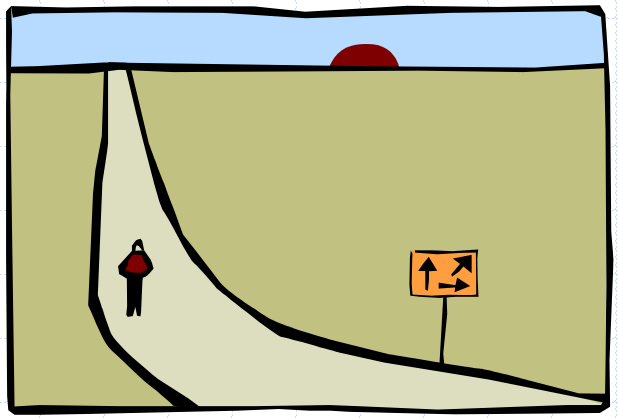
그래프 순회



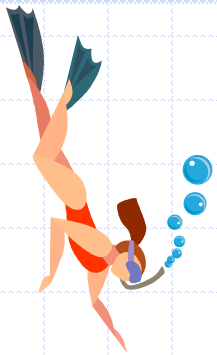
Outline

- 14.1 그래프 순회
- 14.2 깊이우선탐색
- 14.3 너비우선탐색
- 14.4 응용문제

그래프 순회

- **순회**(traversal): 모든 정점과 간선을 검사함으로써 그래프를 탐험하는 체계적인 절차
- 순회 예
  - 수도권 전철망의 모든 역(정점)의 위치를 출력
  - 항공사의 모든 항공편(간선)에 대한 노선 정보를 수집
  - 웹 검색엔진의 데이터 수집 부문은 웹의 하이퍼텍스트 문서(정점)와 문서 내 링크(간선)를 검사함으로써 써핑
- 주요 전략
  - 깊이우선탐색
  - 너비우선탐색

깊이우선탐색



- **깊이우선탐색**(depth-first search, **DFS**): 그래프를 순회하기 위한 일반적 기법
- 그래프 G에 대한 DFS 순회로 가능한 것들
  - G의 모든 정점과 간선을 방문
  - G가 연결그래프인지 결정
  - G의 연결요소들을 계산
  - G의 신장숲을 계산
- n개의 정점과 m개의 간선을 가진 그래프에 대한 DFS는 $\mathbf{O}(n+m)$ 시간 소요
- DFS를 **확장**하면 해결 가능한 그래프 문제들
  - 두 개의 주어진 정점 사이의 경로를 찾아 보고하기
  - 그래프 내 싸이클 찾기
- 그래프에 대한 **깊이우선탐색**은 이진트리에 대한 **선위순회**와 유사

DFS

Alg ***DFS(G)***
input graph ***G***
output labeling of the edges of ***G*** as tree edges and back edges

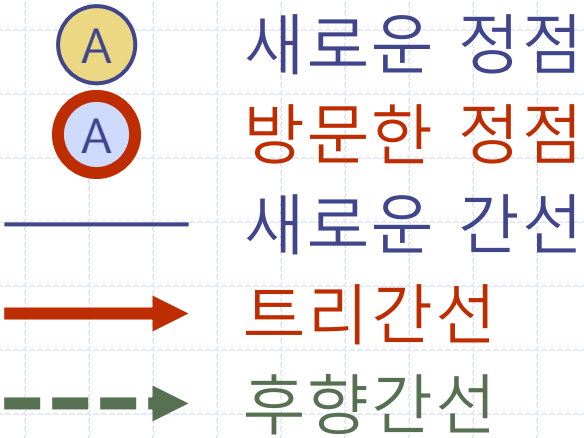
{The algorithm uses a mechanism for setting and getting "labels" of vertices and edges}

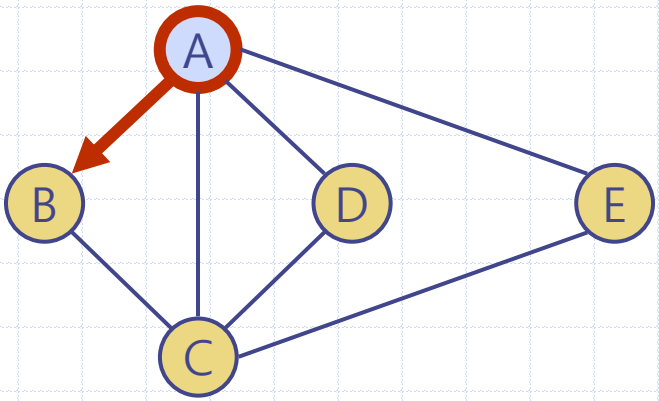
1. **for each** $u \in G.vertices()$
   $l(u) \leftarrow Fresh$
2. **for each** $e \in G.edges()$
   $l(e) \leftarrow Fresh$
3. **for each** $v \in G.vertices()$
   if $(l(v) = Fresh)$
   rDFS(***G***, ***v***)

