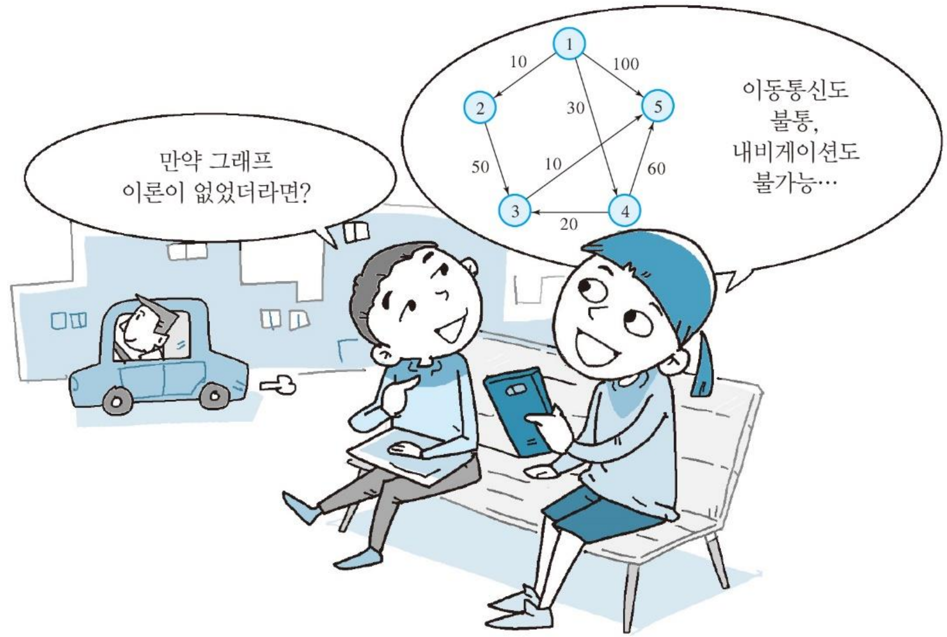
07 CHAPTER

그래프

Graph Theory & Its Applications



단원의 주요 목표

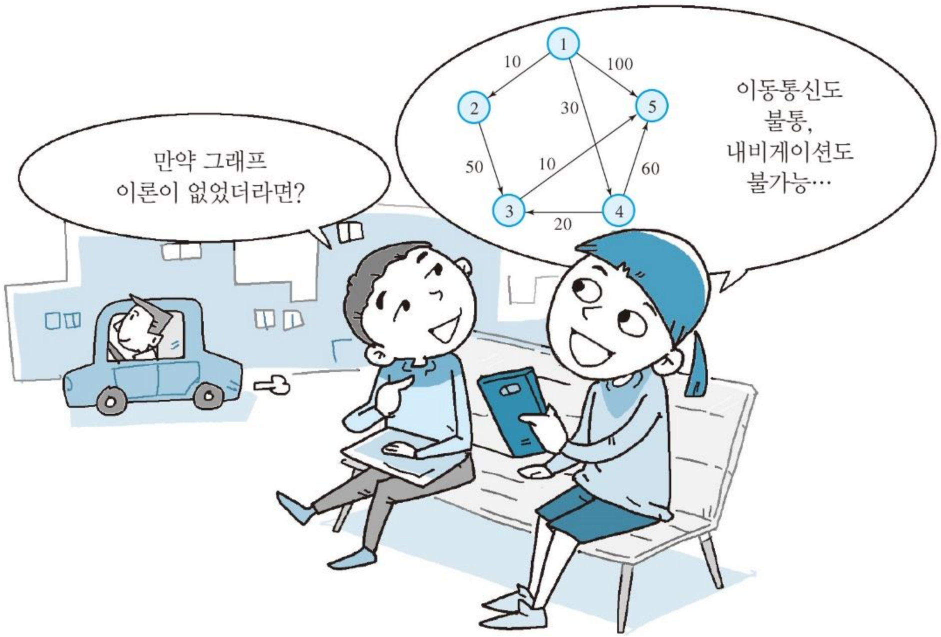
그래프의 의미를 이해하고 그래프와 관련된 다양한 논제들을 고찰한다.

- 그래프의 기본 개념과 용어를 정의한다.
- 그래프에서 사용되는 다양한 용어들을 파악한다.
- 인접 행렬과 인접 리스트를 통한 그래프의 표현 방법을 이해한다.
- 완전 그래프, 오일러 경로와 같은 특수한 형태의 그래프를 알아본다.
- 최단 경로 문제, 그래프 탐색 방법 등을 학습한다.
- 그래프의 응용 분야와 4차 산업혁명과의 관계를 살펴본다.

07 CHAPTER

그래프

Graph Theory & Its Applications

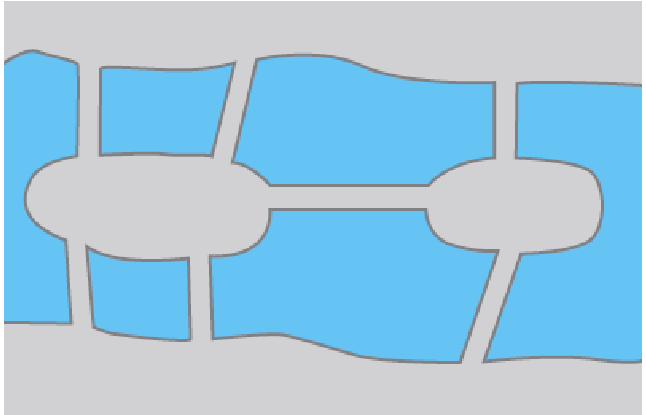


CONTENTS

7.1 그래프의 기본 개념

그래프 이론

- 18세기 스위스 출신의 저명한 수학자 오일러(Leonhard Euler, 1707~1783)에 의해 그래프 이론이 본격적으로 시작됨
- **그래프 이론**의 대표적인 예인 쾨니히스베르크(Königsberg) 다리 문제는 두 개의 섬과 강둑 사이를 연결하는 7개의 다리가 있을 때 각 다리를 꼭 한 번씩만 건너는 경로를 찾는 문제임



〈그림 7.1〉 쾨니히스베르크 다리 문제

7.1 그래프의 기본 개념

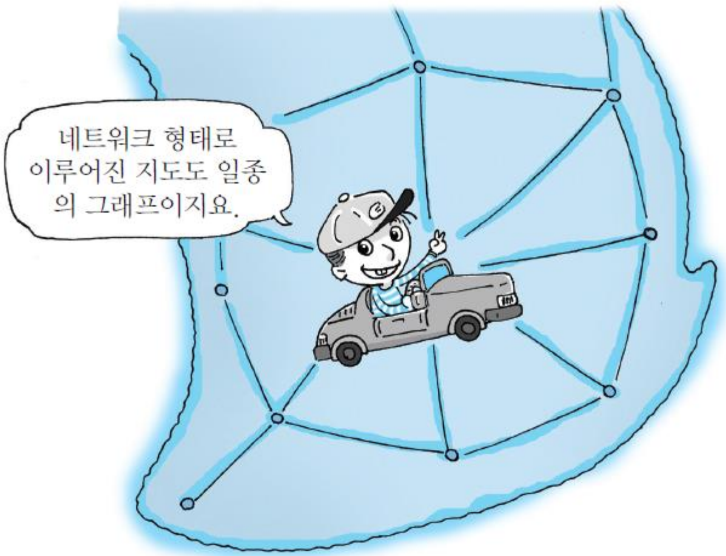
여기서 잠깐!!

그래프의 응용 분야

- 통신 네트워크
- 논리 회로의 설계 및 분석
- 최단 경로 찾기나 최단 거리 순회 문제
- 시스템의 흐름도나 컴파일러에서의 최적화
- 그래프 탐색, 정렬
- 알고리즘
- 전기회로망 설계와 응용
- 순서도를 통한 작업 계획의 분석
- 분자구조식 설계와 화학결합의 표시
- 유한 상태 기계
- 유전학, 언어학, 사회과학 등

7.1 그래프의 기본 개념

정의 7-1 그래프(graph) $G = (V, E)$는 유한한 개수의 정점(vertex) 또는 노드(node)들의 집합인 V와 연결선(edge) 또는 에지라고 불리는 정점들의 쌍들의 집합인 E로 이루어진다.



7.1 그래프의 기본 개념

그래프의 2가지 종류

방향 그래프(directed graph 또는 digraph)

- 방향이 있는 그래프임
- 연결선을 화살표로 표시하여 방향을 나타내는 그래프임

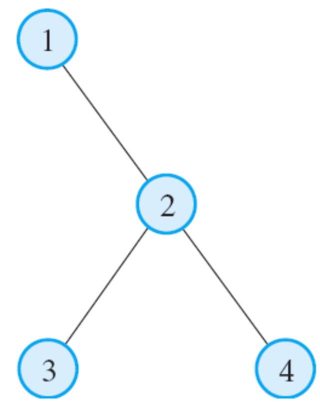
방향이 없는 그래프(undirected graph)

- 방향이 없는 그래프임
- 그래프의 특수한 형태이므로 특별한 언급이 없는 한 그래프는 방향이 없는 그래프를 의미함

7.1 그래프의 기본 개념

그래프의 간단한 예

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



〈그림 7.2〉 그래프의 간단한 예

경로(path)

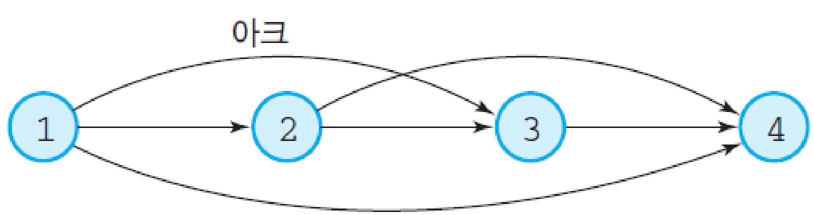
- 모든 $1 \le i < k$에 대해 연결선 (v_i, v_{i+1})이 존재할 때, 정점들의 열(sequence) $v_1, v_2, v_3, \cdots, v_k$라고 함
- $k \geq 1$이며 이 경로의 길이는 $k - 1$임

사이클(cycle)

$v_1 = v_k$ $(k \neq 1)$이면 이러한 경로를 사이클이라고 함

7.1 그래프의 기본 개념

정의 7-2 방향 그래프(directed graph) 또는 다이그래프(digraph)는 $G = (V, E)$로 표시된다. 여기서 V는 정점들의 집합이며, E는 정점들의 순서화된 쌍인 아크(arc) 또는 연결선들의 집합이다. 정점 v에서 정점 w로 가는 아크는 $v \rightarrow w$로 표시한다.



