

Properties

Notation

G_s : connected component of s

Property 1

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

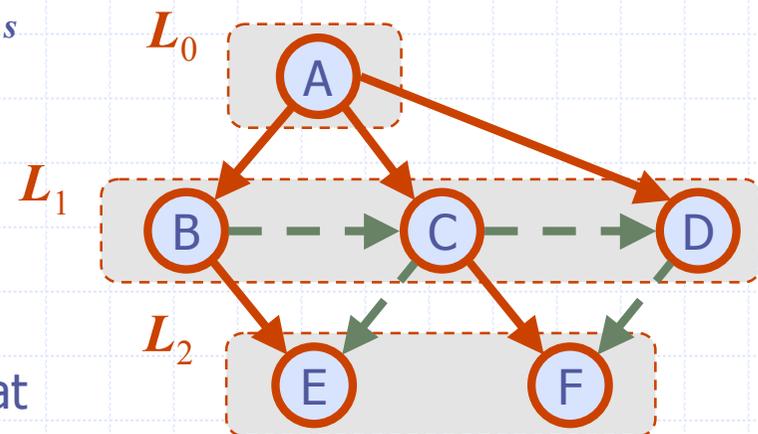
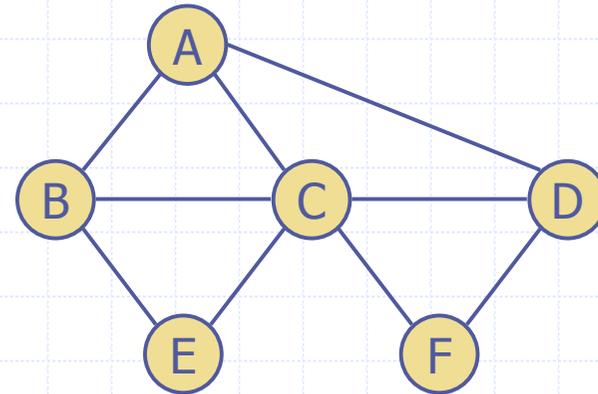
Property 2

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

Property 3

For each vertex v in 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

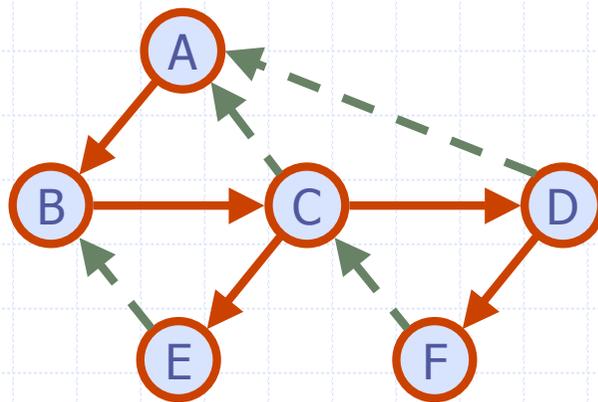
- ◆ Setting/getting a vertex/edge label takes $O(1)$ time
- ◆ Each vertex is labeled twice
 - once as UNEXPLORED
 - once as VISITED
- ◆ Each edge is labeled twice
 - once as UNEXPLORED
 - once as DISCOVERY or CROSS
- ◆ Each vertex is inserted once into a sequence L_i
- ◆ Method incidentEdges is called once for each vertex
- ◆ BFS runs in $O(n + m)$ time provided the graph is represented by the adjacency list structure
 - Recall that $\sum_v \deg(v) = 2m$

Applications

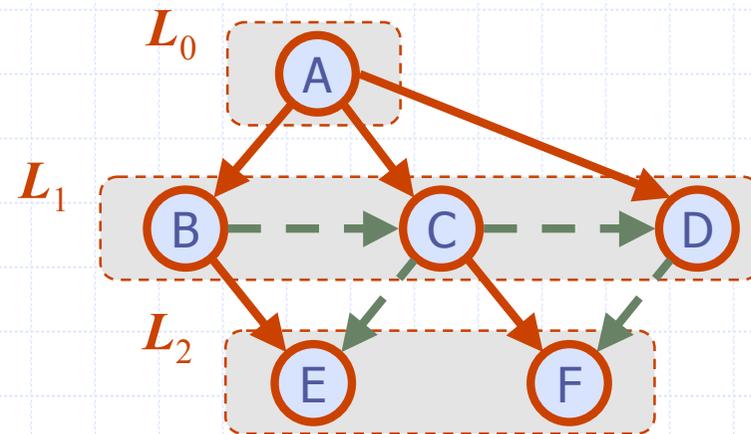
- ◆ Using the template method pattern, we can specialize the BFS traversal of a graph G to solve the following problems in $O(n + m)$ time
 - Compute the connected components of G
 - Compute a spanning forest of G
 - Find a simple cycle in G , or report that G is a forest
 - Given two vertices of G , 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		✓
Minimal use of memory space	✓	



DFS

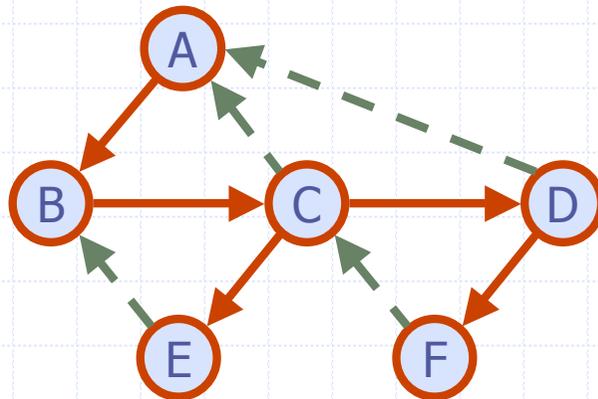


BFS

DFS vs. BFS (cont.)

Back edge (v,w)

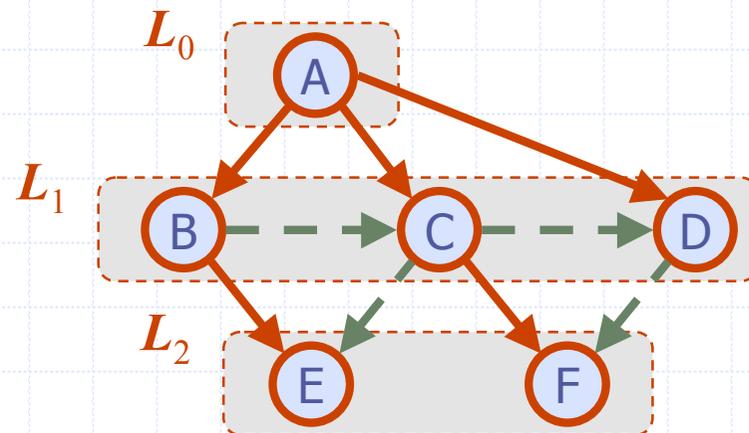
- w is an ancestor of v in the tree of discovery edges



DFS

Cross edge (v,w)

- w is in the same level as v or in the next level in the tree of discovery edges



BFS