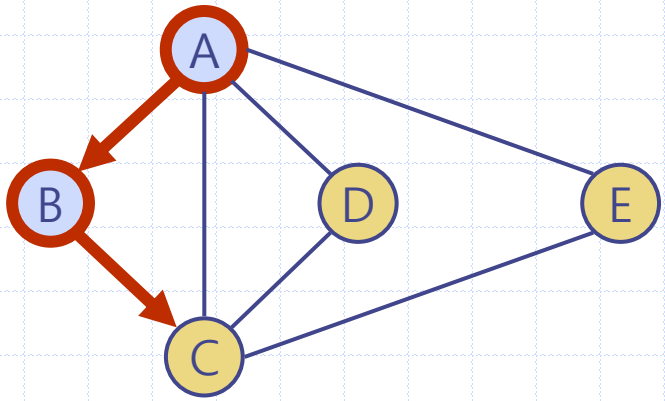
Alg ***rDFS(G, v)***
input graph ***G*** and a start vertex ***v*** of ***G***
output labeling of the edges of ***G*** in the connected component of ***v*** as tree edges and back edges

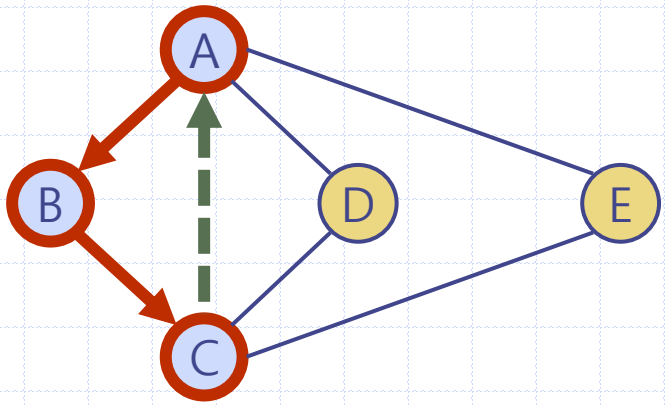
1. $l(v) \leftarrow Visited$
2. **for each** $e \in G.incidentEdges(v)$
   if $(l(e) = Fresh)$
   $w \leftarrow G.opposite(v, e)$
   if $(l(w) = Fresh)$
   $l(e) \leftarrow Tree$
   rDFS(***G***, ***w***)
   else
   $l(e) \leftarrow Back$

