목차

5. 확률과 통계

1. 비둘기집 원리

2. 경우의 수

3. 순열

4. 조합

6. 집합

1. 집합

2. 집합의 연산

1. 이산수학 기초

1. 이산수학의 개요

1. 이산수학

불연속적인 수학 구조에 대해 연구하는 학문. (논리, 집합, 명제, 함수, 트리, 순열 등)

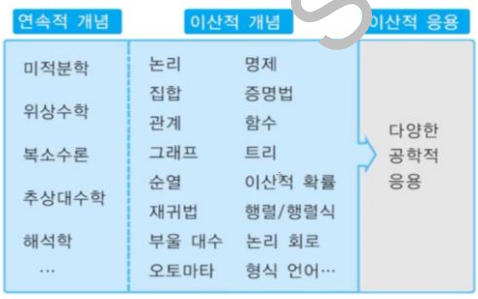
1) 이산수학을 배우는 이유

컴퓨터는 이산적 개념을 사용하기 때문에, 이산수학은 컴퓨터의 이론적 배경임.
이산수학은 프로그래밍을 위해 복잡한 문제를 추상화하는 데에 사용됨.

메모 포함[이1]: 떨어져 있는.

2. 이산수학과 연속수학

이산수학 : 정수영역, 분리된 원소들, 유한 집합, 디지털 컴퓨터
연속수학 : 실수영역, 연속된 원소들, 유한 + 무한 집합, 아날로그 컴퓨터



〈그림 1.2〉 이산수학의 공학적 응용

3. 수학적 모델링

주어진 문제를 수학적 구조에 매핑(mapping)시켜 체계적으로 해결하는 방법론.

모델링 시에 고려하는 여러가지 요소들 중, 수학적 개념을 사용하는 부분을 수학적 모델링이라고 함.
일반 모델링과 수학적 모델링은 다른 개념임.

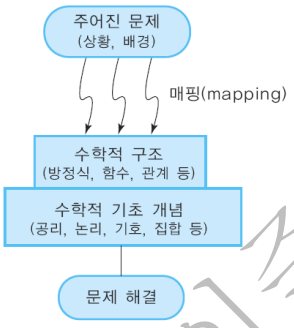
메모 포함[이2]: = 설계.

메모 포함[이3]: 하나의 값을 다른 값으로 대응시키는 것.

메모 포함[이4]: 언어적인 측면, 설계적인 측면 등등.

1) 수학적 모델링 다이어그램의 3가지 핵심 요소

1. 주어진 문제의 상황과 배경
2. 주어진 문제와 수학적 구조 사이의 매핑
3. 수학적 기초 개념을 이용한 문제 해결

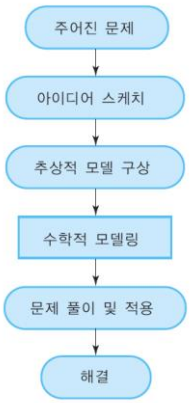


〈그림 1.5〉 수학적 모델링의 구체적인 다이어그램

+ 프로그래밍 이전의 과정

문제 -> 추상화 -> 모델링 -> 결과 도출 -> 프로그래밍 진행

+ 문제 해결을 위한 모델링



〈그림 1.6〉 효과적인 문제 해결 방법

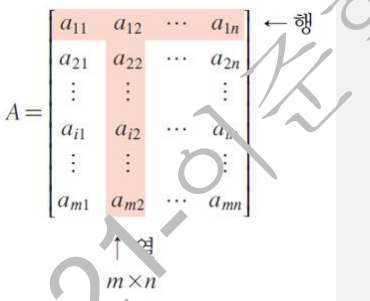
2. 행렬

1. 행렬(matrix)

1. 행렬

숫자 또는 문자의 집합.

행렬의 각 요소들을 '성분'이라고 함.
성분 각각을 '행렬'로 부르기도 함.



1) 표기법

괄호(소, 중, 대 상관없음.)로 묶어서 표기함.

행렬의 각 성분은 a_{mn} 등으로 표현할 수 있음. mn항, mn성분 등으로 부르기도 함.
여기서 앞쪽 숫자가 행을 나타내고, 뒤쪽 숫자가 열을 나타냄.

a_{11}부터 a_{mn}까지의 성분으로 이루어진 행렬을 (m, n) 행렬 또는 m x n 행렬이라고 함.
a_{11}부터 a_{mn}까지의 성분으로 이루어진 행렬을 A = [a_{ij}], $i = 1, \dots, m,\ j = 1, \dots, n$ 이라고 적음.

a_{mn}에서 m, n이 10 이상인 경우, 행과 열을 나타내는 숫자를 , (콤마)로 구분해서 표기함.

메모 포함[이5]: 'm by n 행렬', 'm행 n열 행렬'로 읽는다.
x는 '곱하기'로 읽지 않는다.

2) 성분으로 분해

행렬은 성분으로 나눌 수 있음.

(ex. $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$ 행렬을 행으로 나누면 [1, 2, 3] [4, 5, 6] 이고, 열로 나누면 $\begin{bmatrix} 1 \\ 4 \end{bmatrix}$ $\begin{bmatrix} 2 \\ 5 \end{bmatrix}$ $\begin{bmatrix} 3 \\ 6 \end{bmatrix}$ 임.)

+ 행과 열

행 (row) : 행렬의 가로줄.
열 (column) : 행렬의 세로줄.

2. 행렬의 덧셈과 뺄셈, 스칼라 곱

일반적 연산과 거의 동일함.

덧셈, 뺄셈, 스칼라 곱으로 이루어진 행렬 방정식은 일반 방정식과 동일한 방식으로 처리하면 됨.

1) 덧셈과 뺄셈

두 행렬이 덧셈과 뺄셈은 두 행렬의 크기가 동일할 때만 가능함.

두 행렬을 비교하여 각각 동일한 위치에 있는 성분끼리 더하고 뺌.

(ex. $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix}$, $B = \begin{bmatrix} -1 & 5 & -3 \\ 3 & 4 & 6 \end{bmatrix}$, $A+B = \begin{bmatrix} 1+(-1) & 2+5 & 3+(-3) \\ 4+3 & 5+4 & 6+6 \end{bmatrix} = \begin{bmatrix} 0 & 7 & 0 \\ 7 & 9 & 12 \end{bmatrix}$)

2) 스칼라 곱

행렬과 스칼라 값을 곱한 경우, 행렬의 각 성분에 스칼라 값을 각각 곱함.

3) 덧셈과 뺄셈, 스칼라 곱의 성질

(1) $A+B=B+A$	(덧셈의 교환법칙)
(2) $(A+B)+C=A+(B+C)$	(덧셈의 결합법칙)
(3) $A+O=O+A$	(덧셈의 항등법칙)
(4) $A+(-A)=(-A)+A=O$	(덧셈의 역원)
(5) $c(A+B)=cA+cB$	(스칼라 곱의 배분법칙)
(6) $(c+d)A=cA+dA$	(스칼라 곱의 배분법칙)

일반적인 수의 덧셈, 수식, 스칼라 곱과 동일한 성질을 가짐.

증명은 직접 좌변과 우변을 계산해서 비교하면 됨

메모 포함[이6]: 행렬의 덧셈과 뺄셈, 스칼라 곱을 이용한 방정식을 만들 수도 있다.

메모 포함[이7]: 즉, 행렬 A에 스칼라 값 k를 곱하면 $kA = [ka_{ij}], i = 1, \dots, m, j = 1, \dots, n$ 이다.

3. 행렬의 곱셈

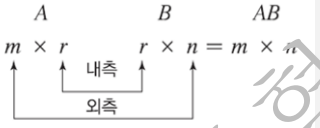
행렬끼리의 곱셈은 행렬의 덧셈/뺄셈과는 달리, 나름의 방식을 사용함.

내측 행렬이 같아야 두 행렬의 곱이 정의되고, 외측 행렬의 값이 결과 행렬의 크기임.

1) 방법

(앞쪽 행렬을 A, 뒤쪽 행렬을 B, 결과 행렬을 C라고 한다.)

1. 두 행렬의 내측 행렬이 같은지 확인한다.
외측 행렬의 값으로 C의 크기를 확인한다.



2. A를 행 단위로 묶고, B를 열 단위로 묶는다.

3. A의 한 행을 B의 모든 열과 순서대로 연산을 진행한다.
연산 방법은 행의 n번째 원소와 열의 n번째 원소를 곱해서, 모든 곱한 값들을 더하는 것이다.
A의 n행과 B의 m열의 연산으로 도출된 값을 C의 nm항으로 작성해 넣는다.
A의 모든 행이 연산이 완료될 때까지 연산한다.

2) 성질

행렬의 곱셈에서는 교환법칙이 성립하지 않음.

A가 $m \times n$ 행렬이고, B와 C는 행렬의 합과 곱에서 정의된 크기를 만족한다고 가정하고 k가 어떤 스칼라 값일 때 다음의 식들이 성립한다.

(1) $A(BC) = (AB)C$ (곱셈의 결합법칙)
(2) $A(B+C) = AB + AC$ (왼쪽 배분법칙)
(3) $(B+C)A = BA + CA$ (오른쪽 배분법칙)
(4) $k(AB) = (kA)B = A(kB)$ (스칼라 곱)
(5) $I_n A = A = AI_n$ (행렬 곱셈의 항등식)

일반적 수의 곱과 동일한데, 행렬끼리의 곱셈 연산 순서(앞뒤)는 바뀌면 안됨.
행렬들과 스칼라 값의 곱셈에서 스칼라 값의 곱셈 연산 순서는 바뀔 수 있음.

