그래프의 개념

◆ 그래프(graph)의 개념

- 문제 정의나 상황을 단순한 그림으로 객체와 객체 사이의 관계로 표현할 때 사용
- 선형 자료 구조나 트리 자료구조로 표현하기 어려운 多:多의 관계를 가지는 원소들 사이의 관계 표현하기 위한 자료구조
- 어떤 공정의 반복이나 상하관계가 명확하지 않은 광범위한 분야에 이용

◆ 그래프 G의 정의

- 객체를 나타내는 정점(노드, vertex)과 객체를 연결하는 간선(연결선, edge)의 집합
- G = (V, E)

  V는 그래프에 있는 정점들의 집합

  E는 정점을 연결하는 간선들의 집합

그래프의 정의 및 용어

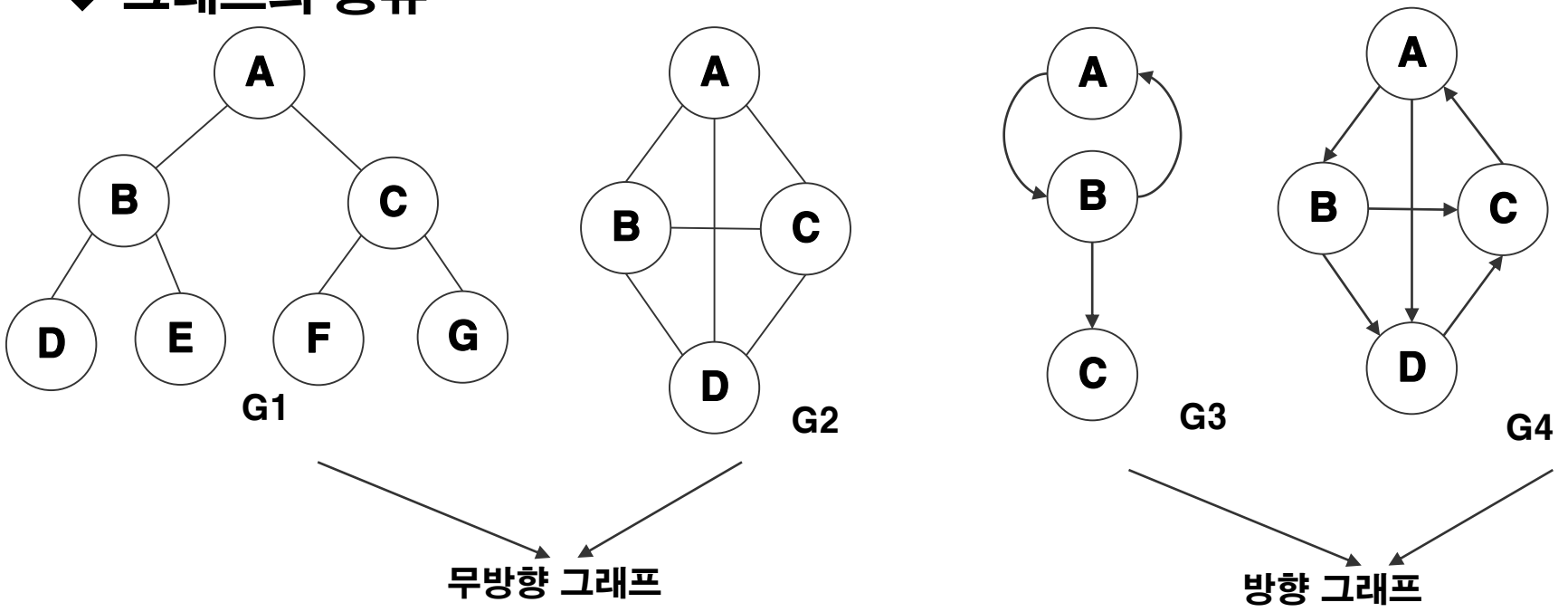
◆ 그래프의 종류

- 무방향 그래프(undirected graph)
  - 두 정점을 연결하는 간선의 방향이 없는 그래프
  - 정점 Vi와 정점 Vj을 연결하는 간선을 (Vi, Vj)로 표현
    - ✓ (Vi, Vj)와 (Vj, Vi)는 같은 간선을 나타낸다.

