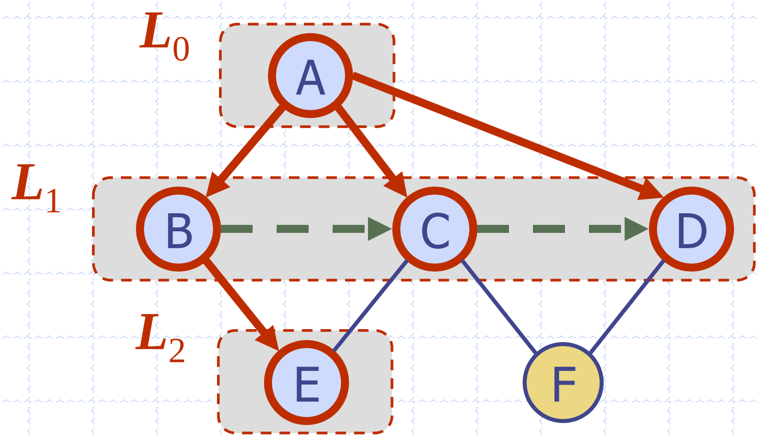
Presentation for use with the textbook Data Structures and Algorithms in Java, 6th edition, by M. T. Goodrich, R. Tamassia, and M. H. Goldwasser, Wiley, 2014

Breadth-First Search



Breadth-First Search

- Breadth-first search (BFS) is a general technique for traversing a graph
- A BFS traversal of a graph G
  - Visits all the vertices and edges of G
  - Can determines whether G is connected
  - Can computes the connected components of G
  - Can computes a spanning forest of G
- BFS can be further extended to solve other graph problems
  - Find and report a path with the minimum number of edges between two given vertices
  - Find a simple cycle, if there is one

BFS Algorithm

```
Algorithm BFS(G, s)
    Q ← new empty queue
    Q.enqueue(s)
    mark(s)
    while Q is not empty do {
        u ← Q.dequeue()
        visit (u)
        for each edge (u,v) incident on u do
                if  (u,v) is not labelled then
                    if  v is not marked then {
                        Label (u,v) as DISCOVERY
                        mark(v)
                        Q.enqueu(v)
                    }
                    else
                        Label (u,v) as CROSS
    }
```

