BFS-Based Algorithms

- reachability
  - can you get from *u* to *v*?
  - what vertices can you get to from *u*?

  bfs(u)
  – is *u* processed?
  – processed vertices are reachable

- unweighted shortest path
  - distance from *u* to all other vertices, in terms of the number of edges in the path
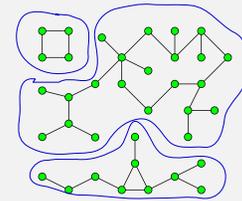
  bfs(u)
  – set dist[v] = dist[u]+1 when *v* is discovered
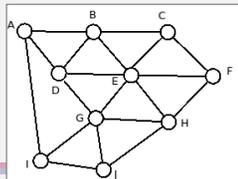
BFS-Based Algorithms

a graph with three connected components (circled)

- connected components
  - a *connected component* is a subgraph where every pair of vertices are connected by a path and there are no connections to other vertices not in the subgraph
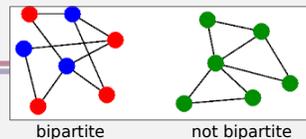
```
c = 0
for each vertex v of graph G
  if v has not been discovered
    run bfs(v), setting comp[u] = c when each vertex u is
     processed
    c++
```

intuition – BFS finds all vertices reachable from *v* along a path

---

BFS-Based Algorithms

bipartite   not bipartite

- bipartite graph detection / two-coloring
  - a *bipartite* graph is one whose vertices can be divided into two sets such that every edge connects a vertex in one set with a vertex in the other
  - *coloring* refers to assigning labels (colors) to vertices so that no two adjacent vertices have the same label (color)
    - a *two-coloring* uses two colors

```
color[s] = 0
run bfs(s), setting color[v] = the opposite color of
  color[u] for each discovery edge (u,v) and checking that
  color[v] is the opposite color of color[u] for each non-
   discovery edge (u,v)
   – if there is an edge (u,v) for which color[u] = color[v], the graph is
     not bipartite / two-colorable
```

intuition – following a path along discovery edges must alternate colors, since those edges are graph edges
  - can't change the color of any vertex without changing them all
  - non-discovery edges are also graph edges, and ends must be opposite colors

CF

46

Takeaways

- BFS algorithm

- BFS-based algorithms
  - graph traversal
  - reachability
  - unweighted shortest path
  - connected components
  - 2-coloring / detecting bipartite graphs