Discrete Mathematics (0034)
Final Exam Lecture Notes

Yulwon Rhee (202211342)

Department of Computer Science and Engineering, Konkuk University

9 Week 9

9.1 관계

- 서로 다른 두 집합에 속하는 원소들 간의 순서$^{\text{Order}}$를 표현
- 순서쌍 집합(곱집합의 부분 집합)에 속하면서 순서쌍을 이루는 원소들은 '관계'가 있다

9.2 이항 관계$^{\text{Binary Relation}}$

집합 A에서 집합 B로 가는 관계
R: $A \times B$의 부분 집합
$a \in A, b \in B$일 때, $(a,b) \in R \to {}_aR_b$; $(a,b) \notin R \to {}_a\not{R}_b$

정의역$^{\text{Domain}}$: $\text{dom}(R) = \{a | a \in A\}$
공변역$^{\text{Codomain}}$: $\text{codom}(R) = \{b | b \in B\}$
치역$^{\text{Range}}$: $\text{ran}(R) = \{b | (a,b) \in R\} \subseteq B$

n항 관계$^{n\text{-ray Relation}}$: $R \subseteq A_1 \times A_2 \times \cdots \times A_n$의 부분 집합

역관계$^{\text{Inverse Relations}}$: B에서 A로의 관계, $R^{-1} = \{(b,a) | (a,b) \in R\}$, ${}_aR_b$ 존재 $\to {}_bR_a^{-1}$ 존재

9.3 관계의 표현: 서술식 방법

e.g.) $A = 1,2,3$에서 원소 a,b가 $a \geq b$인 관계 R

9.4 관계의 표현: 나열식 방법

- 화살표 도표$^{\text{Arrow diagram}}$
- 좌표 도표$^{\text{Coordinate diagram}}$:
  - 집합 A의 원소를 x축 위의 점으로, B의 원소를 y축 위의 점으로 표시
- 방향 그래프$^{\text{Directed graph}}$:
  - 관계 R이 하나의 집합 A에 대한 관계 표현일 때
  - A의 각 원소 $\Rightarrow$ 그래프의 정점$^{\text{Vertex}}$
  - $(a,b) \in R$이면 a에서 b로 화살표가 있는 연결선$^{\text{Edge}}$로 표현

- 관계 행렬 Relation matrix:
  - 부울 Boolean 행렬 이용
  - $A = \{a_1, a_2, \cdots, a_m\}$에서 $B = \{b_1, b_2, \cdots, b_n\}$로 가는 관계 R에 대한 $m \times n$ 행렬 $M_R = [m_{ij}]$

$$m_{ij} = \begin{cases} 1, (a_i, b_j) \in R \\ 0, (a_i, b_j) \notin R \end{cases}$$

  - e.g.) $A = \{1,2,3\}$과 $B = \{a,b\}$의 이항 관계
    $R = \{(1,b),(2,a),(2,b),(3,a)\}$

$$M_R = \begin{array}{c c} & \begin{array}{cc} a & b \end{array} \\ \begin{array}{c} 1 \\ 2 \\ 3 \end{array} & \begin{bmatrix} 0 & 1 \\ 1 & 1 \\ 1 & 0 \end{bmatrix} \end{array} \qquad M_{R^{-1}} = \begin{array}{c c} & \begin{array}{ccc} 1 & 2 & 3 \end{array} \\ \begin{array}{c} a \\ b \end{array} & \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \end{array}$$

9.5 관계의 성질

반사 관계 Reflexive Relation: 모든 $a \in A$에 대해 $(a,a) \in R$인 관계, $\begin{bmatrix} 1 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$

비반사 관계 Irreflexive Relation: 모든 $a \in A$에 대해 $(a,a) \notin R$인 관계, $\begin{bmatrix} 0 & & & \\ & 0 & & \\ & & \ddots & \\ & & & 0 \end{bmatrix}$

반사 관계도 비반사 관계도 아닌 경우: $\begin{bmatrix} 0 & & & \\ & 1 & & \\ & & \ddots & \\ & & & 1 \end{bmatrix}$

대칭 관계 Symmetric Relation:
$\exists a,b \in A$에 대해 $(a,b) \in R$이면 $(b,a) \in R$, (a,b) 존재 $\rightarrow (b,a)$ 존재
관계 행렬에서 대각 성분 기준으로 대칭이면 대칭 관계 성립

반대칭 관계:
$\exists a,b \in A$에 대해 $a = b$이고, $(a,b) \in R$이면 $(b,a) \in R$
$a \neq b$이고, $(a,b) \in R$이면 $(b,a) \notin R$

추이 관계: $\exists a,b,c \in A$에 대해 $(a,b) \in R$이고, $(b,c) \in R$이면 $(a,c) \in R$인 관계

9.6 합성 관계Composite Relation

A에서 B로의 관계 R_1과, B에서 C로의 관계 R_2에 대해서, A에서 C로의 합성 관계 $= R_1 \cdot R_2$ 또는 $R_1 R_2$

$$R_1 \cdot R_2 = \{(a,c) | a \in A, c \in C, (a,b) \in R_1 \text{이고 } (b,c) \in R_2\}$$

합성 관계의 연산: $R \cdot S = M_{R \cdot S} = M_R \odot M_S$

e.g.) $M_R = \begin{array}{c} \\ a \\ b \\ c \\ d \end{array} \begin{array}{c} 1\ 2\ 3 \\ \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \end{array} \qquad M_S = \begin{array}{c} \\ 1 \\ 2 \\ 3 \end{array} \begin{array}{c} x\ y\ z \\ \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \end{array}$

$$\begin{aligned} R \cdot S = M_R \odot M_S &= \begin{bmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \odot \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} (0\wedge1)\vee(1\wedge0)\vee(0\wedge1) & (0\wedge0)\vee(1\wedge1)\vee(0\wedge1) & (0\wedge1)\vee(1\wedge1)\vee(0\wedge0) \\ (0\wedge1)\vee(1\wedge0)\vee(1\wedge1) & (0\wedge0)\vee(1\wedge1)\vee(1\wedge1) & (0\wedge1)\vee(1\wedge1)\vee(1\wedge0) \\ (1\wedge1)\vee(0\wedge0)\vee(0\wedge1) & (1\wedge0)\vee(0\wedge1)\vee(0\wedge1) & (1\wedge1)\vee(0\wedge1)\vee(0\wedge0) \\ (1\wedge1)\vee(1\wedge0)\vee(1\wedge1) & (1\wedge0)\vee(1\wedge1)\vee(1\wedge1) & (1\wedge1)\vee(1\wedge1)\vee(1\wedge0) \end{bmatrix} \\ &= \begin{bmatrix} 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{bmatrix} \end{aligned}$$

합성 관계의 거듭제곱 $R^n = \begin{cases} R & (n = 1) \\ R^{n-1} \cdot R & (n > 1) \end{cases}$

기타 연산

- $R_1 \cap R_2 = M_{R_1 \cap R_2} = M_{R_1} \wedge M_{R_2}$
  $= \{(a,b) \in R_1 \cap R_2 | (a,b) \in R_1 \wedge (a,b) \in R_2\}$
- $R_1 \cup R_2 = M_{R_1 \cup R_2} = M_{R_1} \vee M_{R_2}$
  $= \{(a,b) \in R_1 \cup R_2 | (a,b) \in R_1 \vee (a,b) \in R_2\}$
- $R_1 - R_2 = M_{R_1 - R_2} = \{(a,b) \in R_1 - R_2 | (a,b) \in R_1 \wedge (a,b) \notin R_2\}$

9.7 추이 관계와 합성 관계

[정리] 추이 관계와 거듭제곱의 관계

집합 A에 대한 관계 R이 추이 관계일 필요충분조건은 모든 양의 정수 n에 대하여 $R^n \subseteq R$이다.

