**TÌM KIẾM BFS VÀ DFS**

* Tìm kiếm BFS ( Breadth- First Search ) : Tìm kiếm theo chiều rộng trước và thích hợp cho việc tìm kiếm đường đi ngắn nhất.
* Tìm kiếm DFS ( Depth – First Search ) : Tìm kiếm theo chiều sâu trước và thích hợp cho việc tìm kiếm một giải pháp nhanh chóng.
* Cả hai thuật toán này đều là những thuật toán cơ bản nhất trong lĩnh vực tìm kiếm và duyệt đồ thị. Cả hai thuật toán đều được sử dụng để tìm kiếm trong cấu trúc đồ thị, một tập hợp các đỉnh và cạnh kết nối chúng.
* Tìm kiếm theo chiều rộng:

+ Ý tưởng : BFS tìm kiếm từ đỉnh gốc và theo dõi mỗi đỉnh kề của đỉnh hiện tại trước khi đi sâu vào đỉnh kề tiếp theo. Nó thường được thực hiện bằng cách sử dụng một hàng đợi (queue).

+ Cách hoạt động :

1. Bắt đầu từ một đỉnh gốc
2. Thêm đỉnh gốc vào hàng đợi
3. Lặp lại các bước cho đến khi hàng đợi trống:
   * + - * Lấy đỉnh đầu tiên ra khỏi hàng đợi(Đỉnh này là đỉnh gốc và được thêm vào đầu tiên)
         * Đánh dấu đỉnh này là đã thăm
         * Thêm tất cả các đỉnh kề của đỉnh này vào hàng đợi (Nếu như chưa được thăm và chưa được thêm vào hàng đợi ).
4. Khi không còn đỉnh nào trong hàng đợi thì tìm kiếm kết thúc.

+ Đặc điểm :

+ BFS tìm kiếm theo chiều rộng từ đỉnh gốc

+ BFS tìm kiếm ngắn nhất từ đỉnh gốc đến tất cả các đỉnh khác.

+ Thường được sử dụng để tìm kiếm đường đi ngắn nhất giữa hai đỉnh trong đồ thị không có trọng số hoặc không có trọng số đồng nhất.

* Tìm kiếm theo chiều sâu:

+ Ý tưởng : DFS tìm kiếm đến mức sâu nhất của cây trước khi quay lại và kiểm tra những cành còn lại. Nó thường được thực hiện bằng cách sử dụng đệ quy hoặc stack.

+ Cách hoạt động :

1. Bắt đầu từ một đỉnh gốc
2. Đánh dấu đỉnh này đã được thăm
3. Duyệt qua tất cả các đỉnh kề của đỉnh hiện tại một cách tuần tự.

+ Nếu đỉnh kề chưa được thăm, thăm nó bằng cách lặp lại quá trình này với đỉnh kề làm đỉnh hiện tại

1. Khi không còn đỉnh kề nào chưa được thăm, quay lại đỉnh trước đó và tiếp tục tìm kiếm.
3. Khi không còn đỉnh nào chưa được thăm thì tìm kiếm kết thúc.

+ Đặc điểm :

+ DFS tìm kiếm theo chiều sâu

+ DFS không đảm bảo tìm kiếm đường đi ngắn nhất

+ Thường được sử dụng khi cần tìm kiếm một giải pháp nhanh chóng , nhưng không cần thiết là tối ưu nhất.

==========================================================

DFS : Depth First Search : Thuật toán tìm kiếm theo chiều sâu ưu tiên duyệt xuống xâu nhất có thể trước khi quay lại.Nó cố gắng duyệt đến sâu nhất của đồ thị , dùng nhiều hơn là duyệt theo chiều rộng. Tư tưởng của cái thằng này là nó cứ đi thăm, đi thăm cho đến khi không thể mở rộng được cái thuật toán này nữa thì…. Cái thuật toán DFS này sẽ được code bằng đệ quy.

* Thuật toán này thường được áp dụng để làm các cái bài :

+ Tìm kiếm đường đi hoặc chu trình

+ Kiểm tra tính liên thông của đồ thị

+ Tìm kiếm cốt yếu của đồ thị ( Tức là các các cạnh không thuộc bất kì chu trình nào )

+ Kiểm tra chu trình của đồ thị có hướng

+ Tìm kiếm đỉnh trong đồ thị có hướng

+ Tìm kiếm các thành phần liên thông của đồ thị.