DFS 수행 예

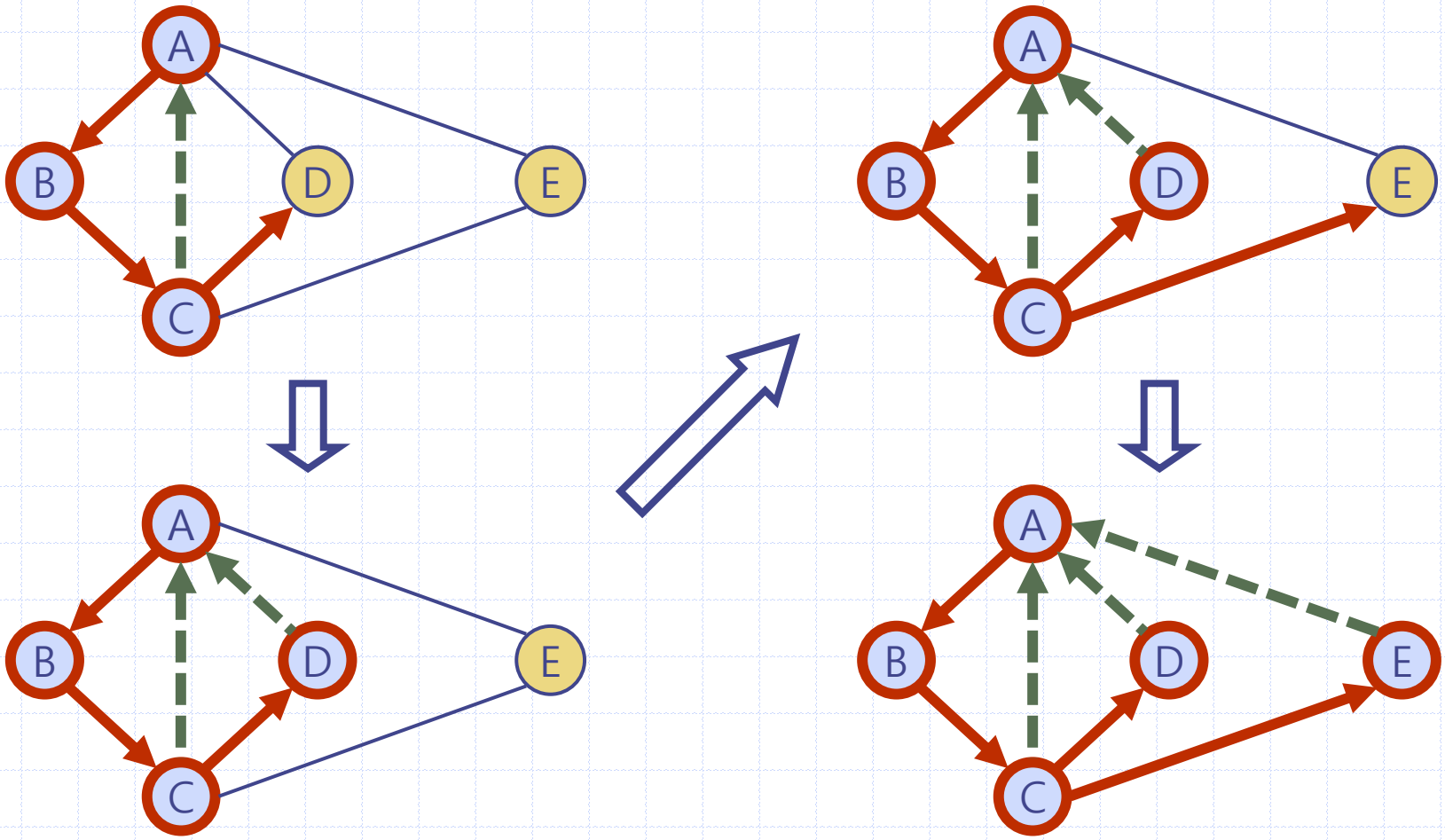






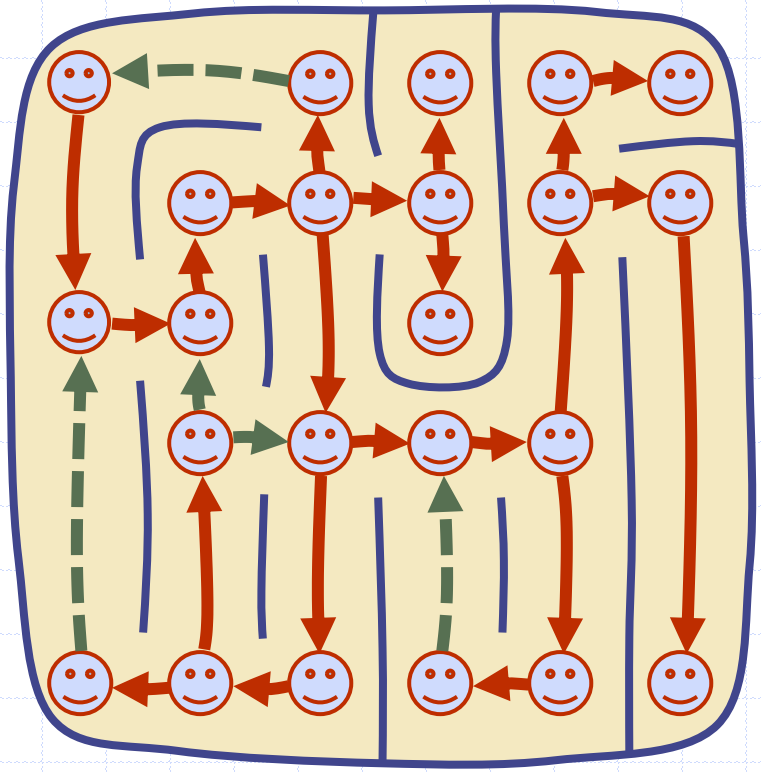


DFS 수행 예 (conti.)



DFS와 미로 순회

- DFS 알고리즘은 **미로**를 탐험하는데 있어서의 고전적이고 **모험적**인 전략과 유사
  - 방문한 **교차로**, **모퉁이**, **막힌 복도**(모두 **정점**)를 표시
  - 순회한 **복도**(모두 **간선**)를 표시
  - 입구(출발정점)로 되돌아가는 경로를 끈(재귀 스택)을 사용하여 추적