9.8 폐포Closure

폐포Closure: A상의 관계 R이 어떤 성질을 만족하지 않을 때, 그 성질을 만족하도록 순서쌍들을 추가하여 R^*(원하는 성질이 만족되는 가장 작은 집합)로 확장

성질 P에 대한 관계 R의 폐포: A에 대한 관계 R에 대해, R^*가 R을 포함하면서 성질 P를 가질 때, R^*는 P에 대한 R의 폐포

9.9 반사 폐포Reflexive Closure

A에 대해, R을 포함하면서 반사 관계를 갖는 관계 S
$S = R \cup \{(a,a) | a \in A\}$

9.10 대칭 폐포Symmetric Closure

A에 대해, R을 포함하면서 대칭 관계를 갖는 관계 S
$S = R \cup \{(b,a) \in A \times A | (a,b) \in R\} = R \cup R^{-1}$

9.11 추이 폐포Transitive Closure

A에 대해, R을 포함하면서 추이 관계를 갖는 관계 S
$S = R \cup \{(a,c) \in A \times A | (a,b) \in R \ \wedge (b,c) \in R\}$

9.12 연결 관계Connectivity Relation R^*

$R^* = \bigcup_{n=1}^{\infty} R^n = R^1 \cup R^2 \cup \cdots \cup R^n$
연결 관계 R^*는 R의 추이 폐포

[정리] R이 n개의 원소를 갖는 집합에 대한 관계이고, M_R을 관계 R에 대한 부울 행렬이라고 했을 때, R의 추이 폐포 R^*는

$$M_{R^*} = M_R \vee M_{R^2} \vee M_{R^3} \vee \cdots \vee M_{R^n}$$

9.13 예제 풀이

양의 정수Positive Integer 집합에서 두 원소 a, b에 대해서 'a가 b를 나눈다' 라는 관계는 어떤 성질을 만족하는가? (관계(1) 강의 참조)

10 Week 10

10.1 동치 관계Equivalence Relation

동치 관계: 반사 관계, 대칭 관계, 추이 관계가 모두 성립하는 경우

10.2 동치류Equivalence Class $[a]$

A에 대한 관계 R이 동치 관계일 때, R에 대한 a의 동치류: a와 순서쌍을 이루는 원소들의 집합
$[a] = \{x|(a,x) \in R\}$

10.3 동치 관계와 분할Partitions

분할: 공집합이 아닌 집합 A의 분할 조건 (A_i는 A의 부분 집합)
- $A_i \neq \varnothing, 1 \leq i \leq n$
- $A = \bigcup_{i=1}^{n} A_i$
- $A_i \cap A_j = \varnothing, i \neq j$

동치류와 분할: A에 대한 관계 R이 동치 관계일 때, 동치류 집합 $S = \{A_1, A_2, \cdots, A_k\}$는 다음 만족함
- $i = 1, 2, \cdots, k$일 때, $A_i \neq \varnothing$
- $S = A_1 \cup A_2 \cup \cdots \cup A_k$
- $i \neq j$면, $A_i \cap A_j = \varnothing$

10.4 부분 순서 관계

순서 관계: 집합 원소들 간 순서를 나타내기 위한 순서 관계
  e.g.)
- 프로젝트에서 수행해야 할 작업들의 집합에서 작업 x, y에 대해 x가 y보다 먼저 수행되어야 한다면 (x, y)의 순서쌍으로 구성되는 순서 관계
- 수학의 대소 관계인 $<$의 순서 관계, x가 y보다 작다 (x, y)

부분 순서 관계Partially Ordered Relation: 집합 A에 대한 관계 R이 반사, 반대칭, 추이 관계일 때 관계 R
부분 순서 집합Partially Order Set, Poset: R이 A에 대한 부분 순서 관계이면 순서쌍 (A, R)

부분Partial 이라는 용어를 쓰는 이유: 집합 A의 원소의 모든 쌍이 관계를 가지는 것은 아니기 때문

부분 순서 집합 기호: $(A, \lesssim)$은 A의 부분 순서 관계

집합 A에 대한 관계 R이 부분 순서 관계
- A의 두 원소 x, y에 대하여 $(x, y) \in R$을 $x \lesssim y$로 표기
- 'x가 y를 선행한다' (x precedes y)라고 읽음

비교 가능[Comparable]: 부분 순서 집합 $(A, \lesssim)$에서 $x, y \in A$가 $x \lesssim y$ 또는 $y \lesssim x$이면 x와 y는 비교 가능

e.g.) 자연수들의 집합 $\mathbb{N}$상의 관계 '⋯은 ⋯의 배수이다'라고 할 때, 자연수 3과 9는 비교 가능한가? 또, 7과 4는 비교 가능한가?

- $3 \lesssim 9$이므로 3과 9는 비교 가능하다.
- $4 \not\lesssim 7$이므로 4와 7은 비교 가능하지 않다.

선형 순서[Linear Order]:

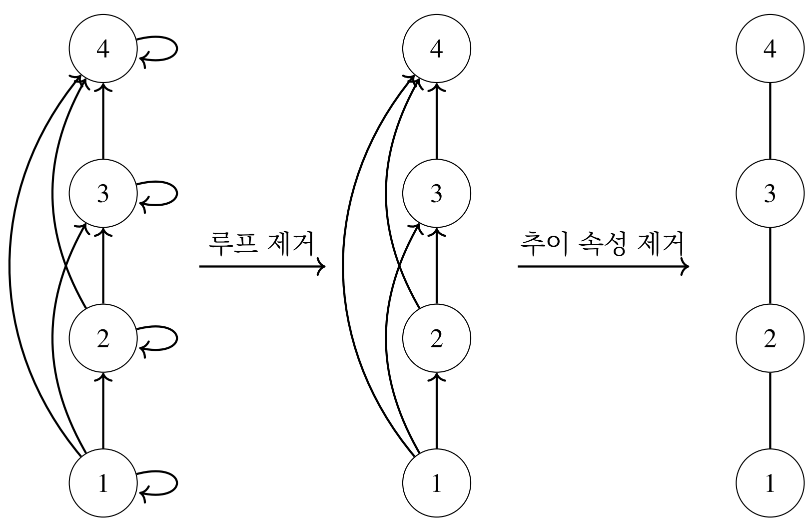
- R이 부분 순서를 만족
- 만약 $a \in A, b \in A$라면 ${}_aR_b, {}_bR_a$ 또는 $a = b$중 하나가 성립

선형 순서 집합[Linearly Ordered Set]: 집합 A의 모든 두 원소가 비교 가능하면 A는 선형 순서 집합

10.5 하세 도표[Hasse Diagram]

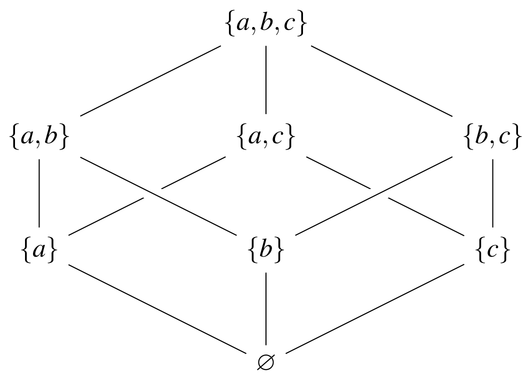
- 부분 순서 집합 $(A, \lesssim)$을 그림으로 표현
- 독일의 수학자 하세가 고안
- 방향 그래프의 일종으로서 화살표는 표시하지 않고 모든 연결선[Edge]을 트리[Tree]와 같이 모두 아래 방향을 향하도록 그림
- 부분 순서 관계에 대한 방향 그래프에서 루프는 생략
- 부분 순서 집합 A의 원소 a, b에 대해 $a \neq b$고, $a \preccurlyeq b$면, 정점 a를 정점 b보다 아래쪽에 그림
- $a \neq b$고 $a \preccurlyeq b$일 때, $a \preccurlyeq k \preccurlyeq b$고 $a \neq k$면서 $k \neq b$인 k가 집합 A에 존재하지 않으면 a에서 b로 가는 선을 그림

e.g.)