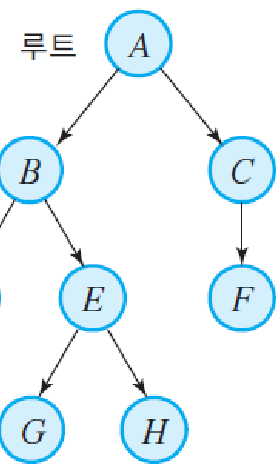
〈그림 7.3〉 방향 그래프$(\{1, 2, 3, 4\}, \{v \rightarrow w \mid v < w\})$

방향 그래프의 예

- 1→ 2 → 3 → 4는 정점 1에서 정점 4로 가는 경로임
- v → w인 아크에 대하여 v를 w의 **선행자(predecessor)**라 하며 w를 v의 **후속자(successor)**라고 함

트리(Tree)

- 사이클(cycle)이 존재하지 않는 그래프임
- **루트(root)**라 불리는 특별한 노드가 한 개 존재하고, 루트로부터 다른 모든 노드로 가는 경로가 항상 유일하게 존재함
- 루트로 들어오는 연결선이 없으므로 루트는 모든 트리의 출발점이 됨

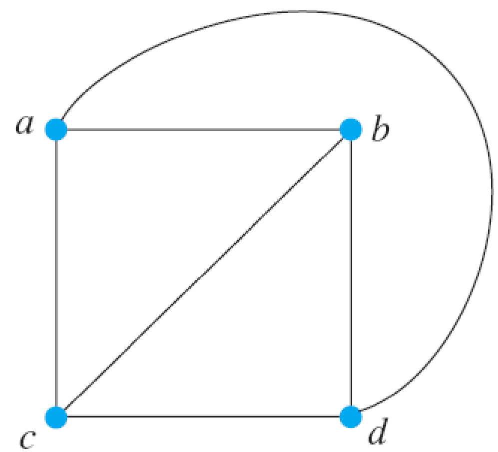


〈그림 7.4〉 트리

7.2 그래프의 용어

정의 7-3 **단순 그래프(simple graph)**

한 쌍의 정점 사이에 많아도 하나의 연결선으로 이루어진, 우리가 통상 다루는 그래프로서 자기 자신으로의 연결선이 없는 그래프이다.

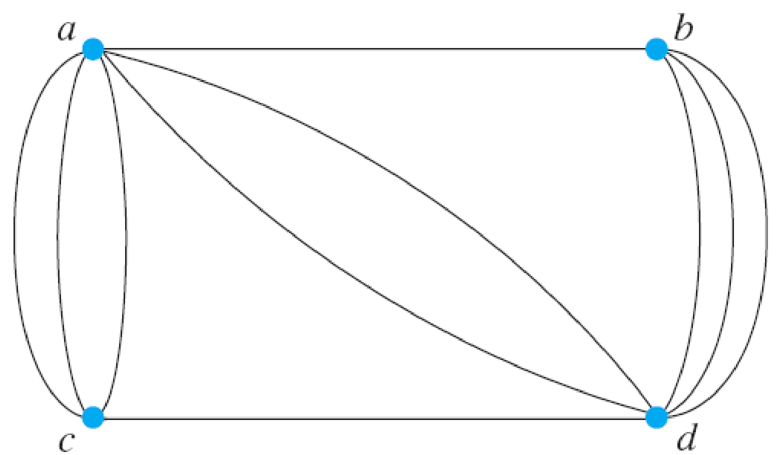


〈그림 7.5〉 단순 그래프의 예

7.2 그래프의 용어

정의 7-4 멀티 그래프(multigraph)

단순 그래프의 확장으로서 한 쌍의 정점 사이에 연결선의 개수의 제한이 없는 일반적인 그래프이다.



〈그림 7.6〉 멀티 그래프의 예

7.2 그래프의 용어

정의 7-5

그래프 $G=(V, E)$에서 순서화된 쌍 E를 그래프의 연결선(edge)이라고 한다. $(u, v) \in E$일 때 u와 v를 연결하는 연결선 e는 u와 v에 '접했다'(incident)라 하고 u와 v를 서로 '인접했다'(adjacent)라고 한다.

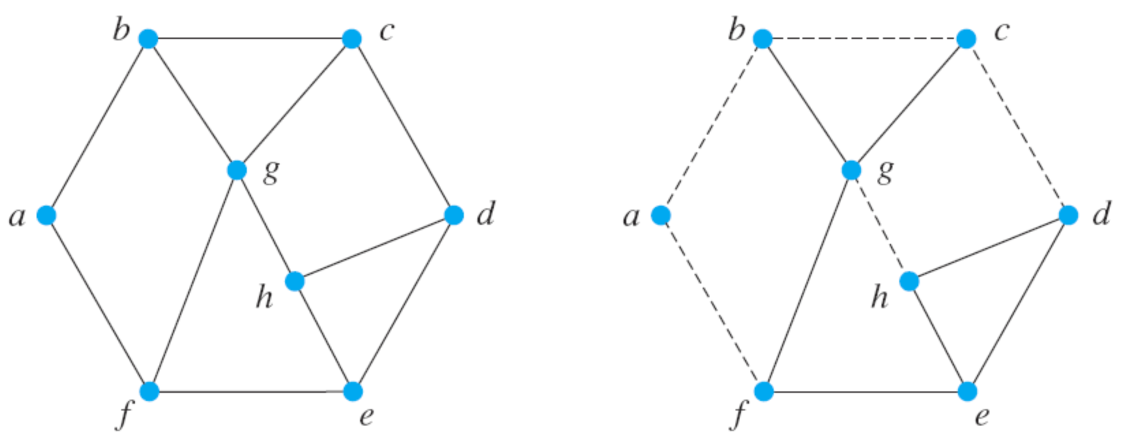
정의 7-6

$G=(V, E)$가 그래프이고 u, v가 정점이라고 할 때 v의 차수(degree) $d(v)$는 v에 인접하는 연결선들의 개수를 말한다.

7.2 그래프의 용어

정의 7−7 두 개의 그래프 $G=(V, E)$, $G'=(V', E')$에서 $V' \subseteq V$, $E' \subseteq E$일 때 그래프 $G'=(V', E')$를 G의 **부분 그래프(subgraph)**라고 한다. 이때 $V=V'$이고 $E' \subset E$이면 G'는 G의 **생성 부분 그래프(spanning subgraph)**라고 한다.

예제 7−1 다음 그림의 (2)는 $V' \subseteq V$, $E' \subseteq E$이므로 (1)의 부분 그래프이다. 또한 (2)는 $V=V'$이고 $E' \subset E$이므로 (1)의 생성 부분 그래프도 된다.



(1) (2)

7.2 그래프의 용어

정의 7-8 그래프에서의 경로는 경로, 단순 경로, 기본 경로, 사이클, 단순 사이클, 기본 사이클 등으로 구분된다.

(1) 경로(path)

- 그래프에서의 경로란 꼭지점의 열 $v_1, v_2, \cdots, v_n$에서 $(v_{k-1}, v_k) \in E$, $1 \le k < n$인 경우를 말함
- 경로의 길이(length)는 n-1임

(2) 단순 경로(simple path)

- 경로가 같은 연결선을 두 번 포함하지 않는 경로를 말함

(3) 기본 경로(elementary path)

- 어떤 정점들도 두 번 만나지 않는 경로를 말함

7.2 그래프의 용어

(4) 사이클(cycle) 또는 순회(circuit)

- 경로 (v_1, v_2, , v_n) 에서 종점 v_n과 시점 v_1이 일치하는 경우를 말함

(5) 단순 사이클(simple cycle)

- 같은 연결선을 반복하여 방문하지 않는 사이클을 말함

(6) 기본 사이클(elementary cycle)

- 시작점을 제외한 어떠한 정점도 반복하여 방문하지 않는 사이클을 말함

7.2 그래프의 용어

정의 7-9 그래프에서 정점들 간의 연결 여부에 따라 연결 그래프, 강한 연결 그래프 등이 정의된다.

(1) 연결 그래프(connected graph)

그래프의 모든 정점들이 연결되어 있는 그래프임

(2) 강한 연결 그래프(strongly connected graph)

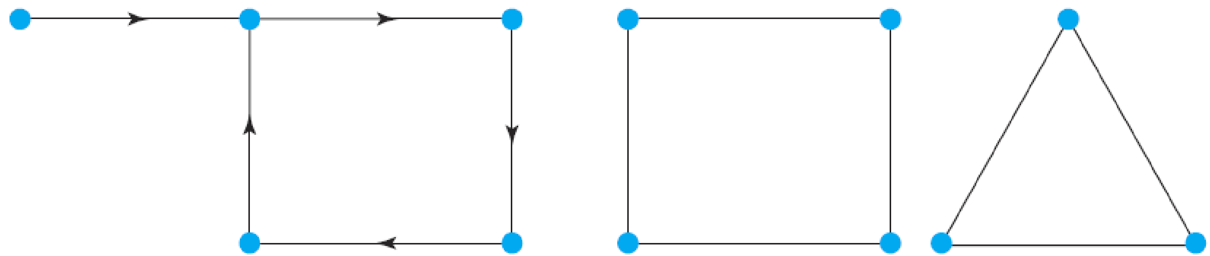
그래프에서 모든 두 정점 a와 b에 대해서 a에서 b로의 경로와 b에서 a로의 경로들이 존재하는 그래프를 말하는데, 특히 방향 그래프에서만 의미를 가짐

(3) 연결 요소(connectivity component)

그래프에서 모든 정점들이 연결되어 있는 부분을 말하며, 연결 수(connectivity number)란 그래프 G에서의 연결 요소의 개수를 말함

7.2 그래프의 용어

예제 7-2 다음 그림이 연결 그래프인지 강한 연결 그래프인지를 판단해보자.

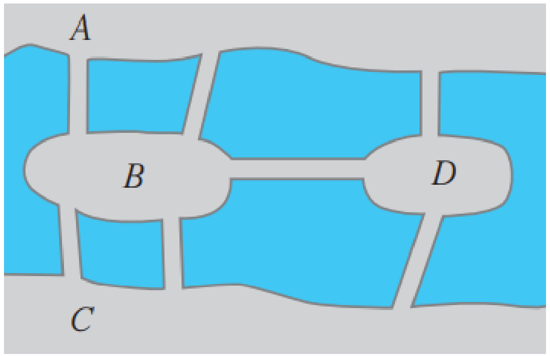


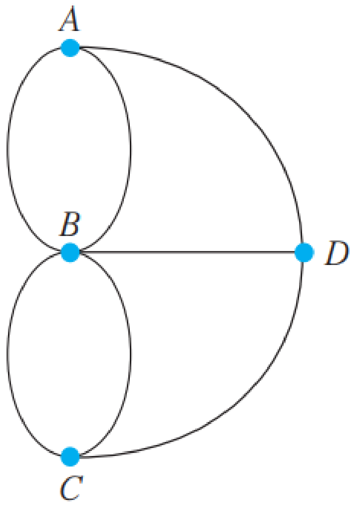
(1) (2)

풀이 (1)은 연결 그래프이나 강한 연결 그래프는 아니다. 왜냐하면 연결선의 반대 방향으로의 경로가 없기 때문이다. (2)는 그래프가 서로 떨어져 있기 때문에 연결 그래프가 아니다.

