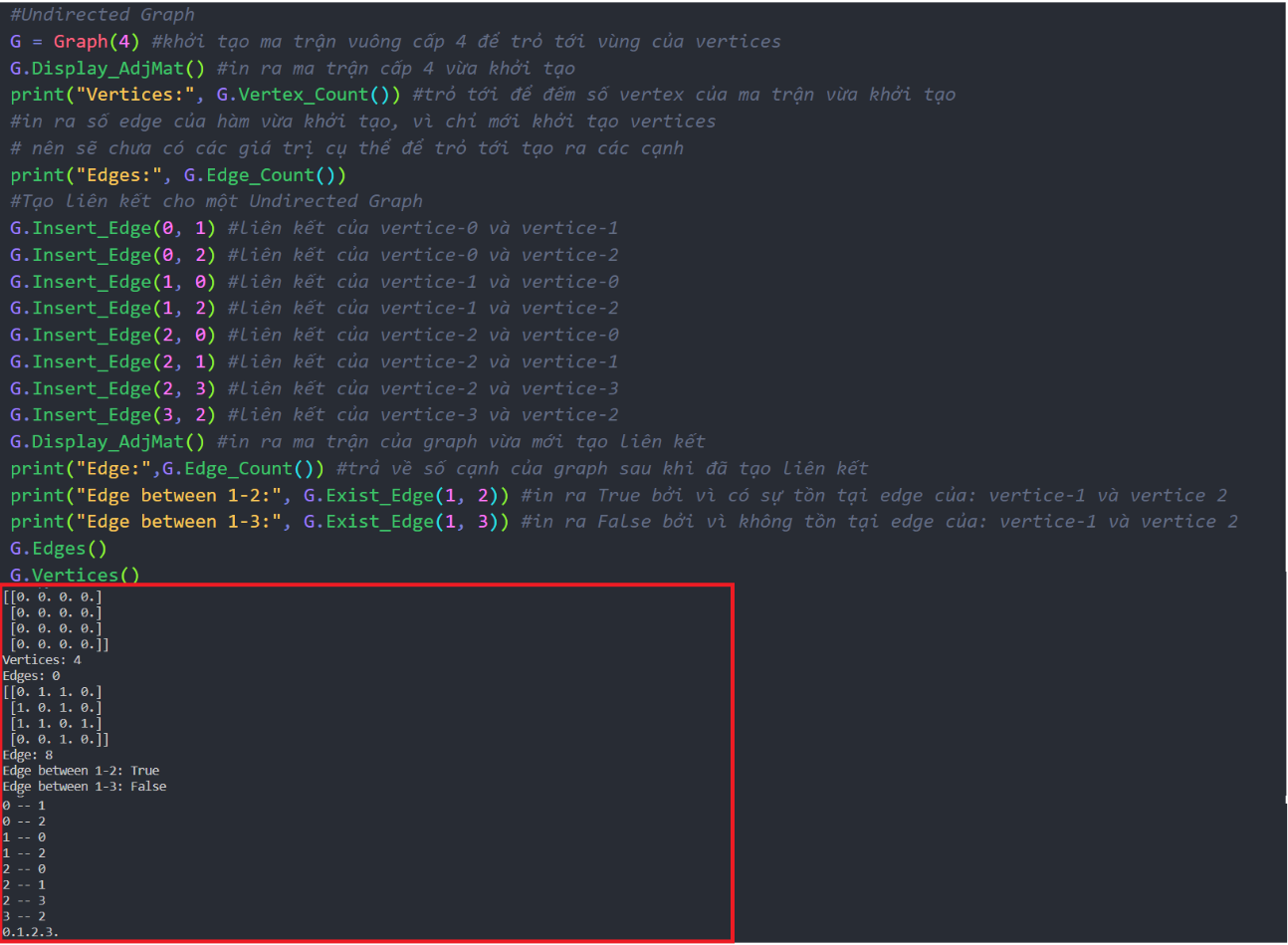
**14.3.3 Lập trình cho một số Graph cơ bản (Python)**