Example

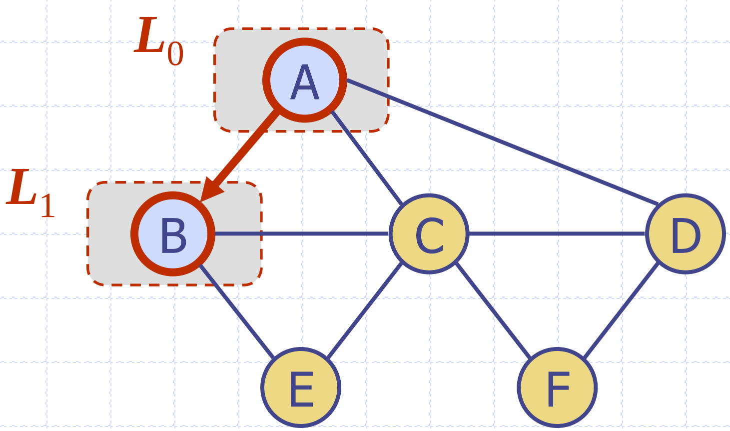
A unexplored vertex

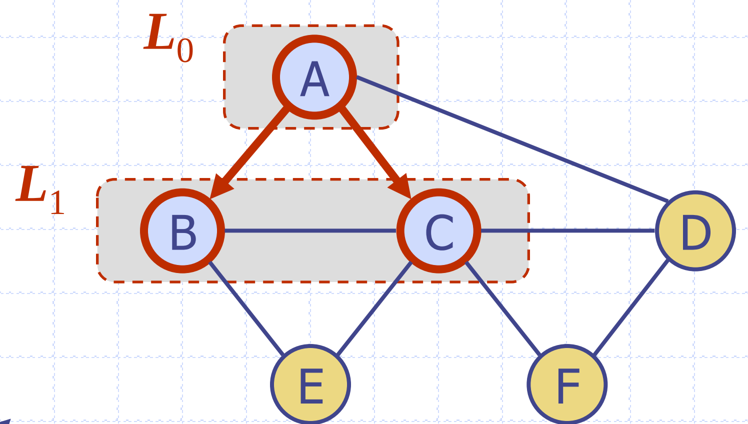
A visited vertex

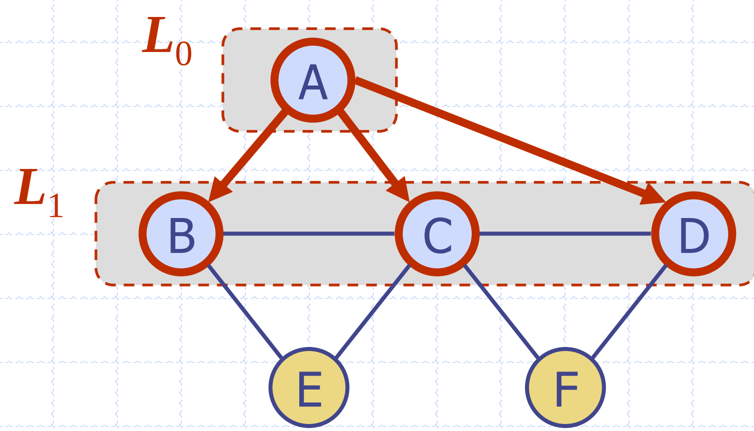
unexplored edge

discovery edge

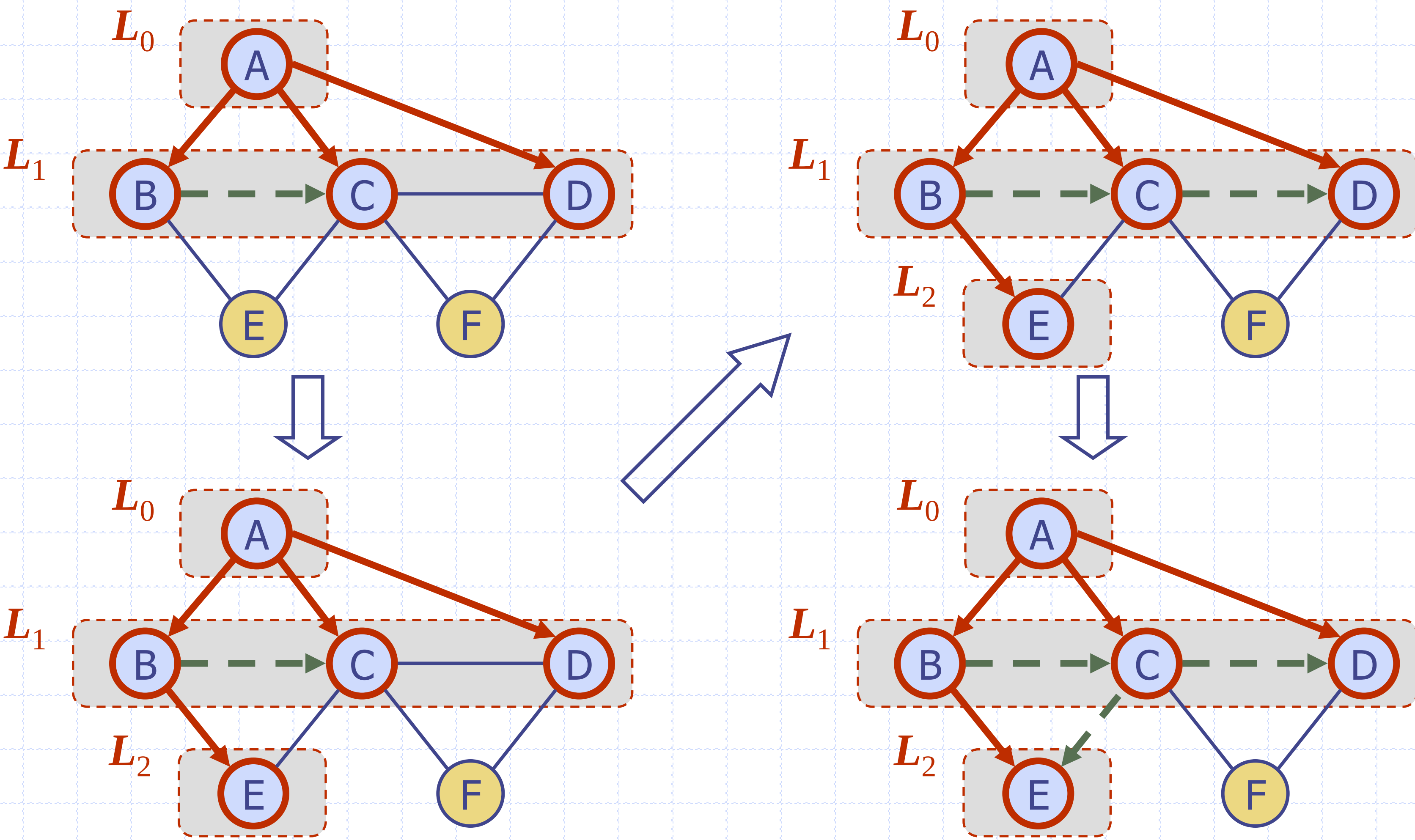
cross edge



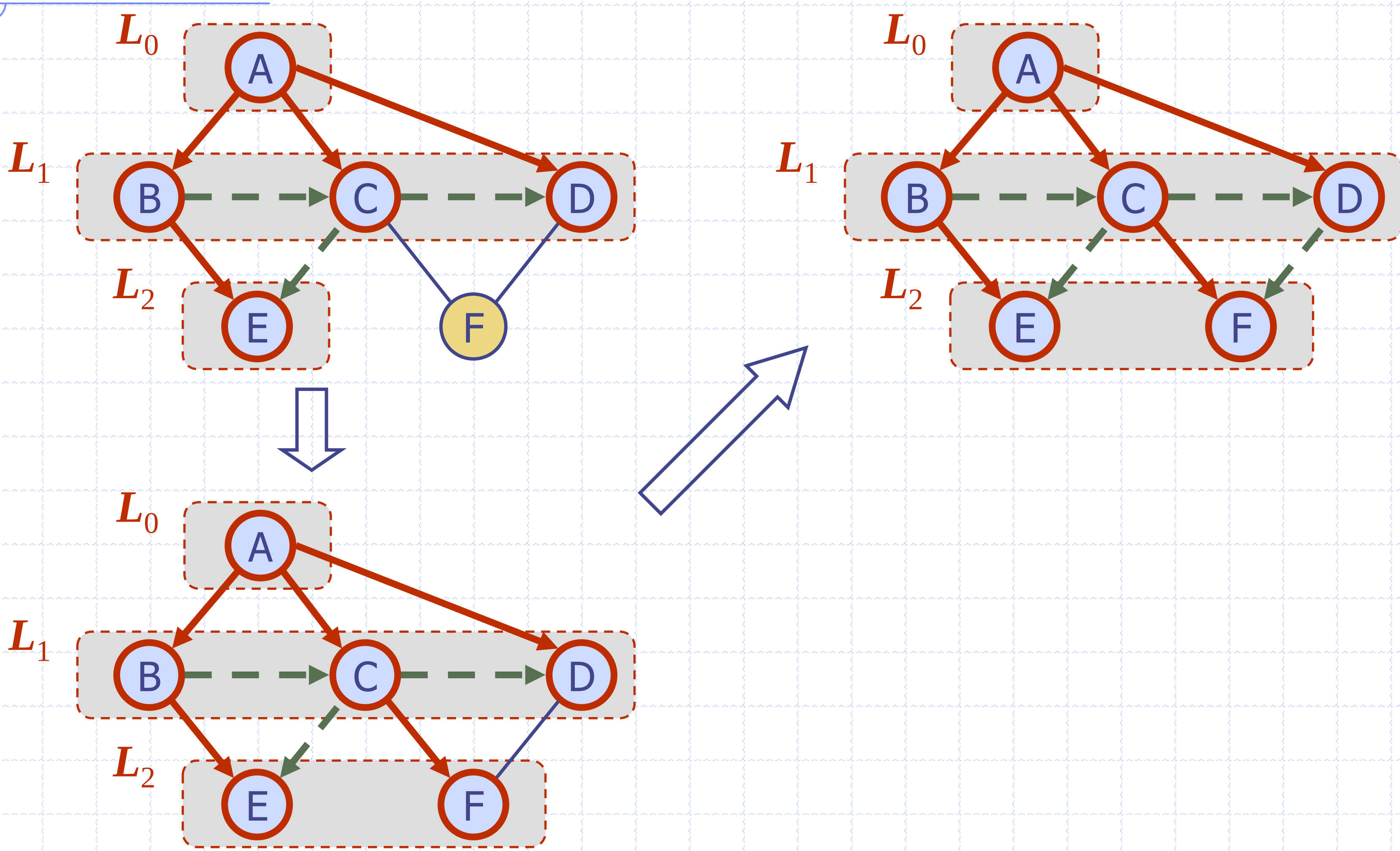




Example (cont.)



Example (cont.)



Properties

Notation

G_s: connected component of s

Property 1

$BFS(G, s)$ visits all the vertices and edges of G_s