집합 $\{a,b,c\}$의 멱집합$^{\text{Power Set}}$ A에 대한 부분 집합 관계의 부분 순서 집합 $(A,\subseteq)$에 대한 하세 도표를 그려라

$$A=\{\varnothing,\{a\},\{b\},\{c\},\{a,b\},\{a,c\},\{b,c\},\{a,b,c\}\}$$



10.6 극대 원소와 극소 원소

부분 순서 집합 $(A,\lesssim)$가 있을 때

- 극대 원소$^{\text{Maximal Element}}$: A의 원소 a에 대하여 $a\lesssim b$인 원소 b가 A에 존재하지 않을 때 원소 a, 위의 하세 도표에서 $\{a,b,c\}$에 해당
- 극소 원소$^{\text{Minimal Element}}$: $b\lesssim a$인 원소 b가 A에 존재하지 않을 때 원소 a, 위의 하세 도표에서 $\varnothing$에 해당

10.7 최대/최소 원소

최대 원소$^{\text{Greatest Element}}$: 어떤 $a\in X$가 X의 모든 원소 x에 대해 $x\lesssim a$이면, a를 X의 최대 원소
최소 원소$^{\text{Least Element}}$: 어떤 $a\in X$가 X의 모든 원소 x에 대해 $a\lesssim x$이면, a를 X의 최소 원소

10.8 함수[Function]: $f: A \to B$

집합 A, B에 대해 집합 A에서 B로 가는 관계가 성립할 때, $a \in A$에 대해 $b \in B$ 하나가 대응되는 관계

$$a \in A, b \in B, (a,b) \in f \text{일 때, } f(a) = b$$

대응[Correspondence]: 집합 A, B가 있을 때, $a \in A$에 $b \in B$가 확정되는 경우 'b는 a에 대응'

$f: A \to B$에서:
- 정의역[Domain]: $\mathrm{dom}(f) = A$
- 공변역[Codomain]: $\mathrm{codom}(f) = B$
- 상[Image]: 함수값, $f(a) = b$
- 치역[Range]: 상의 집합, $\mathrm{ran}(f) = \{f(a) | a \in A\}$

10.9 함수 vs 관계

- 관계
  - B의 원소와 대응하지 않는 A의 원소 존재 가능
  - 하나의 A의 원소가 하나 이상의 B의 원소와 대응 가능
- 함수
  - A의 모든 원소는 B의 원소와 무조건 대응
  - 하나의 A의 원소가 하나의 B의 원소와 대응
  - 하나의 B의 원소가 하나 이상의 A의 원소와 대응 가능

10.10 단사 함수[Injective Function]

- 정의역 A의 모든 원소들이 공변역 B의 서로 다른 원소와 대응
- 일대일 함수[One-to-one Function]
- $a_i, a_j \in A$에 대하여 $a_i \neq a_j$이면 $f(a_i) \neq f(a_j)$ 성립
- 단사 함수에서 함수의 치역은 공변역의 부분 집합
- $f: A \to B$에서 $\mathrm{ran}(f) \subseteq B$

단조 증가 함수[Strictly Increasing Function]: $x, y \in A, x < y \to f(x) < f(y)$

단조 감소 함수[Strictly Decresaing Function]: $x, y \in A, x < y \to f(x) > f(y)$

c.f.) 단조 증가 함수와 단조 감소 함수는 단사 함수

특성: B의 모든 원소가 A의 원소와 반드시 대응하는 것은 아니므로, $|A| \leq |B|$ 성립

10.11 전사 함수 Surjective Function

- 공변역 B의 모든 원소가 정의역에 대응
- 치역 = 공변역, $\mathrm{ran}(f) = B$
- 모든 함수의 관계가 B의 모든 원소에 반영되므로 반영 함수 Onto Function

특성: B의 모든 원소가 A의 원소와 대응되어야 하므로 $|A| \geq |B|$ 성립

10.12 전단사 함수 Bijective Function

- 집합 A의 모든 원소들이 집합 B의 모든 원소와 하나씩 대응
- 일대일 대응 함수 One-to-one Correspondence Function

특성: A의 모든 원소가 B의 모든 원소와 하나씩 일대일 대응되므로 $|A| = |B|$

10.13 합성 함수 Composition Function

$\forall a \in A, (g \circ f)(a) = g(f(a))$
결합 법칙 성립: $h \circ (g \circ f) = (h \circ g) \circ f$
교환 법칙 성립 X
특징:

- $f : A \to B,\ g : B \to C$에 대해 $g \circ f$가 합성함수일 때
- f와 g가 단사 함수면, $g \circ f$도 단사 함수다
- f와 g가 전사 함수면, $g \circ f$도 전사 함수다
- f와 g가 전단사 함수면, $g \circ f$도 전단사 함수다
- $g \circ f$가 단사 함수면, f도 단사함수다
- $g \circ f$가 전사 함수면, g도 단사함수다
- $g \circ f$가 전단사 함수면, f는 단사함수고 g는 단사함수다

10.14 항등 함수 Identity Function

$f(a) = a$
전단사 함수
함수 $f : A \to B$고, 집합 A에 대한 항등 함수가 I_A, 집합 B에 대한 항등 함수가 I_B일 때, $f \circ I_A = I_B \circ f = f$

10.15 역함수 Inverse Function

$\forall a \in A, \forall b \in B, f(a) = b \Rightarrow f^{-1}(b) = a$
전단사 함수일 때만 역함수 존재

10.16 항등 함수와 역함수

전단사 함수 $f : A \to B,\ g : B \to C$에 대해, 다음이 성립

- $f^{-1} \circ f = I_A$
- $f \circ f^{-1} = I_B$
- $(g \circ f)^{-1} = f^{-1} \circ g^{-1}$

10.17 상수 함수Constant Function

$\forall a \in A, \exists b \in B, f(a) = b$

10.18 특성 함수Characteristic Function

$$f_A(x) = \begin{cases} 0, & x \notin A \\ 1, & x \in A \end{cases}$$

10.19 올림 함수Ceiling Function, 내림 함수Floor Function

올림 함수, 천정 함수 / 최소 정수 함수Least Integer Function: $\lceil x \rceil = n \Leftrightarrow n-1 < x \leq n$
내림 함수, 바닥 함수 / 최대 정수 함수Greatest Integer Function: $\lfloor x \rfloor = n \Leftrightarrow n \leq x < n+1$

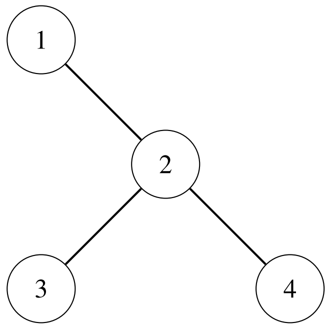
11 Week 11

11.1 그래프^Graph^

공집합이 아닌 정점^Vertex or Node^의 집합 V와 서로 다른 정점의 쌍 (v_i, v_j)를 연결하는 변 또는 연결선^Edge^의 집합 E로 구성되는 구조 G

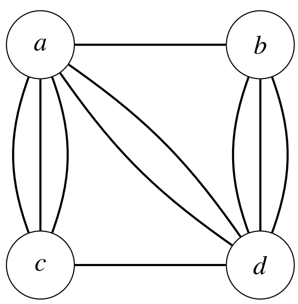
$G = (V, E)$
$V = \{v_1, v_2, \cdots, v_n\}$
$E = \{e_1, e_2, \cdots, e_m\} = \{(v_i, v_j), \cdots\}$

e.g.)



