BFS & Level Order Traversal

Charles Aunkan Gomes
Lecturer, Dept. of CSE
United International University
charles@cse.uiu.ac.bd

Graph Search

- Given: a graph G = (V, E), directed or undirected
- Goal: methodically explore every vertex and every edge
- Ultimately: build a tree on the graph
  - Pick a vertex as the root
  - Choose certain edges to produce a tree
  - Note: might also build a *forest* if graph is not connected

- There are two standard graph traversal techniques:
  - Breadth-First Search (BFS)
  - Depth-First Search (DFS)

Breadth-First Search

- "Explore" a graph, turning it into a tree
  - One vertex at a time
  - Expand frontier of explored vertices across the *breadth* of the frontier

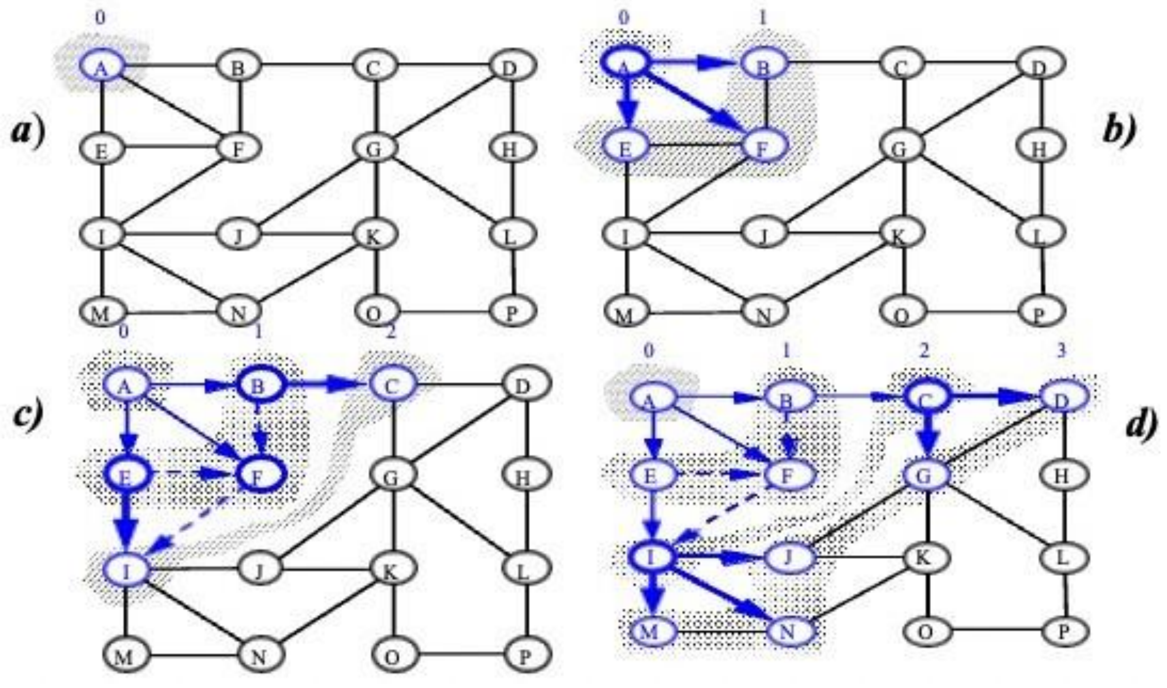
- Builds a tree over the graph
  - Pick a *source vertex* to be the root
  - Find ("discover") its children, then their children, etc.

