**1 Biểu diễn bằng danh sách/ ma trận kề.**

Chúng ta gồm một từ điển. Gồm các cặp key, value. Các key là đỉnh , các đỉnh kề key lưu dưới dạng set().

ADJ = {'A': {'C', 'E', 'D', 'F'},  
 'B': {},  
 'C': {'H'},  
 'D': {'H', 'E'},  
 'E': {'K', 'I'},  
 'F': {'I', 'G'},  
 'G': {},  
 'H': {'K'},  
 'I': {'B', 'K'},  
 'K': {'B'}}

**2 Thao tác duyệt trên đồ thị.**

a)Liệt kê các đỉnh trong đồ thị.

Duyệt qua các key và in ra.

b)Liệt kê các cạnh đồ thị hiển thị dạng danh sách kề.

Duyệt qua key, giá trị trong dict.

Mỗi key, duyệt qua các đỉnh kề nó từ giá trị value.

c)Cho hai đỉnh A, B. Hỏi hai đỉnh A,B có kề nhau không.

Sử dụng try, except .duyệt qua các đỉnh kề trong A. nếu có B thì hai đỉnh đó kề. sau khi duyệt hết đỉnh kề thì A và B không kề. nếu không thể duyệt thì A hoặc B không tồn tại.

d)Cho một đỉnh A. Hỏi danh sách các đỉnh kề với A

Dùng try, except. Thử trả về list. Nếu không trả về được thì trả về None vì A không tồn tại.

**3 Đọc và lưu đồ thị.**

Dòng 1: ghi str(len(G.keys()))

Từ dòng 2 duyệt qua item, duyệt qua value ghi đỉnh và dỉnnh kề trên 1 dòng.

**4Đọc đồ thị từ tập tin.**

Dòng 1 là đọc số đỉnh.

duyệt từ 0 đến số đỉnh hiện có. đọc từng dòng với split(“ “). kí tự đầu là đỉnh, các giá trị kế là đỉnh kề.

**Bài 2 Tìm kiếm đường đi trên đồ thị.**

**1.Tìm kiếm theo chiều rộng (BFS)**

Bỏ start vào s\_open.

khi nào mà s\_open rỗng thì dừng.

cứ lấy phần tử đỉnh của queue và add vào s\_closed.

duyệt các đỉnh kề của phần tử đỉnh queue. nếu mà đỉnh đó chưa có trong path thì add vào s\_open. trường hợp mà đỉnh đó là goal thì dừng

**2.Tìm kiếm theo chiều rộng (DFS)**

tương tự với BFS nhưng thay vì lấy ở đỉnh thì DFS lấy ở cuối