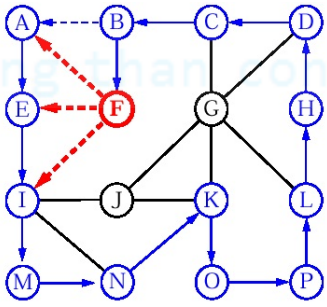
Các phương pháp duyệt đồ thị

Các phương pháp duyệt đồ thị

- Duyệt theo chiều sâu (Depth-First Search)
- Duyệt theo chiều rộng (Breadth-First Search)



Depth-First Search (DFS)

Khái niệm

- **DFS** trên một đồ thị vô hướng cũng giống như khám phá một mê cung với một cuộn chỉ và một thùng sơn đỏ để đánh dấu, tránh bị lạc.
- Ta bắt đầu từ đỉnh **s**, buộc đầu cuộn chỉ vào **s** và đánh dấu đỉnh này là "**đã thăm**". Sau đó ta đánh dấu s là đỉnh hiện hành **u**.

Khái niệm

- Bây giờ, ta đi theo cạnh **(u, v)** bất kỳ.
- Nếu cạnh **(u, v)** dẫn chúng ta đến đỉnh "**đã thăm**" **v**, ta quay trở về u.
- Nếu đỉnh **v** là đỉnh mới, ta di chuyển đến **v** và không quên lăm cuộn chỉ của mình theo. Đánh dấu đỉnh **v** là "**đã thăm**". Đặt **v** thành đỉnh hiện hành và lặp lại các bước trước.

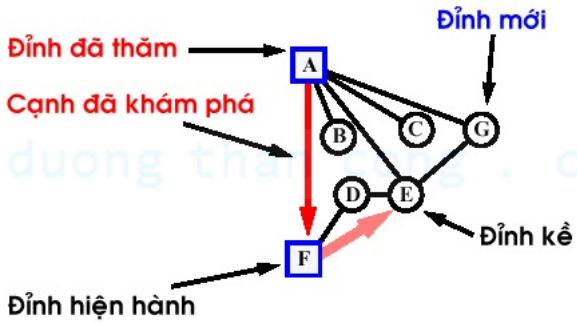
Khái niệm

- Cuối cùng, ta có thể đi đến một điểm mà tại đó tất cả các cạnh kề với nó đều dẫn chúng ta đến các đỉnh "**đã thăm**". Khi đó, ta sẽ quay lui bằng cách cuộn ngược cuộn chỉ và quay lại cho đến khi trở lại một đỉnh kề với một cạnh còn chưa được khám phá. Lại tiếp tục qui trình khám phá như trên.
- Khi chúng ta trở về **s** và không còn cạnh nào kề với nó chưa bị khám phá là lúc **DFS** dừng.

Thuật toán Depth-First Search

```
Algorithm DFS(v);
Input: Moät ñænh v cuûa ñoà thò
Output: Moät caùch gaùn nhaõn cho caùc caïnh ñaõ
  "ñöôïc khaùm phaù" hoaëc "backedge"
for (moïi caïnh e keà vôùi v) do
  if caïnh e chöa ñöôïc khaùm phaù then
     Goïi w laø ñænh khaùc cuûa e
     if ñænh w laø ñænh môùi then
        Gaùn nhaõn e laø "ñöôïc khaùm phaù"
        Goïi ñeä qui DFS(w)
     else
        Gaùn nhaõn e laø "backedge"
```


Thuật toán Depth-First Search

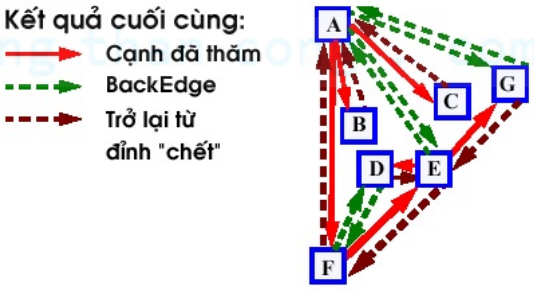


Xác định đỉnh kề trong DFS

- Kết quả của DFS phụ thuộc vào cách ta chọn đỉnh kế tiếp.