증명은 직접 좌변과 우변을 계산해서 비교해 보면 됨.

+ 방정식의 개수와 변수의 개수

변수의 개수만큼의 방정식이 존재해야 각 변수의 값을 구해낼 수 있음.

메모 포함[이8]: 시험에 항상 나온다.

메모 포함[이9]: 값은 연산에 사용 중인 행과 열에 해당하는 위치에 들어간다.
(ex. 2번째 행과 1번째 열을 연산 중이라면 결과 행렬의 2,1 위치에 값이 들어가게 된다.)

현재 연산 중인 행 위치에 오른쪽 열이 하나씩 들어간다고 생각하면 계산이 빨라진다.
(직접 계산해 보면 무슨 말인지 이해가 될 것이다.)

메모 포함[이10]: ex. 행렬 A, B가 있을 때, AB와 BA는 다르다.

4. 행렬의 거듭제곱

일반적인 거듭제곱과 거의 동일한 개념.

행렬 A를 n번 거듭제곱한 것은 A^n으로 표기함.

$$A^k = A^{k-1} \cdot A$$
$$A^0 = I$$

5. 여러 가지 행렬

메모 포함[이11]: 정방행렬>대각행렬>단위행렬
이렇게 집합 관계를 나타내 볼 수 있다.

1) 정방행렬

행의 개수와 열의 개수가 같은 행렬.

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}$$

n개의 행과 n개의 열을 가지는 행렬을 n차 정방행렬이라고 함.

1. 주대각선.
정방행렬의 맨 왼쪽 위부터 맨 오른쪽 아래까지의 선을 주대각선이라고 함.
오른쪽 그림의 색깔로 표시된 부분이 주대각선임.

2. 대각항.
주대각선 상에 있는 성분들을 대각항이라고 함.
대각항들은 행과 열이 같음.

3. 대각합(trace. tr).
대각항들의 합을 대각합(trace. tr)이라고 함.
정방행렬 A의 대각합은 tr(A), trace(A) 등으로 표기함.

4. 성질
대각항과 대각합은 정방행렬에서만 정의됨

A와 B가 같은 크기의 정방행렬일 때 대각합은 다음의 특성들을 가진다.

(1) $\mathrm{tr}(A^T) = \mathrm{tr}(A)$
(2) $\mathrm{tr}(cA) = c\mathrm{tr}(A)$
(3) $\mathrm{tr}(A+B) = \mathrm{tr}(A) + \mathrm{tr}(B)$
(4) $\mathrm{tr}(A-B) = \mathrm{tr}(A) - \mathrm{tr}(B)$
(5) $\mathrm{tr}(AB) = \mathrm{tr}(BA)$

마지막 성질 유의하기.

증명은 직접 좌변과 우변을 계산해서 비교해 보면 됨.

2) 대각행렬

n차 정방행렬에서 주대각선을 제외한 모든 항들이 0인 행렬.

$$D=\begin{bmatrix} d_1 & 0 & \cdots & 0 \\ 0 & d_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & d_n \end{bmatrix}$$

3) 단위행렬(항등행렬)

주대각선의 성분들은 모두 1이고, 나머지 성분들은 모두 0인 정방행렬.

$$I_n \cdot A = A \cdot I_n = A$$

단위행렬에는 어떤 행렬을 곱해도 원래의 행렬이 나옴.

n차 정방행렬인 항등행렬은 I_n으로 표기함.

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

4) 0 행렬

모든 항이 0인 행렬. (정방행렬이 아니어도 됨.)

O으로 표기하거나, $m \times n$ 영행렬인 경우에는 $O_{m\times n}$으로 표기함.

$$[0], \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

5) 전치행렬

기존 행렬의 행과 열을 뒤바꾼 행렬.

n번째 행을 n번째 열로 바꿈.

$A=\begin{bmatrix} 2 & 1 & 0 \\ 1 & 3 & 5 \end{bmatrix}$이면 $A^T=\begin{bmatrix} 2 & 1 \\ 1 & 3 \\ 0 & 5 \end{bmatrix}$

정방행렬의 경우 주대각선을 기준으로 대칭이동한 것으로 볼 수 있음.

A의 전체행렬은 A^T로 표기함.

성질.

전치행렬에서는 다음과 같은 성질들이 성립한다.

(1) $(A^T)^T = A$

(2) $(A \pm B)^T = A^T \pm B^T$

(3) $(cA)^T = cA^T$ (c는 상수)

(4) $(AB)^T = B^T A^T$

특히 4번 성질 유의하기.

6) 대칭행렬

n 차 정방행렬 A 가 $A = A^T$를 만족할 때, A 를 대칭행렬이라고 함.

주대각선을 기준으로 항들이 대칭인 경우 대칭행렬임.

$$A=\begin{bmatrix} 1 & 2 \\ 2 & 3 \end{bmatrix}, \quad B=\begin{bmatrix} 2 & 4 & 0 \\ 4 & 1 & 6 \\ 0 & 6 & 3 \end{bmatrix}, \quad C=\begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 5 \\ 3 & 5 & 6 \end{bmatrix}$$

7) 교대행렬

n차 정방행렬 A가 $A = -A^T$를 만족할 때, A를 교대행렬이라고 함.

주대각선을 기준으로 항들이 부호만 다르고 대칭인 경우 교대행렬임.

$$A=\begin{bmatrix} 0 & 2 & 3 \\ -2 & 0 & -4 \\ -3 & 4 & 0 \end{bmatrix},\quad B=\begin{bmatrix} a & h & g \\ h & b & f \\ g & f & c \end{bmatrix},\quad C=\begin{bmatrix} 0 & a & b \\ -a & 0 & c \\ -b & -c & 0 \end{bmatrix}$$

8) 여인수 행렬

어떤 행렬의 여인수를 성분으로 가지는 행렬.

여인수 행렬의 성분은, 원본 행렬에서 해당 위치 성분의 여인수임.

9) 수반행렬

어떤 행렬의 여인수 행렬의 전치행렬.

행렬 A의 수반행렬은 $Adj(A)$로 표기함.

역행렬을 구할 때 사용함.

10) 첨가행렬

행렬의 오른쪽에 추가적으로 첨가하여 만든 행렬.

2. 행렬식

1. 행렬식(determinant)

하나의 정방행렬과 하나의 스칼라 값 사이의 대응을 통해 생성된 특별한 함수.
행렬으로부터 얻어낼 수 있는 값의 한 종류로 이해하면 될 듯.

det A 또는 | A | 로 표기함. (절댓값 아님.)

메모 포함[이12]: 여기서 배우는 행렬식은 정방행렬만이 가지는 값이다.

1) 1차 정방행렬의 행렬식

성분이 하나뿐이기 때문에 그 성분이 행렬식임.

$$Det(A) = |a_{11}| = a_{11}$$

2) 2차 정방행렬의 행렬식

성분들을 오른쪽과 같은 방식으로 연산한 값이 행렬식임.

$$Det(A) = \begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

3) 3차 정방행렬의 행렬식

메모 포함[이13]: 두 방식 모두 시험에 나온다.

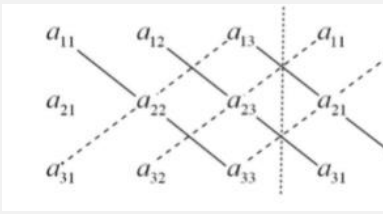
기본 방식. (소행렬, 여인수 사용)

1. 행이나 열 중 하나를 선택한다.
2. 선택한 행/열의 원소를 순서대로 나열한다.
3. 각각의 원소와 그 원소의 여인수를 곱한다.
4. 계산한다.

$$Det(A) = \begin{vmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{vmatrix} = a_{11}M_{11} - a_{12}M_{12} + a_{13}M_{13}$$

사루스의 공식.

1. 첫 번째와 두 번째 열을 행렬의 오른쪽에 복사한다.
2. 왼쪽 위->오른쪽 아래로 세 개의 대각선과 오른쪽 위->왼쪽 아래로 가는 세 개의 대각선을 그린다.
3. 대각선 위에 있는 값들끼리 곱한다.
4. 왼쪽 위->오른쪽 아래 대각선의 값들은 +를, 오른쪽 위->왼쪽 아래 대각선의 값들은 -를 붙인다.
5. 값들을 모두 더한다.



+ 소행렬 M_{ij}과 여인수

소행렬 M_{ij}은 해당 행렬에서, i번째 행과 j번째 열을 제외한 성분들로 만든 행렬임.
소행렬의 행렬식에 위치에 따라 적절한 부호를 붙인 것을 여인수라고 함.
여인수는 $C_{ij} = (-1)^{i+j}M_{ij}$로 표현할 수 있음.

메모 포함[이14]: 위의 과정에서는 단계를 따로 나누어서 썼지만 사실 소행렬의 행렬식을 곱한 이후에 부호를 붙이는 것이 아니라, 소행렬의 행렬식에 위치에 따라 적절한 부호가 붙은 여인수를 곱한 것이다.

4) 행렬식의 특징

1. 하나의 행 또는 열의 공통인수는 행렬식 밖으로 뽑아내도 행렬식의 값은 동일하다..