CODE CHÍNH :

* Bắt đầu duyệt từ đỉnh u

DFS (u){

<Tham Dinh u > - In ra hoặc làm gì đó

visited[u] = true; - Đánh dấu là u đã được thăm

- Duyet các đỉnh kề với đỉnh u –

for (v : adj[u] ){

- Nếu đỉnh kề với u chưa được thăm thì thăm -

if(visited[v] == false){

DFS(v);

- Thăm kết thúc khi các đỉnh kề đã đc thăm hết

}

}

}

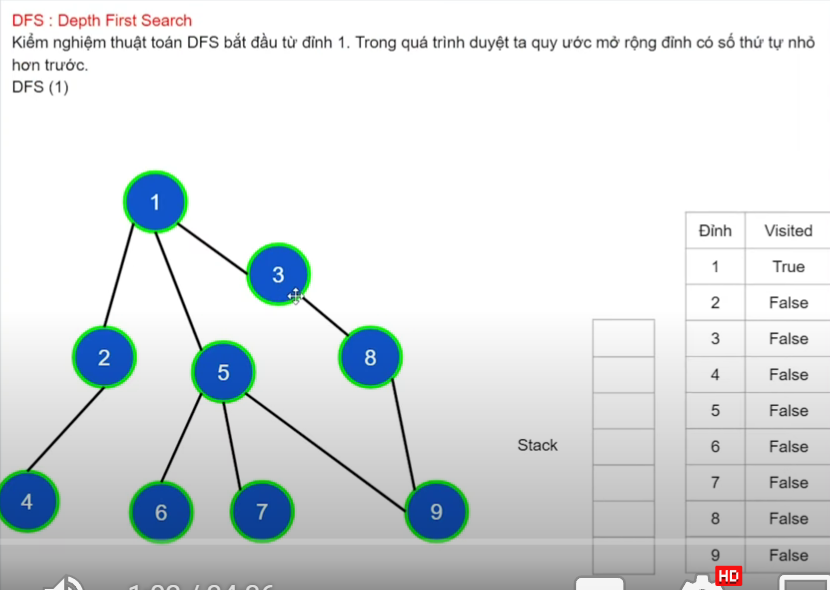
* Mẫu chốt của thuật toán này chỉ có : Thăm một cái đỉnh nào đó. Sau đó sẽ duyệt các cái đỉnh kề với cái đỉnh mà ta vừa thăm và trong quá trình duyệt các cái đỉnh kề đó mà phát hiện ra một cái thằng nào đó chưa được thăm thì ta sẽ tiếp tục gọi lại thuật toán BFS để thăm lại thằng đó.

* Độ phức tạp của thuật toán DFS phụ thuộc vào cách biểu diễn ma trận :
* Đồ thị G = < V, E >

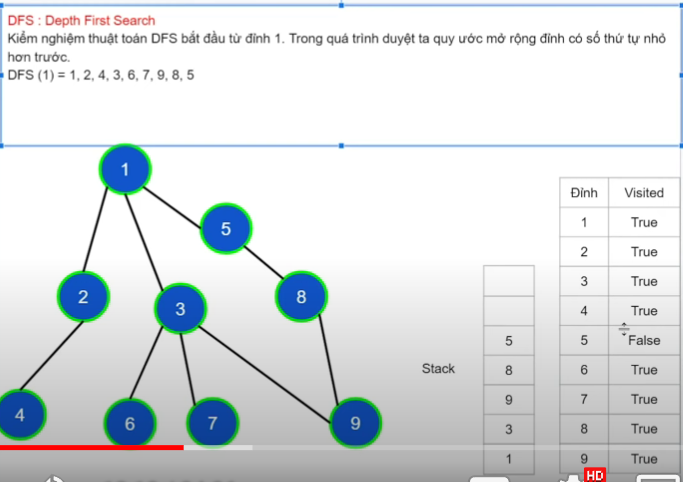
+ Biểu diễn bằng ma trận kề : O (V \* V ) -> Lớn