* Các phương thức cơ bản:

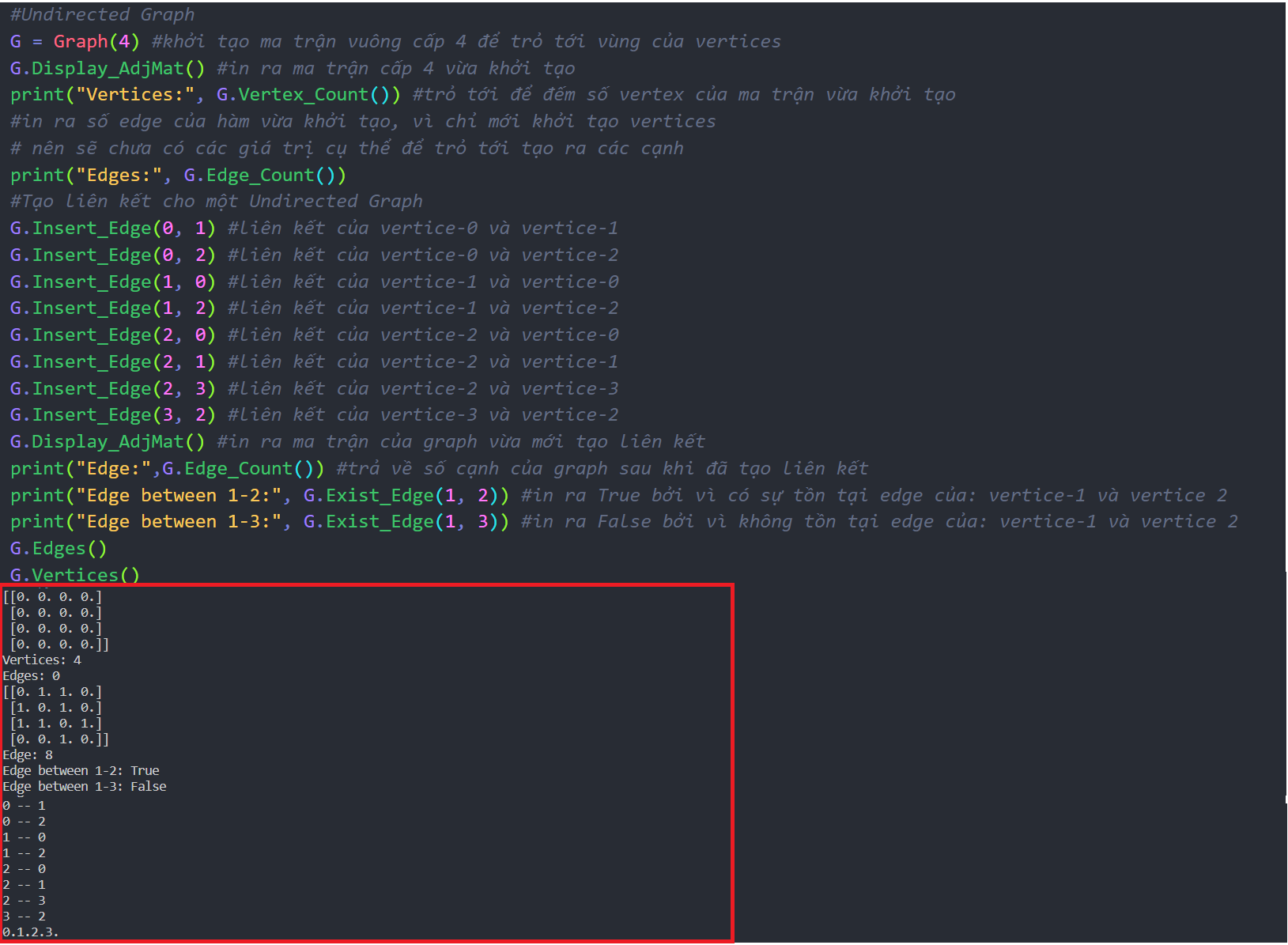