(ex. $\begin{vmatrix} a & kb \\ c & kd \end{vmatrix} = k\begin{vmatrix} a & b \\ c & d \end{vmatrix}$)

2. 두 행끼리 또는 두 열끼리 서로 바꾸면 행렬식의 값은 부호만 바뀐다.

(ex. $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = -\begin{vmatrix} c & d \\ a & b \end{vmatrix}$)

(3차 정방행렬의 행렬식에서도 성립한다.)

3. 하나의 행 또는 열에 실수를 곱한 값을 다른 행 또는 열에 더해도 행렬식의 값은 동일하다.

(ex. $\begin{vmatrix} a & b \\ c & d \end{vmatrix} = \begin{vmatrix} a & b \\ c-ka & b-b \end{vmatrix}$)

4. 곱해진 두 행렬의 행렬식은, 두 행렬 각각의 행렬식을 곱한 것과 같다.

(ex. $|AB| = |A| \cdot |B|$)

5. 어떤 행렬의 행렬식은 그 행렬의 전치행렬의 행렬식과 동일하다.

(ex. $Det(A) = Det(A^T)$)

메모 포함[이15]: 행렬의 특징과 행렬식의 특징을 혼동하지 말자.

메모 포함[이16]: 한 번에 하나의 행 또는 하나의 열에서만 뽑아낼 수 있다. 유의하기.

메모 포함[이17]: n차 정방행렬인 경우, $|kA| = k^n|A|$로 나타낼 수도 있다.

메모 포함[이18]: 수학에서의 실수는 유리수와 무리수를 포괄하는 개념이다.

메모 포함[이19]: $|A^n| = |A|^n$ 임을 쉽게 확인할 수 있다.

+ 인수

정수를 곱 꼴로 나타냈을 때 각각의 구성요소. (약수와 유사한 개념.)

+ 예제

1. $\begin{pmatrix} 2 & 0 \\ 0 & 2 \end{pmatrix} \cdot X = \begin{pmatrix} 1 & 2 \\ 2 & 3 \end{pmatrix}$일 때, $Det(X)$는? -> 행렬식의 6번째 특징을 사용.

-> 행렬식은 값이기 때문에 일반적인 연산을 할 수 있음.

답: 1/4

2. $A = \begin{pmatrix} 1 & 2 \\ 2 & 5 \end{pmatrix}$ 일 때, $Det(A^{100})$은? -> 행렬식의 6번째 특징을 사용.

답: 1

+ 대각선의 방향에 따른 부호

왼쪽 위->오른쪽 아래 : (+)

오른쪽 위->왼쪽 아래 : (-)

3. 역행렬

1. 역행렬(A^{-1})

서로 곱했을 때 항등행렬이 나오는 관계에 있는 두 행렬은 서로에 대해 역행렬이라고 함.

역행렬끼리는 순서를 바꾸어 곱해도 항등행렬이 나옴.

$Det(A) \neq 0$ 은 행렬 A의 역행렬이 존재하기 위한 필요충분조건임. -> 역행렬 나오면 이것부터 확인해야 함.
$Det(A) \neq 0$ 이면 행렬 A의 역행렬이 존재하는 것.
$Det(A) = 0$ 이면 행렬 A의 역행렬이 존재하지 않음.

"A의 역행렬이 존재한다." = "행렬 A가 가역적이다."

행렬 X, A, B에 대해서 $Det(A) = 0$일 때, $AX = B$는 $A^{-1}AX = A^{-1}B,\ X = A^{-1}B$로 전환할 수 있음.

메모 포함[이20]: 유도과정.
1. AB=I이다. (대문자 i. 항등행렬.)
2. $BABB^{-1}$=I이다. (앞뒤로 B, B^{-1}을 곱했다.)
3. $BB^{-1} = I$이므로 $BA = I$이다.

2. 수반행렬로 역행렬 구하기

$$A^{-1} = \frac{1}{Det(A)} Adj(A)$$

1) 일반 공식

행렬 A의 역행렬은 A의 행렬식의 역수와 A의 수반행렬을 곱한 것으로 구할 수 있음.

유도과정.
1. 행렬 A와 A의 수반행렬 Adj(A)를 곱하면 주대각선의 성분 값이 모두 Det(A)인 대각행렬이 나온다.
2. 해당 대각행렬은 $Det(A) \cdot I$ 로 나타낼 수 있으므로, 정리하면 $A \cdot Adj(A) = Det(A) \cdot I$ 이다.
3. $Det(A)$를 좌변으로 넘기면 $A \cdot \frac{Adj(A)}{Det(A)} = I$ 이다.
4. 곱해서 항등행렬이 나오는 것이 역행렬이므로, $\frac{Adj(A)}{Det(A)}$ 는 A의 역행렬이다.

메모 포함[이21]: 실제로 곱해보면 그렇다. 왜 그런지 이해하기보다는 일단 그냥 그렇구나 하고 넘기자.

2) 2차 정방행렬의 역행렬 공식

$$A^{-1} = \frac{1}{Det(A)} \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}$$

행렬 A가 2차 정방행렬인 경우 나름의 공식을 적용할 수 있음.
일반 공식에 2차 정방행렬을 넣으면 오른쪽 공식으로 정리할 수 있음.
2차 정방행렬의 경우

왼쪽 위->오른쪽 아래 대각선(주대각선)은 성분끼리 자리 바꾸기.
오른쪽 위->왼쪽 아래 대각선은 성분의 부호 바꾸기.

3. 가우스-조던 소거법으로 역행렬 구하기

이거 설명해준다면서 안 해줬다. 인터넷에서 찾아서 정리해두자.

이와 같은 방식으로 x_n은 A의 n번째 열 대신에 b가 대체되어 만들어진다.

$$x_n = \frac{1}{\mathrm{Det}(A)} \mathrm{Det} \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1,n-1} & b_1 \\ a_{21} & a_{22} & \cdots & a_{2,n-1} & b_2 \\ \vdots & \vdots & \cdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{n,n-1} & b_n \end{bmatrix}$$

4. 행렬의 방정식 연산

1. 가우스-조던 소거법

연립 일차방정식을 푸는 하나의 방법.

알고리즘에서 자주 사용되는 계산 방법임.

1) 기본 방법

1. 연립 일차방정식을 행렬로 만든다.

2. 기본 행 연산으로 전향단계와 후향단계를 거쳐 사다리꼴을 만든다.

기본 행 연산 -> 1. 두 행을 서로 바꾼다.
-> 2. 한 행에 0이 아닌 상수를 곱한다.
-> 3. 한 행에 상수를 곱한 값을 다른 행에 더한다.

전향단계 : 피벗의 아랫부분이 0이 되도록 연산하는 것. 행 사다리꼴 생성. -> 가우스 소거법.
후향단계 : 피벗의 윗부분까지 0이 되도록 연산하는 것. 기약 행 사다리꼴 생성. -> 가우스-조던 소거법.

+ 연립 일차방정식과 행렬

행렬은 방정식을 더 쉽게 해결하기 위한 도구임.
연립 일차방정식은 행렬로 나타낼 수 있음.

전환 방법.
1. 변수가 포함된 항들을 좌변에, 상수를 우변에 옮겨서 방정식을 정리한다.
2. 변수의 계수와 상수를 행렬의 성분으로 작성한다.
이때 두 방정식의 작성 순서(동일한 변수끼리)를 맞춰야 한다.
변수가 존재하지 않으면 해당 위치에 0으로 작성한다.

(ex. x+2y=5, 2x=8 은 $\begin{pmatrix} 1 & 2 & 5 \\ 2 & 0 & 8 \end{pmatrix}$ 행렬로 나타낼 수 있음.)

메모 포함[이22]: 시험에 나온다.

메모 포함[이23]: 전향단계까지만 진행하면 가우스 소거법이고, 전향단계 이후 후향단계까지 진행하면 가우스-조던 소거법이다.

메모 포함[이24]: 행렬의 성분을 0이나 1로 간단하게 만드는 연산.

이때 초기의 연립 일차방정식과 해가 달라질 만한 계산을 해서는 안 된다. 기본 행 연산으로 작성한 것들만 사용한다면 해가 달라질 수가 없다.

행렬식의 특징으로부터 온 연산 방식. 행렬식의 특징과 동일하지는 않으므로 명확히 구분할 필요가 있다. (행렬식과 이 연산이 무슨 관계인지는 잘 모르겠다.)

메모 포함[이25]: 첫 번째 행부터 시작해서 각 행의 피벗 아래를 모두 0으로 만든다고 생각하면 쉽게 할 수 있다.

2. 피벗(기준점)

여기서는 행렬의 각 행의 0이 아닌 수들 중 가장 앞쪽에 있는 수를 피벗(기준점)으로 삼음.

3. 사다리꼴

메모 포함[이26]: 행 사다리꼴>기약 행 사다리꼴
이런 집합 관계로 이해할 수 있다.

메모 포함[이27]: Row Echelon Form

1) 행 사다리꼴(REF)

기본 행 연산들을 거친 후 3가지 조건을 만족시키면 행 사다리꼴임.

1. 0으로만 이루어진 행들은 행렬의 가장 아래쪽에 작성한다.
2. 0으로만 이루어지지 않은 행에서는, 왼쪽에서부터 처음 나타나는 0이 아닌 수를 피벗으로 한다.
3. 0으로만 이루어지지 않은 연이은 두 행이 있으면, 아래쪽 행의 피벗은 위쪽 행의 피벗보다 오른쪽에 있다.