7.2 그래프의 용어

- **멀티 그래프(Multi-graph)**란 한 쌍의 정점 사이에 연결선의 개수의 제한이 없는 중복된 연결선을 허용하는 특별한 그래프임
- 쾨니히스베르크(Königsberg) 다리 문제를 해결하는 방법은 멀티 그래프를 모델링하는 것임





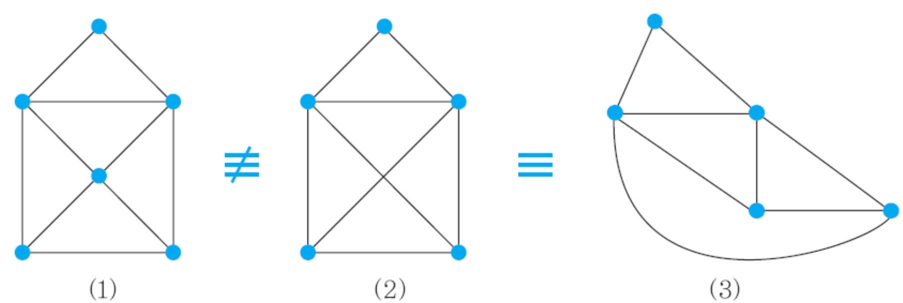
〈그림 7.7〉 쾨니히스베르크 다리 문제의 멀티 그래프적인 표현

7.2 그래프의 용어

- 쾨니히스베르크 다리를 그래프로 나타낸 *A*, *B*, *C*, *D* 로 이름 붙인 점들은 정점을 나타내고, 정점들 사이의 선들은 연결선을 나타냄
- 멀티 그래프에서 모든 연결선들을 꼭 한 번씩만 통과하는 경로를 **오일러 경로(Eulerian path)**라고 하는데, 쾨니히스베르크 다리 문제는 오일러 경로를 찾을 수 있는지 여부와 동치임
- 멀티 그래프에서의 오일러 경로를 판별하는 규칙은 모든 정점에서 그것과 연결된 연결선의 개수가 홀수인 정점(**홀수점**)의 개수가 0 또는 2개인 경우임
- 그래프에서는 *A*, *B*, *C*, *D* 4개의 정점들이 모두 홀수점을 가지므로 (총 4개) 오일러 경로가 없다. 따라서 각 다리를 꼭 한 번씩만 건너는 경로가 존재하지 않음

7.2 그래프의 용어

- 정점들과 연결선으로 이루어진 도형에 대한 초기의 그래프 이론은 단순하고 직관적인 문제 해결에 국한됨
- 복잡한 모델에 대한 이론적인 연구로 발전하게 됨
- 기하학에 위상적인 개념을 접합시킨 응용으로서, **위상 기하학 (Topological Geometry)**적인 관계를 나타냄
- 그래프에서 점의 위치나 선의 길이 등에는 특별한 의미를 부여하지 않고 위상적인 형태에만 중점 둠
- 아래 그림의 (1)과 (2)는 위상적으로 서로 다르지만, (2)와 (3)은 위상적으로 같음



〈그림 7.8〉 위상기하학적 관계

7.3 그래프의 표현

그래프의 표현 방법

- 그래프는 그림을 이용하여 표현하는 것으로 가장 자연스럽고 이해하기에도 가장 쉬운 방법임
- 컴퓨터는 그림으로 표현된 정보를 이용할 수 없기 때문에 통상 인접 행렬이나 인접 리스트에 의해 표현됨
- 이를 통하여 컴퓨터 프로그램으로 구현됨

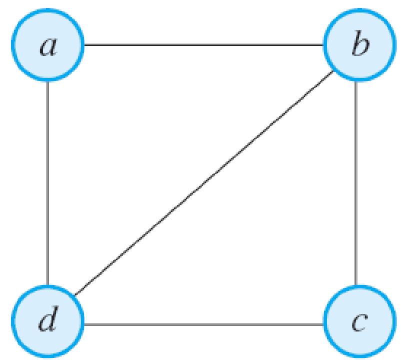
(1) 인접 행렬(Adjacency Matrix)

그래프 G = (V, E)에서 |V| = n일 때 G의 인접 행렬은 n × n 행렬 A로 나타내어지며, 이때 A의 각 원소 a_{ij}는 다음과 같이 정의됨

$$a_{ij} = \begin{cases} 1 & (v_i, v_j) \in E \\ 0 & \text{otherwise} \end{cases}$$

7.3 그래프의 표현

예제 7-3 다음과 같은 그래프의 인접 행렬을 구해보자.



풀이 정점의 개수가 4개이므로 다음과 같은 4×4 행렬로 만들어진다.

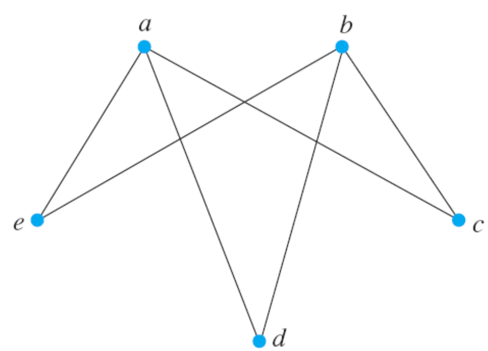
$$
\begin{array}{c c}
 & \begin{array}{cccc} a & b & c & d \end{array} \\
\begin{array}{c} a \\ b \\ c \\ d \end{array} & \begin{bmatrix} 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 1 & 1 & 1 & 0 \end{bmatrix}
\end{array}
$$

7.3 그래프의 표현

예제 7-4 다음과 같은 인접 행렬을 가지는 그래프를 그려보자.

$$
\begin{array}{c c}
 & \begin{array}{ccccc} a & b & c & d & e \end{array} \\
\begin{array}{c} a \\ b \\ c \\ d \\ e \end{array} &
\begin{bmatrix}
0 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 1 & 1 \\
1 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 0 & 0
\end{bmatrix}
\end{array}
$$

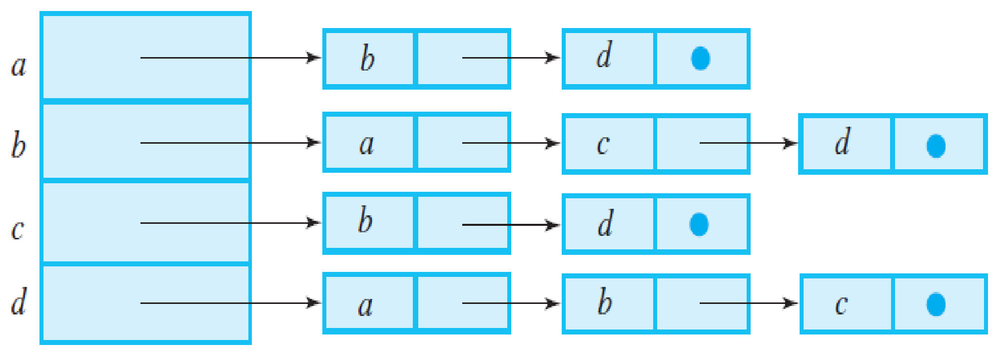
풀이 인접 행렬은 그래프에서 두 꼭지점 사이에 변이 있으면 1로, 그렇지 않으면 0으로 나타내므로, 그것에 따라 다음과 같은 그래프를 그릴 수 있다.



7.3 그래프의 표현

(2) 인접 리스트(Adjacency List)

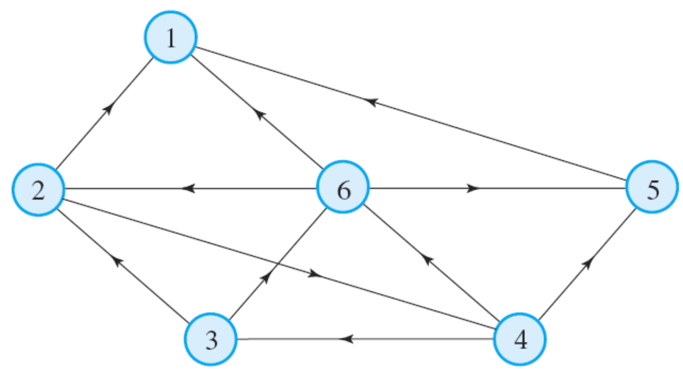
- 각 정점에 대해 **포인터(pointer)**가 주어지고, 그 점으로부터 연결된 정점들을 차례로 **연결 리스트(linked list)**로 표시함
- 같은 리스트 내에서는 순서에 관계가 없음



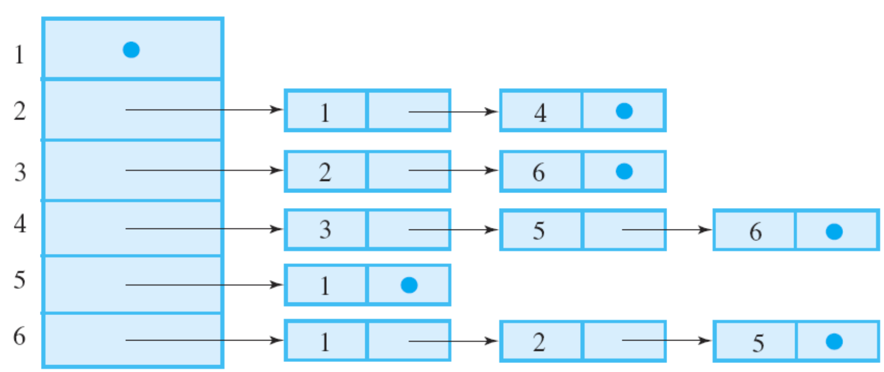
〈그림 7.9〉 인접 리스트

7.3 그래프의 표현

예제 7-5 다음의 방향 그래프에서 인접 리스트를 구해보자.



풀이 여기서는 정점 1에서 나가는 연결선이 없다.

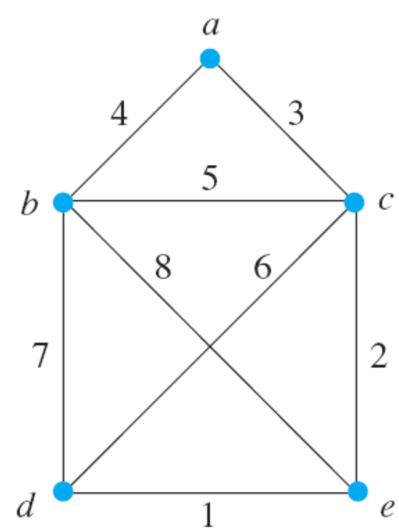


7.4 특수 형태의 그래프

정의 7-10 오일러의 성질을 만족하는 특수한 형태의 그래프인 오일러 경로와 오일러 순회는 다음과 같이 정의된다.

