Graph Traversals

Visit vertices of a graph G to determine some property:

- Is G connected?
- Is there a path from vertex a to vertex b?
- Does G have a cycle?
- Will removal of a single edge disconnect G?
- If G is directed, what are the strongly connected components?
- If G is the WWW, follow links to locate information.
- Most graph algorithms need to examine or process each vertex and each edge.

Breadth-First Search

Find the shortest path from a source node *s* to all other nodes.

Outputs

- the distance of each vertex from *s*.
- a tree of shortest paths called the *breadth-first tree*.

Idea: Find nodes at distance 0, then at distance 1, then at distance 2, …

Queue & Distance

BFS uses a queue Q to store nodes.

dist(v): an estimate of the distance from s to v.

Initialization:

- $Q == \{ s \}$
- dist(s) == 0
- dist(v) == ∞ for every node other than s

Overview of the Algorithm

- ♦ Dequeue a node u.

- ♦ For each neighbor v of u that is being discovered the first time,

    - ♠ $\text{dist}(v) = \text{dist}(u) + 1$.

    - ♠ enqueue v.

Color Map

Keeps track of the status of nodes.

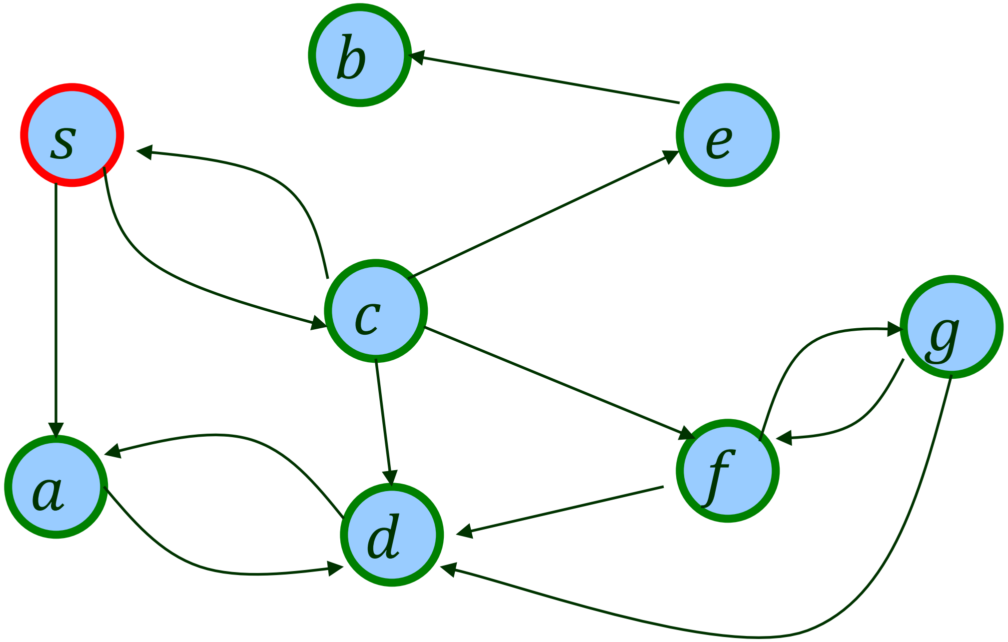
- color(v) == **green**: v is undiscovered and unprocessed
- color(v) == **red**: v has been discovered (in Q) but not processed.
- color(v) == **black**: v has been discovered and processed (no longer in Q).