즉, 행 사다리꼴은 3번 조건을 만족하면서 피벗의 아래쪽 수가 모두 0이면 됨.

$$\begin{bmatrix}1&2&3\\0&1&5\\0&0&1\end{bmatrix}, \begin{bmatrix}1&-1&6&1\\0&1&2&-8\\0&0&0&1\end{bmatrix}, \begin{bmatrix}1&0&2&5\\0&1&2&\end{bmatrix}, \begin{bmatrix}1&2\\0&1\end{bmatrix}, \begin{bmatrix}1&3&2&5\\0&1&3&6\\0&0&0&\end{bmatrix}$$

2) 기약 행 사다리꼴(RREF)

메모 포함[이28]: Reduced Row Echelon Form

행 사다리꼴 중에서 추가 조건을 만족시키면 기약 행 사다리꼴임.

1. 각 행의 피벗을 포함하는 열에는, 피벗 이외의 항들이 모두 0이다.

즉, 기약 행 사다리꼴은 3번 조건을 만족하면서 피벗의 위쪽과 아래쪽 수가 모두 0이면 됨.

$$\begin{bmatrix}1&0&0\\0&1&0\\0&0&1\end{bmatrix}, \begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\end{bmatrix}, \begin{bmatrix}1&0&0&5\\0&0&1&2\end{bmatrix}, \begin{bmatrix}1&0\\0&1\end{bmatrix}, \begin{bmatrix}1&0&2&5\\0&1&3&6\\0&0&0&0\end{bmatrix}$$

4. 계수(Rank)

행렬을 행 사다리꼴로 만들었을 때, 행 전체가 0이 아닌 행의 개수.
피벗의 개수.

어떤 행렬의 Rank를 구하기 위해서는 우선 행 사다리꼴로 만들어야 함.

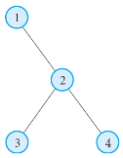
3. 그래프

1. 그래프 기초

1. 그래프

그래프 G=(V, E)는 유한한 개수의 Vertex나 node의 집합인 V와, edge의 집합인 E로 이루어짐.

(ex.



V = {1, 2 ,3 ,4}, E = {{1, 2}, {2, 3}, {2, 4}} 인 그래프)

(ex.

2. 그래프 관련 용어

1) Vertex와 edge

Vertex : 점, 정점

edge : 선, 연결선 -> Vertex의 쌍.

G = (V, E)에서 G는 graph, V는 Vertex, E는 edge임.

2) 방향

그래프의 edge가 가지는 방향은 해당 그래프가 가지는 목적이나 의미를 나타내는 것.

3) 이웃, N(V)

두 Vertex가 하나의 edge로 직접 이어져 있으면 이 두 Vertex는 서로 이웃한다고 함.

$N(v)$

N(V)는 Vertex V의 모든 이웃의 집합을 의미함.
(ex. N(a) = {b, c, f} 등과 같이 집합 형식으로 작성함.)

4) 차수(degree)

Vertex V의 차수 deg(V)는 V에 연결된 edge의 개수를 의미함.

$deg(v)$

루프는 하나 당 2번 연결된 것으로 계산함.

모든 Vertex들의 차수의 합은 edge 개수의 2배임. (악수 정리)

5) walk, trail, path

walk : Vertex들을 지나는 일련의 edge들.
trail : edge들을 한 번씩만 지나는 길.
path : Vertex들을 한 번씩만 지나는 길. -> 단, 시작 Vertex와 끝 Vertex는 같을 수 있음.

메모 포함[이29]: walk > trail > path 순서대로 포함관계를 가진다.

메모 포함[이30]: 한붓그리기와 유사한 개념.

6) 연결성

그래프에서 Vertex들이 edge로 모두 이어져 있으면 연결되었다고 함.

그래프의 연결을 제거하기 위해 제거해야 하는 Vertex들의 최소한의 수(k)가 클수록 연결성이 높다고 함.

7) 폐쇄성

시작 Vertex와 끝 Vertex가 같다면 폐쇄된 그래프라고 함.

+ 악수 정리(handshaking 정리)

방향성이 없는 그래프에서 Vertex 각각의 차수의 합과 edge 개수 사이의 관계를 일반화한 공식.

$$2m = \sum_{v \in V} \deg(v)$$

2. 다양한 그래프

1. 다양한 그래프

메모 포함[이31]: 그래프는 그 그래프를 그리는 목적이나 그래프의 의미 등을 고려하여 그리는 방식을 정한다.

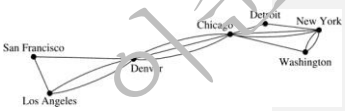
1) 단순 그래프

기본적인 그래프는 오른쪽 그림과 같음.

2) 멀티 그래프

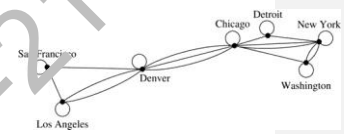
multiple edge를 포함한 그래프.

두 Vertex를 여러 개의 선으로 이은 것을 multiple edge라고 함.



3) 의사 그래프

하나의 Vertex에서 나와서 동일한 Vertex로 들어가는 edge가 있는 그래프.
이런 edge를 루프라고 함.



4) 방향 관련 그래프들

방향 그래프(directed graph, digraph) -> 방향성이 존재하는 그래프.
-> edge를 화살표로 표시해서 방향을 나타냄.
-> 방향성 그래프와 방향성 멀티그래프로 나눌 수 있음.

방향이 없는 그래프(undirected graph) -> 방향성이 존재하지 않는 그래프.
-> 특별한 언급이 없다면 그래프에는 방향이 없음.

+ 그래프 종류별 방향성, 멀티, 루프 여부

TABLE 1 Graph Terminology.

Type	Edges	Multiple Edges Allowed?	Loops Allowed?
Simple graph	Undirected	No	No
Multigraph	Undirected	Yes	No
Pseudograph	Undirected	Yes	Yes
Simple directed graph	Directed	No	No
Directed multigraph	Directed	Yes	Yes
Mixed graph	Directed and undirected	Yes	Yes

2. 방향성 그래프(directed graph)

방향성이 존재하는 그래프.

D = (V, A)로 표기함.

1) 인접

방향성 그래프에서 두 Vertex A, B가 있을 때, A로부터 B로 directed edge가 이어진다면, A는 B에 인접하다고 하고 B는 A에 인접된다고 함.

메모 포함[이32]: 방향성 모서리.

2) 입/출력 차수

해당 Vertex로 들어오는 directed edge의 수를 입력 차수라고 함.
입력 차수는 deg()에 -를 붙여서 표기함.

해당 Vertex에서 나가는 directed edge의 수를 출력 차수라고 함.
출력 차수는 deg()에 +를 붙여서 표기함.

$deg^+(v)$
$deg^-(v)$

루프는 입력 차수와 출력 차수에 모두 반영됨.

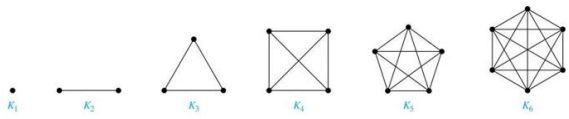
입력 차수의 총합은 출력 차수의 총합과 같음.

3. 완전 그래프(K_n)

모든 Vertex 각각이 자신을 제외한 다른 모든 Vertex와 edge로 연결되어 있는 그래프.
전부 이어져 있는 그래프.

n개의 Vertex를 갖는 완전 그래프는 K_n으로 표기함.

K_n의 모서리 개수는 $\frac{1}{2}(n \times (n-1))$ 개. (중복조합)



4. Cycle/Wheel 그래프

1) Cycle 그래프(C_n)

Vertex의 바깥쪽만 전부 연결한 그래프.

n개의 Vertex로 이루어진 Cycle 그래프는 C_n으로 표기함.

Vertex의 개수와 edge의 개수가 같음.

2) Wheel 그래프(W_n)

내부의 한 점과 나머지 Vertex들을 모두 연결한 Cycle 그래프.

n개의 Vertex로 이루어진 Wheel 그래프는 W_n으로 표기함.

edge의 개수는 Vertex의 개수의 2배임.

5. Cube 그래프

큐브 모양의 그래프.
나름의 규칙대로 만들어지는 그래프.

n차원 Cube 그래프는 Q_n으로 표기함.

1) 1차원 그래프(Q_1)

1. 왼쪽에 0번 Vertex를, 오른쪽에 1번 Vertex를 [illegible]는다.
2. 동일한 자리의 숫자끼리 비교했을 때 다른 숫자가 1개 이하인 경우 두 Vertex를 연결한다.
(2개 이상 다르면 연결하지 않음.)

2) 2차원 그래프(Q_2)

1. 1차원 그래프 2개를 가로로 그대로 나열한다.
2. 아래쪽 그래프에는 번호 앞에 0을, 위쪽 그래프에는 번호 앞에 1을 붙인다.
3. 동일한 자리의 숫자끼리 비교했을 때 다른 숫자가 1개 이하인 경우 두 Vertex를 연결한다.
(2개 이상 다르면 연결하지 않음.)

3) 3차원 그래프(Q_3)

1. 2차원 그래프 2개를 위아래로 그대로 나열한다.
2. 아래쪽 그래프에는 번호 앞에 0을, 위쪽 그래프에는 번호 앞에 1을 붙인다.
3. 동일한 자리의 숫자끼리 비교했을 때 다른 숫자가 1개 이하인 경우 두 Vertex를 연결한다.
(2개 이상 다르면 연결하지 않음.)