Xác định đỉnh kề trong DFS

- Nếu ta bắt đầu tại A và thử cạnh nối đến F, sau đó đến B, rồi đến E, C, cuối cùng là G ta được:



Xác định đỉnh kề trong DFS

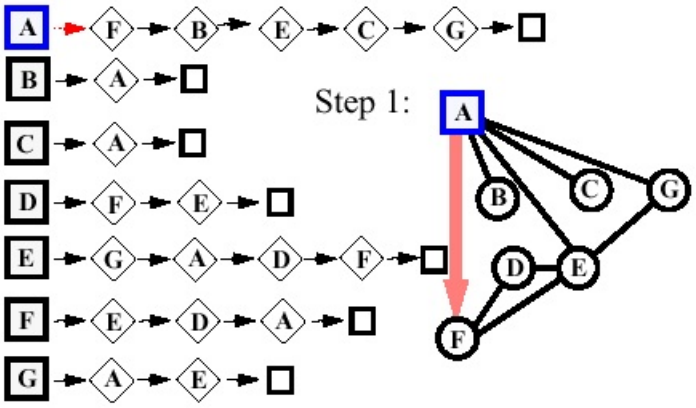
- Nếu cũng bắt đầu từ A nhưng đi theo trình tự:

tập các cạnh đã thăm, backedge và các điểm đệ qui sẽ khác trước. (Hãy tự làm và kiểm chứng).

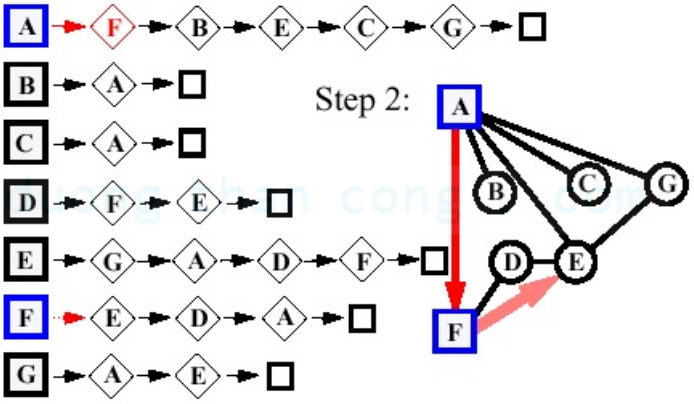
Thuật toán Depth-First Search

- Bây giờ ta sẽ xét từng bước của DFS qua ví dụ trên:

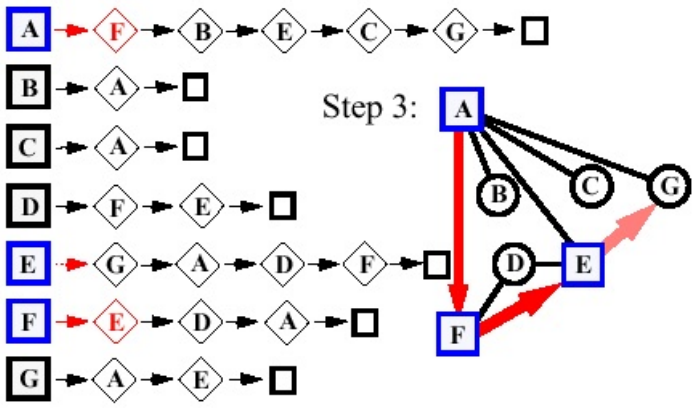
Thuật toán Depth-First Search



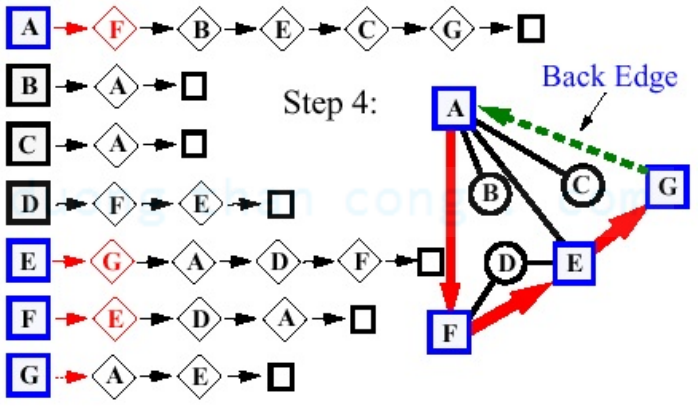
Thuật toán Depth-First Search



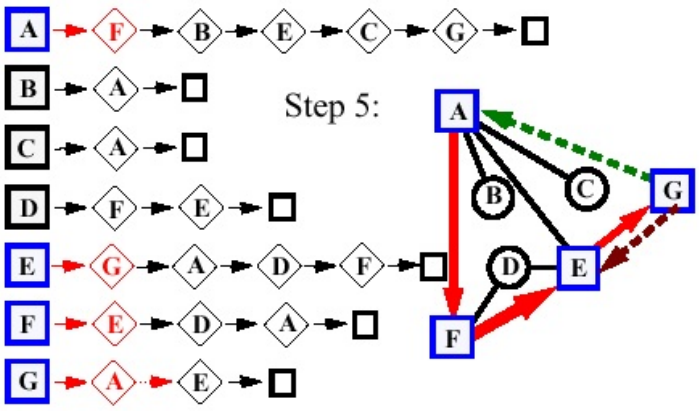
Thuật toán Depth-First Search



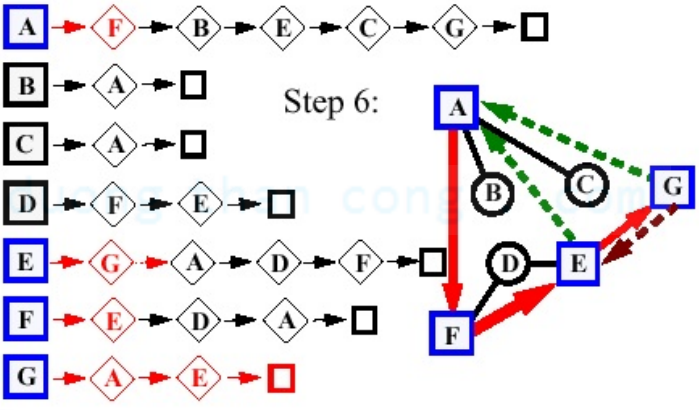
Thuật toán Depth-First Search



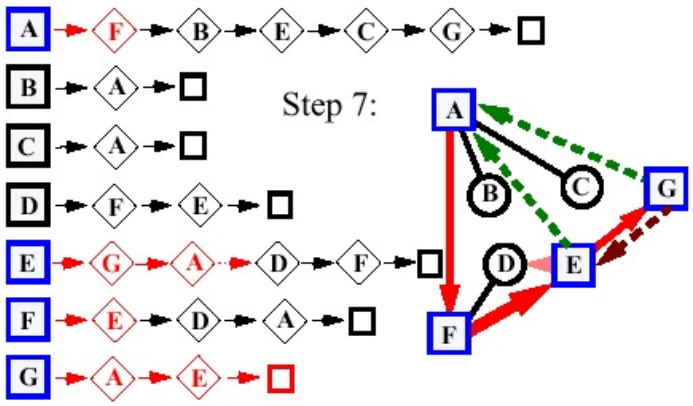
Thuật toán Depth-First Search



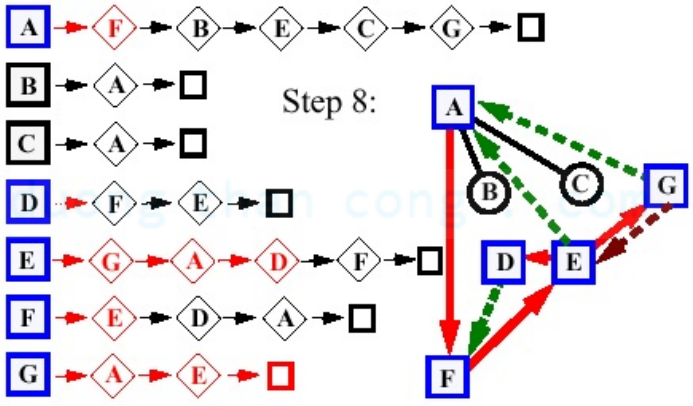
Thuật toán Depth-First Search



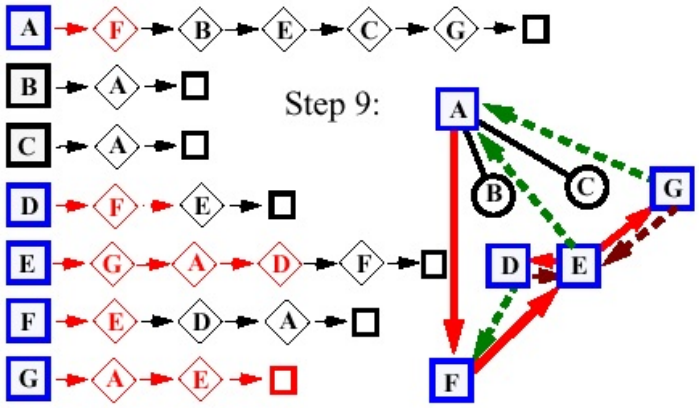
Thuật toán Depth-First Search