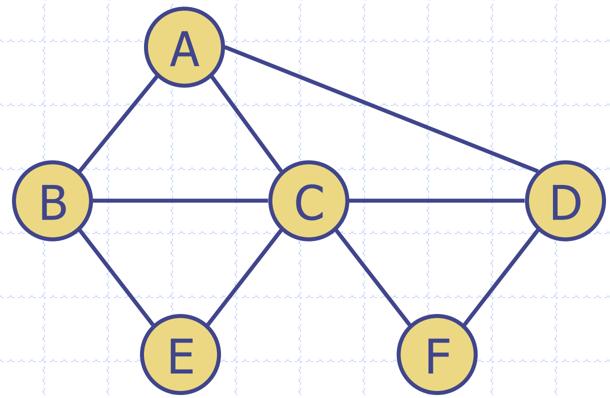
Property 2

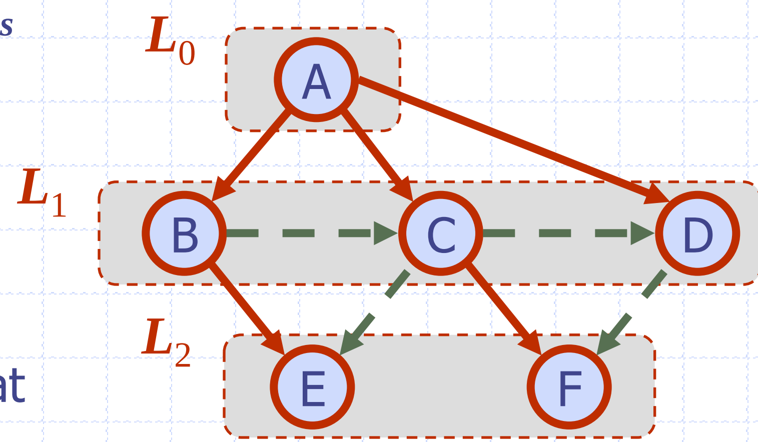
The discovery edges labeled by $BFS(G, s)$ form a spanning tree T_s of G_s called a BFS tree

Property 3

For each vertex v in level L_i

- The path of T_s from s to v has i edges
- Every path from s to v in G_s has at least i edges





Analysis

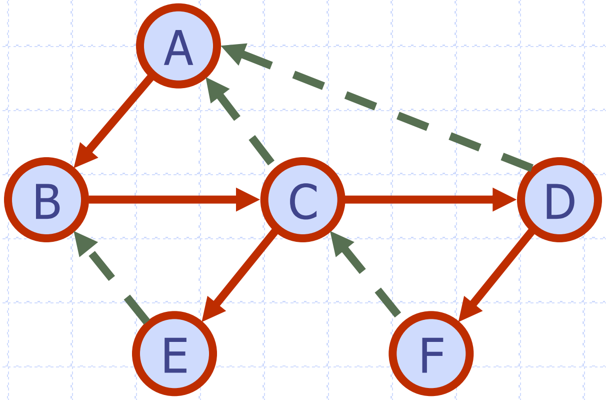
- Setting/getting a vertex/edge label takes $O(1)$ time
- Each vertex is labeled twice
  - once initialized as UNEXPLORED
  - once as VISITED
- Each edge is labeled twice
  - once initialized as UNEXPLORED
  - once as DISCOVERY or CROSS
- Each vertex is inserted once into the queue
- Method incidentEdges is called once for each vertex
- BFS runs in $O(n + m)$ time provided the graph is represented by the adjacency list structure and it runs in $O(n^2)$ time if the graph is represented by the adjacency matrix structure.