The BFS Algorithm

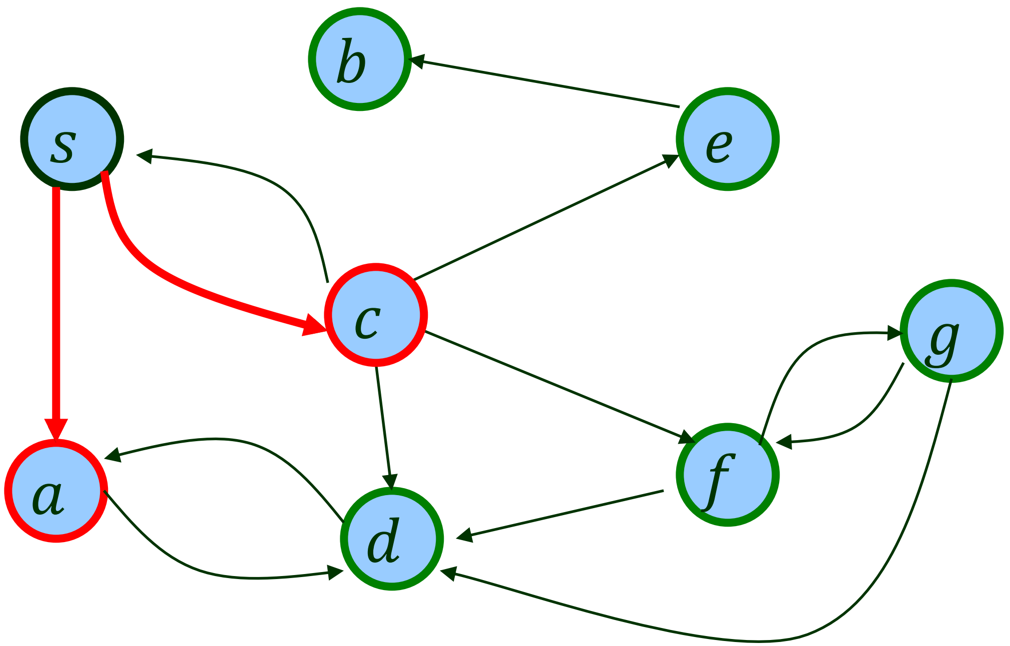
```
BFS(G, s):
    let Q be an empty queue
    foreach node v in G except s
            color(v) = green
            dist(v) = ∞
            pred(v) = null
    color(s) = red
    dist(s) = 0
    pred(s) = null
    Q.enqueue(s)

    while (!Q.isEmpty())
            let u = Q.front()
            foreach neighbor v of u
                 if color(v) == green
                      color(v) = red
                      dist(v) = dist(u) + 1
                      pred(v) = u
                      Q.enqueue(v)
            Q.dequeue()
            color(u) = black
    return dist
```