Breadth-First Search

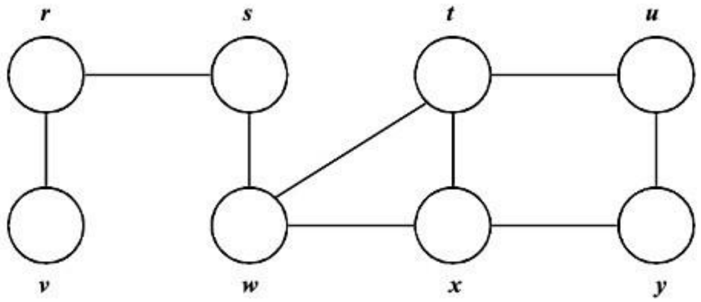
- Again will associate vertex "colors" to guide the algorithm
  - White vertices have not been discovered
    - All vertices start out white
  - Grey vertices are discovered but not fully explored
    - They may be adjacent to white vertices
  - Black vertices are discovered and fully explored
    - They are adjacent only to black and grey vertices

- Explore vertices by scanning adjacency list of grey vertices

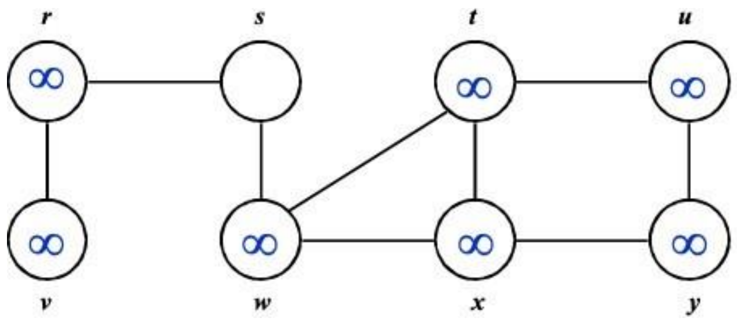
BFS – Graphical Representation



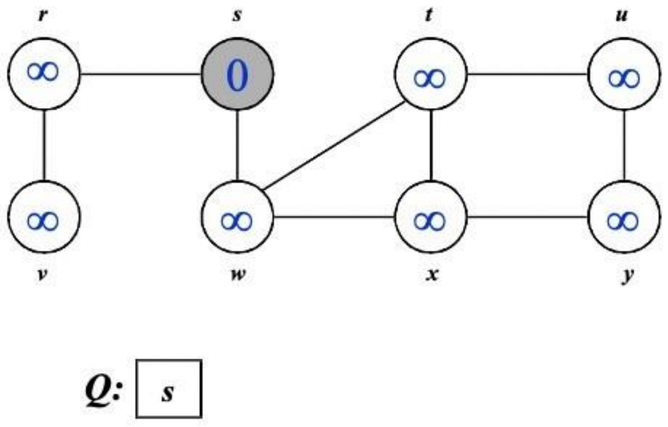
BFS – Example



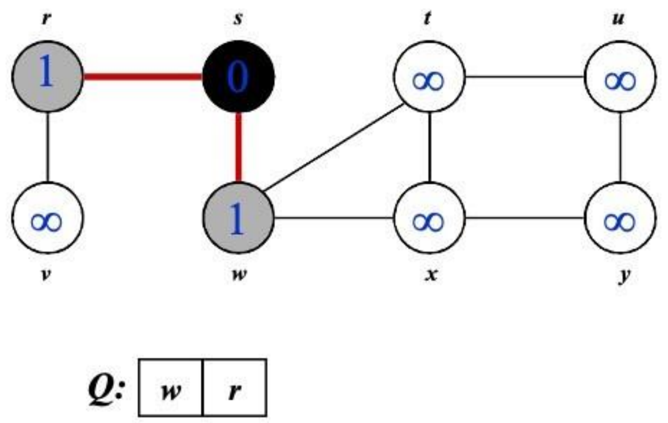
BFS – Example



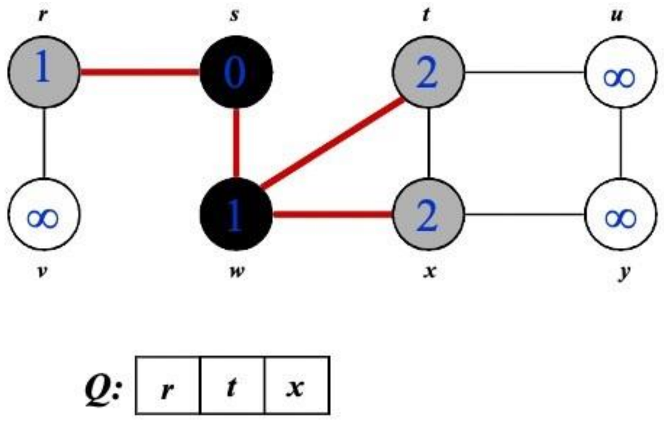
BFS – Example



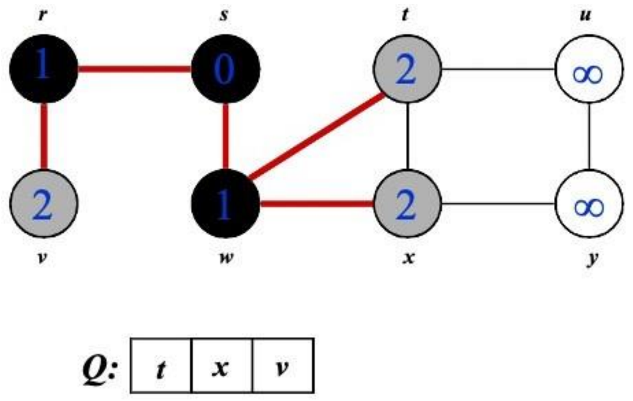
BFS – Example



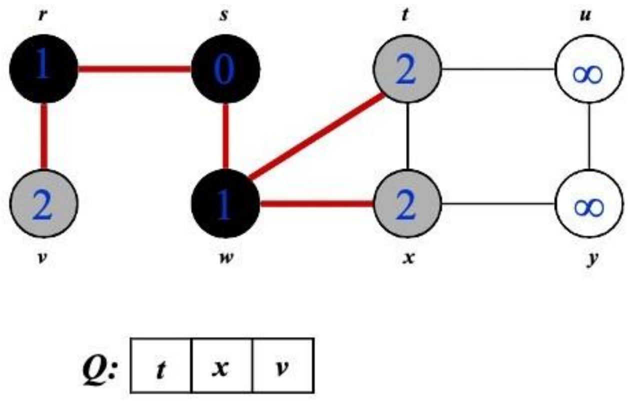
BFS – Example



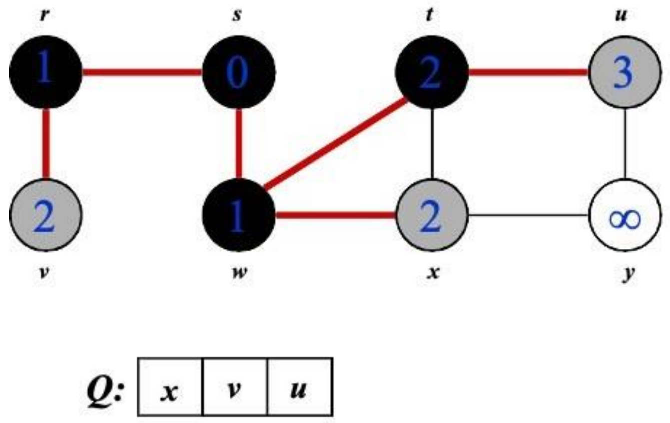
BFS – Example



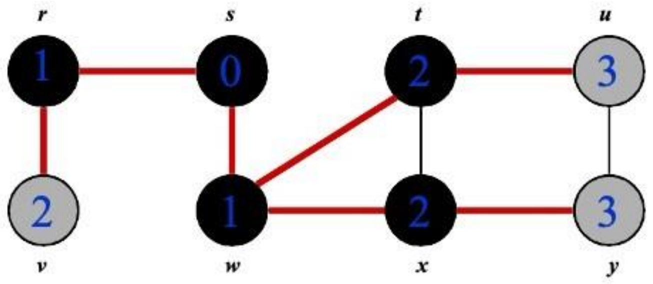
BFS – Example



BFS – Example

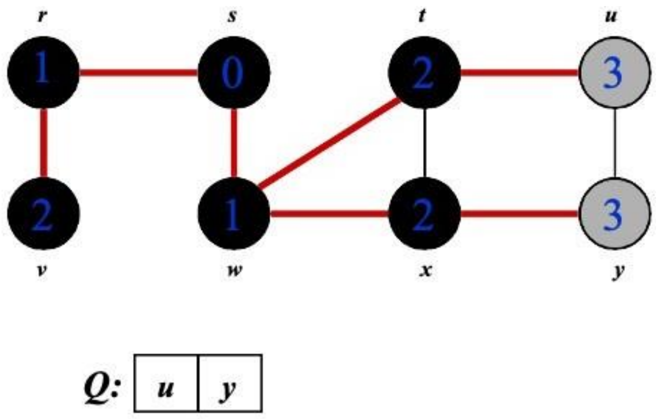