- 방향 그래프(directed graph) , 다이그래프(digraph)
  - 간선이 방향을 가지고 있는 그래프
  - 정점 Vi에서 정점 Vj를 연결하는 간선 즉, Vi→Vj를 <Vi, Vj>로 표현
    - ✓ Vi를 꼬리(tail), Vj를 머리(head)라고 한다.
    - ✓ <Vi, Vj>와 <Vj, Vi>는 서로 다른 간선

그래프의 정의

◆ 그래프의 종류



$V(G_1)$ = {A,B,C,D,E,F,G}

$E(G_1)$ = {(A,B),(A,C),(B,D),(B,E),(C,F),(C,G)}

$V(G_2)$ = {A,B,C,D}

$E(G_2)$ = {(A,B),(A,C),(A,D),(B,C),(B,D),(C,D)}

$V(G_3)$ = {A,B,C}

$E(G_3)$ = {<A,B>,<B,A>,<B,C>}

$V(G_4)$ = {A,B,C,D}

$E(G_4)$ = {<A,B>,<A,D>,<B,C>,<B,D>,<C,A>,<D,C>}

◆ 해당하는 그래프를 그리시오.

- V(G5)={1,2,3,4}
- E(G5) = {(1,3),(2,3),(2,4)}

- V(G6)={1,2,3,4}
- E(G6) = {<1,2>,<2,3>,<1,4>,<4,2>}

그래프 용어

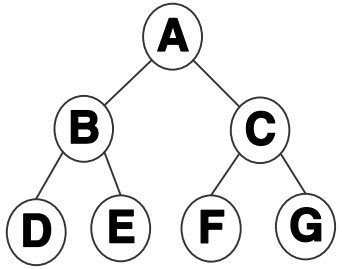
◆ 인접(adjacent)

- 정점의 순서쌍(V1, V2)가 E(G)에 있는 연결선이면,
  정점 V1 과 V2는 인접되어 있다.
  G2에서 정점B에 인접한 정점들은 A, C, D 이다

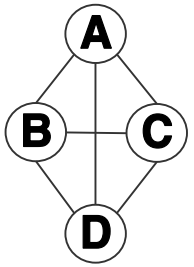
◆ 부속(incident)

- 정점의 순서쌍(v1, v2)가 E(G)에 있는 연결선이면,
  연결선(V1,V2)는 정점 V1과 V2에 부속되었다 한다.

G2에서 연결선 (A,B) (A,C) (A,D)는 정점 A에 부속되어 있는 연결선이다.



G1



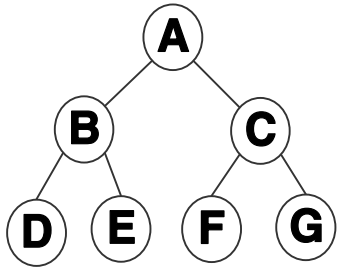
G2

그래프 용어

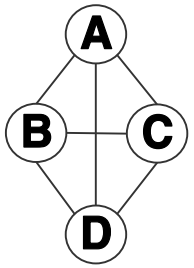
◆경로(path)

- 그래프 G의 정점 V1에서 VN에 이르는 정점들의 순서
- 경로상의 연결선의 수
- G1에서 정점 A에서 E로의 경로는 ABE 고, 이 경로의 길이(연결선의 수)는 2이다.

  G2에서 A에서 D로의 경로는 AD, ABD, ABCD, ACBD, ACD



G1



G2

그래프 용어

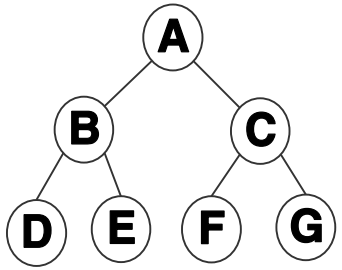
◆ 단순 경로(SIMPLE PATH)

- 경로상에 있는 정점들 중에서 첫 번째와 마지막 정점을 제외하고
  모든 정점들이 서로 다를 때

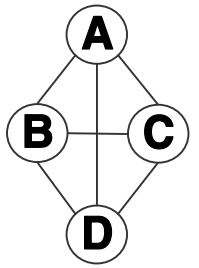
- G2에서 경로 ((AB), (BD), (DC))를 단순경로 (ABDC) OR ABDC 로 표현
- G1에서 정점 A에서 E로의 경로 ABE는 단순경로, ABEBE는 단순 경로가 아니다.