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

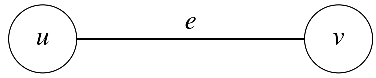
무방향 그래프^Undirected Graph^: 특별한 언급 없으면 무방향 그래프
방향 그래프^Directed Graph, Digraph^: 선행자 $\rightarrow$ 후속자
단순 그래프^Simple Graph^: 루프가 없는 그래프
멀티 그래프^Multigraph^:



연결 그래프^Connected Graph^: 모든 Vertex가 연결된 그래프, 모든 Vertex간 경로 존재
강한 연결 그래프^Strongly Connected Graph^: 방향 그래프에서만, 모든 두 Vertex v_1, v_2에 대해 $v_1 \leftrightarrow v_2$
연결 요소^Connectivity Component^: 그래프에서 모든 Vertex들이 연결되어 있는 부분 그래프
연결 수^Connectivity Number^: G에서 연결 요소 개수

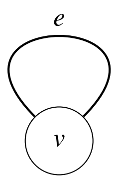
11.2 그래프 용어

인접[Adjacent] 과 근접[Incident]:



u, v는 서로 Adjacent, e는 u, v에 Incident

루프[Loop]:



근접하는 점이 같은 e

경로[Path]: Vertex들의 열[Sequence] $v_1, v_2, \cdots, v_n$에서 $(v_{k-1}, v_k) \in E, 1 \le k < n$, 경로의 길이는 $k-1$
  단순 경로[Simple Path]: 같은 Edge를 두 번 포함하지 않는 경로
  기본 경로[Elementary Path]: 같은 Node를 두 번 포함하지 않는 경로

사이클[Cycle] 또는 순회[Circuit]: $v_1 = v_k (k \neq 1)$인 경로, 종점 == 시점
  단순 사이클[Simple Cycle]: 같은 Edge를 반복해 방문하지 않는 사이클
  기본 사이클[Elementary Cycle]: 시점 제외 어떤 Node도 반복해 방문하지 않는 사이클

길이[Length]: 경로 또는 사이클을 구성하는 Edge의 수

차수[Degree] $d(v)$: Vertex v에 근접하는 Edge의 수, Loop는 두 개로 Count
  홀수점[Odd Vertex]: 차수가 홀수인 Vertex
  짝수점[Even Vertex]: 차수가 짝수인 Vertex

외차수[Out-degree] out$-d(v)$: 방향 그래프에서 Vertex v에서 시작하는 화살표 수

내차수[In-degree] in$-d(v)$: 방향 그래프에서 Vertex v에서 끝나는 화살표 수

[정리] 차수에 대한 정리
- $G = (V, E)$에서, 모든 Vertex의 차수의 합은 Edge의 수의 두 배

$$\sum_{v \in V} d(v) = 2|E|$$

- $G = (V, E)$에서, 차수가 홀수인 정점의 수는 짝수

11.3 오일러 경로Eulerian Path 및 오일러 회로Eulerian Circuit

오일러 경로: 멀티 그래프에서 모든 Edge들을 한 번씩만 통과하는 경로를 찾는 문제
오일러 회로: Node는 여러 번 통과할 수 있지만, Edge는 한 번씩만 통과하는 사이클

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

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

11.4 해밀턴 경로Hamiltonian Path 및 해밀턴 회로Hamiltonian Circuit

해밀턴 경로: 그래프에서 모든 Node를 오직 한 번씩만 지나지만 시점으로 돌아오지 않는 경로
해밀턴 회로: 그래프에서 모든 Node를 오직 한 번씩만 지나는 순회
해밀턴 회로에 대한 충분 조건:

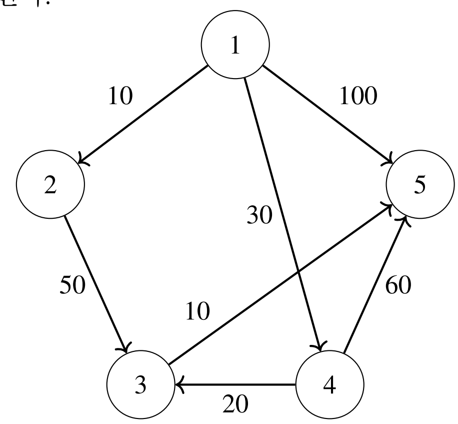
- 차수 1을 갖는 Node를 가진 그래프는 해밀턴 순환을 가질 수 없다.
- 차수가 2인 Node에 근접하는 두 Vertex는 해밀턴 순환에 포함된다.
- 한 Node에 근접하는 두 Vertex가 해밀턴 순환에 포함되면, 그 Node에 근접한 다른 Vertex는 해밀턴 순환에 포함될 수 없다.

Ore's Theorem: $n \geq 3$일 때, n개의 Node를 갖는 단순 연결 그래프 G에서 인접하지 않은 임의의 정점 u, v에 대해 $d(u) + d(v) \geq n$이면 G는 해밀턴 그래프이다.

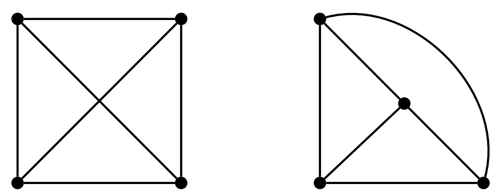
Dirac's Theorem: $n \geq 3$일 때, n개의 Node를 갖는 단순 연결 그래프 G에서 임의의 정점 v에 대해 $2d(v) \geq n$이면 G는 해밀턴 그래프이다.

11.5 특수 형태의 그래프

가중 그래프^Weight Graph^: G의 각 Edge에 0보다 큰 수가 할당되었을 때, 이 값을 가중값^Weight^이라고 하며, 이를 가중 그래프라고 한다.



동형 그래프^Isomorphic Graph^: $G_1 = (V_1, E_1)$과 $G_2 = (V_2, E_2)$가 주어졌을 때, 전단사 함수 $f : V_1 \to V_2$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 그래프라 한다.

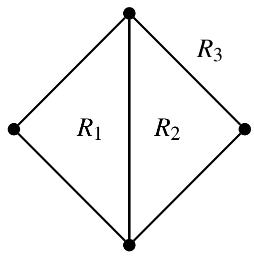


11.6 오일러의 정리

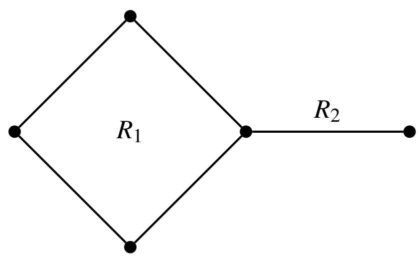
연결된 평면 그래프 G에서 Vertex의 수를 v, Edge의 수를 e, 면^Space^의 수를 s라고 할 때, $v - e + s = 2$

11.7 평면 그래프

평면 그래프[Planar Graph]: $G = (V, E)$를 평면에 그릴 때, 교차하지 않는 그래프
면 f의 차수 $d(f)$: 평면 그래프의 면 f의 경계를 이루는 변의 수
e.g.)



$$d(R_1) = 3,\ d(R_2) = 3,\ d(R_3) = 4$$
$$d(R_1) + d(R_2) + d(R_3) = 2e$$



$$d(R_1) = 4,\ d(R_2) = 6$$
$$d(R_1) + d(R_2) = 2e$$

평면 그래프의 면의 차수의 총 합은 변의 수의 두 배이다.

$$2e = \sum_{f=1}^{n} d(f)$$

연결된 평면 단순 그래프의 Vertex의 수를 v, Edge의 수를 e라 할 때, $v \geq 3$이면 $e \leq 3(v-2)$
모든 면의 차수는 3이상이므로, $2e = \sum_{f=1}^{n} d(f) \geq 3f,\ 2 = v - e + f \leq c - e + \frac{2}{3}e \leq v - \frac{1}{3}e$

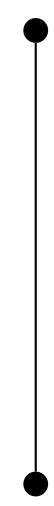
12 Week 12

12.1 특수 형태의 그래프

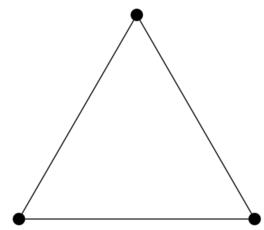
완전 그래프$^{\text{Complete Graph}}$: 모든 n개의 Vertex들의 쌍 사이에 Edge가 존재하는 $G = K_n$

e.g.)