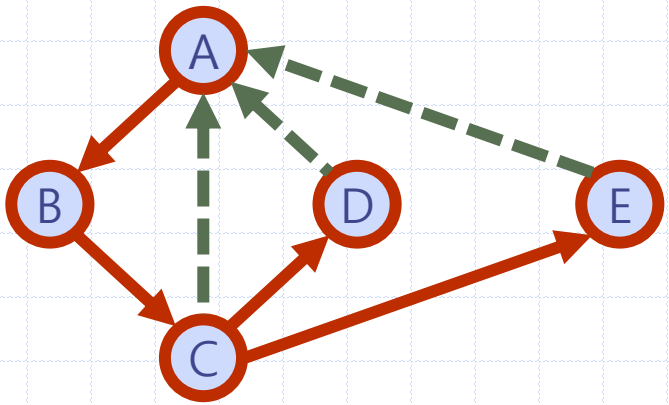
DFS 속성

속성 1

rDFS(G, v)는 v의 연결요소내의 모든 정점과 간선을 방문

속성 2

rDFS(G, v)에 의해 라벨된 트리 간선들은 v의 연결요소의 **신장트리**(**DFS tree**라 불림)를 형성

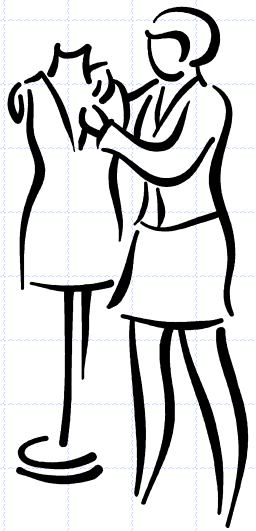


DFS 분석

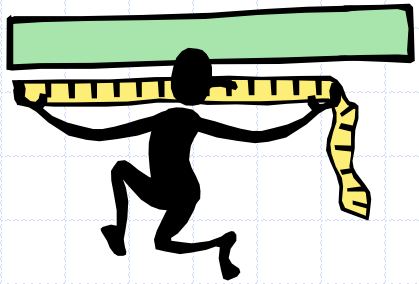
- 정점과 간선의 **라벨**을 쓰고 읽는데 $O(1)$ 시간 소요
  - **참고:** 정점이나 간선을 구현하는 노드 위치의 기능성에 "*Visited*" 플래그를 포함하도록 확장 가능
- 각 **정점**은 두 번 라벨
  - 한 번은 ***Fresh***로, 또 한 번은 ***Visited***로
- 각 **간선**은 두 번 라벨
  - 한 번은 ***Fresh***로, 또 한 번은 ***Tree*** 또는 ***Back***으로
- 메쏘드 incidentEdges는 각 **정점**에 대해 한 번 호출
- 그래프가 **인접리스트 구조**로 표현된 경우, DFS는 $O(n + m)$ 시간에 수행
  - **참고:** $\sum_v deg(v) = 2m$

DFS 템플릿 활용

- DFS 순회로 더욱 흥미있는 작업을 수행코자 한다면, DFS 알고리즘을 확장해야 한다
- 이를 DFS 순회 원형 메쏘드의 **특화**(specialization)라 부른다
- DFS를 확장하여 해결할 수 있는 문제의 예
  - 연결성 검사
  - 경로 찾기
  - 싸이클 찾기



너비우선탐색



- **너비우선탐색**(breadth-first search, **BFS**): 그래프를 순회하기 위한 일반적 기법
- 그래프 G에 대한 BFS 순회로 가능한 것들
  - G의 모든 정점과 간선을 방문
  - G가 연결그래프인지 결정
  - G의 연결요소들을 계산
  - G의 신장숲을 계산
- n개의 정점과 m개의 간선을 가진 그래프에 대한 BFS는 $O(n+m)$ 시간 소요
- BFS를 **확장**하면 해결 가능한 그래프 문제들
  - 두 개의 주어진 정점 사이의 최소 간선을 사용하는 경로를 찾아 보고하기
  - 그래프 내 단순싸이클 찾기
- 그래프에 대한 **너비우선탐색**은 이진트리에 대한 **레벨순회**와 유사

BFS

Alg ***BFS(G)***
input graph ***G***
output labeling of the edges of ***G*** as tree edges and cross edges

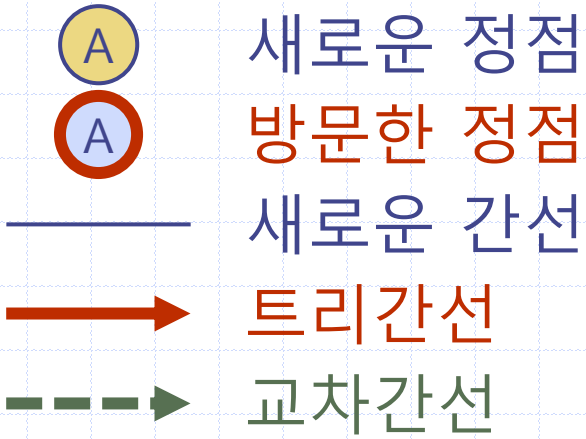
{The algorithm uses a mechanism for setting and getting "labels" of vertices and edges}

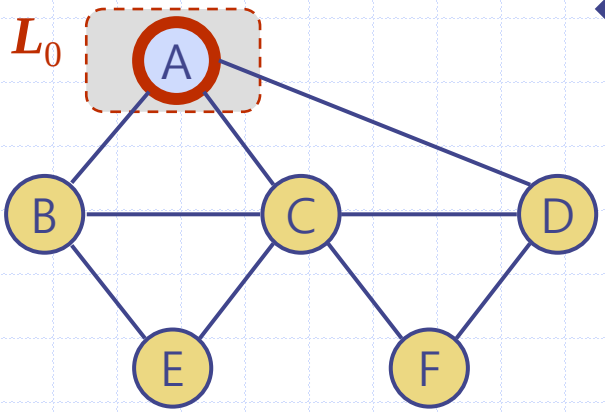
1. **for each** $u \in G.vertices()$
   $l(u) \leftarrow Fresh$
2. **for each** $e \in G.edges()$
   $l(e) \leftarrow Fresh$
3. **for each** $v \in G.vertices()$
   if $(l(v) = Fresh)$
   BFS1(G, v)