3) 4차원 그래프(Q_4)

1. 3차원 그래프 2개를 나열한다.
-> 한 육면체 안에 육면체를 그릴 수도 있고, 떨어진 두 육면체를 그릴 수도 있음.
2. 아래쪽 그래프에는 번호 앞에 0을, 위쪽 그래프에는 번호 앞에 1을 붙인다.
-> 아래, 위의 구분이 어렵다면 그냥 한 육면체에는 0을, 다른 육면체에는 1을 붙이면 됨.
3. 동일한 자리의 숫자끼리 비교했을 때 다른 숫자가 1개 이하인 경우 두 Vertex를 연결한다.
(2개 이상 다르면 연결하지 않음.)

6. 이분 그래프(bipartite graph)

그래프 G에서 Vertex의 집합 V를 V_1과 V_2로 분리했을 때, V_1과 V_2 각각 내부에 있는 Vertex끼리는 edge가 없을 경우, 그래프 G는 이분되었다고 함.

(V_1, V_2)를 V의 이분이라고 함.

odd cycle은 이분될 수 없고, even cycle은 이분될 수 있음.
그래프 내에 odd cycle이 포함되어 있으면 그 그래프는 이분될 수 없음.
odd cycle : Vertex의 개수가 홀수인 cycle 그래프
even cycle : Vertex의 개수가 짝수인 cycle 그래프

1) 이분 가능 여부 확인법

방법1. Vertex들을 두 개의 집합으로 구분하여 확인한다.
-> 차수가 가장 높은 것부터 시작해서 하나의 집합에 들어갈 수 없는 것들을 분리.

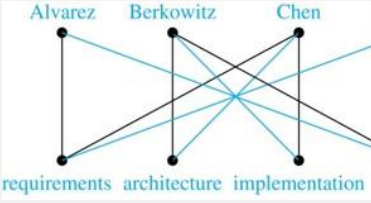
방법2. 그림 상에서 Vertex들에 색깔을 지정하여 확인한다.
-> 차수가 가장 높은 것부터 시작해서 하나의 집합에 들어갈 수 없는 것들을 분리.
-> 그림으로 표시하면 실수를 줄일 수 있음.

방법3. 그래프의 모든 사이클이 even cycle인지 확인한다.

2) 사용 예시

이분 그래프는 Vertex들을 성질이나 특성에 따라 나누어야 할 때 사용함. (ex. 작업할당)

메모 포함[이33]: 아래의 방법들은 그래프가 이분되기 위한 필요충분 조건들이다.



7. 매칭

그래프 G = (V, E)에서 매칭 M은 같은 Vertex에 연결되어 있지 않는 edge들의 집합임.

크기와 개수 구분 유의하기.

1) 매칭(집합)의 크기

매칭은 집합이므로, 크기가 n인 매칭은 원소의 개수가 n개인 집합을 의미함.
같은 Vertex에 연결되어 있지 않은 n개의 edge들로 이루어진 집합인 것.

어떤 그래프에서 크기가 1인 매칭의 개수는 edge의 개수와 같음.

2) 완전매칭, 최대매칭

완전매칭 : 매칭의 각 edge들에 연결된 Vertex들의 집합이 해당 그래프의 모든 Vertex들을 포함하는 매칭.
-> Vertex의 개수가 짝수인 경우에만 완전매칭이 존재할 가능성이 있음. (모든 Vertex를 포함해야 하므로.)

최대매칭 : 해당 그래프에서 크기가 최대인 매칭. 즉, edge의 개수가 최대인 매칭.

8. 동형 그래프(Isomorphic graph)

Vertex 와 edge 의 연결 상태가 같으면 Vertex 의 이름이나 전체적인 모양이 다르더라도 동형이라고 함.
서로 동형인 두 그래프 각각을 동형 그래프라고 함.

1) 동형 여부 판별 방법

그래프 문제에는 정해진 해법이 없지만, 아래의 방법들을 하나씩 시도해볼 수 있음.

간단한 그래프인 경우.
1. 직접 머릿속으로 그림을 움직이고 상상해서 판별한다.

차수로 판별하기.
1. Vertex 각각의 차수를 파악한다.
2. 두 그래프의 Vertex들이 가지는 차수의 분포가 서로 다를 경우 동형이 아니다. (반례를 드는 것.)

Vertex들이 이루는 도형으로 판별하기.
1. Vertex들이 이루는 도형들의 종류와 개수를 파악한다.
2. 두 그래프의 Vertex들이 가지는 도형들의 분포가 다를 경우 동형이 아니다. (반례를 드는 것.)

메모 포함[이34]: 그냥 머리 잘 굴리는 게 답이다.

메모 포함[이35]: ex.
그래프 A에는 차수가 4인 Vertex가 3개 있는데, 그래프 B에는 차수가 4인 Vertex가 2개밖에 없다면, 이 두 그래프는 동형이 아니다.

메모 포함[이36]: ex.
그래프 A에는 사각형이 3개 존재했지만, 그래프 B에는 사각형이 2개밖에 없다면 이 두 그래프는 동형이 아니다.

3. 그래프의 수치화

컴퓨터가 확인할 수 있게 하려면 그래프를 수치화해야 함.

1. 인접목록

그래프를 수치화한 표.

방향성이 없는 그래프 -> Vertex와 인접 Vertex로 나타냄.
방향성이 있는 그래프 -> 시작 Vertex와 종료 Vertex로 나타냄.

TABLE 1 An Adjacency List for a Simple Graph.

정점	인접 정점들
a	b, c, e
b	a
c	a, d, e
d	c, e
e	a, c, d

TABLE 2 An Adjacency List for a Directed Graph.

시작	종료 정점들
a	b, c, d, e
b	b, d
c	a, c, e
d	
e	b, c, d

2. 인접행렬(단순 그래프의 인접행렬)

그래프를 수치화한 행렬.

특별한 언급이 없다면 인접행렬은 단순 그래프에 대한 것임.

메모 포함[이37]: 인접행렬을 그래프로, 그래프를 인접행렬로 전환할 수 있어야 한다.

1) 인접행렬 작성법

1. Vertex를 행의 위와 열의 옆에 작성하여 성분에 의미를 부여한다.
2. 두 Vertex 사이를 잇는 edge의 개수를 성분으로 작성한다.

$$\begin{array}{c} \\ A \\ B \\ C \\ D \end{array}\begin{array}{c} \begin{array}{cccc} A & B & C & D \end{array} \\ \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

2) 그래프로의 변환방법

1. 주어진 Vertex들을 임의로 그린다.
2. 행렬의 원소에 맞춰 edge를 그린다.

3) 인접행렬의 성질

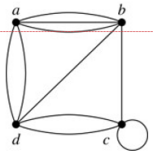
인접행렬은 대칭행렬이면서 정방행렬임.

3. 멀티/의사 그래프의 인접행렬

멀티 그래프 -> 두 Vertex 사이의 2개 이상의 edge가 있는 경우에도 성분으로 edge의 개수를 작성하여 표현할 수 있음..

의사 그래프 -> 하나의 Vertex에서 edge가 나가서 들어오는 경우에는 자기 자신에 해당하는 성분에 해당 edge의 개수를 작성하여 표현할 수 있음.

그래프는 작성자의 의도에 따라 표현 방식이 달라질 수 있음.



$$\begin{bmatrix} 0 & 3 & 0 & 2 \\ 3 & 0 & 1 & 1 \\ 0 & 1 & 1 & 2 \\ 2 & 1 & 2 & 0 \end{bmatrix}$$

메모 포함[이38]: ex. 두 Vertex 사이에 edge가 3개 있는 경우, 성분을 3으로 할 수도 있지만, 이 두 Vertex 사이의 관계는 하나이기 때문에 1로 할 수도 있다. 이 경우, 별 말이 없는 경우는 3으로 한다.

4. 방향성 그래프의 인접행렬

방향성 그래프의 인접행렬은 단순 그래프의 인접행렬과는 달리, 대칭행렬이 아님.

1) 작성법

1. Vertex를 행의 위와 열의 옆에 작성하여 성분에 의미를 부여한다.
2. 두 Vertex 사이를 잇는 edge중, 해당 Vertex로부터 나가는 edge의 개수를 성분으로 작성한다.
(열 옆에 작성한 Vertex를 기준으로 함.)

메모 포함[이39]: ex. 열 옆에 작성한 Vertex 목록이 a, b, c, d라면, a에 해당하는 행에는 Vertex a에서 edge가 나가서 들어가는 Vertex에 1을 작성한다.

5. 결합행렬(incidence matrix)

그래프를 Vertex와 edge의 관계로 수치화한 행렬.

그래프 G에 대한 결합행렬은 M, M_G, M_{ij}로 표기함.

메모 포함[이40]: 인접행렬은 Vertex끼리의 관계로 수치화했지만, 결합행렬은 Vertex와 edge의 관계로 수치화한 것이다.