(1) **오일러 경로(Eulerian path)**란 그래프에서 각 연결선을 단 한 번씩만 통과하는 경로를 말한다.

(2) **오일러 순회(Eulerian circuit)**란 그래프에서 정점은 여러 번 지날 수 있지만, 각 연결선을 단 한 번씩만 통과하는 순회를 말한다.



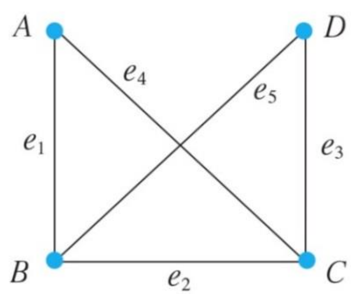
〈그림 7.10〉 오일러 경로

7.4 특수 형태의 그래프

정리 7-1

정점 v와 연결된 연결선의 개수를 정점 v의 차수(degree)라고 하며 $\deg(v)$로 나타낸다. 이 경우 각 연결선은 그래프의 차수를 계산할 때 두 번 반복해서 계산되므로 그래프에서 모든 정점들의 차수의 합은 연결선들 개수의 2배가 된다.

예제 7-6 다음 그래프에서 모든 정점들의 차수의 합은 연결선들 개수의 2배임을 보이자.



풀이 $\deg(A)=2,\ \deg(B)=3,\ \deg(C)=3,\ \deg(D)=2$

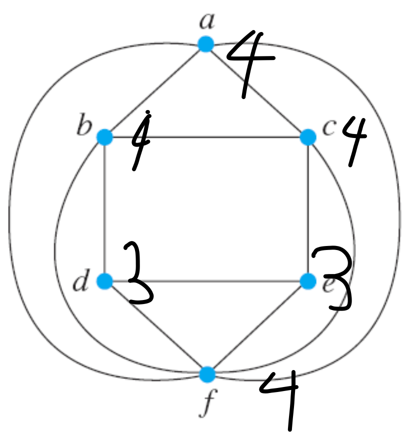
이므로, 차수의 합은 10이고 연결선은 5개이므로 차수의 합은 연결선 수의 두 배와 같다.

7.4 특수 형태의 그래프

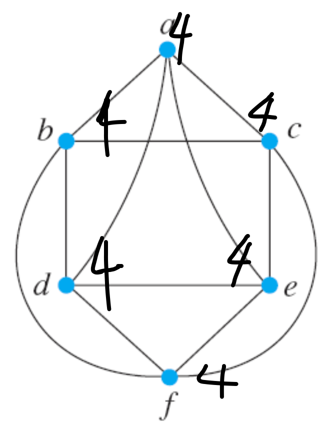
정리 7-2

어떤 그래프 G가 오일러 경로를 가지기 위한 필요충분조건은 G가 연결 그래프이고, 홀수 차수의 개수가 0 또는 2인 경우이다.

예제 7-7 다음과 같은 두 그래프들이 오일러 경로를 가지는지를 확인해보자.



(1)



(2)

풀이 (1)의 홀수 차수인 정점은 d와 e로서 그 개수가 2이고, (2)의 경우에는 홀수 차수인 정점의 개수가 0이므로 둘 다 오일러 경로를 가진다.

7.4 특수 형태의 그래프

정리 7-3

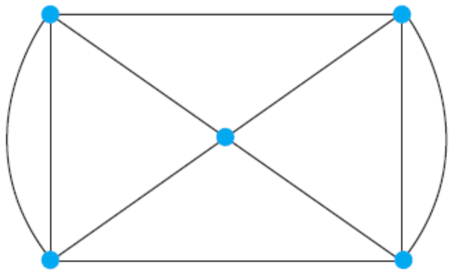
어떤 그래프 G가 오일러 순회를 가지기 위한 필요충분조건은 G가 연결 그래프이고, 모든 정점들이 짝수 개의 차수를 가지는 경우이다.

정의 7-11

그래프에서 연필을 떼지 않고 모든 변을 오직 한 번만 지나는 것을 **한붓그리기(traversable)**라고 한다. 연결된 그래프에서 한붓그리기가 가능하려면 시작점과 끝점을 제외한 모든 정점의 차수가 짝수이어야 한다.

예제 7-8

다음 그래프는 한붓그리기가 가능한지를 판별해보자.



풀이 각 정점의 차수가 모두 짝수 차수로서 홀수점이 존재하지 않는 오일러 경로이므로 한붓그리기가 가능하다.

7.4 특수 형태의 그래프

정의 7-12 해밀턴의 성질을 만족하는 특수한 형태의 그래프인 해밀턴 경로와 해밀턴 순회는 다음과 같이 정의된다.

(1) 해밀턴 경로(Hamiltonian path)란 그래프에서 모든 정점을 오직 한 번씩만 지나지만 시작점으로 돌아오지 않는 경로를 말한다.

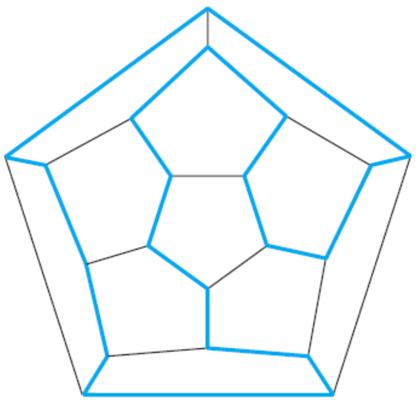
(2) 해밀턴 순회(Hamitonian circuit)란 그래프에서 모든 징점들을 오직 한 번씩만 지나는 순회를 말한다.

여기서 잠깐!!

그래프의 한 정점에서 이어진 변을 따라 또 다른 정점으로 이동할 때, 지나간 변을 반복해서 통과하지 않을 경우에 지나온 정점들을 순서대로 나열한 것을 경로라고 하고, 이때 한 정점에서 출발하여 다시 출발한 정점으로 돌아오는 경로를 순회라는 점에 유의하자.

7.4 특수 형태의 그래프

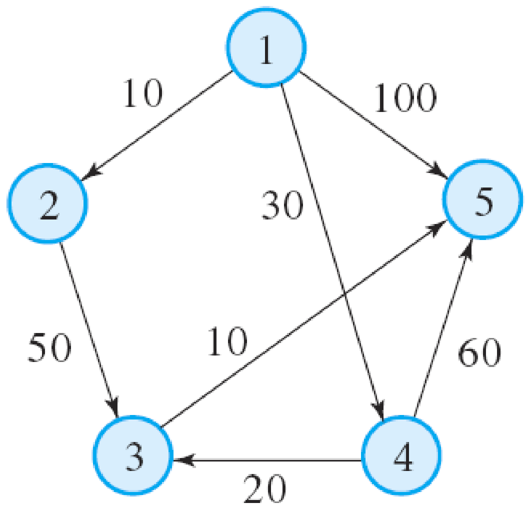
예제 7-9 다음 그래프에서 굵은 색선으로 표현된 연결선은 해밀턴 순회를 나타냄을 알아보자.



풀이 모든 정점들을 한 번씩만 거치면서 순회를 하기 때문에 해밀턴 순회이다.

7.4 특수 형태의 그래프

정의 7-13 그래프 G의 각 연결선에 0보다 큰 수가 할당되었을 때 이 값을 가중값(weight)이라고 하며, 이와 같은 그래프를 가중 그래프(weight graph)라고 한다.

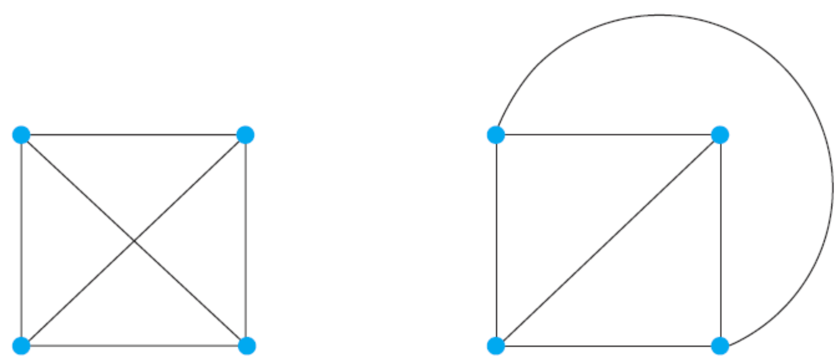


〈그림 7.11〉 가중 그래프

7.4 특수 형태의 그래프

정의 7-14 그래프 $G_1=(V_1, E_1)$과 $G_2=(V_2, E_2)$가 주어졌을 때, 전단사 함수 $f: V_1 \rightarrow V_2$가 존재하여 $\{u, v\} \in E_1 \Leftrightarrow \{f(u), f(v)\} \in E_2$이면 f를 동형(isomorphism)이라고 하고 G_1과 G_2를 **동형 그래프(isomorphic graph)**라고 한다.

정점, 차수의 개수가 같으면 동형

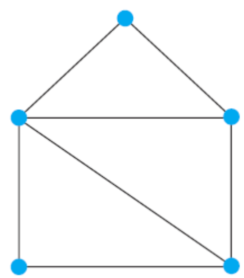


〈그림 7.12〉 동형 그래프의 예

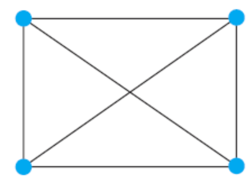
정의 7-15 평면상에서 어떠한 연결선들도 서로 교차할 수 없도록 그려진 하나의 그래프를 **평면 그래프(planar graph)**라고 한다.

7.4 특수 형태의 그래프

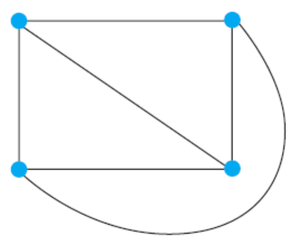
예제 7-10 다음 그래프들이 평면 그래프인지의 여부를 알아보자.



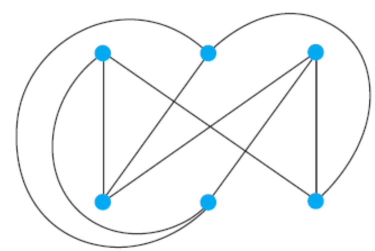
(1)



(2)



(3)



(4)

풀이 (1)은 어떤 연결선들도 서로 교차하지 않으므로 평면 그래프이다. (2)는 (3)과 같이 동형으로 변경할 수 있으므로 평면 그래프이다. 그러나 (4)는 평면 그래프가 아니다.

