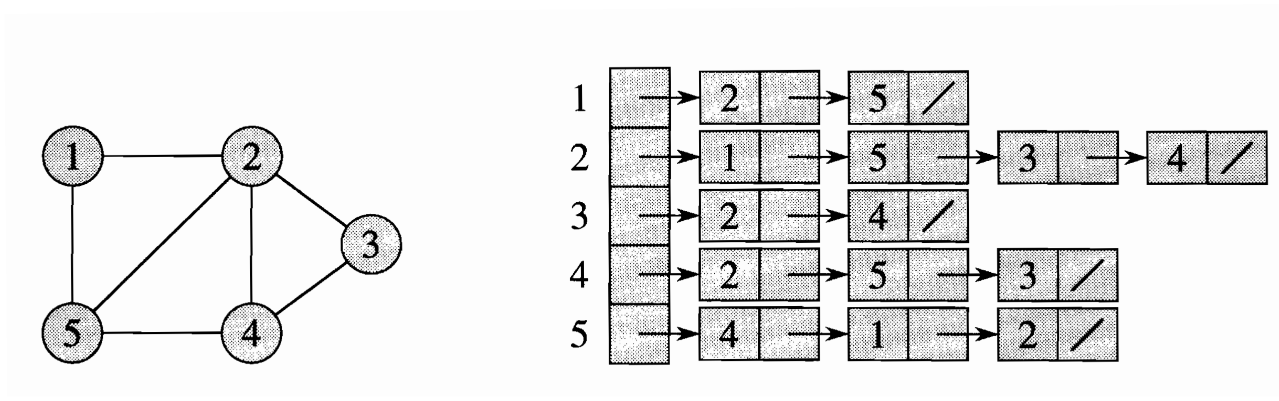
22.1 Representations of graphs

There are two standard ways to represent a (directed or undirected) graph $G = (V, E)$, where V is the set of vertices (or nodes) and E is the set of edges (or links).

- Adjacency list
- Adjacency matrix



$$\begin{array}{c|ccccc|} & 1 & 2 & 3 & 4 & 5 \\ \hline 1 & 0 & 1 & 0 & 0 & 1 \\ 2 & 1 & 0 & 1 & 1 & 1 \\ 3 & 0 & 1 & 0 & 1 & 0 \\ 4 & 0 & 1 & 1 & 0 & 1 \\ 5 & 1 & 1 & 0 & 1 & 0 \\ \hline \end{array}$$

22.2 Basic Search Techniques for Graphs

Graph searching techniques:

- ➤ Breadth First Search (BFS)
- ➤ Depth First Search (DFS)

Both techniques start by visiting a single vertex. Then they repeatedly choose **some vertex with an unvisited neighbor** and visit that neighbor.

This procedure continues until all neighbors of visited vertices are also visited. At this point we have visited all vertices reachable from the starting vertex.

If this cannot cover all vertices in the graph, we start a new search at an unvisited vertex.

22.2 Breadth First Search

Breadth-First Search(BFS) always visits a neighbor of the least recently visited vertex with an unvisited neighbor. This causes it to visit vertices in increasing order of their distance from the starting vertex.

To keep track of BFS search progress, we color each vertex in a graph with one of following three colors.

- WHITE – not visited yet
- GRAY – visited but don't check all its neighbors yet
- BLACK – visited and finished looking at all its neighbors

22.2 Breadth First Search (continued)

Given a vertex v in graph G, we use $G.Adj[v]$ (a linked list) to represent the set of vertices w such that there is an edge from v to w. We need data structures:

- A queue Q with operations
  - ENQUEUE(Q, v) : insert vertex v at the tail of Q
  - DEQUEUE(Q) : remove the vertex at the head of Q and returns its value
- Attributes of vertices
  - $v.color$: WHITE, GRAY or BLACK
  - $v.d$: an integer (the distance of v from the starting vertex)
  - $v.\pi$: the parent pointer that points to another vertex