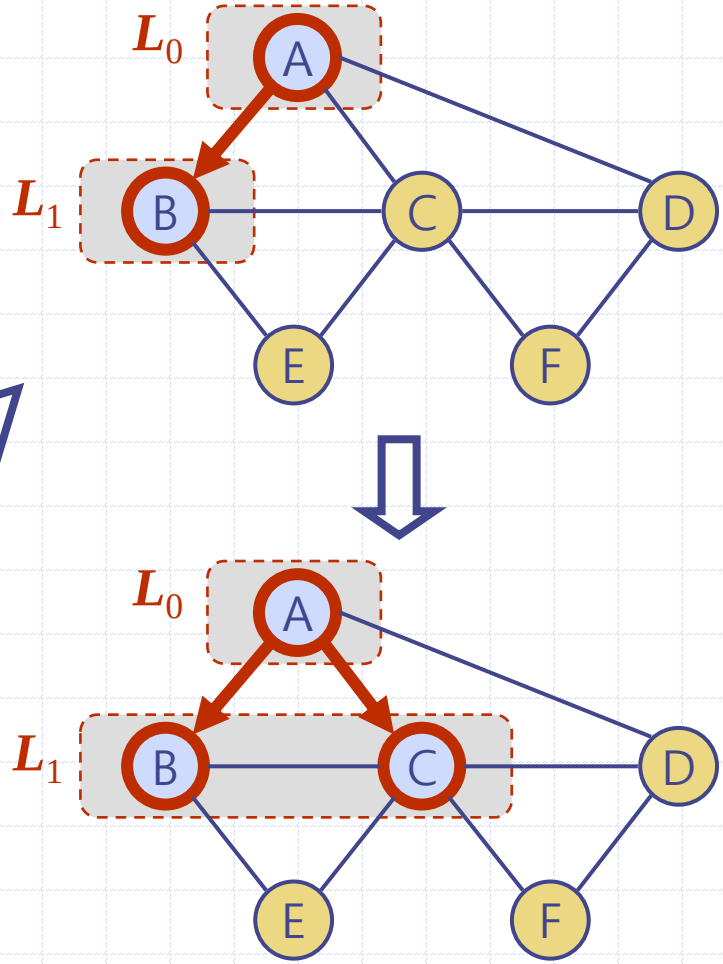
Alg ***BFS1(G, v)***
input graph ***G*** and a start vertex ***v*** of ***G***
output labeling of the edges of ***G*** in the connected component of ***v*** as tree edges and cross edges

1. $L_0 \leftarrow$ *empty list* {level container}
2. $L_0.addLast(v)$
3. $l(v) \leftarrow Visited$
4. $i \leftarrow 0$
5. **while** $(!L_i.isEmpty())$
   $L_{i+1} \leftarrow$ *empty list*
   for each $v \in L_i.elements()$
   for each $e \in G.incidentEdges(v)$
   if $(l(e) = Fresh)$
   $w \leftarrow G.opposite(v, e)$
   if $(l(w) = Fresh)$
   $l(e) \leftarrow Tree$
   $l(w) \leftarrow Visited$
   $L_{i+1}.addLast(w)$
   else
   $l(e) \leftarrow Cross$
   $i \leftarrow i + 1$