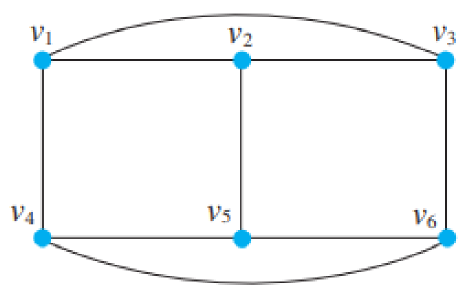
7.4 특수 형태의 그래프

정리 7−4

오일러의 정리

연결된 평면 그래프에서 정점의 수를 v, 연결선의 수를 e, 면의 수를 f라고 할 때 $v - e + f = 2$이다.

예제 7−11 다음 그래프에서 오일러의 정리가 성립하는지를 살펴보자.



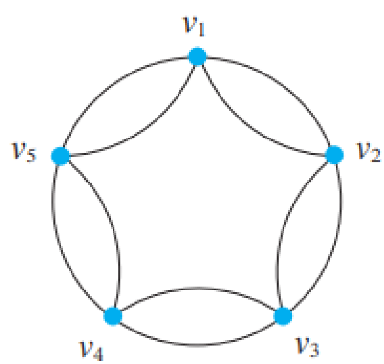
풀이 이 그래프에서 정점의 개수 $v = 6$, 연결선의 개수 $e = 9$, 면의 개수 $f = 5$이므로 $v - e + f = 2$인 오일러의 공식이 성립한다.

? $f+1$

여기서 잠깐!! 평면의 개수를 셀 때 그래프 바깥에 있는 평면도 세어야 함에 유의하자.

7.4 특수 형태의 그래프

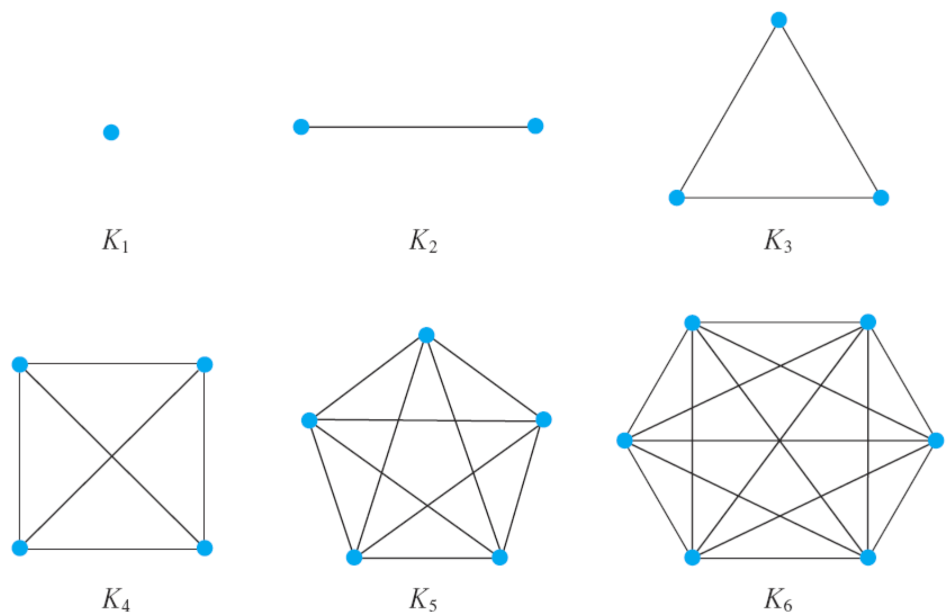
예제 7-12 다음 그래프에서 오일러의 공식을 이용하여 오일러의 정리가 성립하는지를 살펴보자.



풀이 이 그래프에서 정점의 개수 $v = 5$, 연결선의 개수 $e = 10$, 면의 개수 $f = 7$이므로 $v - e + f = 2$인 오일러의 공식이 성립한다.

7.4 특수 형태의 그래프

정의 7-16 그래프 $G=(V, E)$의 모든 정점들의 쌍 사이에 연결선이 존재하면 G를 **완전 그래프(Complete graph)**라고 한다. 즉, 각 정점이 다른 모든 정점들과 연결되는 그래프를 말하는데, n개의 정점으로 구성된 완전 그래프는 K_n으로 표기한다.



〈그림 7.13〉 완전 그래프의 예

K_6

$\frac{6\times5}{2}=15$

7.4 특수 형태의 그래프

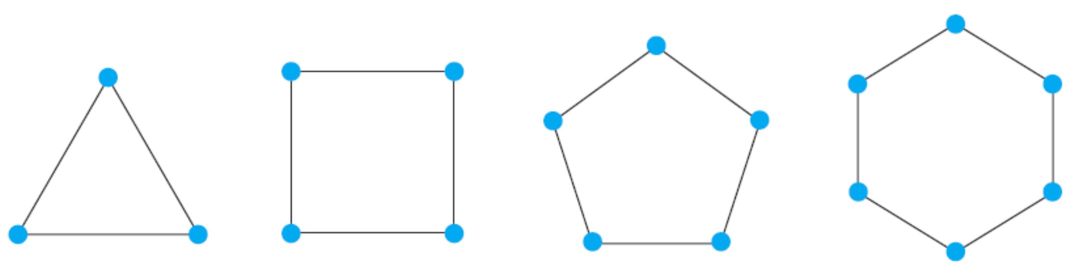
여기서 잠깐!!

n개의 정점을 가지는 완전 그래프에서의 연결선의 개수는 $\frac{n(n-1)}{2}$개이다. 만약 $n=5$인 경우에는 $\frac{5 \cdot 4}{2}=10$개의 연결선을 가진다.

정의 7−17 그래프 $G=(V, E)$의 모든 정점의 차수가 k이면 G를 k차 **정규 그래프**라고 한다.

(1) 0차 정규

(2) 1차 정규



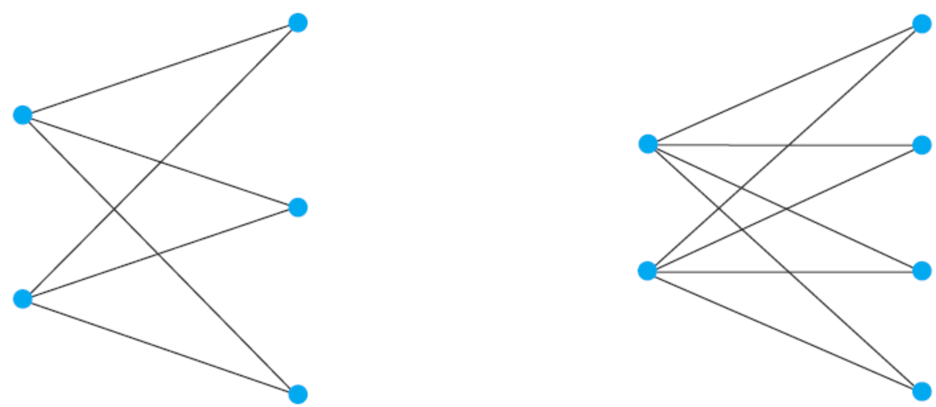
(3) 2차 정규

〈그림 7.14〉 k차 정규 그래프

7.4 특수 형태의 그래프

정의 7-18 그래프 $G=(V, E)$에서 V가 두 부분 집합 X와 $Y=V-X$로 나누어져 각 연결선이 X 내의 정점과 Y 내의 정점의 쌍으로 연결되면 그래프 G를 이분 그래프(bipartite graph)라고 한다. 이때 X 내의 모든 정점들과 Y 내의 모든 정점들 사이에 연결선이 존재하면 G를 완전 이분 그래프(complete bipartite graph)라고 한다.

X의 정점의 개수가 m이고, Y의 정점의 개수가 n일 때 완전 이분 그래프를 $K_{m,n}$으로 나타낸다. 완전 이분 그래프는 〈그림 7.15〉에 나타나 있다.

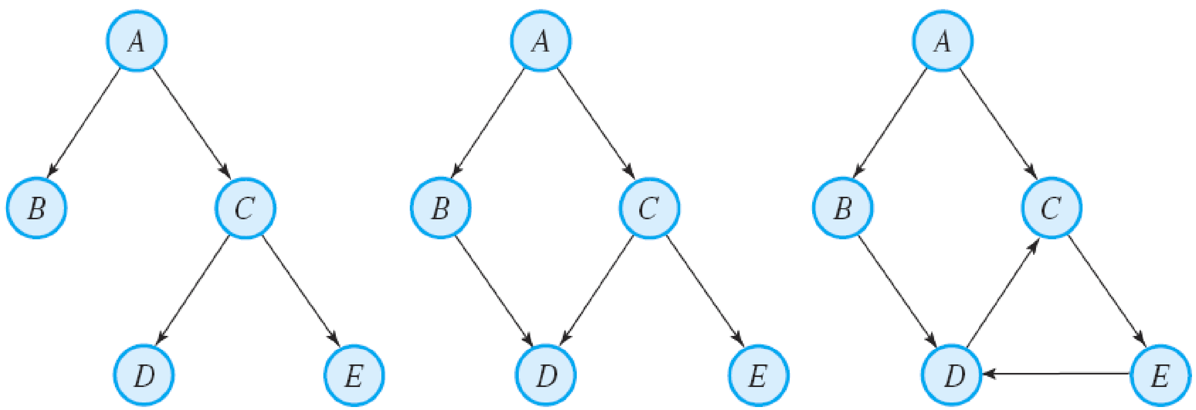


(1) $K_{2,3}$ (2) $K_{2,4}$

〈그림 7.15〉 완전 이분 그래프의 예

7.4 특수 형태의 그래프

정의 7-19 방향 비사이클 그래프(Directed Acyclic Graph : DAG)는 사이클이 없는 방향 그래프를 말하는데, 트리보다는 일반적이나 임의의 방향 그래프보다는 제한적이다.

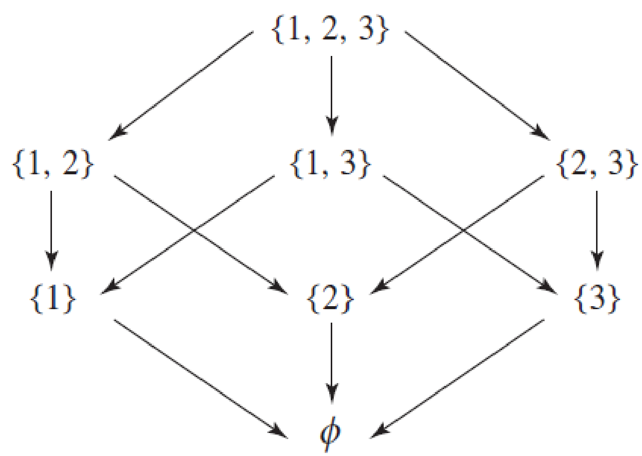


〈그림 7.16〉 트리, DAG, 사이클을 가진 방향 그래프

7.4 특수 형태의 그래프

예제 7-13 집합 {1, 2, 3}의 진부분 집합을 DAG로 표현해보자.