1) 작성법

1. Vertex는 행(왼쪽에 세로로)에 해당되게, edge는 열(위쪽에 가로로)에 해당되게 작성한다.
2. 해당 Vertex가 해당 edge에 연결되어 있다면 1을, 연결되어 있지 않으면 0을 성분으로 작성한다.

2) 결합행렬의 특징

1. 인접행렬과는 달리, 정방행렬이 아닐 수 있음.-> Vertex와 edge의 개수가 항상 같은 것은 아니기 때문.
2. 루프가 아닌 열에는 1이 2개 존재함. -> 하나의 edge는 두 개의 Vertex와 연결되어 있기 때문.
3. 루프인 열에는 1이 1개 존재함. -> 루프는 Vertex에서 나와서 들어가기 때문.

4. 오일러리언과 해밀토니언

1. 오일러리언(Eulerian)

연결된 그래프 G 의 모든 edge 를 포함하는 폐쇄된 trail 이 있다면, G 는 오일러리언이라고 함.
edge 들을 한 번만 지나면서 모든 edge 들을 지나고 시작 Vertex 로 돌아오도록 그린 것.

1) 오일러리언 판별 방법 : 오일러 공식

연결된 그래프가 오일러리언일 필요충분 조건은, 해당 그래프의 각 Vertex 들의 차수가 전부 짝수인 것임.
-> 들어오는 edge 가 있으면 나가는 edge 도 있어야 하기 때문.

2. 세미-오일러리언(semi-Eulerian)

연결된 그래프 G 의 모든 edge 를 포함하는 trail 이지만, 폐쇄되지 않은 그래프는 세미-오일러리언이라고 함.
edge 들을 한 번만 지나면서 모든 edge 들을 지나도록 그린 것. 시작 Vertex 로 돌아올 필요는 없음.

1) 세미-오일러리언 판별 방법

연결된 그래프가 세미-오일러리언일 필요충분 조건은, 해당 그래프의 Vertex 중 차수가 홀수인 것이 정확히 2 개인 것임.
-> 시작 Vertex 와 끝 Vertex 에는 각각 들어오는 edge 와 나가는 edge 만 있으면 되기 때문.

+ 오일러 경로

오일러리언과 세미-오일러리언 모두 오일러 경로를 가진다고 함.

메모 포함[이41]: 한붓그리기.

메모 포함[이42]: = 시작 Vertex와 끝 Vertex가 다른.

2. 해밀토니언(Hamiltonian)

그래프 G 의 모든 Vertex 들을 정확히 한 번씩 지나는 폐쇄된 cycle 을 해밀턴 사이클이라고 함.
그래프 G 가 해밀턴 사이클을 가지고 있다면, G 는 해밀토니언이라고 함.
Vertex 들을 한 번만 지나면서 모든 Vertex 들을 지나고 시작 Vertex 로 들어오도록 그린 것.

메모 포함[이43]: = 해밀턴 순회

1) 해밀토니언 판별 방법

1. Ore 의 정리
그래프 G 가 N(N >= 3)개의 Vertex 를 가진 단순 그래프일 때, 인접하지 않은 Vertex v, w 에 대해서 deg(v) + deg(w) >= N 이면 G 는 해밀토니언임.

메모 포함[이44]: 해밀토니언이 되는 필요충분 조건은 아니지만, 너무 깊게 파헤치지는 않기로 했다.

두 방법 중 뭘 쓰든 상관없다고 한다.

2. 응용 정리
그래프 G 가 N(N >= 3)개의 Vertex 를 가진 단순 그래프일 때, Vertex v 에 대해서 deg(v) >= N/2 이면 G 는 해밀토니언임. (?)

3. 세미-해밀토니언(semi-Hamiltonian)

그래프 G 의 모든 Vertex 를 정확히 한 번씩 지나는 경로를 해밀턴 경로라고 함.
그래프 G 가 해밀턴 경로를 가지고 있다면, G 는 세미-해밀토니언이라고 함.
Vertex 들을 한 번만 지나면서 모든 Vertex 들을 지나도록 그린 것.

+ 해밀턴 순회와 해밀턴 경로

해밀턴 순회 : 그래프에서 모든 꼭짓점들을 정확히 한 번씩 지나는 순회.
(순회란 시작 Vertex 와 끝 Vertex 가 동일한 경로를 의미함.)

해밀턴 경로 : 그래프에서 모든 꼭짓점들을 정확히 한 번씩 지나지만 시작 Vertex 로 돌아오지 않는 경로.

+ 최단 경로

이 수업에서 다루는 최단 경로 문제들은 단순히 갈림길마다 경우를 나누어서 계산하는 것이 전부임.

edge 별 가중치 등을 고려하여 계산함.

메모 포함[이45]: ex.
해당 edge를 지나면 가중치가 더해지는 방식. 총 가중치 값이 최소가 되는 경로를 찾는 문제 등이 있다.

이때 경로는 목적지 반대 방향으로 갈 수도 있다는 것 조심하기. (ppt 예제에 그런 문제 있음.)

4. 함수

1. 함수

1. 함수(사상, mapping)

관계의 특수한 형태.

첫 번째 원소가 같지 않은 순서쌍들의 집합.

A 와 B 를 집합이라고 했을 때, A 의 모든 원소 각각에 B 의 원소를 단 하나만 대응시킨 것.

정의역의 범위에 따라 함수식이 다른 경우, 중괄호로 나누어 표기함.

1) 함수의 곱셈, 덧셈 표기

덧셈 -> (f1 + f2)(x) = f1(x) + f2(x) 로 표기함.

곱셈 -> (f1f2)(x) = f1(x) f2(x) 로 표기함.

2. 함수 관련 용어들

1) 상(image), 원상

f : X -> Y 를 함수라고 할 때, y = f(x)인 경우 y 는 함수 f 에 의한 상(image) 또는 함수값이라고 함.

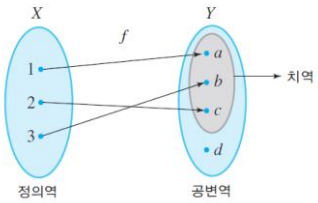
f : X -> Y 를 함수라고 할 때, y = f(x)인 경우 x 는 y 의 원상이라고 함.

2) 정의역, 공역, 치역

정의역(domain) -> Dom(f)로 표기.

공역(공변역, codomain)

치역(range) -> Ran(f)로 표기.



〈그림 6.1〉 함수의 정의역, 공변역, 치역

메모 포함[이46]: 함수는 관계로 나타낼 수 있다.
ex.
{(1, a), (2, b). (3, c), (4, d)}

함수를 f : A -> B 등으로 표기할 때, A와 B 위치에는 문자가 아니라 집합 자체가 올 수도 있다.
ex.
f : {a, b, c} -> {d}

메모 포함[이47]: 함수 f가 A와 B에 대한 함수라면, f는 A에서 B로 사상(대응)한다고 표현한다.

메모 포함[이48]: 함수에서 하나의 원소는 반드시 단 하나의 원소를 가리켜야 한다.

메모 포함[이49]: 이 경우, 부분집합 A에 대한 부분은 f_A로 표기하기도 한다.

3. 함수 그래프

함수 f : A->B 에 대한 함수 그래프 G 는 $x \in A$ 이고 y = f(x) 인 순서쌍 (x, y)의 집합을 나타냄.

함수 그래프는 주로 좌표 평면에 나타냄.

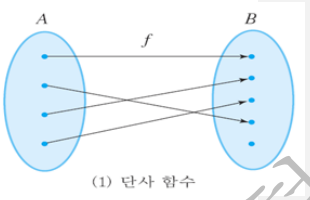
4. 컴퓨터 언어에서의 함수

컴퓨터 언어에서의 함수도 수학적인 함수의 기능과 유사한 역할을 수행함.

프로그램에서는 함수를 호출하여 매개변수(정의역)를 함수에 전달하고, 값(상)을 리턴 받음.

2. 여러 가지 함수

1. 단사 함수(일대일 함수)

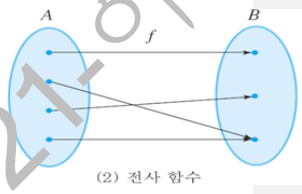


(1) 단사 함수

정의역에 속한 모든 a, b 에 대해서 f(a) = f(b) 이면 a = b 인 함수.
정의역의 모든 원소들이 서로 다른 원소로 대응되는 함수.

함수 f : A -> B 가 단사 함수인 경우, B 의 크기가 A 의 크기보다 항상 크거나 같음.

2. 전사 함수(onto function)

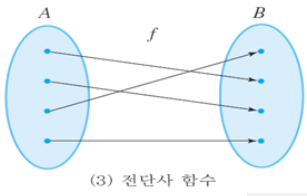


(2) 전사 함수

공역과 치역이 같은 함수.
공역에 가리켜지지 않은 원소가 존재하지 않는 함수.

함수 f : A -> B 가 단사 함수인 경우, A 의 크기가 B 의 크기보다 항상
크거나 같음.

3. 전단사 함수(일대일 대응 함수)



(3) 전단사 함수

단사함수이면서 전사함수인 함수.
정의역의 모든 원소들이 서로 다른 원소로 빠짐없이 대응되는 함수.

함수 f : A -> B 가 단사 함수인 경우, A 와 B 의 크기가 같음.

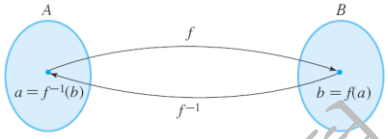
+ 전사/단사/전단사 함수 판별

집합 관계를 직접 그림으로 그려보는 것이 좋음.

4. 역함수

f : A -> B 가 전단사 함수(일대일대응)일 때,
f 의 역함수 f^{-1}은 f^{-1} : B -> A 이고, x ∈ A, y ∈ B, f(x) = y -> f^{-1}(y) = x 로 정의함.