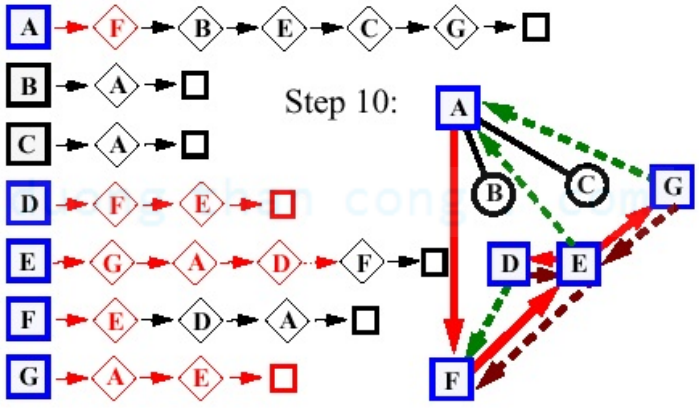
Thuật toán Depth-First Search



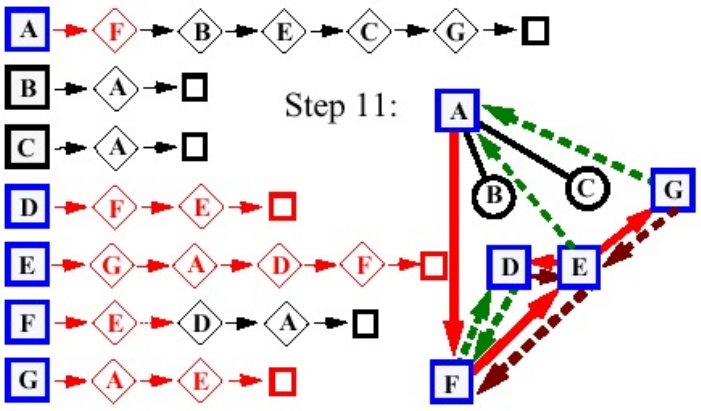
Thuật toán Depth-First Search



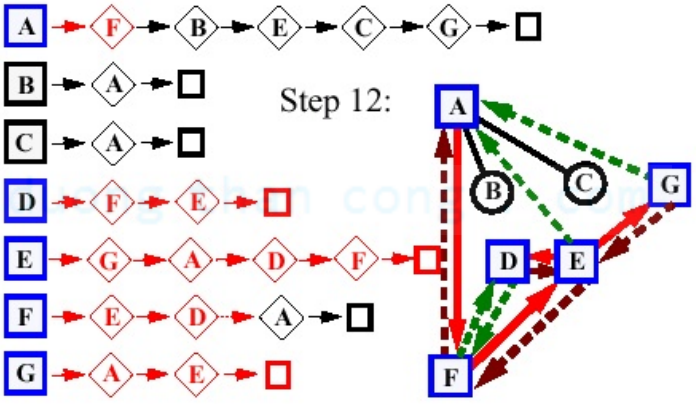
Thuật toán Depth-First Search



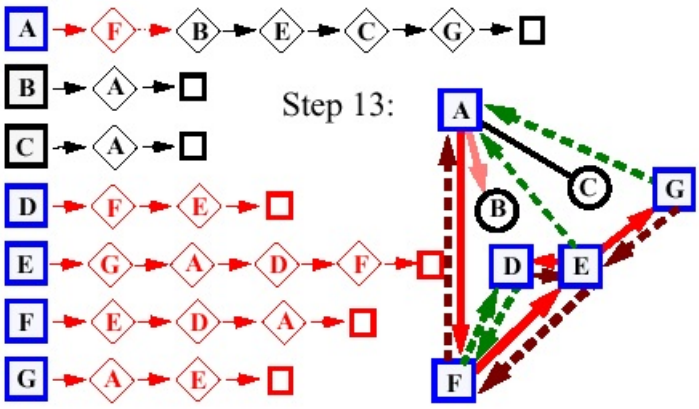
Thuật toán Depth-First Search



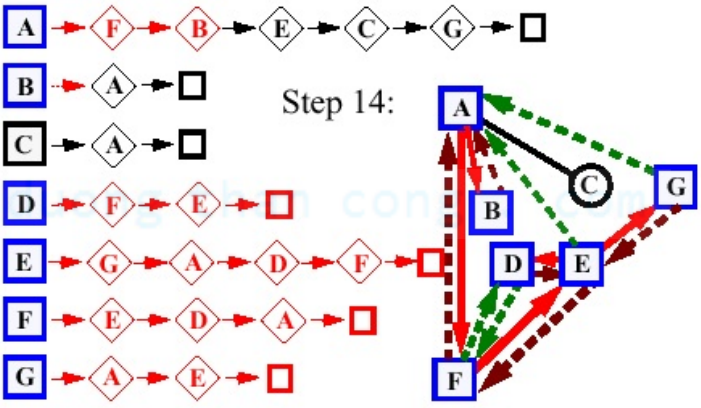
Thuật toán Depth-First Search



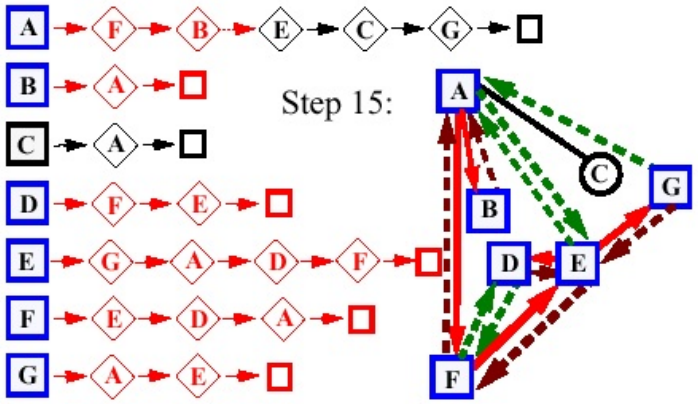
Thuật toán Depth-First Search



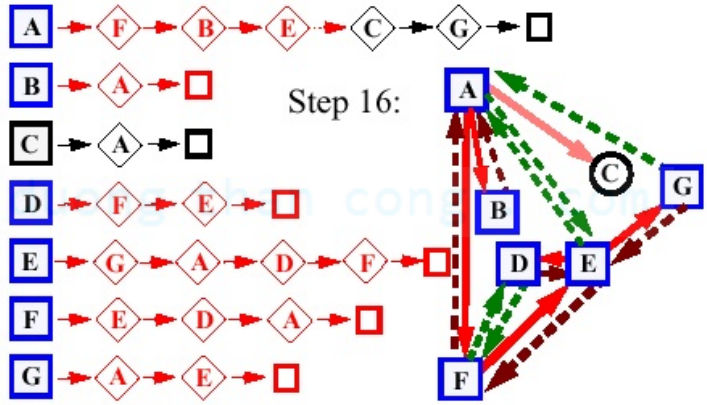
Thuật toán Depth-First Search