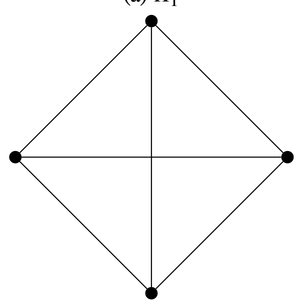
(a) K_1



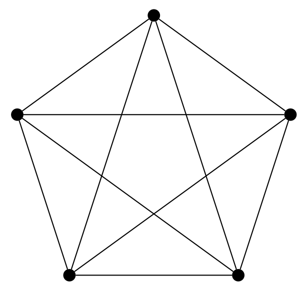
(b) K_2



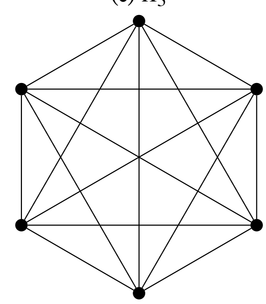
(c) K_3



(d) K_4



(e) K_5

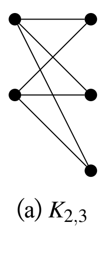


(f) K_6

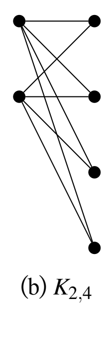
이분 그래프$^{\text{Bipartite Graph}}$: V가 X와 $Y = V - X$로 나누어져, 각 Edge가 X 내의 Vertex와 Y 내의 Vertex의 쌍으로 연결될 때, $G = (V, E)$

완전 이분 그래프$^{\text{Complete Bipartite Graph}}$: X 내의 모든 Vertex와 Y 내의 모든 Vertex 사이에 Edge가 존재할 때, 그래프 G

e.g.)



(a) $K_{2,3}$



(b) $K_{2,4}$

12.2 그래프의 표현 방법

인접 행렬Adjacency Matrix: $G = (V,E)$에서, $|V| = n$일 때, G의 인접 행렬은 $n \times n$ 행렬 A

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

인접 리스트Adjacency List: 각 Vertex에 대해 포인터가 주어지고, 그 Vertex로 부터 인접한 Vertex들을 모두 Linked List에 담음 (List 내에서는 순서에 관계가 없음)

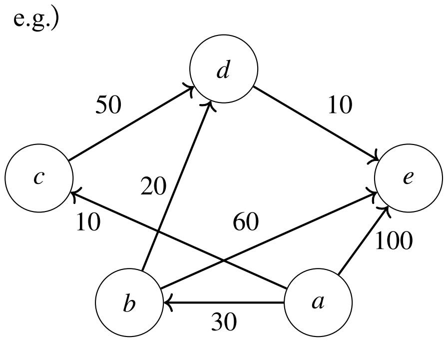
Linked List는 Node의 연결로 표현

- Head: Linked List의 시작 Node (그래프의 각 Vertex들)
- Node: 데이터 필드 + 포인트 필드 (다음에 연결된 Node의 주소 저장)
- 마지막 Node의 포인트 필드는 `null`

12.3 그래프의 응용 및 활용

최단 경로 찾기Shortest Path Problem: 다익스트라Dijkstra 알고리즘

e.g.)



$$\begin{array}{c} \\ a \\ b \\ c \\ d \\ e \end{array} \begin{array}{c} \begin{array}{ccccc} a & b & c & d & e \end{array} \\ \begin{bmatrix} 0 & 30 & 10 & \infty & 100 \\ \infty & 0 & \infty & 20 & 60 \\ \infty & \infty & 0 & 50 & \infty \\ \infty & \infty & \infty & 0 & 10 \\ \infty & \infty & \infty & \infty & 0 \end{bmatrix} \end{array}$$

위와 같은 그래프가 주어졌을 때,

S		`D[a]`	`D[b]`	`D[c]`	`D[d]`	`D[e]`
$\{a\}$	a	0	30	10	∞	100
$\{a,c\}$	c		30	10	60	100
$\{a,c,b\}$	b		30		50	90
$\{a,c,b,d\}$	d				50	60

해밀턴 순회의 응용: 일반적인 해결 알고리즘 존재 X → 최근접 이웃 방법Nearest Neighbour Method(Greedy 알고리즘)

12.4 그래프의 탐색(Traversal)

깊이 우선 탐색^Depth First Search; DFS^: Stack or 재귀로 구현
- 시작 Vertex v에서 인접 Vertex 중 방문하지 않은 Vertex w 방문
- w에서 다시 인접 Vertex 중 방문하지 않은 Vertex u 방문 반복
- 어떤 Vertex $\boldsymbol{v}$ 방문 후 $\boldsymbol{v}$에 인접한 모든 Vertex 방문한 경우, 바로 이전 Vertex로 돌아가 위 반복
- 모든 Vertex 방문 후 탐색 종료

너비 우선 탐색^Breadth First Search; BFS^: Queue 사용
- 시작 Vertex v에서 인접한 Vertex 모두 차례로 방문
- 더 이상 방문할 Vertex가 없을 때, 다시 v에 인접한 Vertex중 처음 방문한 Vertex와 인접한 Vertex 방문
- v에 인접한 Vertex중 두 번째 방문한 Vertex와 인접한 Vertex 방문 반복
- 모든 Vertex 방문 후 탐색 종료

12.5 트리

트리(Tree):
- A connected undirected graph with no circuits
- An undirected graph is a tree iff there is a unique simple path between any two of its vertices
- 특별히 지정된 노드인 루트가 있고, 나머지 노드들은 다시 각각 트리이면서 연결되지 않는(disjoint) $T_1, T_2, \cdots, T_n (n \geq 0, T_n$은 루트의 서브 트리^Subtree^) 으로 나누어 진다.

루트^Root^: 주어진 트리의 시작 노드, 트리의 가장 높은 곳에 위치
차수^Degree^: 각 노드의 서브 트리의 개수
잎 노드 or 단말 노드^Leaf Node^: 차수가 0인 노드
자식 노드^Children Node^: 어떤 노드의 서브 트리의 루트 노드
부모 노드^Parent Node^: 자식 노드의 반대
형제 노드^Sibling Node^: 동일한 부모를 가지는 노드
중간 노드^Internal Node^: 자식 노드를 갖는 노드
조상^Ancestor^: 루트로부터 각 노드에 이르는 경로 상에 나타난 모든 노드들
자손^Descendant^: 각 노드부터 잎 노드에 이르는 경로 상에 나타난 모든 노드들
레벨^Level^: 루트의 레벨 $= 0$, 자손 노드로 내려가며 ++
트리의 높이 or 깊이^Height or Depth^: 트리에서 노드가 가질 수 있는 맥스 레벨
숲^Forest^: 서로 연결되지 않는 트리들의 집합, 트리에서 루트를 제거 $\rightarrow$ 숲 생성

G는 트리
$\equiv G$는 연결되어 있고, $M = n - 1$
$\equiv G$는 연결되어 있고, 어느 한 연결선만을 제거하더라도 G는 연결되지 않음
$\equiv G$는 사이클을 가지지 않고, $m = n - 1$
$\equiv G$는 어느 한 연결선만 첨가하더라도 사이클 형성

12.6 이진 트리

이진 트리$^{\text{Binary Tree}}$: 노드들의 유한 집합, 공집합이거나, 루트와 왼쪽 서브 트리, 오른쪽 서브 트리로 이루어짐
사향 이진 트리$^{\text{Skewed Binary Tree}}$: 왼쪽 or 오른쪽으로 편향된 트리
완전 이진 트리$^{\text{Complete Binary Tree}}$: 높이가 k일 때 레벨 1부터 $k-1$까지는 모두 차있고 레벨 k에서는 왼쪽 노드부터 차례로 차있는 이진 트리
포화 이진 트리$^{\text{Full Binary Tree}}$: 잎 노드가 아닌 것들은 모두 2개씩 자식 노드를 가지며 트리의 높이가 일정할 때

완전 n-ary 트리: 레벨이 k일 때, 레벨 1부터 $k-1$까지는 모두 n개의 자식 노드를 가지고, 레벨 k에서는 왼쪽 노드부터 차례로 차있는 트리
포화 n-ary 트리: 모든 중간 노드가 n개의 자식 노드를 가지는 트리
  포화 n-ary 트리에서 m개의 중간 노드를 가질 때, 전체 노드 개수 $= (m+1)*n+1$(루트 노드)

이진 트리가 레벨 i에서 가질 수 있는 최대 노드 수 $= 2^i$
높이가 k인 이진 트리가 가질 수 있는 최대 전체 노드 수 $= 2^{k+1}-1$
잎 노드 개수 $= n_0$, 차수가 2인 노드 개수 $= n_2$ 일 때, $n_0 = n_2 + 1$ 항상 성립

12.7 이진 트리의 표현

배열:

- 트리의 중간에 새로운 노드를 삽입하거나 기존의 노드를 지울 때 비효율적
- 높이가 h인 이진 트리는 각 노드 번호를 인덱스로 하여 1차원 배열로 구현 가능 (인덱스는 1부터 시작)
- 노드 인덱스 n의 부모 인덱스 $= \left\lfloor \frac{n}{2} \right\rfloor$
- 노드 인덱스 n의 왼쪽 자식 인덱스 $= 2n$, 오른쪽 자식 인덱스 $= 2n+1$

연결 리스트: 일반적으로 가장 많이 사용. 중간 데이터, 왼쪽 자식 포인터, 오른쪽 자식 포인터 저장

13 Week 13

13.1 이진 트리의 탐방

트리의 각 노드를 꼭 한 번씩만 방문Traversal 하는 방법

- 각 노드와 그 노드의 서브 트리를 같은 방법으로 탐방
- 전순위, 중순위, 후순위 탐방 기법
- D: 노드, L: 노드의 왼쪽 서브 트리, R:노드의 오른쪽 서브 트리
- 왼쪽을 오른쪽보다 항상 먼저 방문한다고 가정
- 중순위: LDR
- 전순위: DLR
- 후순위: LRD
- 수식 표현에서 중순위 표기Infix, 전순위 표기Prefix, 후순위 표기Postfix 와 각각 대응

탐방의 결과, 각 노드에 들어있는 데이터를 차례로 나열

중순위:

```cpp
void inOrder(TREE* currentNode) {
    if (currentNode != NULL) {
        inOrder(currentNode->leftChild);
        std::cout << currentNode->data;
        inOrder(currentNode->rightChild);
    }
}
```

전순위:

```cpp
void preOrder(TREE* currentNode) {
    if (currentNode != NULL) {
        std::cout << currentNode->data;
        preOrder(currentNode->leftChild);
        preOrder(currentNode->rightChild);
    }
}
```

후순위:

```cpp
void postOrder(TREE* currentNode) {
    if (currentNode != NULL) {
        postOrder(currentNode->leftChild);
        postOrder(currentNode->rightChild);
        std::cout << currentNode->data;
    }
}
```

13.2 순회 표기 & 수식 트리

중순위: $(a+b)\times(c-d)$
전순위: $\times + ab - cd$
후순위: $ab + cd - \times$

전순위 수식 $+ - \times\ 2\ 3\ 5 \div \wedge\ 2\ 3\ 4$ ⟹ 중순위 수식 $2\times 3 - 5 + 2^3 \div 4$
후순위 수식 $7\ 2\ 3 \times - 4 \wedge 9\ 3 \div +$ ⟹ 중순위 수식 $(7-2\times 3)^4 + 9 \div 3$

후순위 표기식과 스택을 활용하여 수식 트리 생성:
 후순위 표기식이 주어지면 스택에 피연산자 저장
 연산자를 만나면 스택에서 두 개의 피연산자 `pop()` 후 연산 결과(트리) 다시 저장

13.3 생성 트리와 최소 비용 생성 트리

생성 트리[Spanning Tree]: $\exists G$에서 모든 노드들을 포함하는 트리
 비용[Cost]: 트리 연결선의 값의 합
최소 비용 생성 트리(Minimum Spanning Tree: MST): 생성 트리 중 최소 비용

Prim's Algorithm:

- $G=(V,E)$에서 $V=\{1,2,\cdots,n\}$
- 노드의 집합 U를 1로 시작
- $u\in U,\ v\in V-U$일 때, U와 $V-U$를 연결하는 사이클 형성 X인 가장 짧은 연결선 (u,v)를 찾아 v를 U에 포함시킴
- 위를 $U-V$까지 반복

```
void prim(graph G: set_of_edges T) {
    set_of_vertices U;
    vertex u, v;
    T = NULL;
    U = {1};
    while (U != V) {
        let (u, v) be a lowest cost edge such that u is in U and v is in
↪   V - U;
        if ((u, v) does not create a cycle) {
            T = T ∪ {(u, v)};
            U = U ∪ {v};
        }
    }
}
```

Kruskal Algorithm:

- $G = (V, E)$ 에서 $V = \{1, 2, \cdots, n\}$, $T =$ (연결선의 집합)
- Let $T = \varnothing$
- E를 비용이 적은 순서로 정렬
- 가장 최솟값 가진 연결선 (u, v) 차례로 찾아 사이클 형성 X이면 T에 포함
- 위를 $|T| = |V| - 1$까지 반복

```
void kruskal(graph G: set_of_edges T) {
    T = NULL;
    while (T contains less than n - 1 edges and E is not empty) {
        choose an edge (v, w) from E of lowest cost;
        delete (v, w) from E;
        if ((v, w) does not create a cycle in T)
            add (v, w) to T;
        else discard (v, w);
    }
    if (T contains fewer than n - 1 edges)
        std::cout << "No Spanning Tree";
}
```

13.4 트리의 활용

문법의 파싱$^{\text{Parsing}}$

허프만 코드$^{\text{Huffman Code}}$:

- 알파벳 문자를 0과 1의 비트 코드로 Encoding
- 문자의 발생 빈도에 따라 코드의 길이를 다르게 → 통신의 효율성
- 접두어 성질: 어떤 문자 코드도 다른 문자 코드의 접두어 코드가 아님

Huffman Algorithm:

- 발생 빈도가 가장 낮은 두 문자를 선택해 하나의 이진 트리로 연결
  - 왼쪽 노드에는 빈도수 낮은 문자, 오른쪽 노드에는 빈도수 높은 문자
  - 그 두 문자의 루트 노드는 두 문자의 빈도의 합
  - 문자들을 이진 트리로 연결
  - 그 후에 이진 트리들을 연결
- 위 과정을 모든 문자가 하나의 이진 트리로 묶일 때까지 반복
- 생성된 이진 트리의 왼쪽 노드는 0, 오른쪽 노드에는 1 부여
- 루트부터 해당 문자까지 0 또는 1을 순서대로 나열한 것이 해당 문자의 허프만 코드

14 Week 14

14.1 이진 탐색 트리

이진 탐색 트리Binary Search Tree:

- 모든 노드 x에 대하여 노드 y가 노드 x의 왼쪽 서브 트리에 있을 때 $y < x$이고, 노드 z가 노드 x의 오른쪽 서브 트리에 있으면 $x < z$인 이진 트리
- $<$는 순서 관계를 의미
- 높이 균형 이진 트리인 경우 탐색 시간은 $\log n$

14.2 결정 트리

결정 트리Decision Tree: 8개의 동전 문제

14.3 부울식

0과 1의 조합으로 연산
$A = \{0, 1\}$에 대해 이항 연산자$^{\text{Binary Operator}}$ $+^{\text{OR}}$과 $\cdot^{\text{AND}}$ 및 단항 연산자$^{\text{Unary Operator}}$ $'^{\text{Complement}}$로 표현되는 식
연산 우선 순위: $' > \cdot > +$

두 부울식이 같은 진리표$^{\text{Truth Table}}$을 가질 때 동치$^{\text{Equivalent}}$, 연산자는 $=$

Table 1: 부울식의 법칙

이름	법칙
멱등 법칙$^{\text{Idempotent Law}}$	$p \cdot p = p$ $p + p = p$
항등 법칙$^{\text{Identity Law}}$	$p + 0 = p$ $p \cdot 1 = p$
교환 법칙$^{\text{Commutative Law}}$	$p + q = q + p$ $p \cdot q = q \cdot p$
결합 법칙$^{\text{Associative Law}}$	$p + (q + r) = (p + q) + r$ $p \cdot (q \cdot r) = (p \cdot q) \cdot r$
분배 법칙$^{\text{Distributive Law}}$	$p + (q \cdot r) = (p + q) \cdot (p + r)$ $p \cdot (q + r) = (p \cdot q) + (p \cdot r)$
흡수 법칙$^{\text{Absorption Law}}$	$p + (p \cdot q) = p$ $p \cdot (p + q) = p$
역 법칙$^{\text{Inverse Law}}$	$p + p'$ $p \cdot p' = 0$
보 법칙$^{\text{Complement Law}}$	$(p')' = p$
우등 법칙$^{\text{Dominance Law}}$	$p + 1 = 1$ $p \cdot 0 = 0$
드 모르간의 법칙$^{\text{De Morgan's law}}$	$(p + q)' = p' \cdot q'$ $(p \cdot q)' = p' + q'$