풀이 집합 {1, 2, 3}의 진부분 집합 중 2개의 원소로 이루어진 것은 {1, 2}, {1, 3}, {2, 3}이고, 이들의 진부분 집합은 {1}, {2}, {3}이다. 또한 ϕ은 모든 집합의 부분 집합이므로 DAG의 가장 마지막 부분에 위치하게 된다.

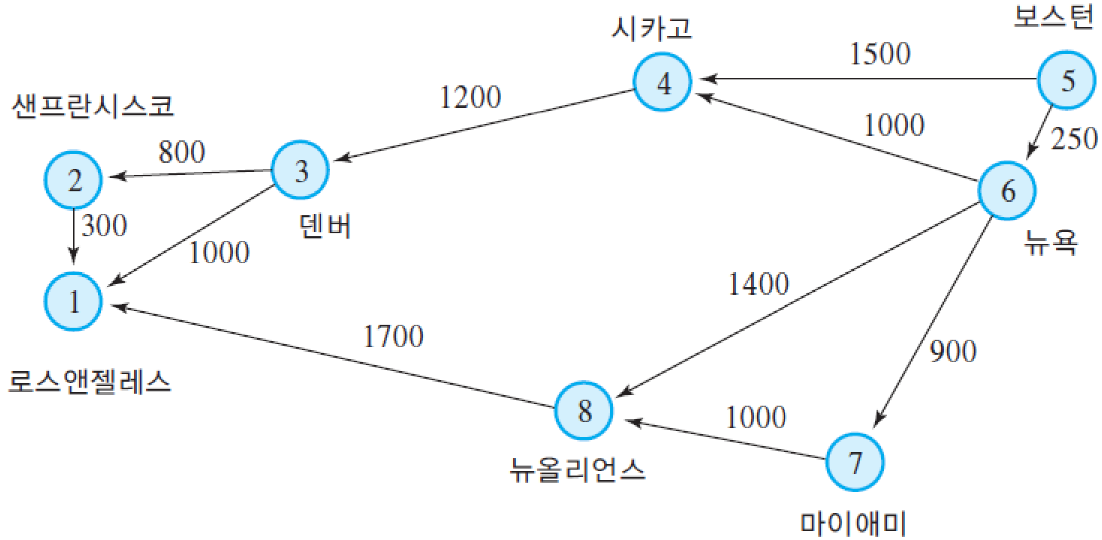


7.5 그래프의 응용

(1) 최단 경로

- 그래프는 지도에서 도시를 나타내는 점과 그들 도시 간의 거리를 나타내는 연결선의 값으로 표현됨
- 도시 A에서 도시 B로 가기 위한 방법
- 첫째, A에서 B로 가는 경로가 있느냐는 점
- 둘째, A에서 B로 가는 경로가 여러 개 있을 경우 어떤 경로로 가는 것이 가장 짧은 거리인가 하는 점
- 가장 짧은 거리의 경로를 찾는 문제를 **최단 경로 문제(shortest path problem)**라 함
- 주어진 방향 그래프에서 경로의 시작점을 **출발점(source)**이라 하며 목적지를 **도착점(destination)**이라고 함
- 주어진 연결선의 길이는 0 이상인 경우를 가정함

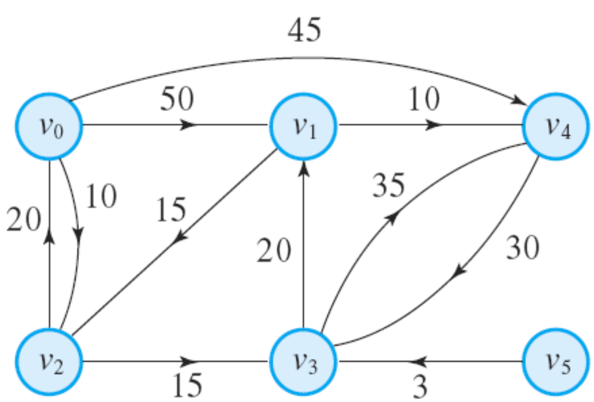
7.5 그래프의 응용



〈그림 7.17〉 지도에서의 방향 그래프

7.5 그래프의 응용

- 연결선의 값이 두 점 사이의 거리를 나타낼 때, v_0가 출발점이라 하면 v_0에서 v_1까지의 최단 경로는 $v_0v_2v_3v_1$임
- 이때 최단 거리는 10 + 15 + 20 = 45가 됨
- 경로의 길이는 3이지만 v_0에서 v_1로 직접 가는 거리인 50보다 짧음
- v_0에서 v_1, v_2, v_3, v_4로 가는 최단 경로의 값을 나타냄



〈그림 7.18〉 가중 그래프

	경로	거리
1)	$v_0v_2v_3v_1$	45
2)	v_0v_2	10
3)	$v_0v_2v_3$	25
4)	v_0v_4	45

〈그림 7.19〉 v_0에서의 최단 경로와 거리

7.5 그래프의 응용

- 최단 경로의 거리 문제를 해결할 수 있는 방법을 **다익스트라 알고리즘 (Dijkstra algorithm)**이라고 함
- 출발점으로부터 거리가 최소로 알려진 점들의 집합S를 유지함으로써 가장 짧은 거리를 가지는 나머지 점 v를 차례로 S에 포함시킴

다익스트라 알고리즘

- 주어진 방향 그래프 G = (V, E)에서 v = {1, 2, ..., n}이고 점 {1}이 출발점이라고 가정함
- 점 i에서 j로 가는 거리를 C[i, j]로 나타내는데, 만약 i에서 j로 가는 경로가 없으면 거리는 ∞가 됨
- D[i]는 출발점에서 현재점 i에 이르는 가장 짧은 거리를 나타냄

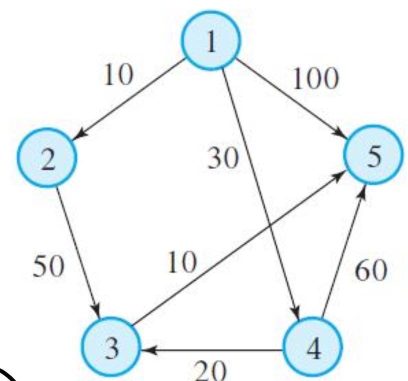
7.5 그래프의 응용

알고리즘 1 다익스트라 알고리즘

```
void Dijkstra()
   /* 이 알고리즘은 정점 1에서 방향 그래프의 모든 정점으로의 최단 거리를 계산한다.*/
{
   S = {1};
   for( i = 2 ; i <= n ; i++ )
      D[i] = C[1, i];   /* D값을 초기화한다 */
   for( i = 1 ; i <= n-1 ; i++ )
   {
      choose a vertex w in V - S such that
         D[w] is a minimum;
      add w to S;
      for(each vertex v in V - S)
         D[v] = min (D[v], D[w] + C[w, v]);
   }
} /* Dijkstra */
```

7.5 그래프의 응용

예제 7-14 다익스트라 알고리즘을 다음과 같은 가중 그래프에 적용해보자.



0	10	∞	30	100

X	X	60	~~30~~	10

X	X	50	X	60

	1	2	3	4	5
1	0	10	∞	30	100
2	∞	0	50	∞	∞
3	∞	∞	∞	∞	10
4	∞	∞	20	∞	60
5	∞	∞	∞	∞	0

풀이 처음 단계에서 $S = \{1\}$, $D[2] = 10$, $D[3] = \infty$, $D[4] = 30$, $D[5] = 100$이 된다. for 루프의 첫 번째 반복에서 $w = 2$가 최소의 D값을 가진 점으로 선택된다. 그러면 $D[3] = \min(\infty, 10 + 50) = 60$이 되고 $D[4]$와 $D[5]$는 변동이 없다. 왜냐하면 점 1에서 직접 가는 값이 점 2를 경유하는 것보다 값이 작기 때문이다. for 루프에서의 각 반복 과정 후의 D값은 다음 표와

7.5 그래프의 응용

같으며 4번째 반복 후 점 1에서 점 2, 점 3, 점 4까지의 거리가 각각 10, 50, 30, 60으로 최종 결정된다.

단계	S	w	$D[2]$	$D[3]$	$D[4]$	$D[5]$
0	{1}	—	10	∞	30	100
1	{1, 2}	2	10	60	30	100
2	{1, 2, 4}	4	10	50	30	90
3	{1, 2, 3, 4}	3	10	50	30	60
4	{1, 2, 3, 4, 5}	5	10	50	30	60

7.5 그래프의 응용

(2) 해밀턴 순회의 응용

- 해밀턴 순회의 응용 문제로는 **순회판매원 문제(Traveling salesperson problem)**가 있음
- 순회판매원 문제란 방문해야 할 도시들과 이들 사이의 거리가 주어졌을 경우, 순회판매원이 어떤 특정한 도시를 출발하여 어떠한 도시도 두 번 방문함이 없이 모든 도시들을 거쳐 처음 출발한 도시로 되돌아올 때, **총 여행 거리가 최소가 되는 경로를 찾는 문제임**

- 최소의 경로가'최적의 경로라고 할 수 있음
- 일반적인 해결 알고리즘이 존재하지 않음
- **최근접 이웃 방법(nearest neighbor method)**을 통하여 최소값은 아니더라도 근사값은 구할 수 있음
- 임의로 선택한 꼭지점에서 출발하여 그 꼭지점과 가장 가까운 꼭지점을 찾아서 연결하고 있음
- 경로를 첨가하는 과정을 반복하며 마지막에 순회를 형성하도록 하는 것임

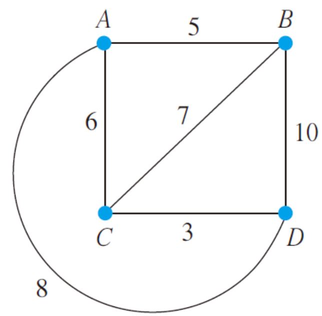
7.5 그래프의 응용

알고리즘 2 최근접 이웃 방법 알고리즘

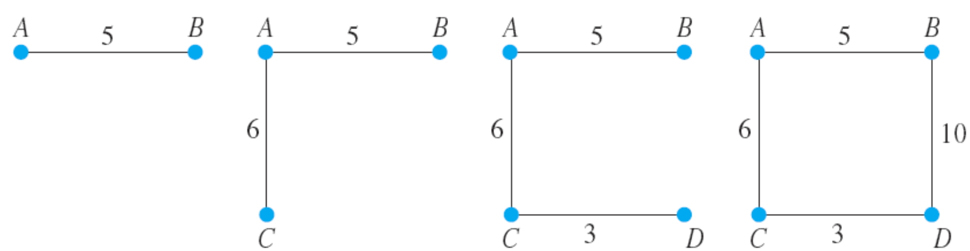
```
begin
    Choose any v1 ∈ V.
    v' ← v1
    w ← 0
    Add v' to the list of vertices in the path.
    while unmarked vertices are remained do
        begin
        Mark v'.
        Choose any unmarked vertex, u, that is closest to v'.
        Add u to the list of vertices in the path.
        w ← w + the weight of the edge (v', u)
        v' ← u
        end
    Add v1 to the list of vertices in the path.
    w ← w + the weight of the edge {v', v1}
end.
```

7.5 그래프의 응용

예제 7-15 다음의 가중 그래프에서 최근접 이웃 방법을 적용한 해밀턴 순회를 만들어 보자.