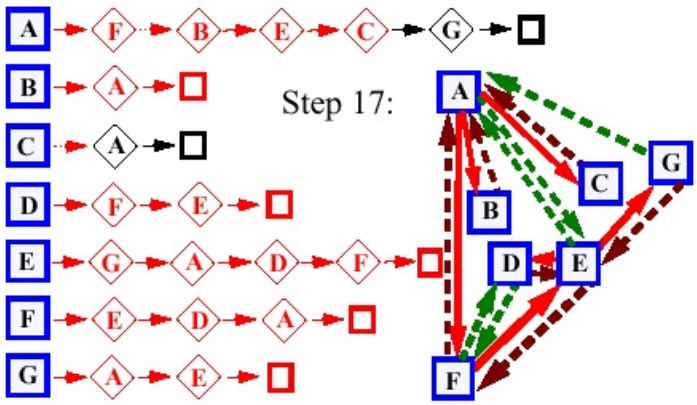
Thuật toán Depth-First Search



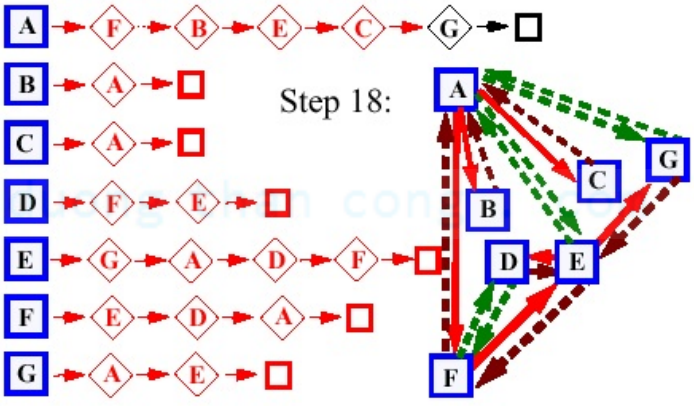
Thuật toán Depth-First Search



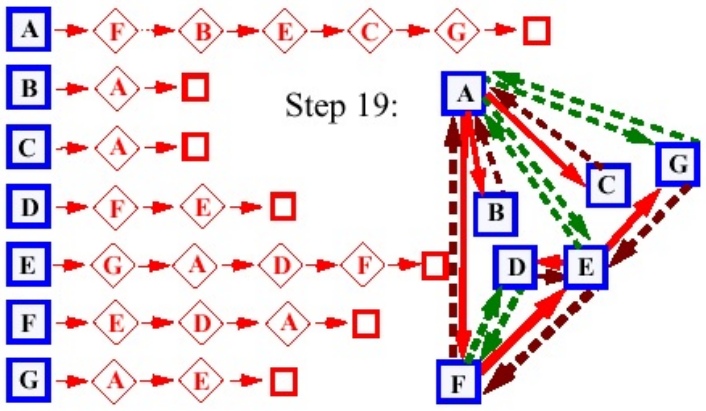
Thuật toán Depth-First Search



Thuật toán Depth-First Search



Thuật toán Depth-First Search



Thuật toán Depth-First Search

- **Mệnh đề:** Gọi G là một đồ thị vô hướng, trên đó ta sẽ thực hiện thao tác DFS với đỉnh bắt đầu là **s** thì:
  1. Phép duyệt sẽ thăm tất cả các đỉnh cùng thành phần liên thông với **s**.
  2. Các cạnh có nhãn "**đã thăm**" sẽ tại ra một cây tối đại của thành phần liên thông chứa **s**.

Thuật toán Depth-First Search

- Chứng minh:
  - Khẳng định 1. là hiển nhiên vì DSF duyệt qua tất cả các đỉnh kề với đỉnh hiện hành. (Có thể chứng minh hoàn chỉnh hơn bằng phản chứng).
  - Khẳng định 2. đúng do ta chỉ đánh dấu các cạnh đi đến một đỉnh mới nên không thể tạo ra chu trình. Như vậy DFS tạo ra một cây. Hơn nữa, DFS thăm tất cả các đỉnh thuộc thành phần liên thông nên cây này là cây tối đại.