22.2 Breadth First Search (continued)

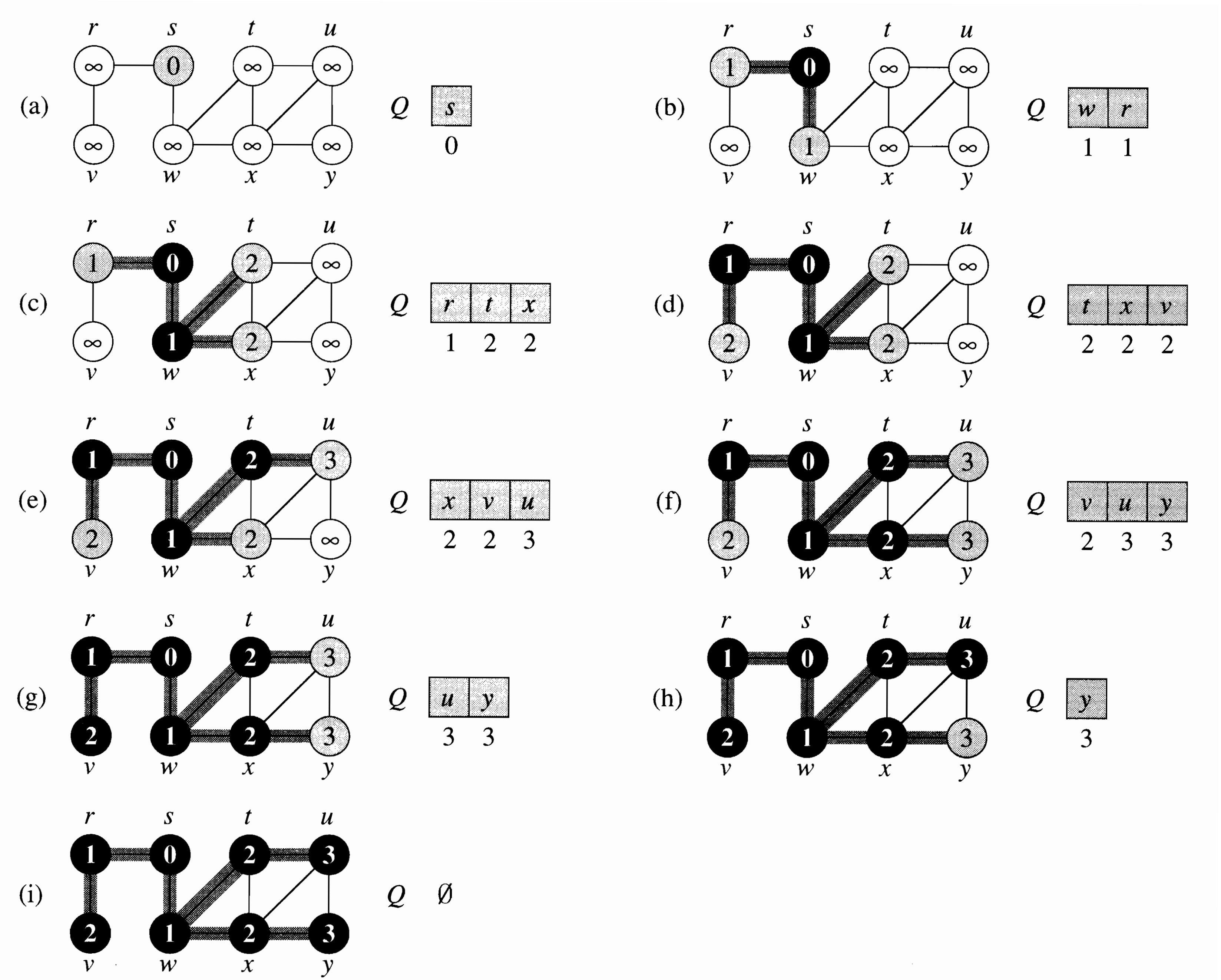
BFS pseudo-code:

```
BFS($G, s$) /* starting vertex is $s$ */
    1    for each vertex $u \in V[G] - \{s\}$ do
    2        $u.color \leftarrow$ WHITE;
    3        $u.d \leftarrow \infty$; /* the distance to the source node $s$ */
    4        $u.\pi \leftarrow NIL$; /* parent pointer */
    5    $s.color \leftarrow$ GRAY;
    6    $s.d \leftarrow 0$;
    7    $s.\pi \leftarrow NIL$;
    8    $Q \leftarrow \emptyset$ /* a queue holds gray nodes only */
    9    ENQUEUE($Q, s$)
    (continued)
```

22.2 Breadth First Search (continued)

BFS(G, s) (continued)

```
while  $Q \neq \emptyset$
        $u \leftarrow$ DEQUEUE($Q$);
            for  each $v \in G.Adj[u]$ do
                if  ($v.color ==$ WHITE)
                    $v.color \leftarrow$ GRAY;
                    $v.d \leftarrow u.d + 1$;
                    $v.\pi \leftarrow u$;
                    ENQUEUE($Q, v$);
        $u.color \leftarrow$ BLACK;
```


(a) Q s 0
(b) Q w r 1 1
(c) Q r t x 1 2 2
(d) Q t x v 2 2 2
(e) Q x v u 2 2 3
(f) Q v u y 2 3 3
(g) Q u y 3 3
(h) Q y 3
(i) Q ∅
r s t u
v w x y

22.2 Breadth First Search (continued)

Theorem: The **BFS** procedure has the following properties:

(a) All the vertices reachable from s are visited in order of their distance from s.

(b) The final value of $v.d$ is ∞ if v is not reachable from s, and the distance
from s to v otherwise.

(c) If v is reachable from s, then a shortest path from s to v (in the number of edges) is

$$s \leftarrow \cdots \leftarrow v.\pi.\pi.\pi \leftarrow v.\pi.\pi \leftarrow v.\pi \leftarrow v$$

(d) The edges $(v.\pi,\ v)$ for all $v \in V$ together form a tree.

Time complexity: Each vertex is initialized once and put into the queue once. Each edge is examined twice from each of its two endpoints. Therefore, the total time for BFS traversal on a graph $G(V,E)$ is $O(|V|+|E|)$, where $|V|$ is the number of vertices and $|E|$ is the number of edges.

22.2 BFS Applications

Distances in a graph.
As described, BFS finds a shortest path from s to every other reachable vertex. (Note that this is only for graphs where all edges have the same length.)

Connected components of an undirected graph.
BFS visits all the vertices reachable from a given vertex. Use it to find one connected component at each time.
It takes $O(|V| + |E|)$ time to find all connected components, why?

Coloring a bipartite graph.
An undirected graph $G(V, E)$ is a bipartite graph if its vertex set can be colored with two colors such that every edge has one endpoint of each color.
Exercise: apply BFS to a graph G find such a two-color coloring or prove it doesn't exist. Under which condition such a coloring does not exist?