BFS – Example

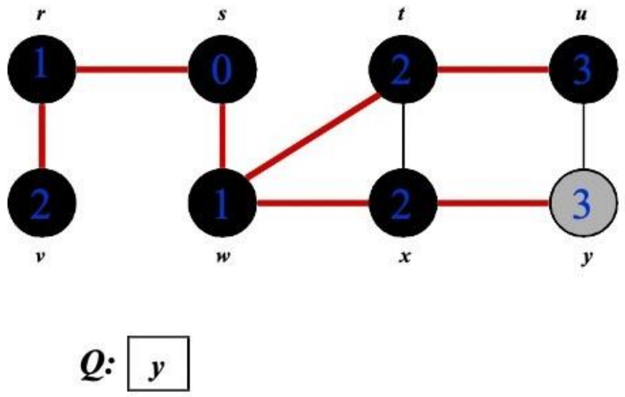


Q:	*v*	*u*	*y*

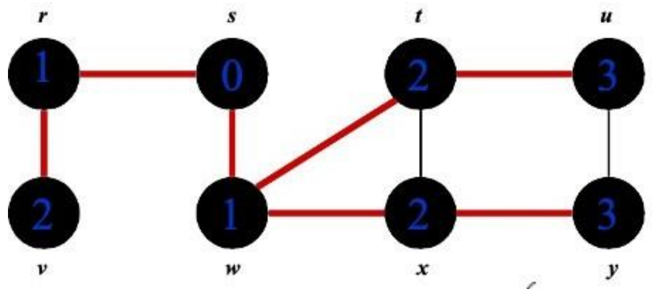
BFS – Example



BFS – Example

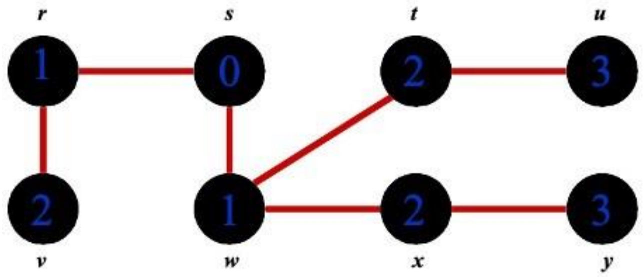


BFS – Example



Q: ø

BFS – Example



Output: BFS Spanning Tree

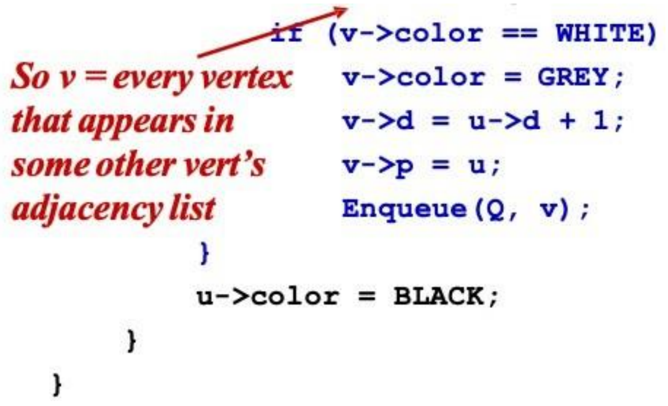
BFS – Code

```
BFS($G, s$)
for each vertex $u \in V[G] - \{s\}$
    do $color[u] \leftarrow$ WHITE
       $d[u] \leftarrow \infty$
       $\pi[u] \leftarrow$ NIL
$color[s] \leftarrow$ GRAY
$d[s] \leftarrow 0$
$\pi[s] \leftarrow$ NIL
$Q \leftarrow \emptyset$
ENQUEUE($Q, s$)
while $Q \neq \emptyset$
    do $u \leftarrow$ DEQUEUE($Q$)
       for each $v \in Adj[u]$
           do if $color[v] =$ WHITE
                 then $color[v] \leftarrow$ GRAY
                      $d[v] \leftarrow d[u] + 1$
                      $\pi[v] \leftarrow u$
                      ENQUEUE($Q, v$)
       $color[u] \leftarrow$ BLACK
```

```
BFS(G, s) {
  initialize vertices;
  Q = {s};