풀이 임의의 정점 B로부터 시작하자. B와 가장 가까운 방문되지 않은 정점은 A이므로 다음 그림 (1)과 같이 A를 경로에 포함시킨다. 또한 C를 포함시키고 D와 B를 연결한다.



(1) (2) (3) (4)

7.6 그래프의 탐색

(1) 깊이 우선 탐색(Depth First Search : DFS)

- 깊이 우선 탐색은 시작점 v부터 방문함
- v에 인접한 정점 중에서 방문하지 않은 정점 w를 방문하고 다시 w로부터 탐색을 시작함
- 어떤 정점 u를 방문하고 u에 인접한 모든 정점들을 이미 방문한 경우에는 그 전에 마지막으로 방문한 정점으로 되돌아가서 위의 과정들을 반복함
- 모든 정점들을 방문한 후 탐색을 종료함

➢ **순차적인 프로그램보다는 DFS 알고리즘을 재귀(recursive) 알고리즘으로 구현하는 것이 좋음**

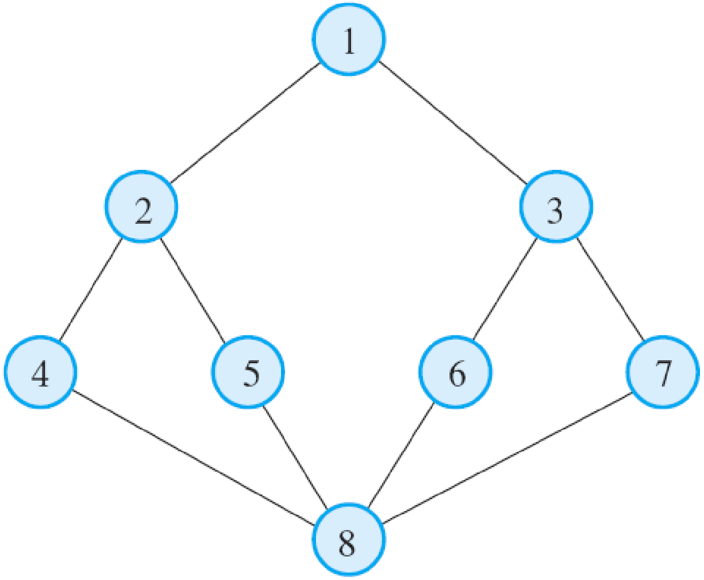
➢ **재귀 알고리즘으로 구현할 경우에는 스택(stack)을 사용함**

7.6 그래프의 탐색

알고리즘 3 DFS

```
void dfs(int v)
   /* G = (V, E)가 n개의 정점을 가진 그래프이고 처음에는 False값으로 행렬
   visited[n]이 주어졌다고 할 때, 이 알고리즘은 정점 v로부터 도달 가능한 모든 정
   점들을 방문한다. */
{
   int w;
   visited[v] = TRUE;
   for(each vertex w adjacent to v)
      if(!visited[w]) dfs(w);
} /* dfs */
```

7.6 그래프의 탐색

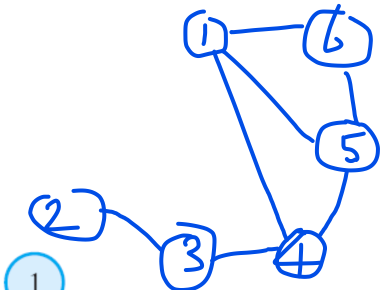


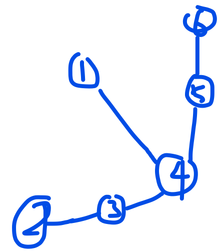
〈그림 7.20〉 주어진 그래프

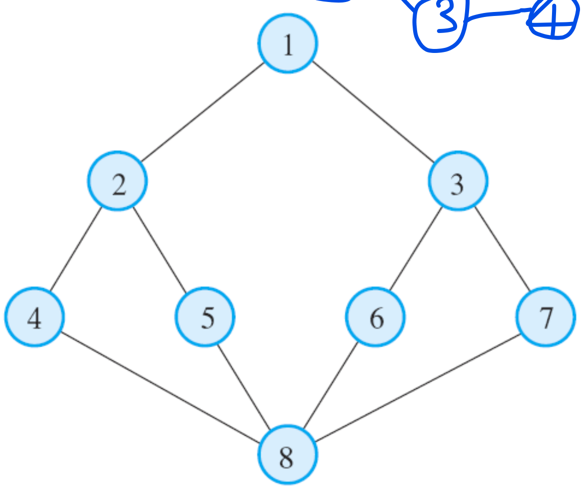
7.6 그래프의 탐색

DFS의 결과

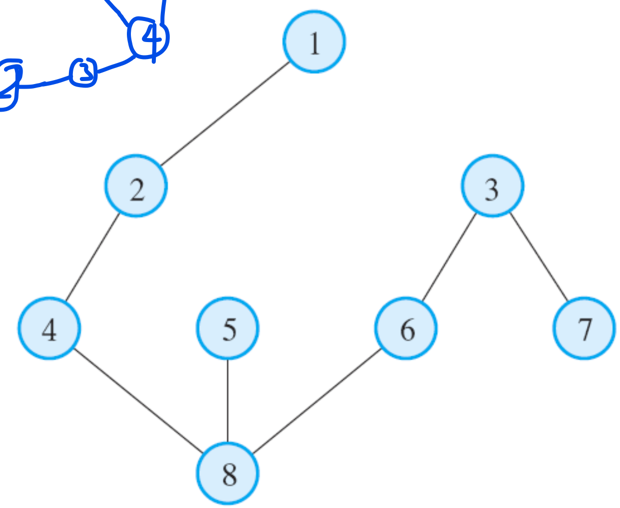
1-4-3-2-3-4-5-6 — 5-4-1





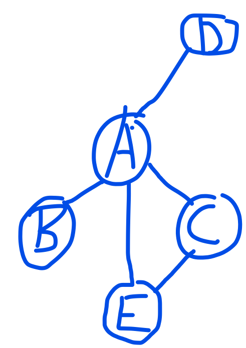


〈그림 7.20〉 주어진 그래프



〈그림 7.21〉 DFS의 결과

DFS의 결과는 탐색은 1,2, 4, 8, 5, 6, 3, 7의 순으로 이루어짐



A-B-A-C-E-C-A-D-A

A-B-C-E-D

7.6 그래프의 탐색

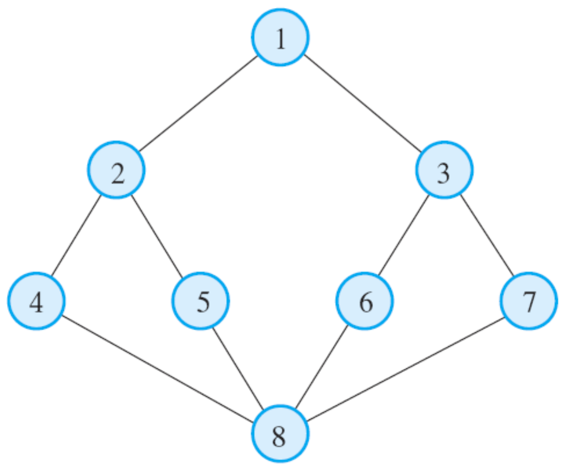
(2) 너비 우선 탐색(Breadth First Search : BFS)

- 처음에 방문한 정점과 인접한 정점들을 차례로 방문한다는 점에서 깊이 우선 탐색과 차이가 있음
- 먼저 시작점 v를 방문한 후 v에 인접한 모든 정점들을 차례로 방문함
- 더 이상 방문할 정점이 없는 경우 다시 v에 인접한 정점 가운데 맨 처음으로 방문한 정점과 인접한 정점들을 차례로 방문함
- v에 인접한 정점 중 두 번째로 방문한 정점과 인접한 정점들을 차례로 방문하는 과정을 반복함
- 모든 정점들을 방문한 후 탐색을 종료함

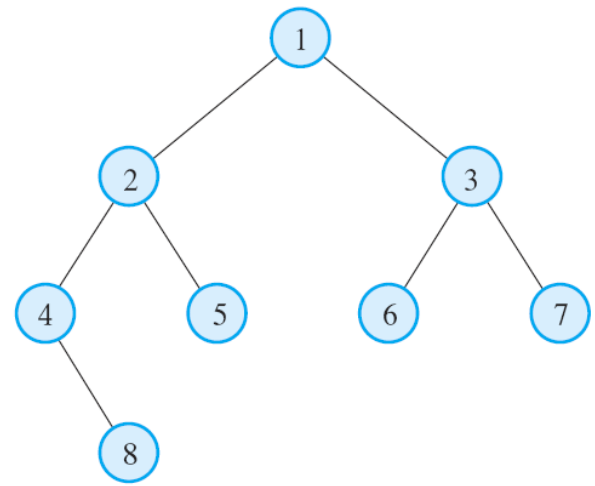
➢ **깊이 우선 탐색이 스택(stack)을 사용하는데 비해 너비 우선 탐색은 큐(queue)를 사용함**

7.6 그래프의 탐색

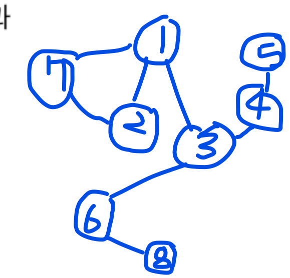
BFS의 결과



〈그림 7.20〉 주어진 그래프



〈그림 7.22〉 BFS의 결과



BFS의 결과는 1, 2, 3, 4, 5, 6, 7, 8의 순으로 이루어짐

1-2-3-4-7-6-5-8

7.7 그래프의 색칠 문제

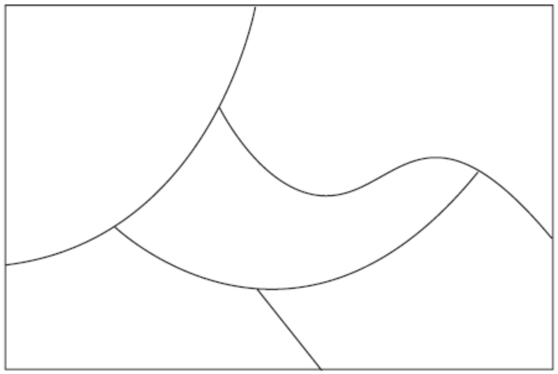
그래프 색칠 문제

정의 7−20 어떤 주어진 그래프 G에 대해 인접한 어느 두 영역도 같은 색이 안 되도록 각 정점에 색을 칠하는 문제를 그래프 G의 **색칠 문제(coloring problem)**라고 한다. 그래프 G를 색칠하는 데 필요한 최소한의 색의 수를 $\chi(G)$로 표현하고 G의 **색칠 수(chromatic number)**라고 한다.