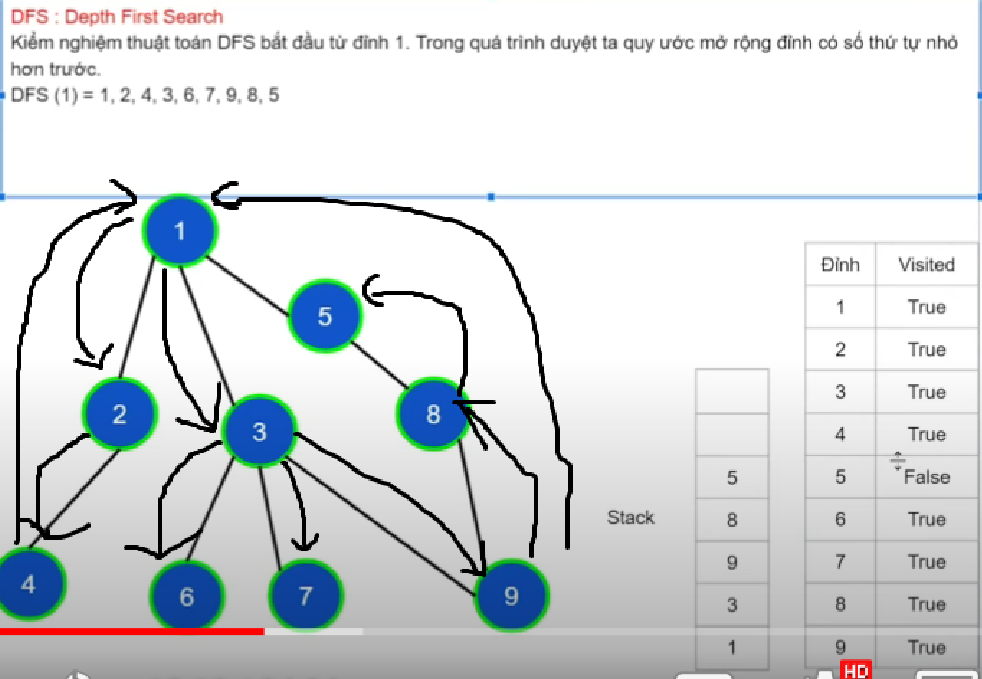
+ Biểu diễn bằng danh sách cạnh : O ( V \* E ) -> Lớn

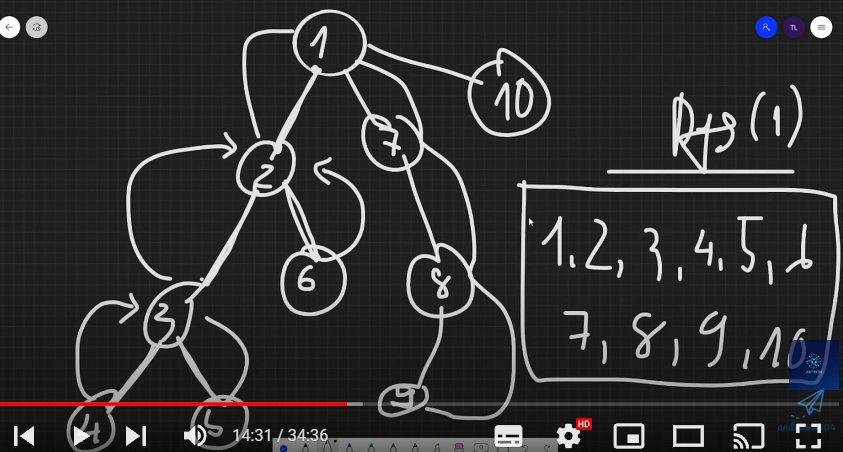
+ Biểu diễn bằng danh sách kề : O (V + E ) -> Độ phức tạp chỉ là V + E thôi nên ta thường sử dụng danh sách kề để cho thuật toán là tốt nhất.



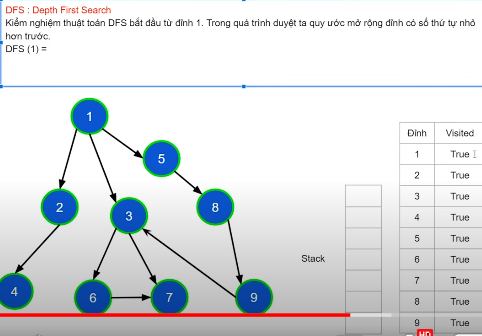
* Duy trì một cái mảng visited để đánh dấu tất cả các cái mảng từ đỉnh 1 cho tới 9 là false để sau khi thăm thằng nào xong thì mới đánh dấu là true.