함수 f가 전단사 함수(일대일 대응 함수)일 때, f는 역함수가 존재함.
함수 f의 역함수가 존재하는 경우, 함수 f는 일대일 대응 함수임.



〈그림 6.4〉 함수 f의 역함수 f^{-1}

5. 상수 함수(constant function)

정의역의 모든 원소가 전부 하나의 원소로 대응되는 함수.

6. 합성 함수

두 함수를 합성한 함수.

$$(f \circ g)(x) = f(g(x))$$

여러 함수들을 합성할 경우, 결합법칙이 성립함.

7. 항등 함수

f : A -> A, $a \in A$, f(a) = a 인 경우 함수 f는 항등함수이고, I_A로 표기함.

메모 포함[이50]: ex.
f = {(1, a), (2, b), (3, c)} 일 때,
f^{-1}={(a, 1), (b, 2), (c, 3)} 이다.

메모 포함[이51]: ex.
$f \circ (g \circ h) = (f \circ g) \circ h$

메모 포함[이52]: 즉, $a \in A$, $I_A(a) = a$ 이다.

5. 확률과 통계

1. 비둘기집 원리

1. 비둘기집 원리

n 개의 비둘기 집에 (n + 1) 마리 이상의 비둘기가 들어갔다면, 두 마리 이상의 비둘기가 들어간 비둘기 집이 적어도 하나 있다는 원리.

여러 상황에 적용할 때는, 대상이 전부 고르게 분포하는 경우를 떠올리면 쉽게 이해할 수 있음.
또는, 조건이 달성되지 않는 최악의 상황을 떠올리면 쉽게 이해할 수 있음.

1) 증명(기본)

1. 귀류법
두 마리 이상의 비둘기가 들어간 비둘기 집이 없다고 가정하면, 비둘기의 수는 n 마리임.
-> 모순. 비둘기는 (n + 1)마리임.

2. 평균값
한 비둘기 집에 들어가 있는 비둘기는 (n + 1)/n 마리임. (n + 1)/n > 1 이므로 두 마리 이상의 비둘기가 들어간 비둘기 집이 적어도 하나 있음.
-> 1 보다 큰 값이 존재하므로 평균이 1 보다 큼.

2) 예시

1. 원소의 개수가 (k + 1)보다 크거나 같은 집합으로부터 원소의 개수가 k 인 집합으로의 함수는 일대일 대응이 될 수 없음.

2. 100 명의 사람들이 있으면 태어난 사람이 9 명 이상인 달이 적어도 하나는 있음. (100/12)

메모 포함[이53]: 시험에서는 계산 전부 직접 할 필요 없고, 7! x 3P2 등으로만 써도 된다.

메모 포함[이54]: 확률과 통계 문제라고 하기보다는 논리적인 사고가 필요한 문제이다.

메모 포함[이55]: 시험 전에 영상에 나온 비둘기집 원리 관련 문제 다시 풀어 보기. 강의자료에 없어서 영상을 봐야 한다.
10주차 두 번째 영상.
11주차 첫 번째 영상.

메모 포함[이56]: 모순되는 것으로 증명하는 방법.

3) 문제풀이

1. 비둘기집 원리를 사용해야 하는 문제임을 판단해야 함.
-> ~가 반드시 존재한다, 항상 ~하도록 등의 형태이면 비둘기집 원리를 사용하는 문제임.
2. 비둘기와 비둘기집을 구분해야 함.
3. 평균 사용 고려하기 / 최댓값, 최솟값 구해보기 / 가능한 전체 경우의 수 구해보기 / worst case 를 생각해보기

2. sever-workstation 예시 문제

1) 문제 상황

서버 10 대와 워크스테이션 15 대가 있음.
각각의 서버에는 한 번에 하나의 워크스테이션만 접속할 수 있음.
워크스테이션은 서버와 연결선으로 연결되어야 해당 서버에 접속할 수 있음.

15 대의 워크스테이션 중 임의로 정한 10 대(또는 그 이하)의 워크스테이션이 모두 서버에 접속할 수 있음을 보장하려 할 때, 필요한 연결선의 최소 개수는?

2) 풀이

더 간단하게 서버 10 대와 워크스테이션 11 대인 것으로 생각해 보면 이해가 쉬움.

서버 10 대와 워크스테이션 10 대를 각각 연결선으로 연결하고, 나은 워크스테이션 5 대는 각 서버와 모두 연결함. 그러면 총 10 + 50 = 60 개의 연결선이 필요함.

이 방법으로 60 개인 것을 알았으니, 59 개 이하인 경우 접속이 보장되지 않는다는 것을 증명해야 함.

3) 접속 보장 실패 증명

1. 평균을 이용해서 반례 찾기
59 개인 경우, 59 / 10 = 5.9 임. 각 서버에 연결된 연결선의 개수의 평균이 5.9 이므로 어떤 서버에는 5 개의 연결선이 연결되어 있다는 것을 알 수 있음. 5 개의 연결선이 연결된 서버가 존재하는 상황을 생각해 보면, 5 개가 모두 추가 워크스테이션 5 대에 연결된 경우 하나의 기본 워크스테이션(10 대 중 하나)에는 서버와 연결선으로 이어져 있지 않음. 기본 워크스테이션 10 대가 선택된 경우 접속 보장이 실패한다는 것을 알 수 있음.

2. 경우의 수

1. 경우의 수

1) 곱셈법칙/덧셈법칙

곱셈법칙(and) : 하나의 과정이 두 개의 연속된 작업으로 분리될 때, 각 작업에 대한 경우의 수를 서로 곱하면 전체 과정에 대한 경우의 수임.

덧셈법칙(or) : 하나의 과정이 두 개의 연속되지 않은 작업으로 분리될 때, 연속되지 않은 두 작업에 대한 경우의 수를 서로 더하면 전체 과정에 대한 경우의 수임.

3. 순열

1. 순열(nPr)

$$nPr = \frac{n!}{(n-r)!}$$

서로 다른 원소를 순서를 고려하여 일렬로 배열하는 것.

서로 다른 n 개의 원소를 한 줄로 배열하는 순열의 수는 n!임.

집합 A 의 순열이란 A 의 원소들의 순서적인 배열임. 이 배열의 개수가 집합 A 의 순열의 수임.
집합 A - 2 의 순열이란 A 의 원소들 중 2 개를 사용한 순서적인 배열임. 이 배열의 개수가 A - 2 순열의 수임.

메모 포함[이57]: ex. S = {1, 2, 3}의 순열과 순열의 수는?

메모 포함[이58]: ex. S = {1, 2, 3}일 때 S - 2 순열의 수는?

2. 계승

1 부터 n 까지의 모든 자연수의 곱을 n 의 계승 또는 n factorial 이라고 읽으며, !으로 나타냄.

0! = 1
1! = 1
n! = n(n - 1)!

+ 적어도 ~
전체에서 특정 경우를 빼는 형식으로 접근을 시도하는 것이 좋음.

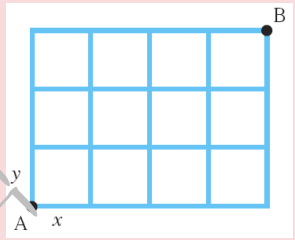
3. 같은 것을 포함하는 순열

$$\frac{n!}{p!\,q!\,r!}$$

n 개 중에 같은 것이 p 개, q 개, r 개 있는 순열은 p!, q!, r!으로 나누어 줌.

특히 최단거리에서 활용함.

메모 포함[이59]: ex.



이 경우 A에서 B로 가는 경우의 수는 7!/(3!4!)이다. 4개의 x와 3개의 y를 일렬로 배열하는 것과 같다.

4. 원순열

$$(n-1)!$$

회전시켰을 때 같아지는 경우의 순열.

원순열 문제를 풀 때에는 회전시켰을 때 같아지지 않도록 원소를 배정해야 함.
기본 원순열 계산을 할 때 (n-1)!인 것도 동일한 원리임.
(첫 번째 원소는 경우에 포함되지 않기 때문에 -1 한 것.)

5. 중복순열($n\Pi r$)

$$n\Pi r = n^r$$

원소를 중복해서 일렬로 배열하는 순열.

4. 조합

1. 조합(nCr)

$$nCr = \frac{n!}{r!\,(n-r)!}$$

서로 다른 n 개의 원소 중에서 순서를 생각하지 않고 서로 다른 r 개를 선택하는 방법의 수.

2. 중복조합(nHr)

$$nHr = (n+r-1)Cr$$

서로 다른 n 개의 원소 중에서 순서를 생각하지 않고 중복해서 r 개를 선택하는 방법의 수.

3. 문제 예시

1. x + y + z = 10 를 만족하는 x, y, z 순서쌍의 개수는? (단, x, y, z 는 음이 아닌 정수)
-> 중복조합 사용.

2. 12 명을 2 명, 2 명, 2 명, 6 명으로 나누는 경우의 수는? -> 조 나누기
-> 조합으로 사람을 뽑은 뒤에 1/3! 곱해줘야 함. (5 명인 조가 2 개인데, 어느 쪽에 뽑히든 같으므로.)

6. 집합

1. 집합

1. 집합(set)

원소(element)라고 불리는 서로 다른 수학적 객체들의 모임.