- 단순 그래프의 색칠은 각각의 정점을 서로 다른 색깔로 색칠함으로써 알 수 있음
- 대부분의 그래프에서는 정점의 수보다 적은 색깔을 사용해도 색칠이 가능함
- 그렇다면 몇 가지 색으로 모든 경계가 구별되도록 색칠할 수 있을까?

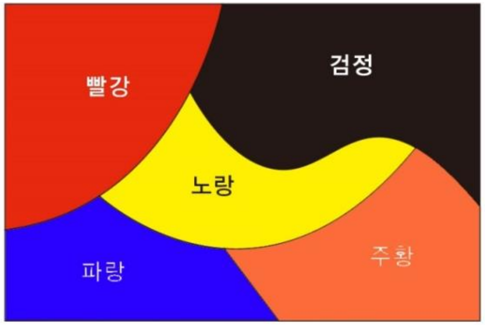
7.7 그래프의 색칠 문제

예제 7-16 다음 그림과 같이 경계가 주어진 지도를 색칠해보자. 서로 경계가 만나는 부분이 있으면 서로 다른 색을 사용해야 한다.

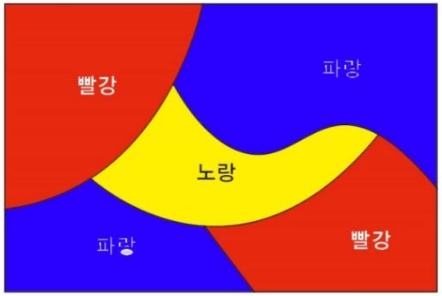


풀이 다음 그림은 주어진 지도를 5가지의 색을 사용하여 색칠한 예이다.

7.7 그래프의 색칠 문제



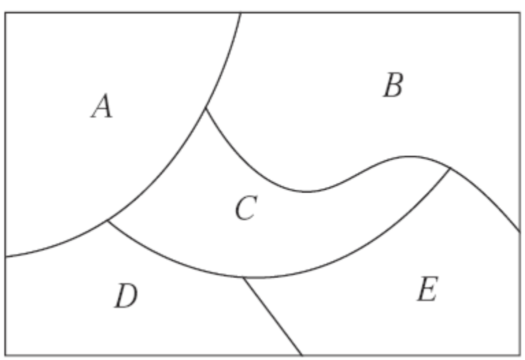
그러나 이 지도는 왼쪽 그림과 같이 4가지 색으로도 가능한 것을 알 수 있으며, 최종적으로 오른쪽 그림과 같이 3가지 색만으로도 가능하다.



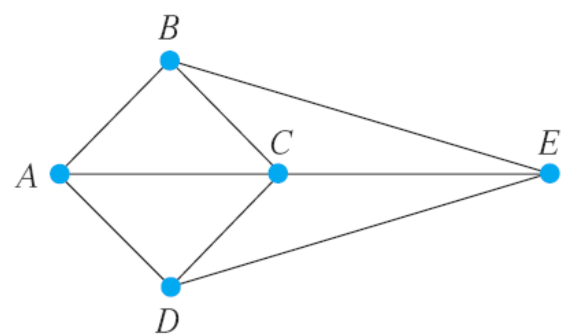
지도를 색칠하는 문제는 그래프의 색칠 문제와 연관시킬 수 있다. 예를 들어, (1)의 그래프는 (2)의 그래프와 동형이 되는데, 이것을 **쌍대 그래프**라고 한다.

7.7 그래프의 색칠 문제

- 지도를 색칠하는 문제는 그래프의 색칠 문제와 연관시킬 수 있음
- 예를 들어, (1)의 그래프와 동형이 되는 (2)의 그래프를 **쌍대 그래프**라고 함
- 지도의 영역을 색칠하는 문제는 쌍대 그래프의 정점들을 색칠하는 문제와 동일하므로 그래프의 인접한 어떤 정점들도 같은 색깔을 가지지 않도록 해야 함



(1)



(2)

7.7 그래프의 색칠 문제

정리 7−5

그래프 G에 대해 다음은 서로 동치이다.

(1) G는 두 가지 색으로 칠할 수 있다.

(2) G는 이분 그래프이다.

(3) G의 모든 순환은 짝수의 길이를 가진다.

정리 7−6

모든 평면 그래프는 네 가지 색으로 색칠이 가능하다. 이것을 4색 문제(four coloring problem)라고 한다.

7.7 그래프의 색칠 문제

그래프 색칠하기의 3가지 응용 분야

1) 화학물질을 창고에 저장할 때, 만일의 사고를 방지하기 위해 서로 반응하는 물질끼리는 다른 장소에 배치하는 문제
2) 방송국들의 주파수가 같아서 난시청 지역이 발생하지 않도록 하는 문제
3) 애완동물을 판매할 때 사이가 좋지 않은 동물들을 다른 전시관에 배치하는 문제

7.8 그래프의 응용과 4차 산업혁명과의 관계

(1) 그래프의 응용 분야

① 모델링과 다양한 학문 분야에서의 응용

그래프는 주어진 문제를 모델화하는 데 유용한 도구로서 그 사용이 증가하고 있으며, 최근에는 기하학과 그래픽 등 광범위한 범위에서 응용되고 있다. 또한 사회과학, 유전학, 언어학, 시스템의 흐름도, 분자구조식 설계, 화학 결합의 표시, 반도체 회로 설계, 내비게이션에서의 빠른 길 찾기, 지능형 교통 시스템, 이동통신을 위한 중계기 설치 등에도 널리 응용되고 있다.

② 컴퓨터 관련 분야에서의 응용

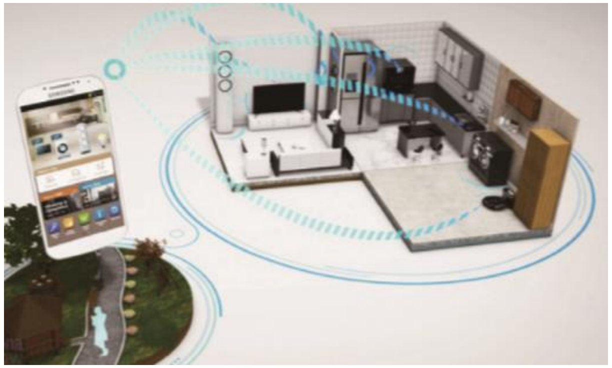
그래프 이론은 컴퓨터 관련 분야에 필수적이다. 주요 응용 예로는 논리 회로의 설계 및 분석, 시스템의 흐름도, 컴파일러에서의 최적화, 플로우 차트를 통한 작업 계획의 분석, 통신 네트워크, 그래프의 탐색과 정렬, 알고리즘, 유한상태 기계 등을 들 수 있다.

7.8 그래프의 응용과 4차 산업혁명과의 관계

(2) 그래프와 4차 산업혁명과의 관계 : 사물인터넷

사물인터넷(Internet of Things: IoT)은 인터넷에 연결된 기기가 사람의 개입 없이도 상호간에 알아서 정보를 주고받으면서 처리하는 기술이다. 사물인터넷은 인간을 둘러싼 사물들이 〈그림 7.23〉과 같이 네트워크로 서로 연결되면서 인간에게 새로운 편의와 가치를 부여한다.

4차 산업혁명의 한 기술 분야인 사물인터넷은 네트워크 연결이 중심이 되므로 그래프와 관련이 깊다. 따라서 그래프 이론과 관련된 깊이 있는 지식은 사물인터넷 관련 개발에 초석이 될 수 있을 것이다.



〈그림 7.23〉 사물인터넷의 연결

요약 및 생활 속의 응용

- 그래프 G=(V, E)는 유한한 개수의 정점 또는 노드들의 집합인 V와 연결선 또는 에지라고 불리는 정점들의 쌍들의 집합인 E로 이루어진다.
- 그래프는 방향이 있는 그래프인 방향 그래프와 방향이 없는 그래프의 2가지로 구분된다.
- 멀티 그래프는 단순 그래프의 확장으로서 한 쌍의 꼭지점 사이에 연결선의 개수의 제한이 없는 일반적인 그래프이다.
- 멀티 그래프에서 모든 연결선들을 꼭 한 번씩만 통과하는 경로를 오일러 경로라고 하는데, 쾨니히스베르크 다리 문제는 오일러 경로를 찾을 수 있는지 여부와 동치이다.
- 그래프의 표현 방법으로는 인접행렬과 인접 리스트가 있다.
- 어떤 그래프 G가 오일러 순회를 가지기 위한 필요충분조건은 G가 연결 그래프이고, 모든 정점들이 짝수 개의 차수를 가지는 경우이다.

요약 및 생활 속의 응용

- 오일러의 정리는 연결된 평면 그래프에서 꼭지점의 수를 v, 연결선의 수를 e, 면의 수를 f라고 할 때 $v - e + f = 2$이다.
- 그래프 탐색에는 깊이 우선 탐색과 너비 우선 탐색이 있다.
- 그래프는 주어진 문제를 모델화하는 데 유용한 도구로서 그 사용이 증가하고 있으며, 최근에는 기하학과 그래픽 등 광범위한 범위에서 응용되고 있다.
- 그래프는 통신 네트워크, 논리 회로의 설계 및 분석, 최단 거리 찾기나 최단 거리 순회 문제, 시스템의 흐름도나 컴파일러에서의 최적화, 그래프 탐색, 정렬, 알고리즘, 전기회로망 설계와 응용, 순서도를 통한 작업 계획의 분석, 분자구조식 설계와 화학결합의 표시, 유한 상태 기계, 유전학, 언어학, 사회과학 등에 다양하게 응용된다.
- 그래프와 4차 산업혁명과의 관계로는 사물인터넷을 들 수 있다.

CHAPTER 7
FINISH