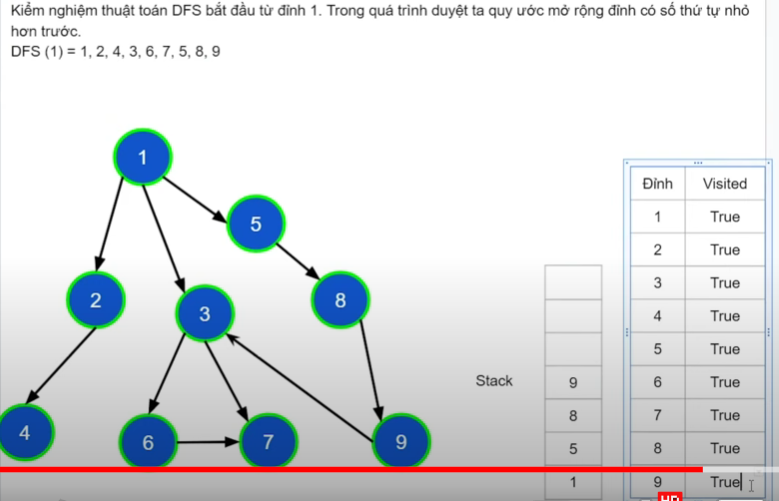




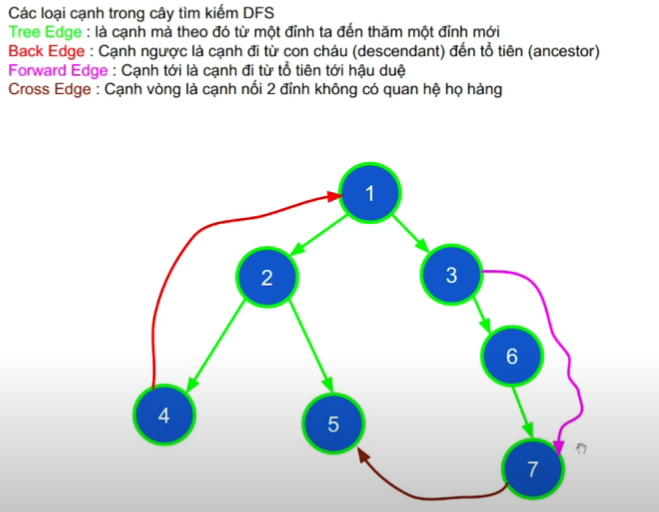


* Đối với đồ thị có hướng thì nếu nó không thăm được thằng nào nữa thì nó sẽ bị đẩy ra khỏi stack.

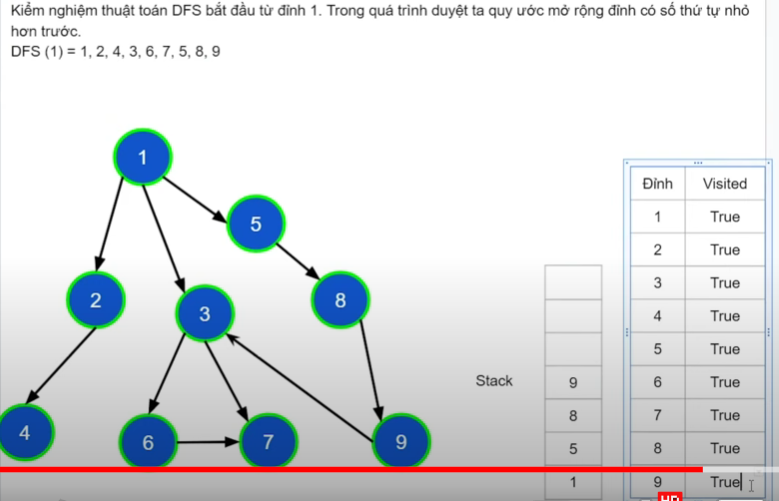




* Thằng nào đến đường cùng rồi hoặc xung quanh các cái thằng đó không còn thằng nào để thăm cả thì nó sẽ được đẩy ra khỏi stack.