  while (Q not empty) {
    u = RemoveTop(Q);
    for each v ∈u->adj {
     if(v->color == WHITE)
          v->color = GRAY;
          v->d = u->d + 1;
          v->p = u;
          EnQueue(Q,v);
    }
    u->color = BLACK;
  }
```

BFS – Code

```
BFS(G, s) {
  initialize vertices;
  Q = {s};
  while (Q not empty) {
    u = RemoveTop(Q);
    for each v ∈u->adj {
```



```
        if (v->color == WHITE)
          v->color = GREY;
          v->d = u->d + 1;
          v->p = u;
          Enqueue(Q, v);
      }
      u->color = BLACK;
  }
}
```

So v = every vertex that appears in some other vert's adjacency list

Touch every vertex: O(V)

u = every vertex, but only once

(Why?)

What will be the running time?

Total running time:

$O(V + \Sigma(\text{degree}(v))) = O(V+E)$

BFS – Code again

```
BFS(G, s) {
  initialize vertices;
  Q = {s};
  while (Q not empty) {
    u = RemoveTop(Q);
    for each v ∈ u->adj {
        if (v->color == WHITE)
          v->color = GREY;
          v->d = u->d + 1;
          v->p = u;
          Enqueue(Q, v);
    }
    u->color = BLACK;
  }
}
```

Touch every vertex: O(V)

u = every vertex, but only once
(Why?)

So v = every vertex that appears in some other vert's adjacency list

What will be the storage cost in addition to storing the tree?

Total space used:
$O(V + \Sigma(\text{degree}(v))) = O(V + E)$

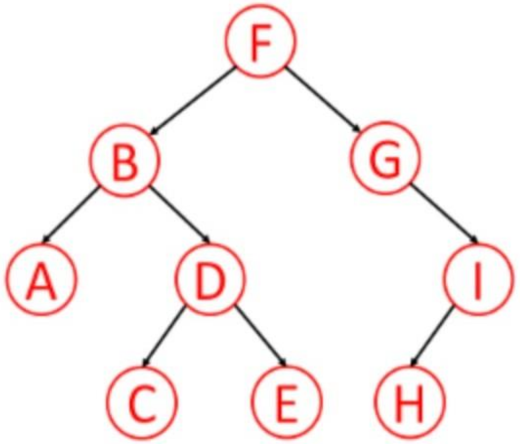
BFS – Properties

- BFS calculates the *shortest-path distance* to the source node
  - Shortest-path distance ⍰(s, v) = minimum number of edges from s to v, or ⍰ if v not reachable from s

- BFS builds *breadth-first spanning tree (forest)*, in which paths to root(s) represent shortest paths in G
  - Thus can use BFS to calculate shortest path from one vertex to another in $O(V + E)$ time in an unweighted graph

Level Order Traversal – Using Queue

- In a level order traversal, every node on a level is visited before going to a lower level



Solution?

Start a BFS traversal from the tree root !!

THANK YOU