Độ phức tạp thuật toán

- Hãy nhớ rằng:
  - DFS được gọi đúng 1 lần ứng với mỗi đỉnh.
  - Mỗi cạnh được xem xét đúng 2 lần, mỗi lần từ một đỉnh kề với nó

Độ phức tạp thuật toán

- Với n_S đỉnh và m_S cạnh thuộc thành phần liên thông chứa **s**, một phép DFS bắt đầu tại s sẽ chay với thời gian $O(n_S + m_S)$ nếu:
  - Đồ thị được biểu diễn bằng CTDL dạng danh sách kề.
  - Đặt nhãn cho một đỉnh là "**đã thăm**" và kiểm tra xem một đỉnh "**đã thăm**" chưa tốn chi phí O(degree).
  - Bằng cách đặt nhãn cho các đỉnh là "**đã thăm**", ta có thể xem xét một cách hệ thông các cạnh kề với đỉnh hiện hành nên ta sẽ không xem xét một cạnh quá 1 lần.

Breadth-First Search (BFS)

Khái niệm

- Cũng giống như DFS, Breadth-First Search (BFS) duyệt qua toàn bộ các đỉnh thuộc một thành phần liên thông của đồ thị và xác định được một cây tối đại của nó với một số thuộc tính hữu ích:

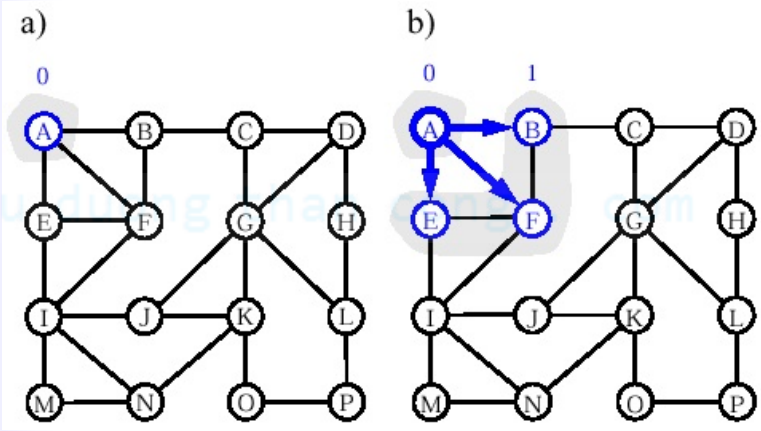
Khái niệm

- Đỉnh xuất phát **s** ở mức 0, và cũng như trong DFS, được xem như một điểm mốc trong quá trình tìm kiếm.
- Ở lượt thứ nhất, cuộn chỉ được mở ra dọc theo chiều dài một cạnh, và tất cả các đỉnh kề với điểm mốc (cách điểm mốc đúng một cạnh) đều được thăm.
- Các đỉnh này được đặt ở mức 1 (các cạnh tương ứng cũng vậy)

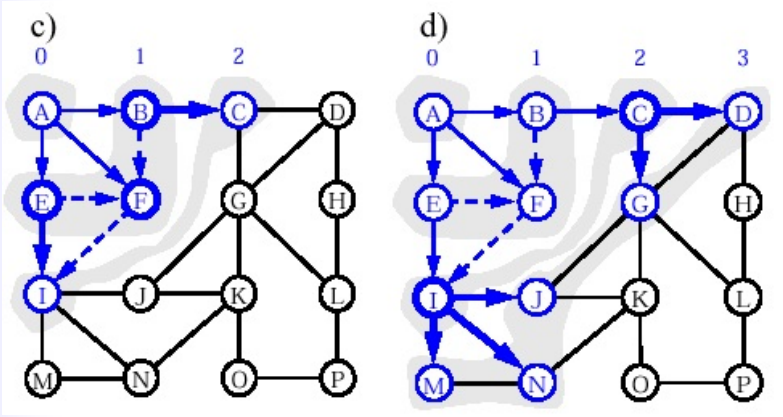
Khái niệm

- Ở lượt thứ hai, tất cả các đỉnh mới cách mốc 2 cạnh sẽ được thăm và được đặt ở mức 2
- Qui trình này tiếp tục cho đến khi tất cả các đỉnh được thăm (được gán vào một mức nào đó).
- Nhãn của mọi đỉnh **v** tương ứng với đường đi qua ít cạnh nhất (ngắn nhất) từ **s** đến **v**.