* Chung quy lại ta sẽ thử làm biểu diễn một bài bằng thuật toán theo chiều sâu:



* Khi khởi tạo thì ta khởi tạo tất cả các phần tử trong mảng visited là false hết.
* Ban đầu đi từ đỉnh 1, cho nó vào stack. , đánh dấu nó đã được thăm.
* Duyệt qua các cái đỉnh kề với đỉnh 1 là đỉnh 2 , thấy 2 chưa thăm , đánh dấu thăm và cho nó vào stack và tiếp tục thăm những cái thằng giáp với cái thằng 2 là đỉnh 4.
* Thấy thằng 4 chưa được thăm thì đánh dấu đã thăm và cho nó vào stack , tiếp tục thăm thằng kề với 4 nhưng hết mẹ rồi nên coi như đến thằng 4 là đã xong và nó sẽ đẩy ra khỏi stack.
* Ta sẽ quay về 2 và thấy cả 2 và nhưng thằng kề với 2 đã thăm hết rồi nên đẩy 2 ra khỏi stack

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | 7 |  |  |  |  |  | 9 |  |  |  |
| 4 |  |  |  | 6 | 6 | 6 |  |  |  | 8 | 8 | 8 |  |  |
| 2 | 2 |  | 3 | 3 | 3 | 3 | 3 |  | 5 | 5 | 5 | 2 | 2 |  |
| 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |

* Ta sẽ tiếp tục quay về thằng đỉnh 1 thấy thằng 3 chưa thăm nên thăm đỉnh 3 , đánh dấu đã thăm và cho nó vào stack , tiếp tục đến cái đỉnh kề với thằng 3 là đỉnh 6
* Từ đỉnh 6 thấy chưa thăm nên đánh dấu thăm và cho nó vào stack, tiếp tục thăm đến thằng kề với 6 là đỉnh 7 , thấy đỉnh 7 chưa thăm, đánh dấu là đã thăm , cho nó vào stack duyệt tiếp , thấy thằng 7 cục mẹ rồi không còn thằng đéo nào để duyệt nữa nên là dừng , đẩy thằng 7 khỏi stack.
* Thằng 6 cũng hết nên là đẩy ra khỏi stack , thằng 3 cũng hết nên là đẩy ra khỏi stack. .. . . ..

-------------------------------------------------------

* THUẬT TOÁN BFS : Thuật toán tìm kiếm theo chiều rộng : Tư tưởng của thuật toán BFS là tìm kiếm ưu tiên chiều rộng hơn là chiều sâu . Thuật toán sẽ tìm kiếm xung quanh để mở rộng trước khi đi xuống sâu hơn.
* Nếu như thuật toán tìm kiếm theo chiều sâu sử dụng stack thì thuật toán tìm kiếm theo chiều rộng lại sử dụng hàng đợi. Cái hàng đợi thì ta cũng biết rồi, phương trâm của cái hàng đợi như kiểu đợi thanh toán tiền của BIGC vậy, thằng nào vào thanh toán tiền trước thì sẽ được thanh toán tiền trước, nó cũng như kiểu thằng nào được nạp vào hàng đợi trước thì sẽ được cho ra ngoài trước.
* Thứ tự PUSH vào hàng đợi : 1 2 3 4 5
* Thứ tự RA KHỎI hàng đợi : 1 2 3 4 5

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ====> | 6 | 5 | 4 | 3 | 2 | 1 | ====🡺 |

* CODE CHÍNH :

BFS (int u) {

STEP1: KHỞI TẠO

- Khởi tạo một hàng đợi trống

queue = rỗng;

- Đẩy đỉnh u vào hàng đợi

push(queue, u );

- Đánh dấu đỉnh u đã được thăm

visited[u] = true;

STEP2: LẶP KHI MÀ HÀNG ĐỢI VẪN CÒN PHẦN TỬ

while(queue != rỗng) {

- Lấy phần tử đầu hàng đợi -

v = queue.front();

queue.pop(); - Xóa đỉnh khỏi đầu hàng đợi

<Thăm đỉnh v : IN RA >

- Duyệt đỉnh kề với v mà chưa được thăm và đẩy nó vào hàng đợi

for (int x : ke[v]) {

if(visited[x] == false){

push(queue,x);

visited[x] = true;