◆ 사이클(CYCLE)

- 처음 정점과 마지막 정점이 같은 단순 경로
- ABCDA 는 단순경로로 사이클



G1



G2

그래프 용어

◆차수(degree)

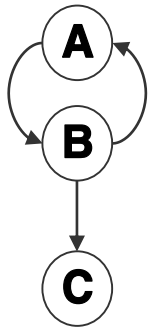
- 그래프 G(V,E)에서 한 정점에 교차된 연결선들의 수
- G2에서 정점 B에 교차된 연결선의 수는 3, 정점 B의 차수는 3

◆진입 차수(indegree) = 내차

- 방향성 그래프에서 한 정점에 들어오는 연결선의 수 G3에서 B의 진입차수는 1

◆진출차수(outdegree) = 외차

- 방향성 그래프에서 한 정점에서 나가는 연결선의 수 G3에서 B의 진출차수는 2
- 방향 그래프의 정점의 차수 = 진입차수 + 진출차수

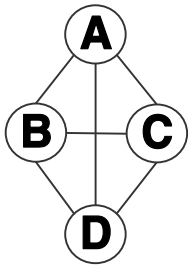


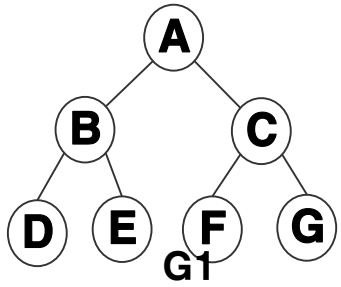
G3

◆연결(connected)

- 무방향그래프에서 두 정점 Vi와 Vj사이에 경로가 존재하면 Vi와 Vj는 연결되었다고 정의함
- 즉 G1에서 A와 D는 연결되었다.

G2



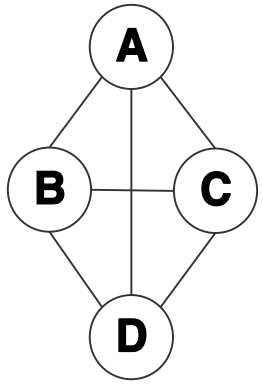


G1

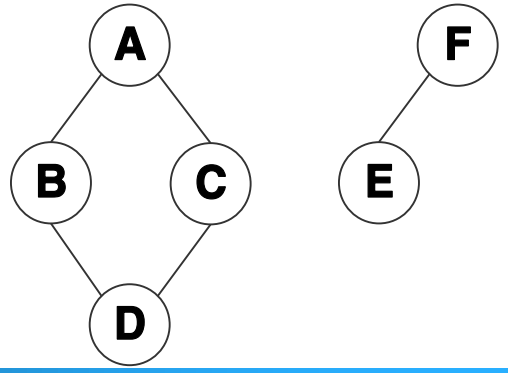
그래프 용어

◆ 연결 그래프(connected graph)

- 그래프 G에 속하는 모든 정점들이 연결되어 있어서 임의의 두 정점 Vi와 Vj 에 대하여 경로가 존재하는 그래프



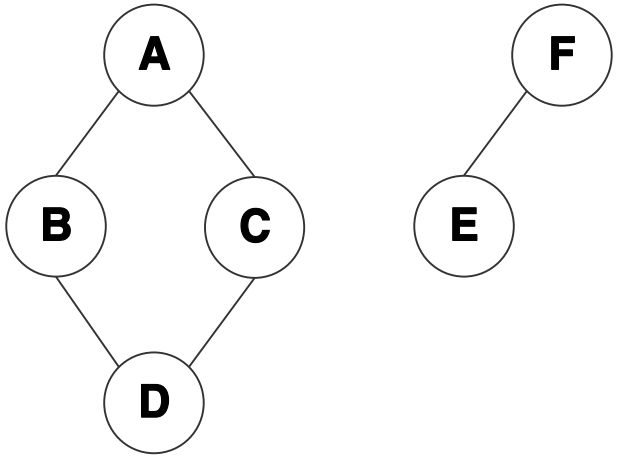
◆ 단절그래프 (disconnected graph)



그래프 용어

◆ 연결 요소 (connected component)