Applications

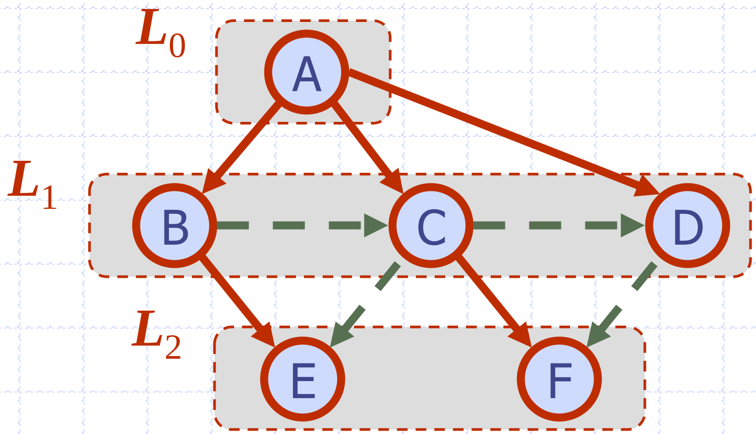
- We can use a BFS traversal of a graph G to solve the following problems in $O(n + m)$ time
  - Compute the connected components of G
  - Compute a spanning forest of G
  - Find a simple cycle in G, or report that G is a forest
  - Given two vertices of G, find a path in G between them with the minimum number of edges, or report that no such path exists

DFS vs. BFS

Applications	DFS	BFS
Spanning forest, connected components, paths, cycles	√	√
Shortest paths		√



DFS



BFS

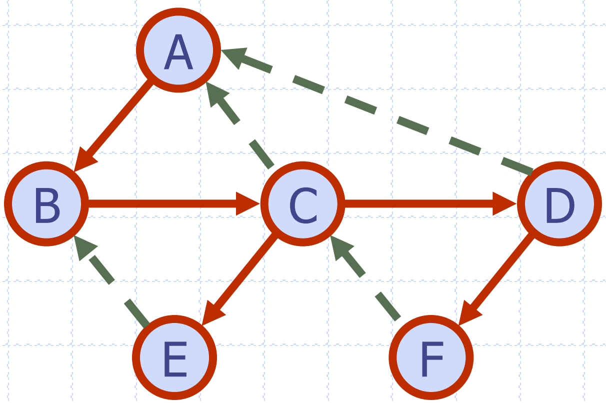
DFS vs. BFS (cont.)

Back edge (v,w)

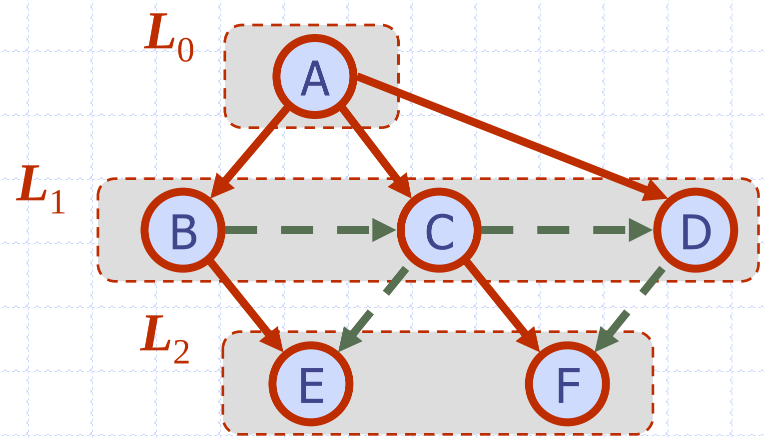
- w is an ancestor of v in the tree of discovery edges

Cross edge (v,w)

- w is in the same level as v or in the next level



DFS



BFS