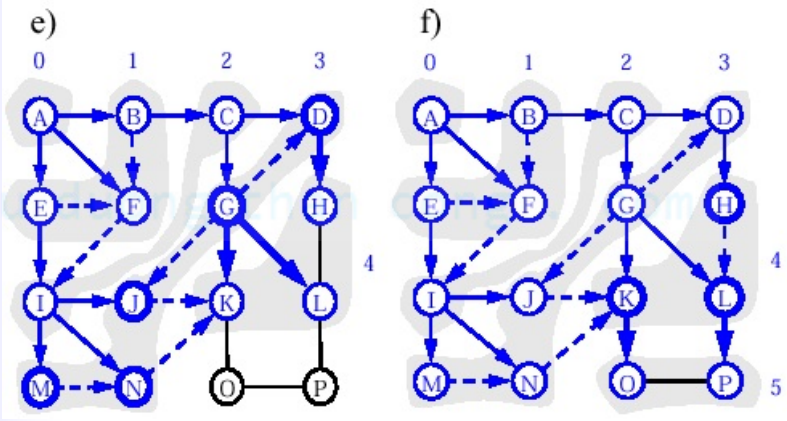
Breadth-First Search (BFS)



Breadth-First Search (BFS)



Breadth-First Search (BFS)



Thuật toán Breadth-First Search

```
Algorithm BFS(v);
Input: Moät ñænh v cuûa ñoà thò
Output: Moät caùch gaùn nhaõn cho caùc caïnh ñaõ "ñöôïc khaùm phaù" hoaëc
   "crossedge"
Khởi tạo hàng đợi L0 để chứa đỉnh s
i ← 0;
while Li ≠ ∅ do
   Tạo Li+1 = ∅
   for mỗi đỉnh v ∈ Li do
      if cạnh e kề với v then
         Gọi w là đỉnh khác của e
      if đỉnh w là đỉnh mới then
         Gán nhãn e là "được khám phá"
         Chèn w vào Li+1
      else
         Gaùn nhaõn e laø "crossedge"
   i ← i + 1
```


Các tính chất của BFS

- Mệnh đề: Gọi G là một đồ thị vô hướng trên đó thực hiện phép duyệt BFS bắt đầu từ đỉnh **s**. Ta có:
  - Phép duyệt sẽ thăm tất cả các đỉnh cùng thành phần liên thông với **s**.
  - Các cạnh có nhãn "**đã thăm**" sẽ tại ra một cây tối đại của thành phần liên thông chứa **s** mà ta sẽ gọi là cây BFS.

Các tính chất của BFS

- Với mỗi cạnh v tại mức i, đường đi trên cây BFS T giữa **s** và **v** qua i cạnh, và bất kỳ đường đi nào khác trên G giữa **s** và **v** dài tối thiểu i cạnh.
- Nếu (u, v) là một cạnh không nằm trên cây BFS, thì mức của **u** và **v** sai lệch không quá 1.

Các tính chất của BFS

- Mệnh đề: Gọi G là một đồ thị vô hướng với **n** đỉnh và **m** cạnh. Một phép duyệt BFS trên *G* tốn chi phí O(n + m). Ngoài ra, tồn tại các thuật toán O(*n* + m) dựa trên nền tảng BFS để giải các bài toán sau:
  - Kiểm tra tính liên thông của *G*.
  - Xác định cây tối đại *G*.
  - Xác định các thành phần liên thông của *G*.
  - Xác định, với mỗi cạnh v ∈ G, số cạnh tối thiểu cần đi từ **s** đến **v**.