**Câu 1:**

Tìm đường đi từ S đến G theo các phương pháp. Trình bày các bước biến đổi của fringe (stack, queue)

**a. BFS (Breadth-First Search)**

BFS là thuật toán duyệt đồ thị theo chiều rộng, sử dụng hàng đợi (queue) để lưu trữ các đỉnh sẽ được duyệt.

+ Khởi tạo: Đặt đỉnh S vào hàng đợi.

+ Lặp lại:

- Lấy đỉnh đầu hàng đợi ra và kiểm tra.

- Nếu đỉnh đó là G, kết thúc quá trình và trả về đường đi.

- Nếu không, thêm tất cả các đỉnh kề chưa được duyệt vào cuối hàng đợi.

\* Bước 1: Hàng đợi = [S]

\* Bước 2: Hàng đợi = [A, D] (đã duyệt S) => chưa thấy G

\* Bước 3: Hàng đợi = [D, B] (đã duyệt A) => chưa thấy G

\* Bước 4: Hàng đợi = [B, E] (đã duyệt D) => chưa thấy G

\* Bước 5: Hàng đợi = [E, C] (đã duyệt B) => chưa thấy G

\* Bước 6: Hàng đợi = [C, G] (đã duyệt E) => chưa thấy G

\* Bước 7: Hàng đợi = [G] (đã duyệt C) => chưa thấy G

\* Bước 8: Hàng đợi = [] (đã duyệt G) => Thấy G và kết thúc

Kết quả: **S**->**D**->**B**->**C**->**G**

**b. DFS (Depth-First Search)**

DFS là thuật toán duyệt đồ thị theo chiều sâu, sử dụng ngăn xếp (stack) để lưu trữ các đỉnh sẽ được duyệt.

+ Khởi tạo: Đặt đỉnh S vào ngăn xếp.

+ Lặp lại:

- Lấy đỉnh đầu ngăn xếp ra và kiểm tra.

- Nếu đỉnh đó là G, kết thúc quá trình và trả về đường đi.

- Nếu không, thêm tất cả các đỉnh kề chưa được duyệt vào ngăn xếp.

\* Bước 1: Ngăn xếp = [S]

\* Bước 2: Ngăn xếp = [D, A] (đã duyệt S) => chưa thấy G

\* Bước 3: Ngăn xếp = [E, B, A] (đã duyệt D) => chưa thấy G

\* Bước 4: Ngăn xếp = [G, B, A] (đã duyệt E) => chưa thấy G

\* Bước 5: Ngăn xếp = [B, A] (đã duyệt G) => Thấy G và kết thúc

Kết quả: **S**->**D**->**E**->**G**

**Câu 2:**

Tìm đường đi từ A đến G theo các phương pháp. Trình bày các bước biến đổi của fringe (stack, queue)

**a. BFS (Breadth-First Search)**

BFS là thuật toán duyệt đồ thị theo chiều rộng, sử dụng hàng đợi (queue) để lưu trữ các đỉnh sẽ được duyệt.

+ Khởi tạo: Đặt đỉnh S vào hàng đợi.

+ Lặp lại:

- Lấy đỉnh đầu hàng đợi ra và kiểm tra.

- Nếu đỉnh đó là G, kết thúc quá trình và trả về đường đi.

- Nếu không, thêm tất cả các đỉnh kề chưa được duyệt vào cuối hàng đợi.

\* Bước 1: Hàng đợi = [A]

\* Bước 2: Hàng đợi = [B, C] (đã duyệt A) => chưa thấy G

\* Bước 3: Hàng đợi = [C, D] (đã duyệt B) => chưa thấy G

\* Bước 4: Hàng đợi = [D] (đã duyệt C) => chưa thấy G

\* Bước 5: Hàng đợi = [E, F, G] (đã duyệt D) => chưa thấy G

\* Bước 6: Hàng đợi = [F, G] (đã duyệt E) => chưa thấy G

\* Bước 7: Hàng đợi = [G] (đã duyệt F) => chưa thấy G

\* Bước 8: Hàng đợi = [] (đã duyệt G) => Thấy G và kết thúc

Kết quả: **A**->**C**->**D**->**F**->**G**

**b. DFS (Depth-First Search)**

DFS là thuật toán duyệt đồ thị theo chiều sâu, sử dụng ngăn xếp (stack) để lưu trữ các đỉnh sẽ được duyệt.

+ Khởi tạo: Đặt đỉnh S vào ngăn xếp.