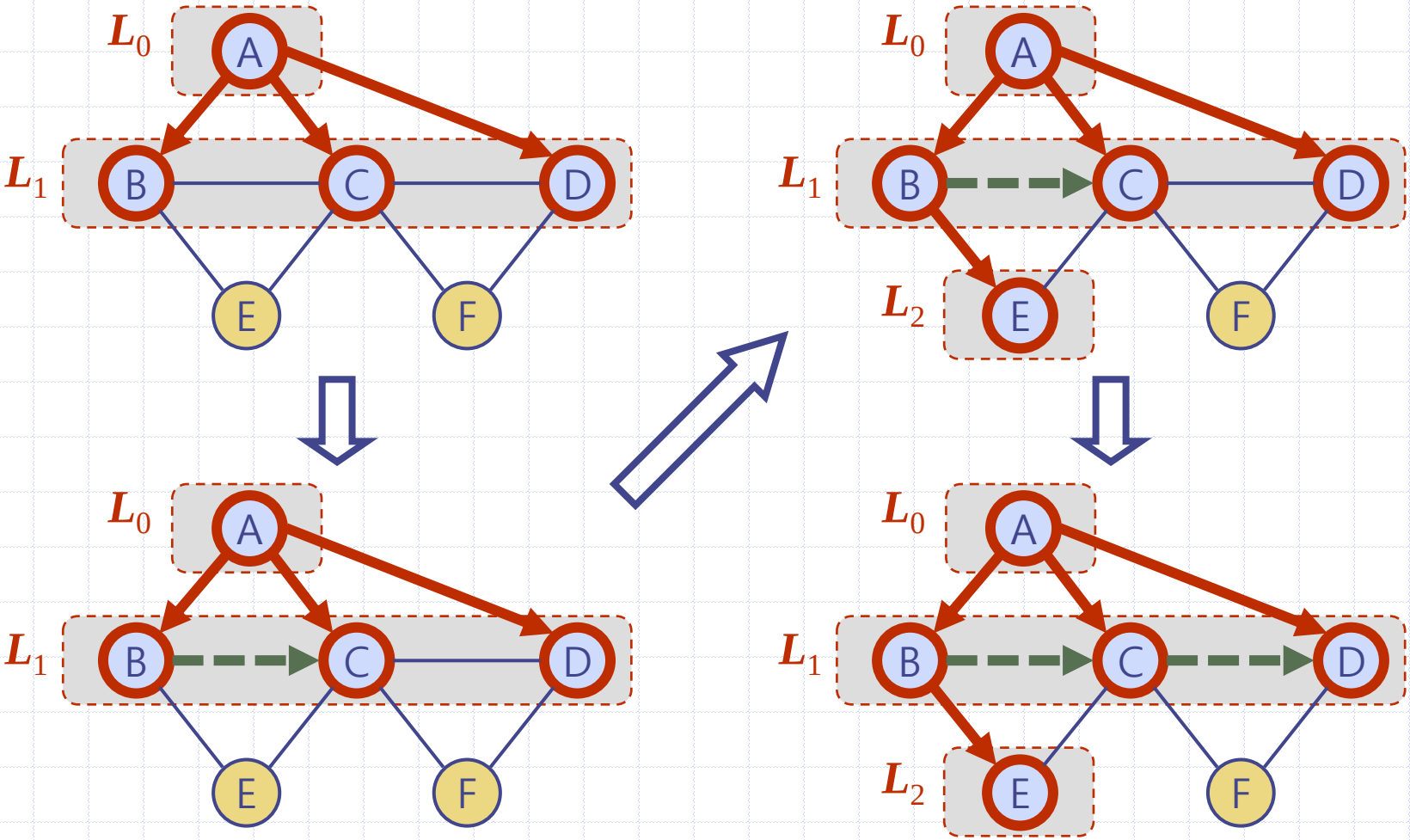
BFS 수행 예


A
새로운 정점
A
방문한 정점
새로운 간선
트리간선
교차간선

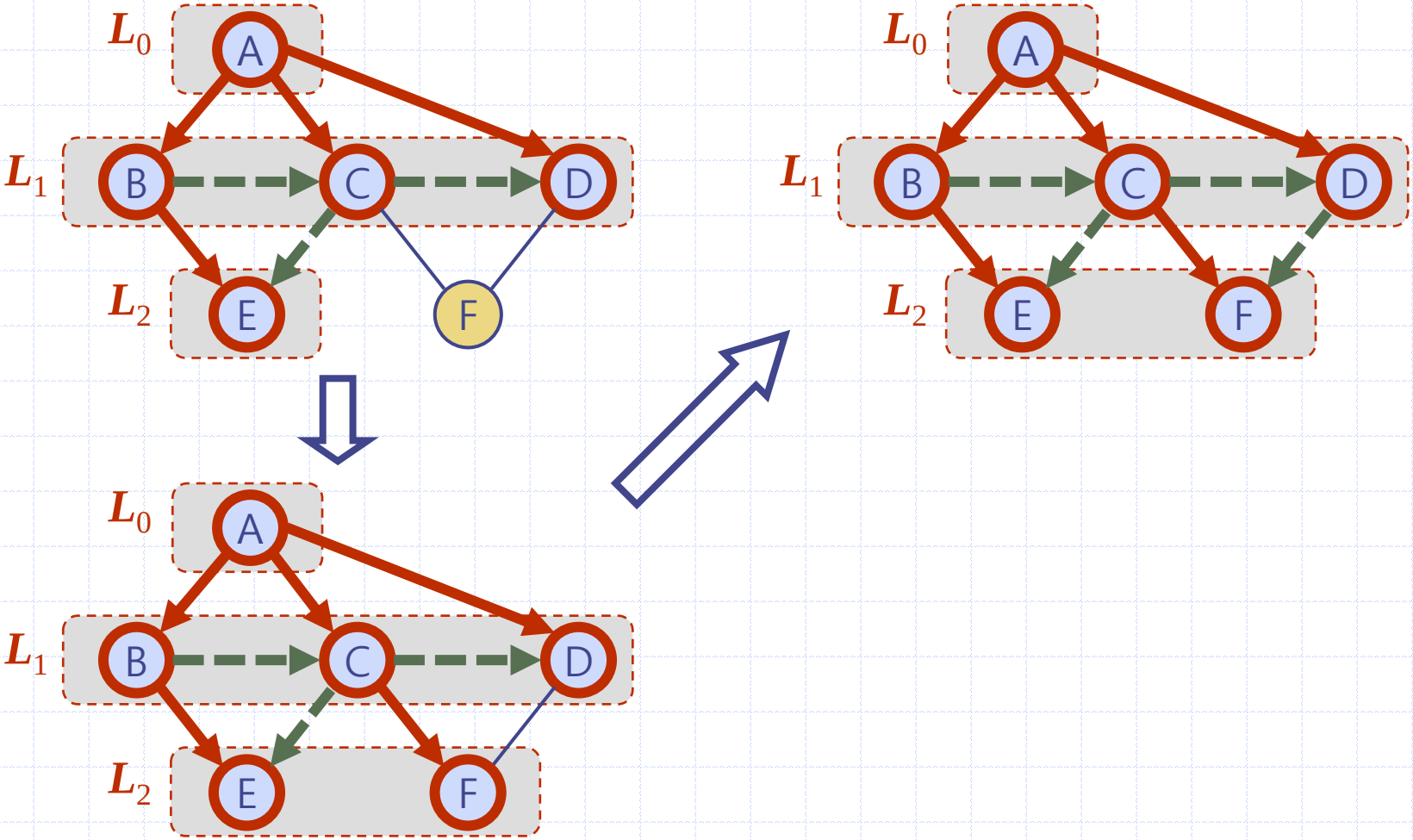

L_0
A
B
C
D
E
F


L_0
A
L_1
B
C
D
E
F
L_0
A
L_1
B
C
D
E
F

BFS 수행 예 (conti.)

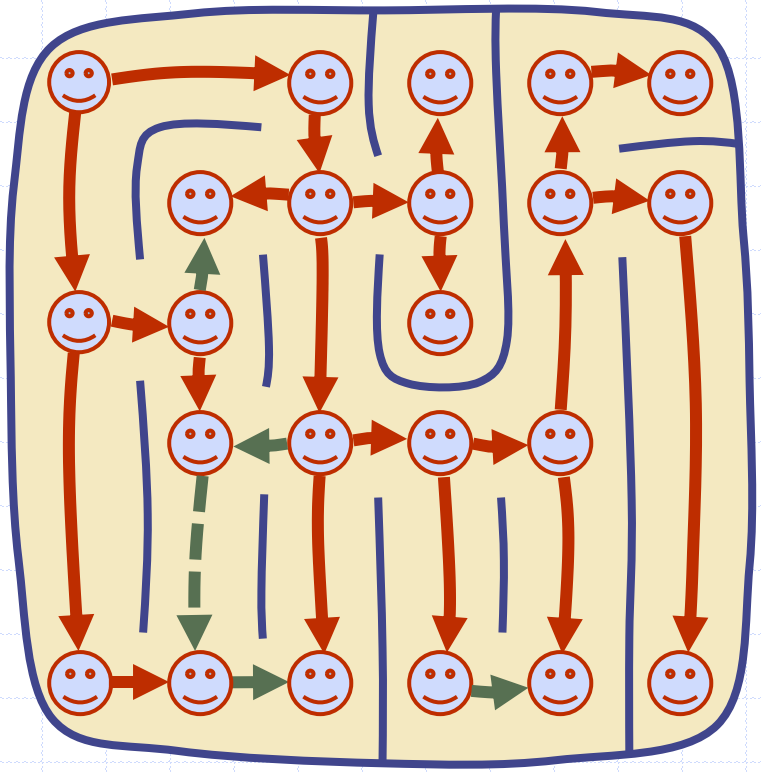


BFS 수행 예 (conti.)



BFS와 미로 순회

- BFS 알고리즘은 **미로**를 탐험하는데 있어서의 다소 **보수적**인 전략과 유사