14.4 부울 함수Boolean Function

부울 변수와 부울 연산자로 구성된 부울식
n개의 부울 변수 $x_1, x_2, \cdots, x_n$에 대한 부울 함수는 $f(x_1, x_2, \cdots, x_n)$으로 표현
n개의 부울 변수가 있을 때, 그 변수들로부터 얻을 수 있는 조합은 2^n개

리터럴Literal: 부울 함수를 구성하는 부울 변수 또는 그의 보수
최소항Minterm: n차 부울함수 $f(x_1, x_2, \cdots, x_n)$을 구성하는 논리곱 항들 중 n개의 리터럴 곱으로 구성된 항

14.5 최소항

부울 함수는 부울 변수에 대한 최소항들 중에서 1의 값을 가지는 최소항들의 부울 합을 식으로 표현하는 함수
e.g.)

x	y	z	$f(x,y,z)$
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

부울 함수는 $f(x,y,z) = 1$인 최소항들의 합이므로 $f(x,y,z) = x'y'z + xy'z' + xyz'$이다.

14.6 정규식Disjunctive Normal Form; DNF

최소항들의 부울합으로 표현된 부울 함수
최소항들은 부울 변수의 곱으로 표현
곱의 합(Sum of Products) 또는 논리합 표준형

정규식이 아닌 부울 함수를 정규식으로 표현하는 방법:
- 진리표를 이용한 정규식 변환
- 부울 법칙을 이용한 정규식 변환
    - 각 항에 포함되지 않은 부울 변수를 파악
    - 각 항에 포함되지 않은 부울 변수에 대해
    - 논리곱에 대한 항등 법칙 $x \cdot 1 = x$와
    - 논리합에 대한 보수 법칙 $x + x' = 1$을 적용
    - 각 항에 없는 부울 변수 추가
    - 분배 법칙 등을 이용해 식을 풀고, 중복되는 항은 멱등 법칙에 의해 제거

e.g.)

$$\begin{aligned}
f(x,y) &= x+y' \\
&= x\cdot 1 + y'\cdot 1 && (\because \text{항등 법칙}) \\
&= x(y+y') + y'(x+x') && (\because \text{보수 법칙}) \\
&= xy+xy'+y'x+y'x' && (\because \text{분배 법칙}) \\
&= xy+xy'+xy'+x'y' && (\because \text{교환 법칙}) \\
&= xy+xy'+x'y' && (\because \text{멱등 법칙})
\end{aligned}$$

$$\therefore f(x,y) = xy+xy'+x'y'$$

14.7 예제 풀이

1. $(x+y)(y+z)(z+x) = xy+yz+zx$

$$\begin{aligned}
(x+y)(y+z)(z+x) &= (xy+xz+yy+yz)(z+x) \\
&= (xy+xz+y+yz)(z+x) \\
&= (xy+xz+y(1+z))(z+x) \\
&= (xy+xz+y)(z+x) \\
&= xyz+xxy+xzz+xxz+yz+xy \\
&= xyz+xy+yz+zx \\
&= xy(z+1)+yz+zx \\
&= xy+yz+zx
\end{aligned}$$

2. $xy+yz+x'z = xy+x'z$

$$\begin{aligned}
xy+yz+x'z &= xy+1\cdot yz+x'z \\
&= xy+(x+x')yz+x'z \\
&= xy+xyz+x'yz+x'z \\
&= xy(1+z)+x'z(y+1) \\
&= xy+x'z
\end{aligned}$$

15 Week 15

15.1 부울 함수의 간소화

더 적은 변수와 연산자를 사용하여 같은 기능 또는 결과 도출
부울 함수를 간소화 하는 방법:

- 부울식의 기본 법칙 사용 → 과정이 복잡하고 간소화에 대한 확인이 쉽지 않음
- 카르노맵을 사용

카르노맵[Karnaugh Map]:

- 부울 함수가 가질 수 있는 모든 경우의 최소항들을 사각형 형태의 표에 배열한 후, 1의 값을 가지는 최소항들만 1로 표시
  - 1을 갖는 항들을 연결하여 최소화
- 사각형 내 배치 시, 인접한 최소항들은 변수 하나 차이만 있도록 배치
  - xy와 xy'은 인접 가능
  - xy와 $x'y'$은 인접 불가

두 변수에 대한 카르노맵: 변수를 x, y라 할 때, 모든 가능한 최소항 조합

- $x'y', x'y, xy', xy$
- x가 0이고, y가 0일 때, 부울식은 $x'y'$

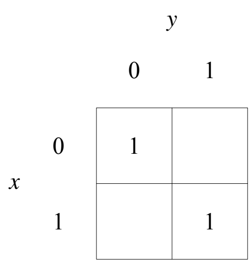
		y = 0	y = 1
x	0	$x'y'$	$x'y$
	1	xy'	xy

(a) 두 변수에 대한 카르노 맵

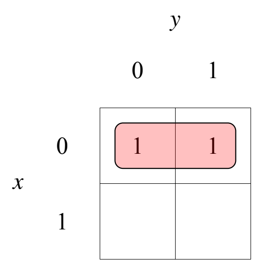
		y = 0	y = 1
x	0	1	1
	1	1	

(b) $f(x,y) = x'y' + x'y + xy'$의 카르노 맵

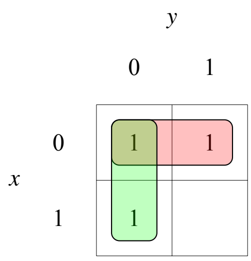
간소화: 1로 표시된 사각형이 인접할 경우, 함께 묶을 수 있는 경우
e.g.)



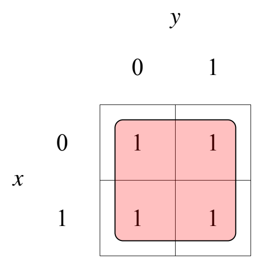
(a) 인접한 경우 없음: $x'y' + xy$



(b) y의 값에 관계없이 x의 값이 0: x' (x'일 때 1)



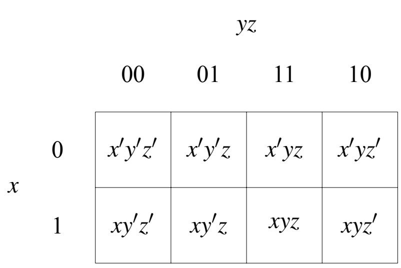
(c) (b)의 경우 $+\,x$의 값에 관계 없이 y': $x' + y'$



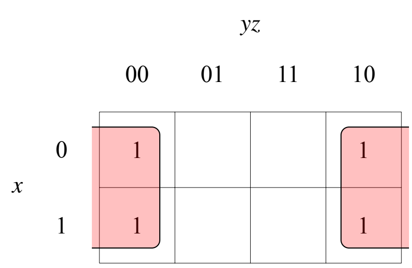
(d) x와 y의 값에 관계없이 항상 1: 1

세 변수에 대한 카르노 맵:

- yz에 대한 사각형에서 00, 01 다음에 11 (인접하는 행 또는 열에서 1비트 차이만 나야함)
- 왼쪽 끝은 오른쪽 끝과 인접하다 생각하기



(a) 세 변수에 대한 카르노 맵

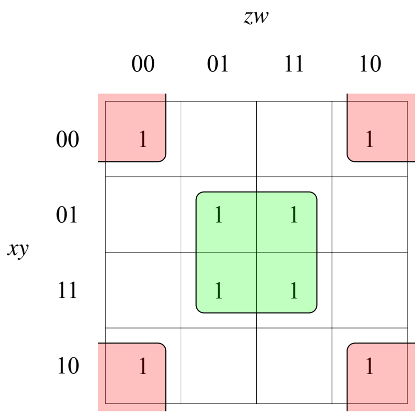


(b) $f(x,y,z) = x'y'z' + x'yz + xy'z' + xyz = z'$

네 변수에 대한 카르노 맵:
- 01 다음 11
- 왼쪽 끝과 오른쪽 끝 인접
- 위쪽 끝과 아래쪽 끝 인접

xy \ zw	00	01	11	10
00	$x'y'z'w'$	$x'y'z'w$	$x'y'zw$	$x'y'zw'$
01	$x'yz'w'$	$x'yz'w$	$x'yzw$	$x'yzw'$
11	$xyz'w'$	$xyz'w$	$xyzw$	$xyzw'$
10	$xy'z'w'$	$xy'z'w$	$xy'zw$	$xy'zw'$