predecessor of v on the path from s

A BFS Example

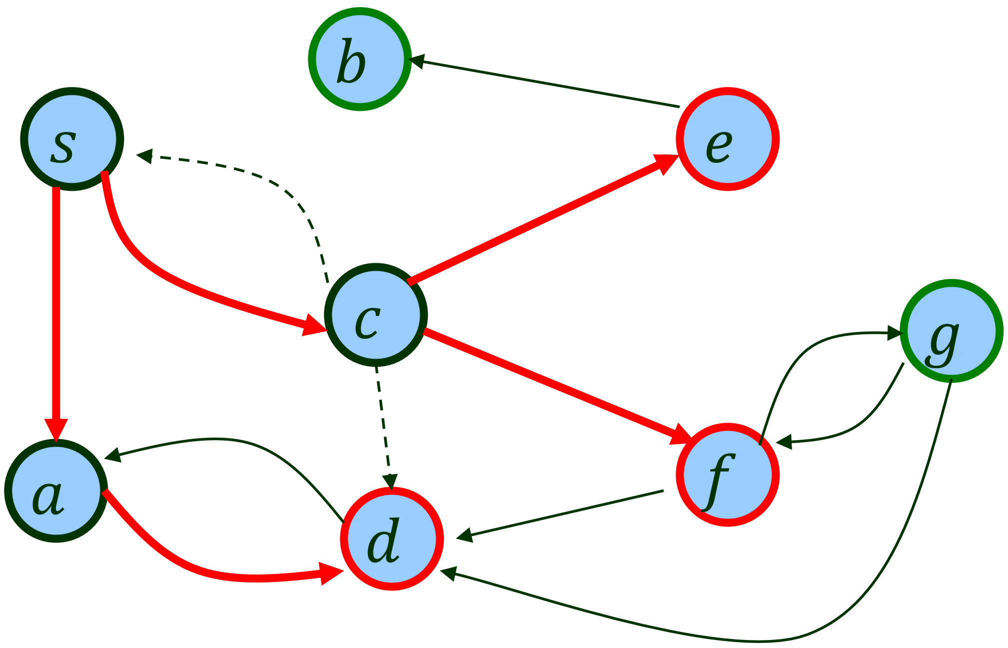


$$Q = \langle\, s \,\rangle$$



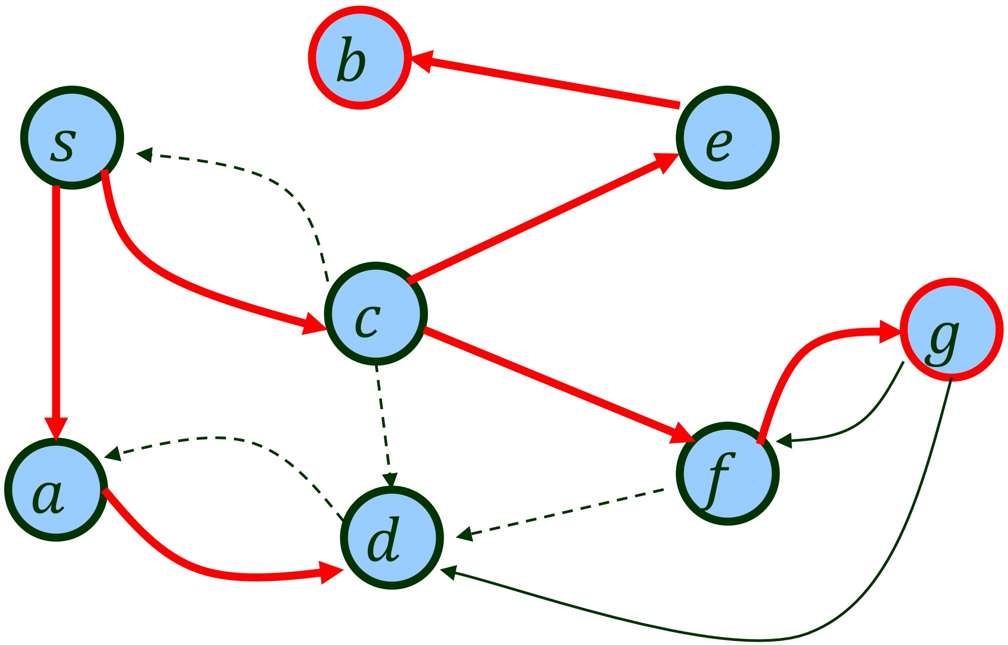
$Q \;= \langle\, a, c \,\rangle$ ⟸ frontier of exploration

Visualized as many *simultaneous explorations* starting from *s* and spreading out independently.



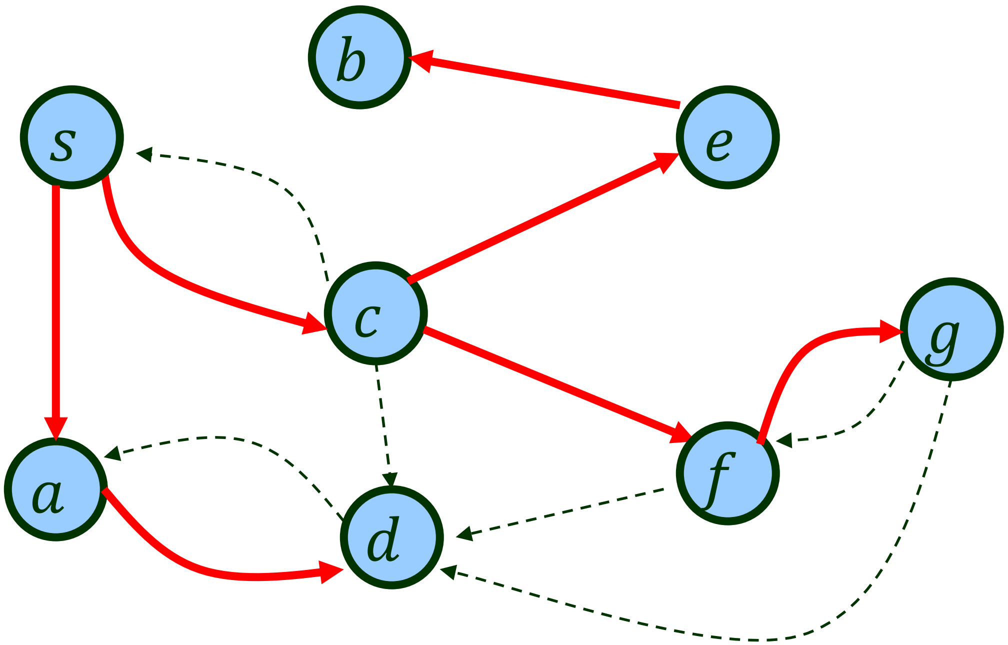
Dashed edges were explored but had resulted in the discovery of no new vertices.

$$Q = \langle c, d \rangle \Longrightarrow \langle d, e, f \rangle \Longrightarrow \langle e, f \rangle$$



$\langle\, e, f \,\rangle \Longrightarrow \langle\, f, b \,\rangle \Longrightarrow \langle\, b, g \,\rangle$