  - 방문한 **교차로**, **모퉁이**, **막힌 복도**(모두 **정점**임)를 표시
  - 순회한 **복도**(모두 **간선**임)를 표시
  - **레벨**을 하나씩 증가시키면서 진행



BFS 속성

표기

G_v: v의 연결요소

속성 1

BFS1(G, v)는 G_v의 모든 정점과 간선을 방문

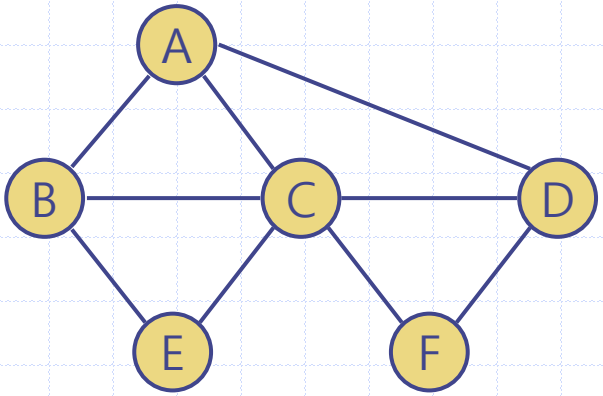
속성 2

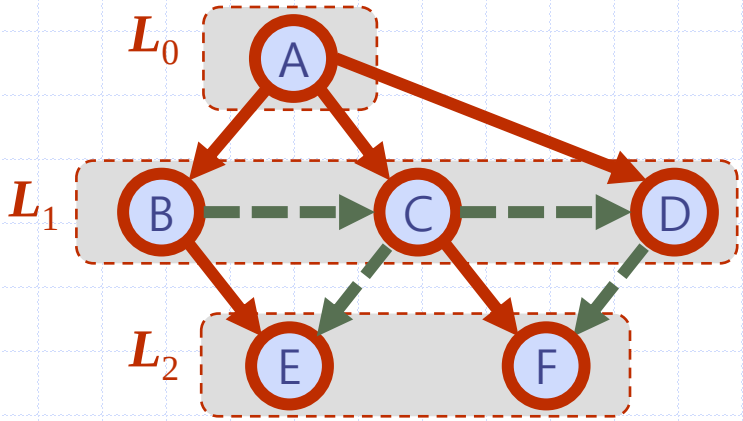
BFS1(G, v)에 의해 라벨된 트리 간선들은 G_v의 **신장트리**(**BFS tree**라 불림) T_v를 형성

