Ch 1 Logic-introduce

Ch 2 Set description

Ch 3 Complextiy introduction

Ch 4 Inductive Definitions/Functions/Proofs

Ch 7-4 Generating functions

Ch 8 relations

Ch 9 grap

PigeonHole

이산수학이란?

**이산수학**(Discrete mathematics, 離散數學)은 [이산](https://ko.wikipedia.org/wiki/%EC%9D%B4%EC%82%B0_%EA%B3%B5%EA%B0%84)적인 수학 구조에 대해 연구하는 학문으로, [연속](https://ko.wikipedia.org/wiki/%EC%97%B0%EC%86%8D)되지 않는 공간을 다룬다. **유한수학**이라고도 하며, [전산학](https://ko.wikipedia.org/wiki/%EC%A0%84%EC%82%B0%ED%95%99)적인 측면을 강조할 때는 **전산수학**이라고도 한다.

이산수학에서는 실수 같이 연속적인 성질이 있는 대상이 아니라 주로 [정수](https://ko.wikipedia.org/wiki/%EC%A0%95%EC%88%98), [그래프](https://ko.wikipedia.org/wiki/%EA%B7%B8%EB%9E%98%ED%94%84), 논리 연산 같이 서로 구분되는 값을 가지는 대상을 연구한다. 따라서 이산수학에서는 미분적분학이나 수치 해석같이 '연속적'인 분야에서 다루는 주제는 다루지 않는다. 이산적인 대상은 정수로 개수가 열거되는 경우가 많다. [공식적으로, 이산수학은 가산집합을](https://ko.wikipedia.org/wiki/%EA%B0%80%EC%82%B0%EC%A7%91%ED%95%A9) 다루는 수학의 한 부류로 특징지을 수 있다. 하지만 이산수학이라는 용어에 대해 정확한 정의는 내려져 있지 않다. 사실, 이산수학은 포함된 주제에 의해서 정의되기 보다는, 이산수학이 다루는 주제가 아닌 것들에 의해서 정의된다.

이산수학에서 연구하는 집합의 종류는 무한 혹은 유한집합이다. 이산 수학중에서도 유한 집합을 다루는 한 분야에 대해서 가끔씩 유한 수학이라는 용어가 쓰이기도 한다.

이산적인 과정을 통해서 데이터를 저장하고, 동작하는 디지털 컴퓨터의 개발으로 인해 20세기 후반에 이산수학에 대한 연구가 점점 활기를 띄기 시작했다. 이산수학에 포함된 개념과 기호들은 컴퓨터 [알고리즘](https://ko.wikipedia.org/wiki/%EC%95%8C%EA%B3%A0%EB%A6%AC%EC%A6%98), [프로그래밍 언어](https://ko.wikipedia.org/wiki/%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D_%EC%96%B8%EC%96%B4), 암호학, 자동 이론 증명, 소프트웨어 개발 등의 문제를 연구하는 데 유용하다. 반대로, 컴퓨터의 구현은 운영연구라고 불리는, 이산 수학의 개념들을 현실 세계에 적용하는 방법이 중요하다.

이산수학 개요 요약

1. 이산적 개념과 연속적 개념

|  |  |  |
| --- | --- | --- |
|  | 이산수학 | 연속수학 |
| 영역 | 정수 | 실수 |
| 연속성 | 분리됨 | 연속적임 |
| 집합 | 유한집합 | 유한+무한집합 |
| 컴퓨터 | 디지털 컴퓨터 | 아날로그 컴퓨터 |

1. 수학적 모델링  
   모델링에는 3가지 요소가 있다.  
   문제의 상황 및 배경, 명제와 수학적 구조의 매핑, 문제해결
2. 응용분야  
   네트워크(그래프) 분석, 알고리즘 분석, 문법과 언어 분석 등

논리와 명제

1. 명제: 참,거짓을 명확히 구분할 수 있는 문장
2. 진리값: 참(True) 또는 거짓(False)
3. 예제   
   1) 콜라는 검은색이다. (True)  
   2) 축구경기는 라켓을 사용한다 (False)

논리 연산

1. 단순 명제: 하나의 문장 혹은 식으로 구성된 명제
2. 합성 명제: 여러 개의 단순 명제들을 논리연산자들로 연결해 만든 명제
3. 논리 연산자: 단순 명제들을 연결하는데 사용

|  |  |  |  |
| --- | --- | --- | --- |
| 연산자의 이름 | 기호 | 연산자의 의미 | 예시 |
| 부정 | ~ [not p, p가 아니다] | NOT | ~p |
| 논리곱 | ∧ [p and q, p 그리고 q] | AND | p&q, pq |
| 논리합 | ∨ [p or q, p 또는 q] | OR | p+q |
| 배타적 논리합 | ⊕ [익스클루시브 OR, XOR] | Exclusive OR | 논리 회로, 패리티 비트 등에 쓰임 |
| 조건 | → [p이면 q이다, p는 q를 함축한다] | if … then | p→q |
| 쌍방 조건 | ↔ [p이면 q이고, q이면 p이다] | if and only if (iff) | p↔q |

[표] 논리 연산자들을 정리한 표.