집합은 중복된 원소를 여러 개 가지지 않음. (하나만 표기함.)

1) 집합 관련 기호(연산자)

$a \in A$: a 가 집합 A 의 원소라는 것을 나타냄.

$a \notin A$: a 가 집합 A 의 원소가 아니라는 것을 나타냄.

$B \subset A$: B 가 집합 A 의 부분집합이라는 것을 나타냄.

$B \not\subset A$: B 가 집합 A 의 부분집합이 아니라는 것을 나타냄.

2. 집합 표현법

1) 원소나열법

집합의 원소들을 {} 안에 하나씩 나열하는 방법.

A = { 1, 2, 3 }

의미가 명확한 경우 모든 원소를 나열하는 대신, ...을 작성함.

+ 원소의 위치

두 집합이 각 원소의 위치가 다르게 표현되어도, 원소의 구성이 같다면 동일한 집합임.
단, 순서쌍 내부의 원소의 위치는 바뀔 수 없음.

2) 조건제시법

집합의 원소들이 가지고 있는 특정한 성질을 작성하는 방법.

A = { x | p(x) }

중괄호를 | 로 구분하여, 앞에는 원소의 형태를 작성하고, 뒤에는 조건을 작성함.

조건에 종종 작성하는 공인 기호들.

Z : 정수의 집합. (첨자로 + 붙이면 양의 정수, - 붙이면 음의 정수.)

N : 자연수의 집합.

R : 실수의 집합.

Q : 유리수의 집합.

C : 복소수의 집합.

S_n: 1 ~ n 자연수의 집합.

메모 포함[이60]: 여러 집합들을 조건제시법으로 나타낼 수 있어야 한다.

ex.

A ∩ B = { x | x ∈ A and x ∈ B }

A ∪ B = { x | x ∈ A or x ∈ B }

메모 포함[이61]: ex.

A = {x | x는 자연수이고, 1< x < 6}

= {2, 3, 4, 5}

ex.

{1, 4, 9, 16, 25, ...} 을 조건제시법으로 나타내면

{x^2 | x는 자연수}

{x | x = k^2, k는 자연수}

등이다.

ex.

유리수의 집합을 조건제시법으로 작성하면

{a/b | a, b는 정수, b는 0이 아님}

이다.

3. 카디날리티(Cardinality)

집합 내의 서로 다른 원소의 개수.

집합 S 의 카디날리티는 |S|로 표기함.

+ 공집합(∅)

원소를 갖지 않는 집합.

∅로 표기함.

집합 A = {∅} 일 때, A 는 공집합이 아님.

+ 유한/무한집합

유한집합 : 원소의 개수가 유한개인 집합.

무한집합 : 원소의 개수가 무한개인 집합.

+ 닫힌/열린구간

$[a,b] = \{x \mid a \le x \le b\}$ 닫힌구간 (closed interval)

$[a\ b) = \{x \mid a \le x < b\}$

$(a\ b] = \{x \mid a < x \le b\}$

$(a\ b) = \{x \mid a < x < b\}$ 열린구간 (closed interval)

4. 부분집합(subset)/진부분집합

부분집합 : A 의 모든 원소가 B 의 원소에 속하면 A 는 B 의 부분집합임.
진부분집합 : A 의 모든 원소가 B 의 원소에 속하고 A 와 B 가 다른 집합일 때 A 는 B 의 진부분집합임.

어떤 두 집합이 부분집합/진부분집합 관계에 있다는 것을 증명하는 방법.
1. 벤 다이어그램으로 포함관계를 나타낸다.
2. 한 집합에서 고른 임의의 원소가 다른 집합에 존재한다는 것을 보인다.

어떤 두 집합이 부분집합/진부분집합 관계에 있지 않다는 것을 증명하는 방법.
1. 포함되지 않는 원소를 찾아 반례를 든다.

5. 멱집합

집합 S 의 멱집합은 집합 S 의 모든 부분집합을 원소로 가지는 집합임.

집합 S 의 멱집합은 P(S)로 표기함.

공집합(ø) 또한 원소로 포함하고 있어야 함.

|S| = n 인 경우, |P(S)| = 2^n 임.

메모 포함[이62]: ex.
S = {1, 2, 3}
$P(S) = \{\emptyset, \{1\}, \{2\}, \{3\}, \{1,2\}, \{2,3\}, \{3,1\}, \{1,2,3\}\}$

6. 서로소(Disjoint)

집합 A 와 집합 B 가 공통된 원소를 하나도 가지지 않는 경우.

2. 집합의 연산

1. 벤 다이어그램

집합 관계를 그림으로 나타낸 것.

집합의 연산에 사용하면 이해가 편리함.

이 수업에서 나오는 집합 증명의 대부분은 벤 다이어그램을 사용함.

메모 포함[이63]: 시험에 집합 관련 증명이 나오면, 벤 다이어그램을 단계별로 그리면 된다.

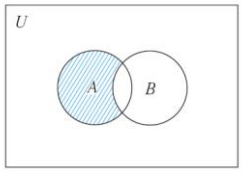
2. 집합의 연산

1) 기본 연산

합집합(∪)(Union) : 두 집합의 원소들을 모두 가지고 있는 집합.

교집합(∩)(Intersection) : 두 집합의 원소들 중 겹치는 원소들의 집합.

차집합(-, /)(Difference) : A 차집합 B 는 A 에는 속하고 B 에는 속하지 않는 원소들의 집합.



〈그림 3.7〉 차집합 $A-B$

2) 여집합(complement)

A 의 여집합은 전체집합의 원소들 중 A 의 원소들을 제외한 집합임.

A 의 여집합은 A^c로 표기함.

U 가 전체집합인 경우에만 사용 가능.

3) 곱집합(Cartesian Product)

집합 A, B 에 대해서 A 곱집합 B 는 $A \times B = \{(x, y) \mid x \in A, y \in B\}$

4) 포함 배제의 원리

1. $|A \cup B| = |A| + |B| - |A \cap B|$
2. $|A \cup B \cup C| = |A| + |B| + |C| - |A \cap B| - |B \cap C| - |C \cap A| + |A \cap B \cap C|$
3. $|A \cup B \cup C \cup D| = \ldots$ (동일한 원리)

증명은 벤 다이어그램으로 함.

3. 집합 연산의 성질

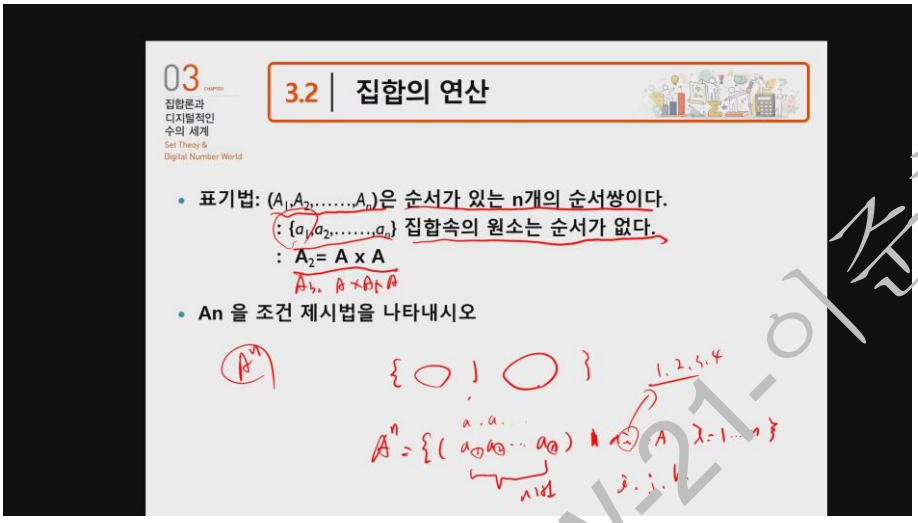
메모 포함[이64]: 법칙 이름 기억할 필요 없다.

분배 법칙, 흡수 법칙, 드 모르간의 법칙 정도만 제대로 기억하기.

증명은 벤 다이어그램으로 함.

$A \cup \phi = A,\ A \cap \phi = \phi$ $A \cup U = U,\ A \cap U = A$	항등 법칙 (identity law)
$A \cup B = B \cup A$ $A \cap B = B \cap A$	교환 법칙 (commutative law)
$(A \cup B) \cup C = A \cup (B \cup C)$ $(A \cap B) \cap C = A \cap (B \cap C)$ $(A \oplus B) \oplus C = A \oplus (B \oplus C)$	결합 법칙 (associative law)
$A \cup (B \cap C) = (A \cup B) \cap (A \cup C)$ $A \cap (B \cup C) = (A \cap B) \cup (A \cap C)$	분배 법칙 (distributive law)
$(A \cap B) \cup A = A$ $(A \cup B) \cap A = A$	흡수 법칙 (absorption law)
$\overline{\overline{A}} = A$	보 법칙 (complement law)
$A \cup \overline{A} = U,\ A \cap \overline{A} = \phi$ $\overline{U} = \phi,\ \overline{\phi} = U$	역 법칙 (inverse law)
$\overline{(A \cup B)} = \overline{A} \cap \overline{B}$ $\overline{(A \cap B)} = \overline{A} \cup \overline{B}$	드 모르간의 법칙 (De Morgan's law)
$A - B = A \cap \overline{B}$ $A - A = \phi$ $A - \phi = A$	기타 법칙

의문..?



이 문제. 다시 봐보기..?