Định nghĩa:

Thuật toán Breadth First Search (BFS) và Depth First Search (DFS) là hai thuật toán tìm kiếm đồ thị rất phổ biến. Trong đó, BFS tìm kiếm theo chiều rộng và DFS tìm kiếm theo chiều sâu. Dưới đây là định nghĩa và code demo cho cả hai thuật toán trong Java.

**Định nghĩa BFS và DFS:**

* BFS (Breadth First Search): Thuật toán này sử dụng hàng đợi (queue) để duyệt qua các đỉnh của đồ thị theo cấp độ. Ban đầu, đỉnh gốc được đưa vào hàng đợi, sau đó ta lần lượt lấy các đỉnh trong hàng đợi ra và đưa vào hàng đợi tất cả các đỉnh kề của nó chưa được duyệt. Việc duyệt sẽ dừng lại khi không còn đỉnh nào trong hàng đợi.

**Triển khai BFS:**

* + Khởi tạo một mảng boolean để đánh dấu các đỉnh đã duyệt (visited) và một hàng đợi (queue) để lưu các đỉnh đang xét.
  + Bắt đầu từ đỉnh gốc (bắt đầu của BFS).
  + Đánh dấu đỉnh gốc là đã duyệt và đưa nó vào hàng đợi.
  + Lặp qua các bước sau cho đến khi hàng đợi rỗng: a. Lấy một đỉnh từ đầu hàng đợi và in nó ra (hoặc thực hiện các thao tác cần thiết trên đỉnh này). b. Lặp qua tất cả các đỉnh kề của đỉnh đang xét. Nếu đỉnh kề chưa được duyệt (chưa được đánh dấu là đã duyệt), đánh dấu nó là đã duyệt và đưa nó vào hàng đợi.

**Giải thích code BFS:**

1. Lớp **Graph**:
   * Lớp **Graph** là một lớp đại diện cho đồ thị. Nó có hai thuộc tính: **V** (số đỉnh của đồ thị) và **adj** (danh sách kề cho mỗi đỉnh). Danh sách kề (**adj**) được biểu diễn dưới dạng một mảng các danh sách liên kết (LinkedList) chứa các đỉnh kề của từng đỉnh trong đồ thị.
   * Phương thức khởi tạo **Graph(int v)**: Đây là hàm khởi tạo của lớp **Graph**. Khi tạo một đối tượng đồ thị, bạn cần chỉ định số lượng đỉnh (**v**) của đồ thị và đồng thời khởi tạo danh sách kề cho mỗi đỉnh là một danh sách liên kết rỗng.
   * Phương thức **addEdge(int v, int w)**: Đây là phương thức dùng để thêm một cạnh vào đồ thị. Nếu có cạnh từ đỉnh **v** tới đỉnh **w**, ta thêm đỉnh **w** vào danh sách kề của đỉnh **v**.
2. Phương thức **BFS(int s)**:
   * Phương thức này thực hiện thuật toán BFS bắt đầu từ đỉnh **s**. Nó tìm kiếm và in ra các đỉnh trong đồ thị mà có thể tiếp cận từ đỉnh **s** theo cách tìm kiếm theo chiều rộng.
   * Biến **visited**: Mảng boolean để đánh dấu các đỉnh đã duyệt. Ban đầu, tất cả các đỉnh đều chưa được duyệt nên mảng này được khởi tạo với giá trị **false**.
   * Hàng đợi **queue**: Là một hàng đợi (queue) dùng để lưu các đỉnh đang xét trong quá trình duyệt BFS.
   * Bắt đầu từ đỉnh **s**:
     + Đánh dấu đỉnh **s** là đã duyệt và đưa nó vào hàng đợi.
     + Trong vòng lặp, lấy một đỉnh từ đầu hàng đợi (đang xét) và in nó ra (hoặc thực hiện các thao tác cần thiết trên đỉnh này).
     + Lặp qua tất cả các đỉnh kề của đỉnh đang xét. Nếu đỉnh kề chưa được duyệt (chưa được đánh dấu là đã duyệt), đánh dấu nó là đã duyệt và đưa nó vào hàng đợi. Điều này đảm bảo các đỉnh được duyệt theo cấp độ.
3. Phương thức **main** trong lớp **BFSExample**:
   * Trong phương thức **main**, tạo một đồ thị gồm 7 đỉnh và thêm các cạnh để tạo thành đồ thị như trong ví dụ.
   * Tiếp theo, gọi phương thức **BFS** trên đối tượng đồ thị **g** với đỉnh bắt đầu là 0. Kết quả sẽ in ra các đỉnh trong đồ thị theo thứ tự của BFS.
   * Kết quả của chương trình khi chạy là: "BFS starting from vertex 0: 0 1 2 3 4 5 6", đây là kết quả của việc duyệt BFS trên đồ thị bắt đầu từ đỉnh 0.