The Finish

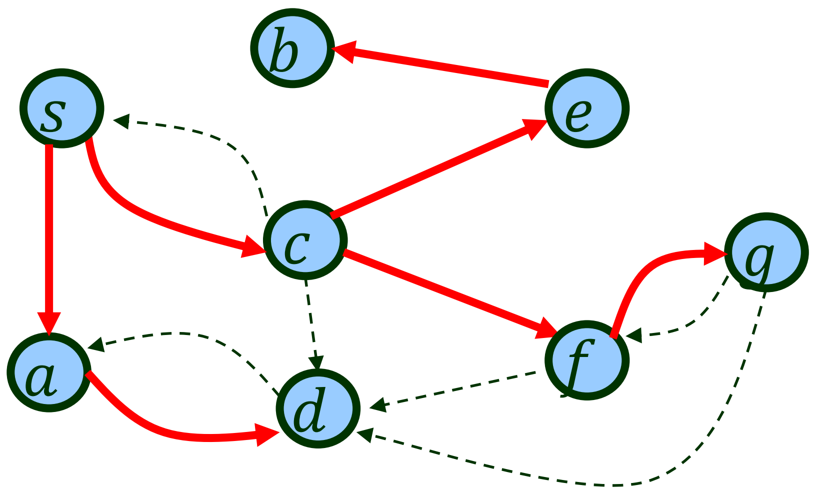


Evolution of Q (with some intermediate statuses omitted)

$\langle s \rangle \Rightarrow \langle a, c \rangle \Rightarrow \langle c, d \rangle \Rightarrow \langle d, e, f \rangle \Rightarrow \langle e, f \rangle$

$\Rightarrow \langle f, b \rangle \Rightarrow \langle b, g \rangle \Rightarrow \langle g \rangle \Rightarrow \varnothing$

Predecessor Table

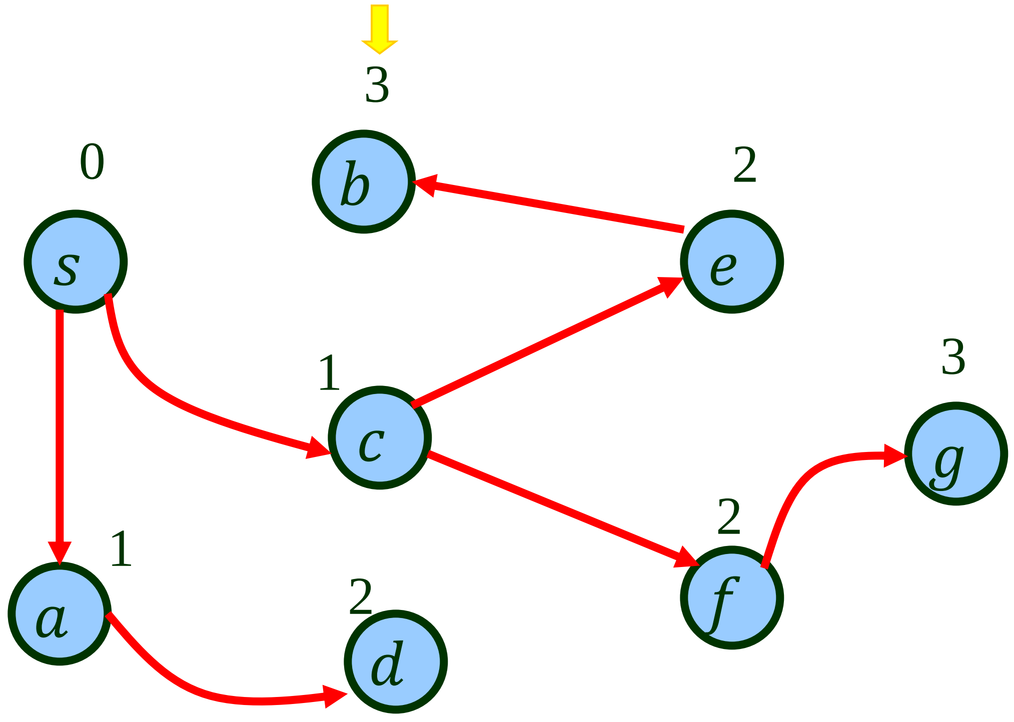


v	s	a	b	c	d	e	f	g
pred(v)	null	s	e	s	a	c	c	f

Breadth-First Tree

Constructed from the predecessor table.

dist(b): shortest distance from s to b



visited vertices = 1 + # tree edges

$\leq$ 1 + # explored edges.

Running Time

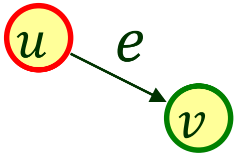
$O(|V| + |E|)$ if G is represented using adjacency lists.

- Every vertex is inserted at most once into Q.

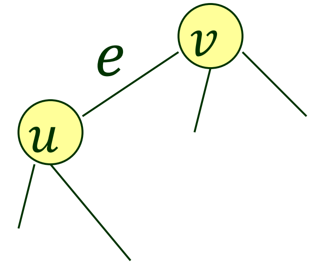
  \# inserted vertices ≤ 1 + # scanned edges

  first vertex

- \# edge scans



  - $\leq\ |E|$ for a directed graph
  - $\leq\ 2\,|E|$ for an undirected graph



$O(|V|^2)$ if represented as an adjacency matrix.