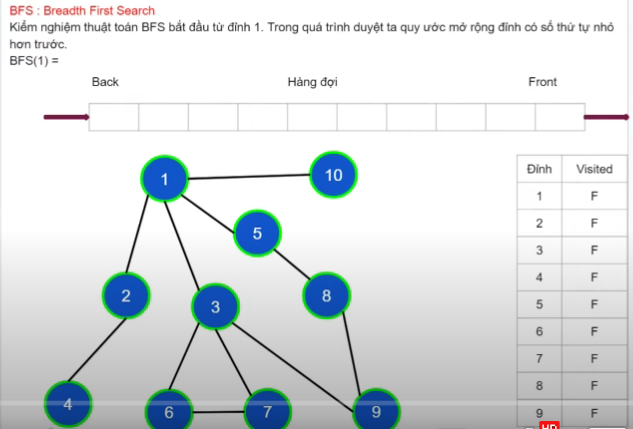
}

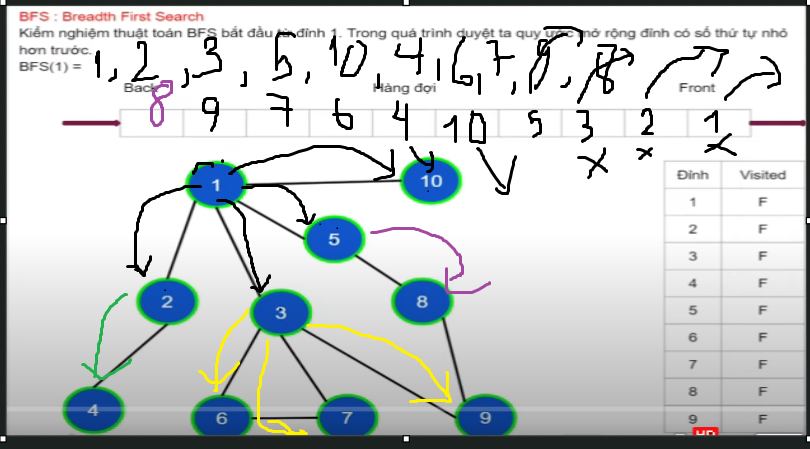
}

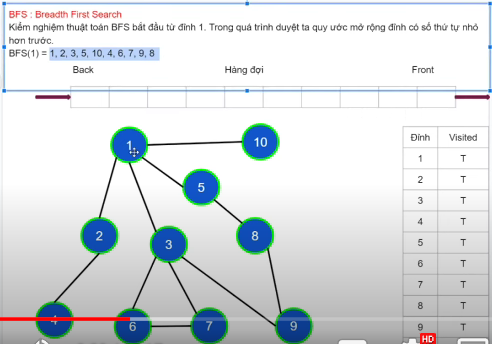
}

}

* Đầu tiên thì ta luôn đẩy thằng đầu vào trong hàng đợi, tiếp theo đó thì ta sẽ dùng một vòng lặp đến khi nào hàng đợi rỗng thì thôi duyệt những cái thằng kề với đỉnh của cái thằng từ trong hàng đợi vừa được lấy ra .
* Nếu như bắt gặp được thằng nào chưa được thăm thì sẽ đẩy nó vào hàng đợi và đánh dấu cho nó đã được thăm
* Với thuật toán BFS thì không có đệ quy gì ở đây cả.







* Đầu tiên , ta đưa thằng 1 vào hàng đợi, đánh dấu là đã được thăm. ( Được một nửa của thuật toán rồi )
* Duyệt đến khi hàng đợi rỗng :

+ Lấy đỉnh 1 ra và xóa khỏi hàng đợi

+ Thăm hết tất cả các đỉnh kề của 1 thấy thằng nào chưa được thăm thì cho nó vào hàng đợi và đánh dấu là đã thăm :

- Ta thăm hết các cái thằng kề với 1 một lượt luôn vì cái vòng for chỉ duyệt qua những thằng kề. Như thế ta sẽ được thứ tự thăm các đỉnh lần lượt là đỉnh 2, đỉnh 3, đỉnh 5, đỉnh 10 và ta phải đánh dấu lần lượt các đỉnh này phải được thăm.

- Tiếp tục thì ta sẽ lấy thằng 2 từ hàng đợi và sau đó sẽ duyệt và xét tiếp các cái thằng kề với thằng 2

- Tương tự như thế cho đến khi mà hàng đợi trống thì thôi.

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  | 1 | => |
|  |  |  |  | 10 | 5 | 3 | 2 | 1 | => |
|  |  |  |  |  | 10 | 5 | 3 | 2 | => |
|  |  |  |  | 4 | 10 | 5 | 3 | 2 | => |
|  |  |  |  |  | 4 | 10 | 5 | 3 | => |
|  |  |  | 9 | 7 | 6 | 4 | 10 | 5 | => |
|  |  |  | 8 | 9 | 7 | 6 | 4 | 10 | => |
|  |  |  |  | 8 | 9 | 7 | 6 | 4 | => |
|  |  |  |  |  |  |  |  |  |  |