- 그래프 G에서 최대 연결부분 그래프

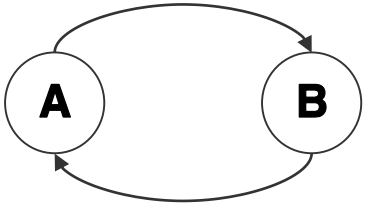


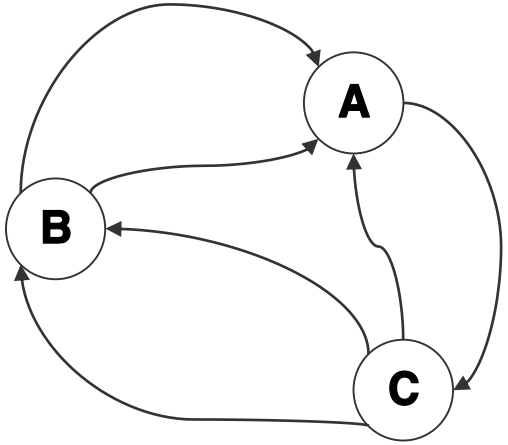
- H1, H2의 두 연결 요소가 있다.

그래프 용어

◆ 강력 연결 그래프 (strongly connected graph)

- 방향그래프 G에서 V(G)에 속한 상이한 두 정점 Vi, Vj의 모든 쌍에 대해 Vi → V와 Vj → Vi 로 경로가 존재하는 그래프

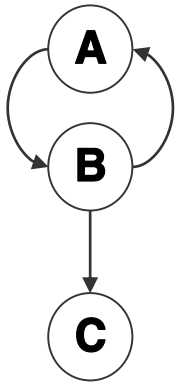




부분 그래프

◆ 부분 그래프(subgraph)

- 원래의 그래프에서 일부의 정점이나 간선을 제외하여 만든 그래프
- 그래프 G와 부분 그래프 G'의 관계
  - $V(G') \subseteq V(G)$, $E(G') \subseteq E(G)$
- 그래프 G1에 대한 부분 그래프의 예



G3

G3의 부분 그래프

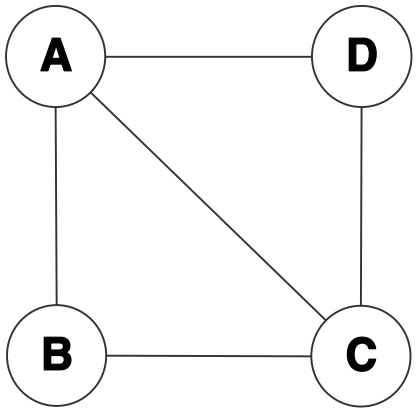
완전 그래프

◆ 완전 그래프(complete graph)

- 각 정점에서 다른 모든 정점을 연결하여 가능한 최대의 간선 수를 가진 그래프
- 정점이 n개인 무방향 그래프에서 최대의 간선 수 : n(n-1)/2개
- 정점이 n개인 방향 그래프의 최대 간선 수 : n(n-1)개
- 완전 그래프의 예
  - G5는 정점의 개수가 4개인 무방향 그래프이므로 완전 그래프가 되려면 4(4-1)/2=6개의 간선 연결
  - G6은 정점의 개수가 4개인 방향 그래프이므로 완전 그래프가 되려면 4(4-1)=12개의 간선 연결

그래프의 표현 – 인접 행렬 (adjacency matrix)

- 정점집합 V(G) ={V1,V2,…, Vn}인 그래프 G=(V(G), E(G))에서 각 정점들간의 인접여부를 n*n의 2차원 배열로 표현한 것.
- A(i,j) = 1 : Vi, Vj가 인접한 경우, 정점간에 연결선이 있는 경우
  0 : 두 정점간에 연결선이 없는 경우



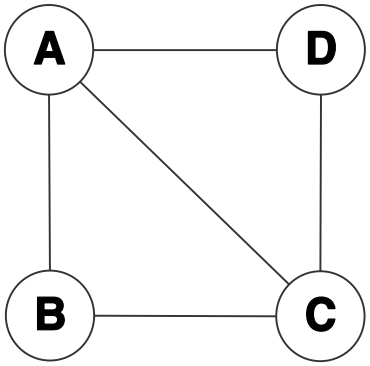
	A	B	C	D
A	0	1	1	1
B	1	0	1	0
C	1	1	0	1
D	1	0	1	0