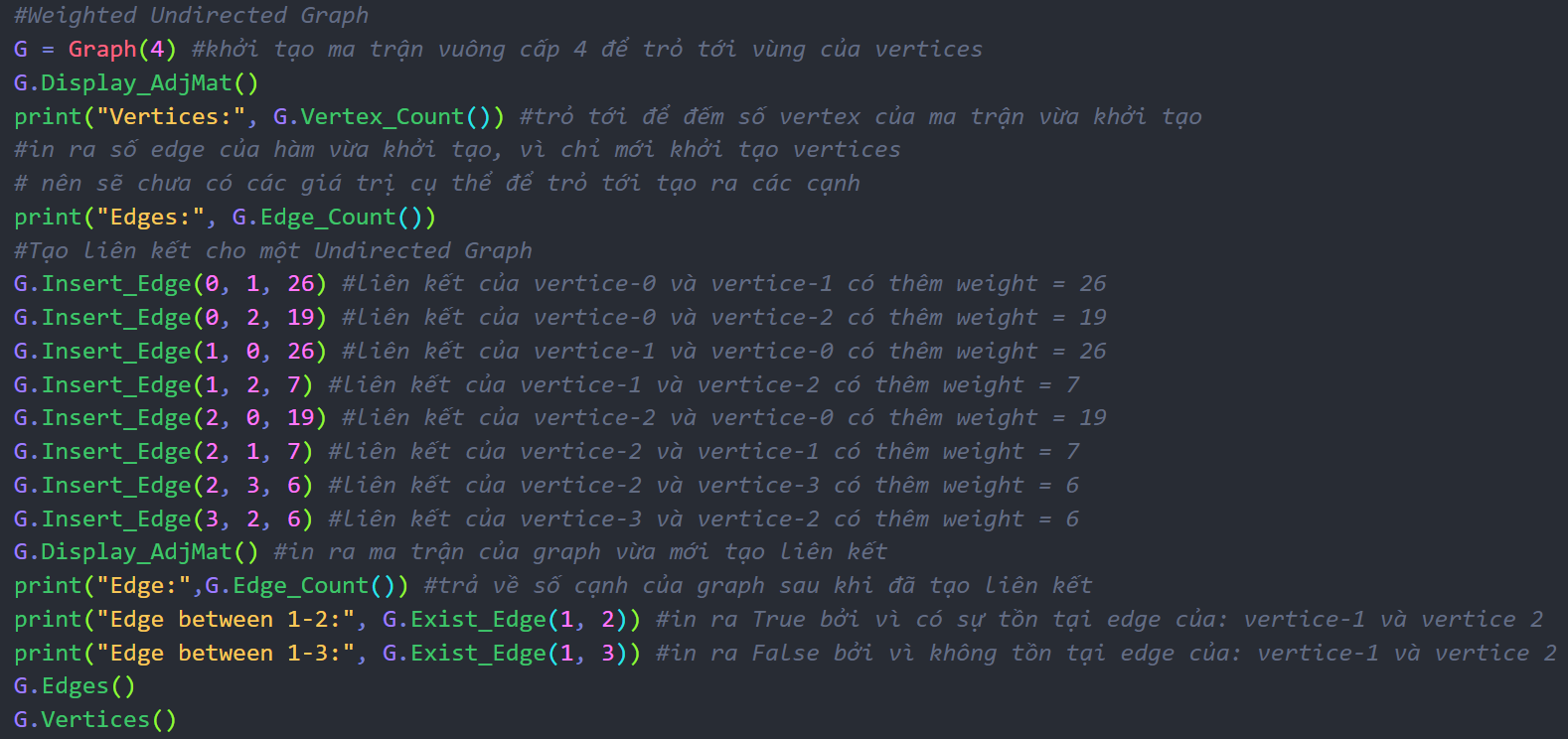
* Undirected Graph:



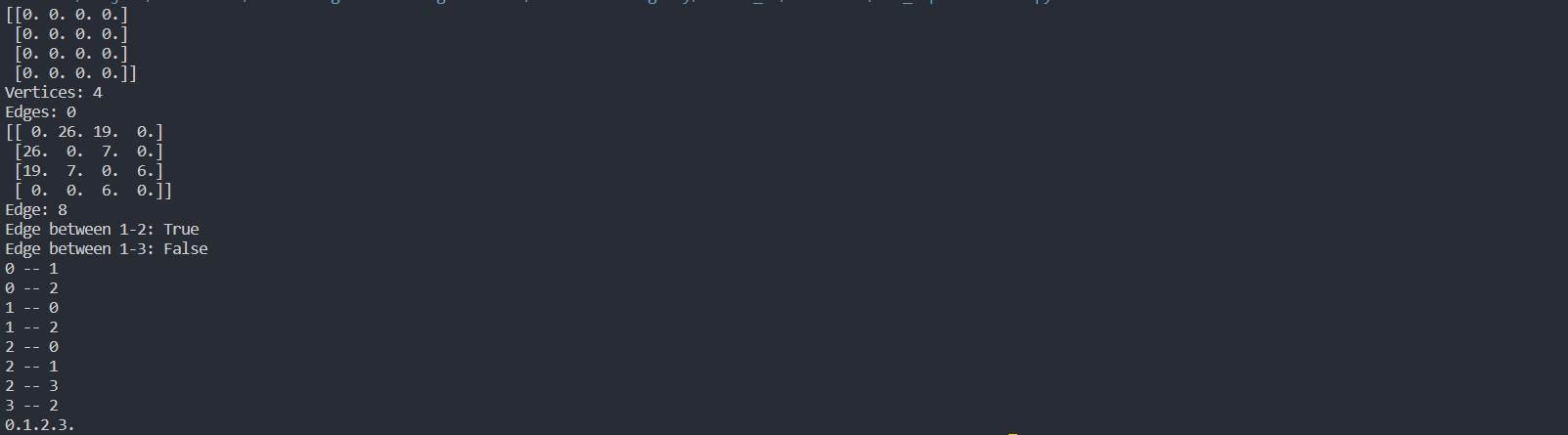
Kết Qủa:



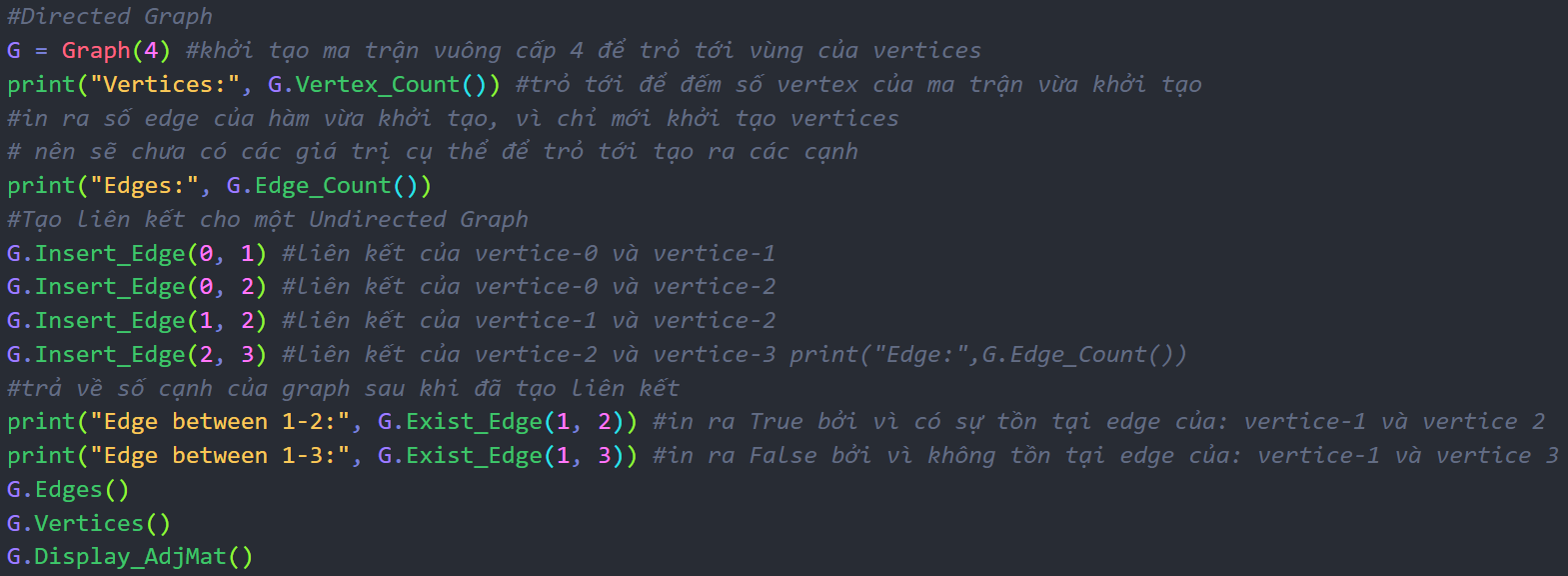
* Weight Undirected Gragh:



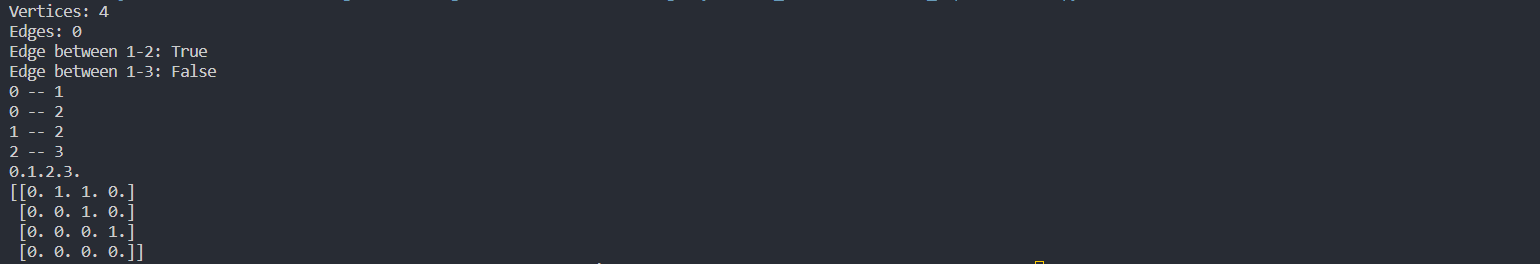
Kết Quả:



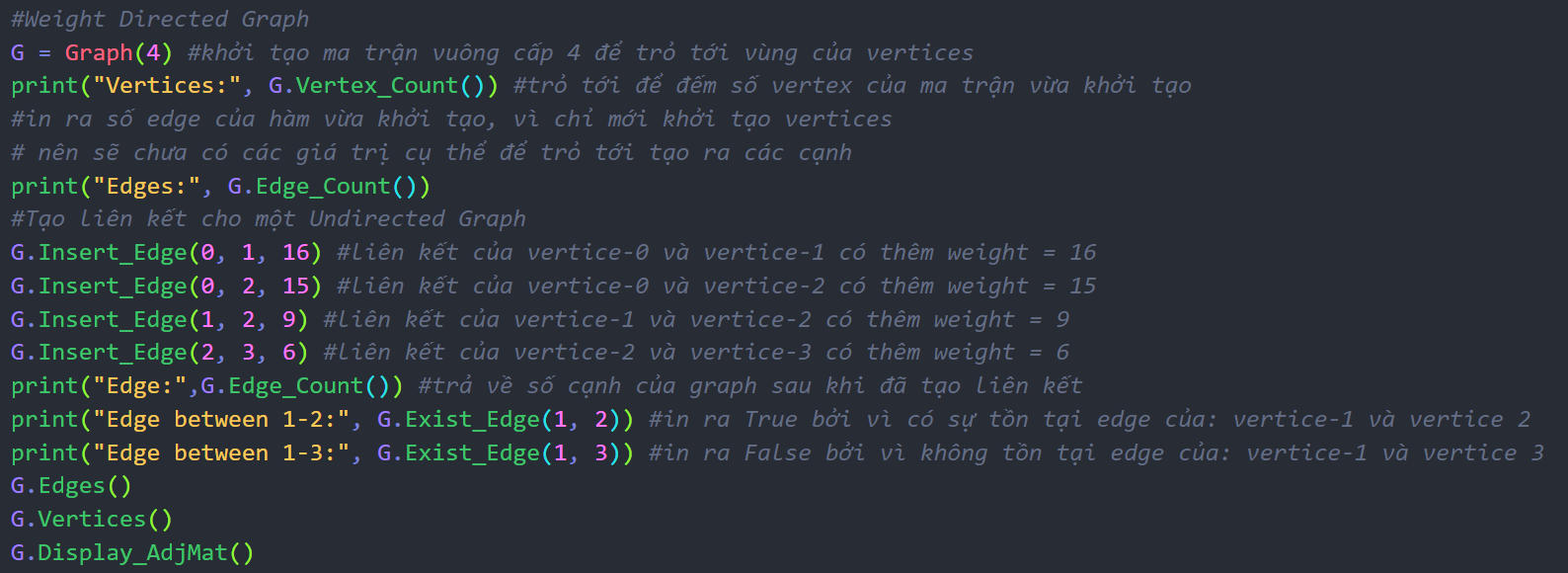
* Directed Graph:



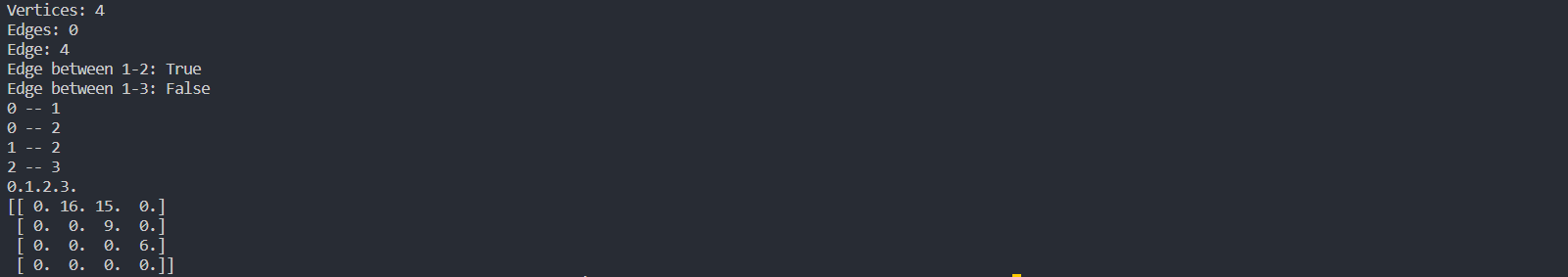
Kết Quả:



* Weight Directed Graph:



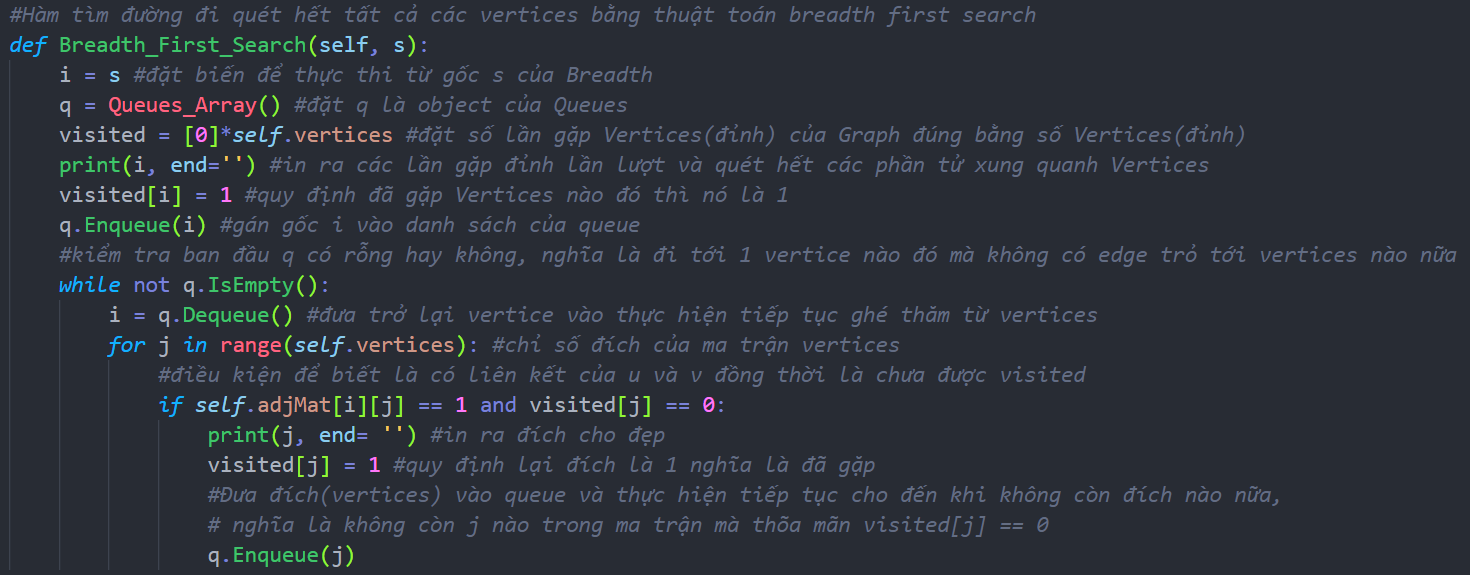
Kết Qủa:



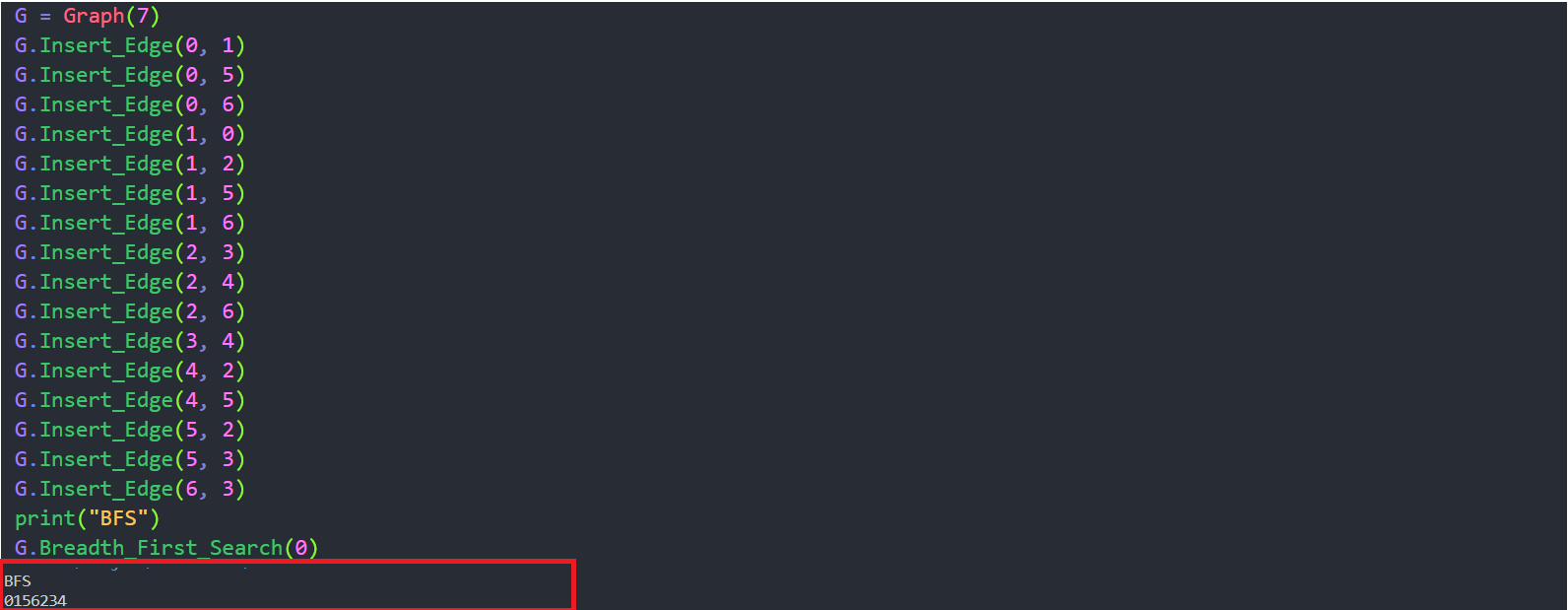
**14.4 Graph Traserval:**

Khái Niệm: Graph Traserval là một thủ tục có hệ thống để quét graph.

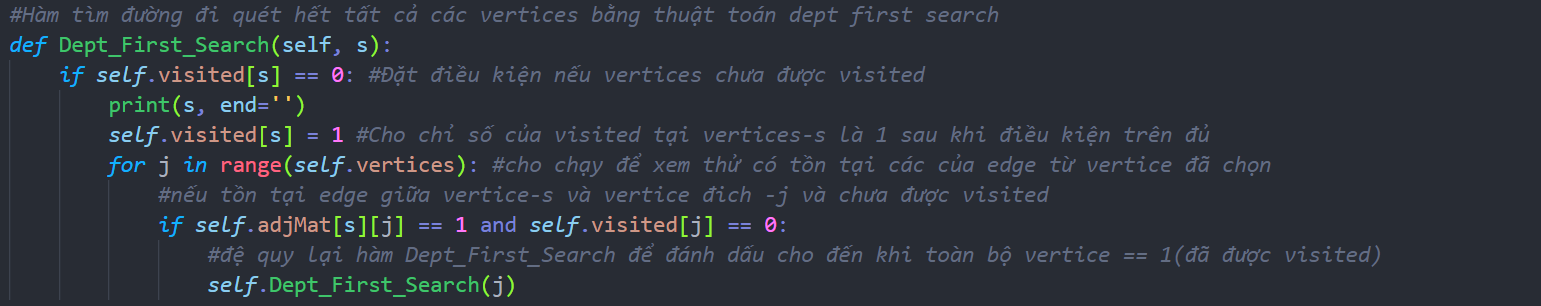
* Exploring: Thực thi toàn bộ Vertice(đỉnh) và Edge(cạnh) của Graph
* Efficient Traversal: Quét được toàn bộ Vertice và Edge có trong Graph
* Thuật Toán Graph Traserval:
* Breadth First Search:



Thực Thi và Kết Qủa: Xét từ vertice-0



* Dept First Search:



Thực Thi và Kết Qủa: Xét từ vertice-0