논리연산자들에 대한 설명 :

1) 부정 : 진리값을 반대로 ( T는 F로 F는 T로)  
  2) 논리곱 : 모두 참인 경우에만 T  
  3) 논리합 : 모두 거짓인 경우에만 F  
  4) 배타적 논리합 : 하나의 명제가 T이고, 다른 하나가 F일 때만 T  
  5) 조건 : T → F일 때만, F  
  - p: q의 충분조건 / q: p의 필요조건.  
  6) 쌍방 조건 : 모두 참이거나 거짓을 때만 T   
  - p는 q의 필요충분조건이다.

명제의 <역, 이 , 대우> .

명제가 ‘A는 B이다’ 라면

역은 ‘B는 A이다’

이는 ‘A가 아니면 B가아니다’

대우는 ‘B가아니면 A가아니다’

명제와 대우는 논리적 동치 관계에 있다.

역과 이도 논리적 동치 관계에 있다 ( 명제와 대우 관계 이므로 )

Ex) 명제가 참이면 대우도 항상 참이고 명제가 거짓이면 대우도 항상 거짓.

항진 명제와 모순 명제

1. 항진 명제 : 합성 명제의 진리값이 항상 참인 경우
2. 모순 명제 : 합성 명제의 진리값이 항상 거짓인 경우

논리적 동치 관계  
1. 논리적 동치  
: 두 개의 명제의 쌍방 조건이 항진 명제일 때  
2. 논리적 동치 관계의 기본 법칙

|  |  |
| --- | --- |
| 논리적 동치 관계 | 법칙 이름 |
| P ∨ P ⬄ P, P ∧ P ⬄ P | 멱등 법칙 |
| P ∨ P ⬄ T, P ∨ F ⬄ P | 항등 법칙 |
| P ∧ P ⬄ P , P ∧ F ⬄ F |
| ~T ⬄ F , ~F ⬄ T | 부정 법칙 |
| ~(~p) ⬄ p |
| P ∨ g ⬄ g ∨ p , p ∧ g ⬄ g ∧ p | 교환 법칙 |
| P <-> g ⬄ g <-> p |
| (p ∨ g) ∨ r ⬄ p ∨ ( g ∨ r ) | 결합 법칙 |
| (p ∧ g) ∧ r ⬄ p ∧ ( g ∧ r ) |
| p ∨ (g ∧ r) ⬄ (p ∨ g ) ∧ ( p ∨ r) | 분배 법칙 |
| p ∨ ( g ∧ r ) ⬄ (p ∧ g) ∨ ( p ∧ r ) |
| p ∨ ( p ∧ g ) ⬄ p , p ∧ ( p ∨ g ) ⬄ p | 흡수 법칙 |
| ~ ( p ∨ g ) ⬄ (~p) ∧ (~g) | 드 모르간의 법칙 |
| P -> g ⬄ ~p ∨ g | 조건 법칙 |
| p -> g ⬄ ~g -> ~p | 대우 법칙 |

[법칙표]

- 두 명제가 논리적 동치임을 입증할 때, 진리표나 기본법칙을 이용해 다른 명제로 유도하는 방법 사용.

추론

1. 추론 : 주어진 명제가 참인 것을 바탕으로 새로운 명제가 참이 되는 걸 유도하는 방법  
   1) 유효 추론 : 주어진 전제가 참, 결론도 참  
    2) 허위 추론 : 주어진 전제가 참, 결론은 거짓  
   2. 전제 : 주어진 명제들  
   3. 결론 : 새로 유도된 명제  
   4. 여러 가지 추론 법칙  
     
     
     
     
     
   \* 잘 알려진 3개의 법칙 : 긍정 법칙, 부정 법칙, 삼단 법칙  
   긍정 법칙 p , p->g g  
   부정 법칙 ~g p->g ~p  
   삼단 법칙 p->g 이고 g->r 이면 p->r 이다.

술어 논리  
1. 명제 술어   
2. 술어 논리 : 명제 논리와 구분하여 명제 술어에 대한 논리  
3. 술어 한정자 : 술어를 나타내는 법 중 변수의 범위를 한정시키는 것  
 1) 전체 한정자(= 모든 것에 대하여, ∀) : 모든 x에 대하여 p(x)는 참이다  
 - 부정 : ~(∀x  p(x)) ⇔ ∃x(~p(x)) [p(x)가 성립하지 않는 x가 존재한다]  
 2) 존재 한정자(= 존재한다, ∃) : 어떤 x에 대하여 p(x)가 참인 x가 존재한다 (적어도 한개)  
 - 부정 : ~(∃x p(x)) ⇔ ∀x (~p(x)) [모든 x는 p(x)가 성립하지 않는다]