* **DFS (Depth First Search**): Thuật toán này sử dụng ngăn xếp (stack) để duyệt qua các đỉnh của đồ thị theo chiều sâu. Ban đầu, đỉnh gốc được đưa vào ngăn xếp, sau đó ta lấy các đỉnh ra khỏi ngăn xếp và đưa vào ngăn xếp tất cả các đỉnh kề của nó chưa được duyệt. Việc duyệt sẽ dừng lại khi không còn đỉnh nào trong ngăn xếp.

**Triển khai DFS:**

* + Khởi tạo một mảng boolean để đánh dấu các đỉnh đã duyệt (visited).
  + Bắt đầu từ đỉnh gốc (bắt đầu của DFS).
  + Gọi hàm đệ quy DFS với đỉnh gốc và mảng visited.
  + Trong hàm DFS, đánh dấu đỉnh đang xét là đã duyệt và in nó ra (hoặc thực hiện các thao tác cần thiết trên đỉnh này).
  + Lặp qua tất cả các đỉnh kề của đỉnh đang xét. Nếu đỉnh kề chưa được duyệt (chưa được đánh dấu là đã duyệt), gọi đệ quy DFS với đỉnh kề và mảng visited.

Giải thích code:

1. Lớp **Graph**:
   * Lớp **Graph** là một lớp đại diện cho đồ thị. Nó có hai thuộc tính: **V** (số đỉnh của đồ thị) và **adj** (danh sách kề cho mỗi đỉnh). Danh sách kề (**adj**) được biểu diễn dưới dạng một mảng các danh sách liên kết (LinkedList) chứa các đỉnh kề của từng đỉnh trong đồ thị.
   * Phương thức khởi tạo **Graph(int v)**: Đây là hàm khởi tạo của lớp **Graph**. Khi tạo một đối tượng đồ thị, bạn cần chỉ định số lượng đỉnh (**v**) của đồ thị và đồng thời khởi tạo danh sách kề cho mỗi đỉnh là một danh sách liên kết rỗng.
   * Phương thức **addEdge(int v, int w)**: Đây là phương thức dùng để thêm một cạnh vào đồ thị. Nếu có cạnh từ đỉnh **v** tới đỉnh **w**, ta thêm đỉnh **w** vào danh sách kề của đỉnh **v**.
2. Phương thức **DFS(int s)**:
   * Phương thức này thực hiện thuật toán DFS bắt đầu từ đỉnh **s**. Nó tìm kiếm và in ra các đỉnh trong đồ thị mà có thể tiếp cận từ đỉnh **s** theo cách tìm kiếm theo chiều sâu.
   * Phương thức **DFSUtil(int s, boolean[] visited)**: Đây là một phương thức hỗ trợ được sử dụng bởi phương thức **DFS**. Nó thực hiện đệ quy để duyệt các đỉnh kề của đỉnh **s**.
   * Biến **visited**: Mảng boolean để đánh dấu các đỉnh đã duyệt. Ban đầu, tất cả các đỉnh đều chưa được duyệt nên mảng này được khởi tạo với giá trị **false**.
   * Bắt đầu từ đỉnh **s**:
     + Gọi phương thức đệ quy **DFSUtil(s, visited)**.
     + Trong phương thức **DFSUtil**, đánh dấu đỉnh **s** là đã duyệt và in nó ra (hoặc thực hiện các thao tác cần thiết trên đỉnh này).
     + Lặp qua tất cả các đỉnh kề của đỉnh **s**. Nếu đỉnh kề chưa được duyệt (chưa được đánh dấu là đã duyệt), gọi đệ quy phương thức **DFSUtil** với đỉnh kề và mảng **visited**. Quá trình này sẽ tiếp tục cho đến khi không còn đỉnh nào chưa được duyệt.
3. Phương thức **main** trong lớp **DFSExample**:
   * Trong phương thức **main**, tạo một đồ thị gồm 7 đỉnh và thêm các cạnh để tạo thành đồ thị như trong ví dụ.
   * Tiếp theo, gọi phương thức **DFS** trên đối tượng đồ thị **g** với đỉnh bắt đầu là 0. Kết quả sẽ in ra các đỉnh trong đồ thị theo thứ tự của DFS.
   * Kết quả của chương trình khi chạy là: "DFS starting from vertex 0: 0 1 3 4 2 5 6", đây là kết quả của việc duyệt DFS trên đồ thị bắt đầu từ đỉnh 0.