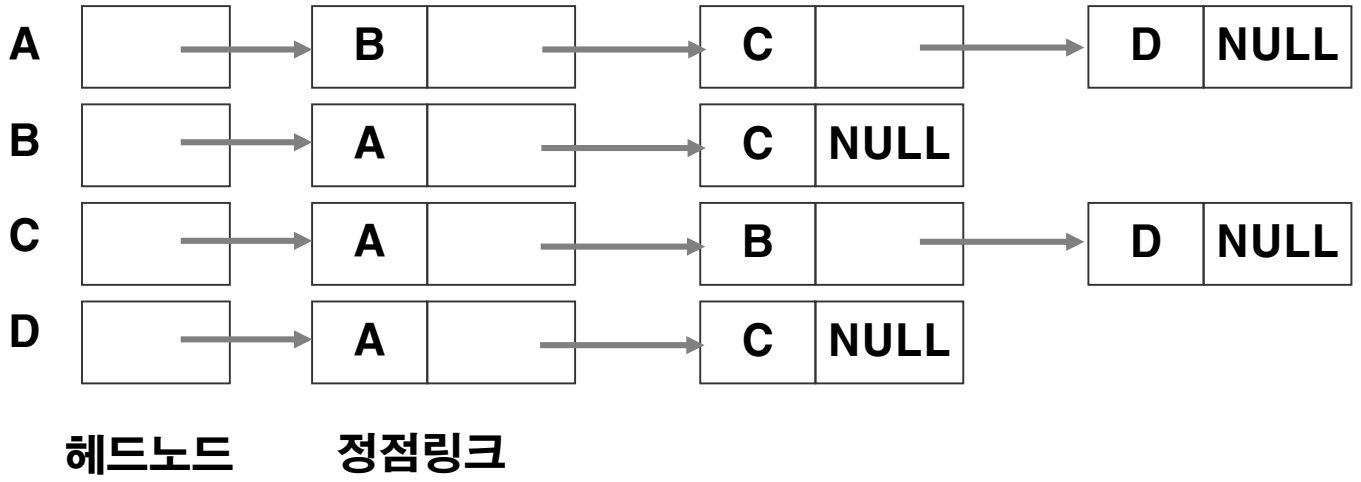
그래프의 표현 – 인접 리스트(adjacency list)

- 그래프 G에 속하는 정점들의 포인터를 1차원배열로 표현한 것
- 각 정점에 대한 인접 정점들을 연결하여 만든 연결 리스트
- 각 정점의 차수만큼 노드를 연결
- 리스트 내의 노드들은 인접 정점에 대해서 오름차순으로 연결

그래프의 표현 – 인접 리스트(adjacency list)

◆ 인접 리스트의 표현





그래프 연산

◆ 그래프 운행(graph traversal), 그래프 탐색(graph search)

- 하나의 정점에서 시작하여 그래프에 있는 모든 정점을 한번씩 방문하여 처리하는 연산
- 그래프 탐색방법
  - 깊이 우선 탐색(depth first search : DFS)
  - 너비 우선 탐색(breadth first search : BFS)

깊이 우선 탐색 (depth first search : DFS)

◆ 운행 방법

- 시작 정점의 한 방향으로 갈 수 있는 경로가 있는 곳까지 깊이 탐색해 가다가 더 이상 갈 곳이 없게 되면 가장 마지막에 만났던 갈림길 간선이 있는 정점으로 되돌아와서 다른 방향의 간선으로 탐색을 계속 반복하여 결국 모든 정점을 방문하는 순회방법

  ➔ 전위 운행과 유사

- 가장 마지막에 만났던 갈림길 간선의 정점으로 가장 먼저 되돌아가서 다시 깊이 우선 탐색을 반복해야 하므로 후입선출 구조의 스택 사용

- 미로 탐색과 유사

깊이 우선 탐색 (depth first search : DFS)

◆ 깊이 우선 탐색의 수행 순서

(1) 시작 정점 v를 결정하여 방문한다.