+ Lặp lại:

- Lấy đỉnh đầu ngăn xếp ra và kiểm tra.

- Nếu đỉnh đó là G, kết thúc quá trình và trả về đường đi.

- Nếu không, thêm tất cả các đỉnh kề chưa được duyệt vào ngăn xếp.

\* Bước 1: Ngăn xếp = [A]

\* Bước 2: Ngăn xếp = [C, B] (đã duyệt A) => chưa thấy G

\* Bước 3: Ngăn xếp = [D, B] (đã duyệt C) => chưa thấy G

\* Bước 4: Ngăn xếp = [G, F, E, B] (đã duyệt D) => chưa thấy G

\* Bước 5: Ngăn xếp = [F, E, B] (đã duyệt G) => Thấy G và kết thúc

Kết quả: **A**->**C**->**D**->**G**

**Câu 3:**

Tìm đường đi từ S đến G theo các phương pháp. Trình bày các bước biến đổi của fringe (stack, queue)

**a. BFS (Breadth-First Search)**

BFS là thuật toán duyệt đồ thị theo chiều rộng, sử dụng hàng đợi (queue) để lưu trữ các đỉnh sẽ được duyệt.

+ Khởi tạo: Đặt đỉnh S vào hàng đợi.

+ Lặp lại:

- Lấy đỉnh đầu hàng đợi ra và kiểm tra.

- Nếu đỉnh đó là G, kết thúc quá trình và trả về đường đi.

- Nếu không, thêm tất cả các đỉnh kề chưa được duyệt vào cuối hàng đợi.

\* Bước 1: Hàng đợi = [S]

\* Bước 2: Hàng đợi = [F, H] (đã duyệt S) => chưa thấy G

\* Bước 3: Hàng đợi = [H, P] (đã duyệt F) => chưa thấy G

\* Bước 4: Hàng đợi = [P, K] (đã duyệt H) => chưa thấy G

\* Bước 5: Hàng đợi = [K, Q] (đã duyệt P) => chưa thấy G

\* Bước 6: Hàng đợi = [Q, C] (đã duyệt K) => chưa thấy G

\* Bước 7: Hàng đợi = [C, R] (đã duyệt Q) => chưa thấy G

\* Bước 8: Hàng đợi = [R, A] (đã duyệt C) => chưa thấy G

\* Bước 9: Hàng đợi = [A, T] (đã duyệt R) => chưa thấy G

\* Bước 10: Hàng đợi = [T, B] (đã duyệt A) => chưa thấy G

\* Bước 11: Hàng đợi = [B, G] (đã duyệt T) => chưa thấy G

\* Bước 12: Hàng đợi = [G, D] (đã duyệt B) => chưa thấy G

\* Bước 8: Hàng đợi = [D] (đã duyệt G) => Thấy G và kết thúc

Kết quả: **S**->**F**->**P**->**Q**->**R**->**T**->**G**

<S,F,H,P,K,Q,C,R,A,T,B,G>

**b. DFS (Depth-First Search)**

DFS là thuật toán duyệt đồ thị theo chiều sâu, sử dụng ngăn xếp (stack) để lưu trữ các đỉnh sẽ được duyệt.

+ Khởi tạo: Đặt đỉnh S vào ngăn xếp.

+ Lặp lại:

- Lấy đỉnh đầu ngăn xếp ra và kiểm tra.

- Nếu đỉnh đó là G, kết thúc quá trình và trả về đường đi.

- Nếu không, thêm tất cả các đỉnh kề chưa được duyệt vào ngăn xếp.

\* Bước 1: Ngăn xếp = [S]

\* Bước 2: Ngăn xếp = [F, H] (đã duyệt S) => chưa thấy G

\* Bước 3: Ngăn xếp = [P, H] (đã duyệt F) => chưa thấy G, mở S bỏ qua

\* Bước 4: Ngăn xếp = [Q, H] (đã duyệt P) => chưa thấy G,

\* Bước 4: Ngăn xếp = [R, H] (đã duyệt Q) => chưa thấy G,

\* Bước 4: Ngăn xếp = [T, H] (đã duyệt R) => chưa thấy G,

\* Bước 4: Ngăn xếp = [G, H] (đã duyệt T) => chưa thấy G,

\* Bước 5: Ngăn xếp = [H] (đã duyệt G) => Thấy G và kết thúc

Kết quả: **S**->**F**->**P**->**Q**->**R**->**T**->**G**

<S,F,P,Q,R,T,G>