(a) 네 변수에 대한 카르노 맵



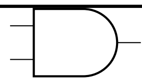
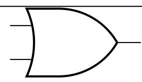
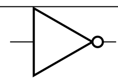
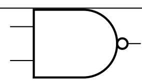
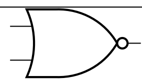
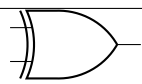
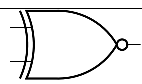
(b) $f(x,y,z,w) = x'y'z'w' + x'y'zw' + x'yz'w + x'yzw + xyz'w + xyzw + xy'z'w' + xy'zw' = yw + y'w'$

15.2 논리 회로$^{\text{Logic Circuit}}$

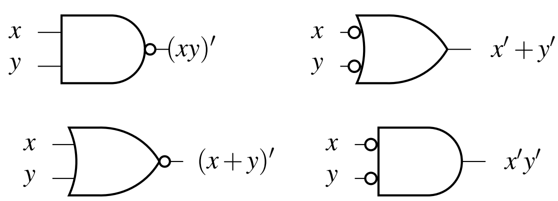
- 논리 회로의 입출력은 논리 게이트$^{\text{Gate}}$ 들을 상호 연결해 구성
- 입력은 부울 변수, 출력은 부울 함수, 부울 연산자는 게이트
- 1 = On, 0 = Off

회로를 설계하기 전에 먼저 부울식을 간소화하는 과정이 필요
논리 함수의 완전성$^{\text{Completeness}}$: 부울 연산자 AND, OR, NOT만으로도 모든 논리 함수들을 나타낼 수 있다.

Table 2: 여러 가지 논리 게이트

게이트	기호	수식
AND		$x = A \cdot B$
OR		$x = A + B$
NOT		$x = A'$
NAND		$x = (A \cdot B)'$
NOR		$x = (A + B)'$
XOR		$x = A \oplus B = A'B + AB'$
XNOR		$x = (A \oplus B)' = AB + A'B'$

NAND와 NOR의 다른 표현: 드 모르간의 법칙을 적용



15.3 가산기^Adder^

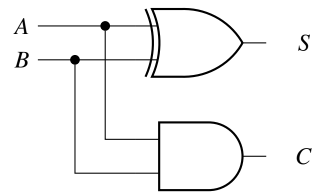
두 개 이상의 입력을 받아서 이들의 합을 출력
반가산기, 전가산기, 병렬 가산기

반가산기^Half Adder^:
- 입력: 1비트 정보 두 개
- 출력: 1비트 정보 두 개, 합^Sum^ + 자리 올림^Carry^

(a) 반가산기의 계산

A	B	올림수(C)	합(S)
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

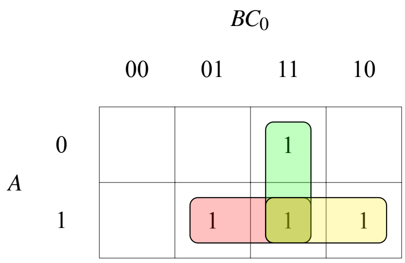
(b) 반가산기 논리 회로



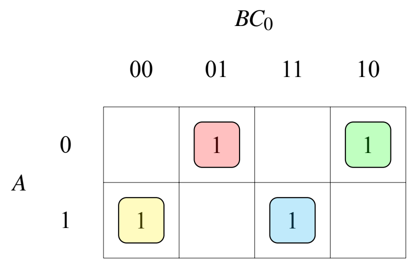
전가산기^Full Adder^: 입력 두 개와 하위 비트에서 발생한 자리 올림수 포함. 2진수 3개 덧셈

Fig. 8: 전가산기의 계산

A	B	C_0	올림수(C)	합(S)
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



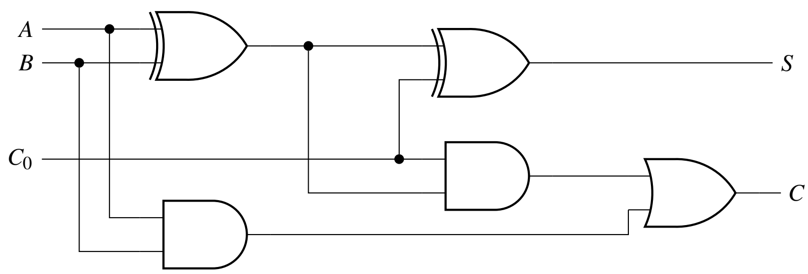
(a) Carry



(b) Sum

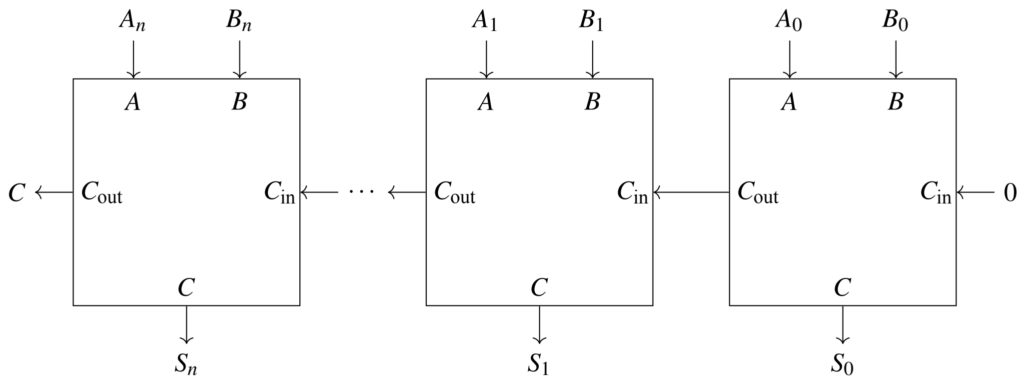
$$
\begin{aligned}
\text{Carry} &= AC_0 + AB + BC_0 \\
&= AB + (A'B + AB)C_0 \\
&= AB + (A \oplus B)C_0
\end{aligned}
$$

$$
\begin{aligned}
\text{Sum} &= AB'C_0' + A'B'C_0 + ABC_0 + A'BC_0' \\
&= A \oplus B \oplus C_0
\end{aligned}
$$



(c) 전가산기 논리회로

병렬 가산기:



15.4 예제 풀이

1. $f(x,y) = x'y + xy + x'y'$

x \ y	0	1
0	1	1
1		1

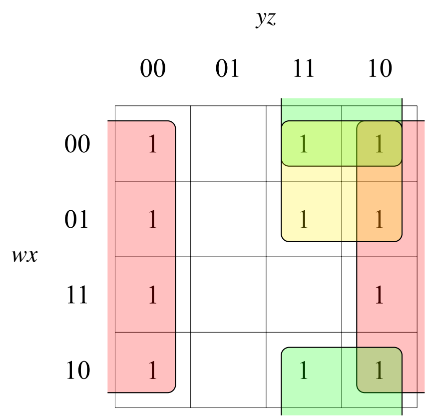
$$\therefore f(x,y) = x' + y$$

2. $f(x,y,z) = xyz, x'yz, xy'z + x'yz'$

x \ yz	00	01	11	10
0			1	1
1		1	1	

$$\therefore f(x,y,z) = x'y + xz$$

3. $f(w,x,y,z) = wxy'z' + wxyz' + wz'yz + wx'y'z' + wx'yz' + w'x'yz + w'x'y'z' + w'x'yz' + w'xyz + w'xy'z' + w'xyz'$



$$\therefore f(w,x,y,z) = y'w + x'y + z'$$