(2) 정점 v에 인접한 정점 중에서

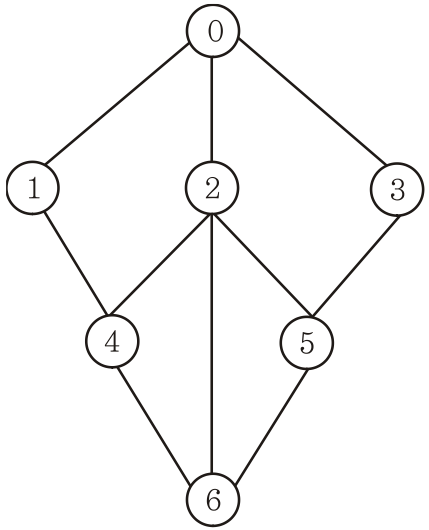
① 방문하지 않은 정점 w가 있으면, 정점 v를 스택에 push하고 w를 방문한다. 그리고 w를 v로 하여 다시 (2)를 반복한다.

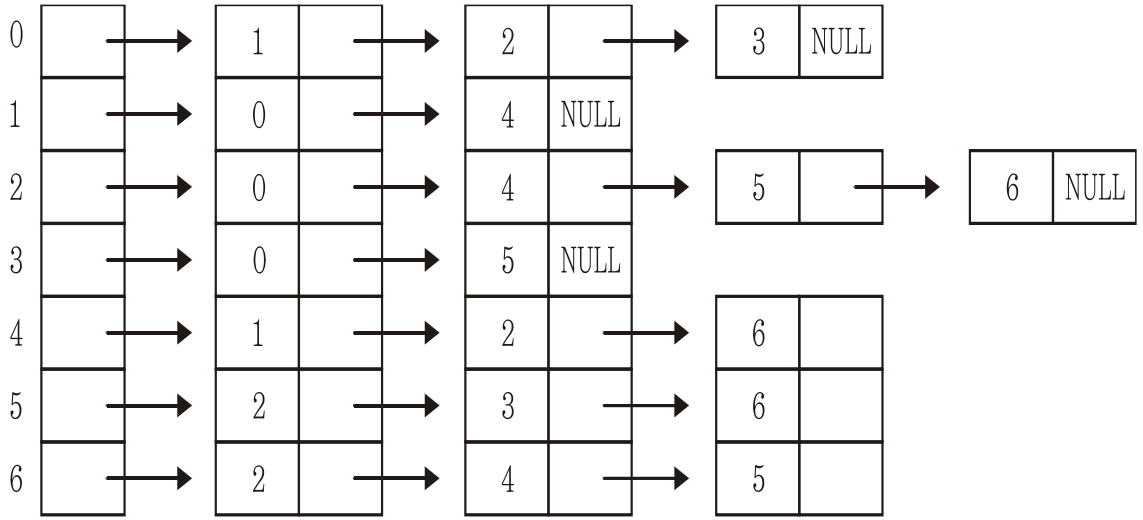
② 방문하지 않은 정점이 없으면, 탐색의 방향을 바꾸기 위해서 스택을 pop하여 받은 가장 마지막 방문 정점을 v로 하여 다시 (2)를 수행한다.

(3) 스택이 공백이 될 때까지 (2)를 반복한다.

깊이 우선 탐색의 예

- **깊이우선탐색 : 0 1 4 2 5 3 6**





너비 우선 탐색 (breadth first search : BFS)

◆ 운행 방법

- 시작 정점으로부터 인접한 정점들을 모두 차례로 방문하고 나서, 방문했던 정점을 시작으로 하여 다시 인접한 정점들을 차례로 방문하는 방식
- 가까운 정점들을 먼저 방문하고 멀리 있는 정점들은 나중에 방문하는 순회방법
- 인접한 정점들에 대해서 차례로 다시 너비 우선 탐색을 반복해야 하므로 선입선출의 구조를 갖는 큐를 사용

너비 우선 탐색 (breadth first search : BFS)

◆ 너비 우선 탐색의 수행 순서

(1) **시작 정점 v를 결정하여 방문한다.**

(2) **정점 v에 인접한 정점들 중에서 방문하지 않은 정점을 차례로 방문하면서 큐에 e 삽입한다.**

(3) **방문하지 않은 인접한 정점이 없으면, 방문했던 정점에서 인접한 정점들을 다시 차례로 방문하기 위해서 큐에서 삭제하여 구한 정점에서 (2)를 반복한다.**

(4) **큐가 공백이 될 때까지 (2)~(3)을 반복한다.**

너비 우선 탐색의 예

◆ 너비 우선 탐색 : 0 1 2 3 4 5 6