속성 3

L_i 내의 각 정점 w에 대해,

- T_v의 v에서 w로 향하는 경로는 i개의 간선을 가진다
- G_v내의 v에서 w로 향하는 모든 경로는 최소 i개의 간선을 가진다

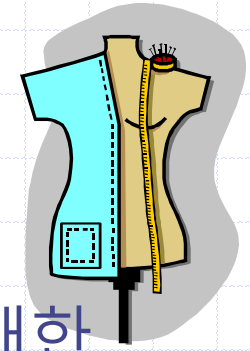




BFS 분석

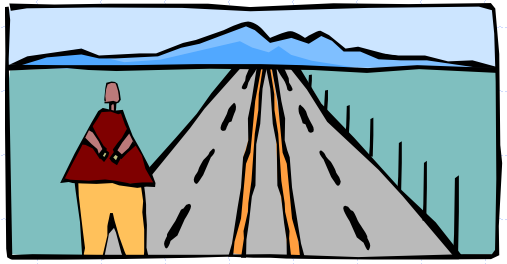
- 정점과 간선의 **라벨**을 쓰고 읽는데 $O(1)$ 시간 소요
  - **참고:** 정점이나 간선을 구현하는 노드 위치의 기능성에 "*Visited*" 플래그를 포함하도록 확장 가능
- 각 **정점**은 두 번 라벨
  - 한 번은 ***Fresh***로, 또 한 번은 ***Visited***로
- 각 **간선**은 두 번 라벨
  - 한 번은 ***Fresh***로, 또 한 번은 ***Tree*** 또는 ***Cross***로
- 각 **정점**은 리스트 L_i에 한 번 삽입
- 메쏘드 incidentEdges는 각 **정점**에 대해 한 번 호출
- 그래프가 **인접리스트 구조**로 표현된 경우, BFS는 $O(n + m)$ 시간에 수행
  - **참고:** $\sum_v deg(v) = 2m$

BFS 템플릿 활용



- 템플릿 메쏘드 패턴을 사용하여, 그래프 G에 대한 BFS 순회를 다음 문제들을 $O(n + m)$ 시간에 해결하도록 **특화**할 수 있다
  - G의 연결요소들을 계산하기
  - G의 신장숲을 계산하기
  - G 내의 단순 싸이클 찾기 또는 G가 숲임을 보고하기
  - G의 주어진 두 정점에 대해, 그 사이의 최소 간선으로 이루어진 G 내의 경로 찾기, 또는 그런 경로가 없음을 보고하기

비트리 간선

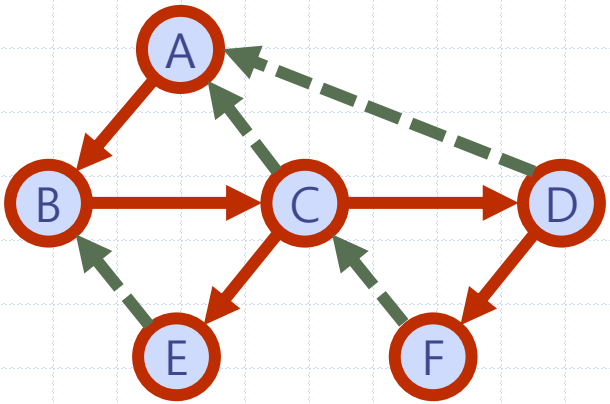


후향간선 (v, w)

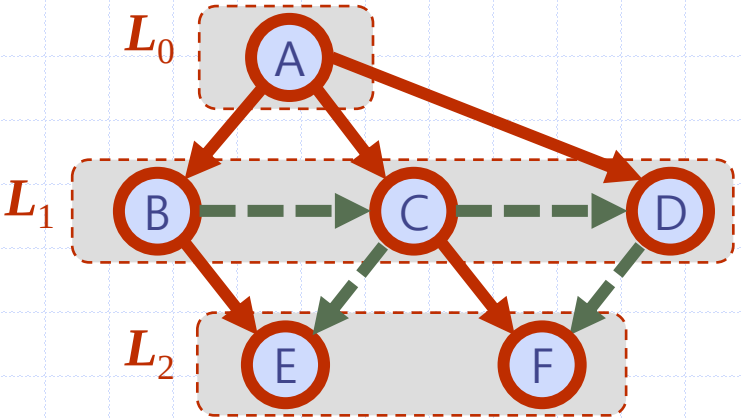
- 트리 간선들의 트리에서 w가 v의 조상

교차간선 (v, w)

- 트리 간선들의 트리에서 w가 v와 동일 또는 다음 레벨에 위치



DFS



BFS

DFS와 BFS 응용

응용	DFS	BFS
신장숲 연결요소 경로 싸이클	√	√
최단경로		√
이중 연결요소	√	