4.1 증명의 방법론

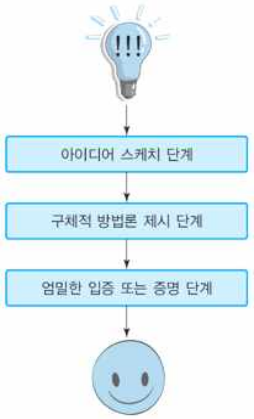
교과목명	이산수학	분반		담당교수	김 외 현
학부(과)		학번		성명	

정의 **증명**

논리적 법칙을 이용하여 주어진 가정으로부터 결론을 유도해내는 추론의 한 방법으로서, 어떠한 명제나 논증이 적절하고 타당한지를 입증하는 작업

- 추론을 통한 수학적 증명은 대부분의 사람들에게 어렵게 느껴질 수 있음
- 그러나 증명과정을 통하여 공학이나 수학을 비롯한 여러분야에서 논리적 바탕에 기반을 둔 학문적 탐구가 가능함

참고 **증명의 단계적 접근 방법**



1. 아이디어 스케치 단계
   - 문제 해결의 핵심적인 실마리를 찾아내어 기술함
   - 문제를 해결할 수 있는 방법론을 구상하게 되며 개략적인 아이디어를 스케치함

2. 구체적인 방법론 제시 단계
   - 아이디어를 묶어서 구체적인 블록 다이어그램(block diagram) 등으로 표현함
   - 프로그래밍의 경우 유사 코드(pseudo code) 단계까지 구체화하는 단계

3. 엄밀한 입증이나 증명의 단계
   - 자기가 내린 결론을 객관적인 증명 방법을 통해 누구나 공감할 수 있게 증명함

4.2 여러 가지 증명 방법

교과목명	이산수학	분반		담당교수	김 외 현
학부(과)		학번		성명	

- 수학이나 공학에서의 증명 문제는 $p \rightarrow q$와 같은 논리 함축을 증명함
- 논리 함축 $p \rightarrow q$가 참이 되기 위해서는 p, q가 모두 참이거나 q에 관계없이 p가 거짓임을 보이면 됨

✓ 증명 방법은 **직접 증명법**과 **간접 증명법** 그리고 **기타 증명법**으로 구분함
- 직접 증명법: $p \rightarrow q$를 직접 증명하는 것
- 간접 증명법: 논리적 동치를 이용하거나 다른 특수한 방법으로 증명

✓ 주어진 문제 유형에 따라 다양한 방법으로 접근하는 것이 효율적임

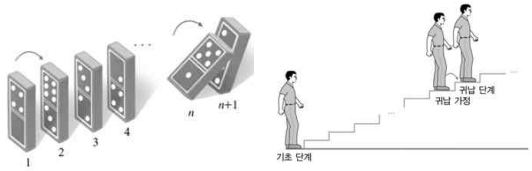
정의 **새로운 결과를 얻는 2가지 방법론**

(1) 연역법(deduction)
주어진 사실(facts)들과 공리(axioms)들에 입각하여 추론(inference)을 통하여 새로운 사실을 도출하는 것

(2) 귀납법(induction)
관찰과 실험에 기반한 가설을 귀납 추론을 통하여 일반적인 규칙을 입증하는 것

정의 **수학적 귀납법**

명제 $p_1, p_2, \cdots, p_n$이 사실이라고 할 때, p_{n+1}의 경우에도 성립함을 보이면 됨



정리 **수학적 귀납법의 원리**

$p(n)$: $n \geq 1$인 모든 정수에 대해 참임을 증명

$\Rightarrow$ ① (기초 단계) $p(1)$: 참임을 보임
② (귀납 가정) $p(n)$: 참이라고 가정
③ (귀납 단계) $p(n+1)$: 참임을 보임

예제

1. $S_n = \sum_{i=1}^{n} i = \frac{n(n+1)}{2}$ 임을 증명해보자.

2. $1^2 + 2^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$ 임을 증명해 보자.

3. $n \geq 4$인 모든 정수에 대하여 $2^n \geq n^2$임을 증명해 보자.

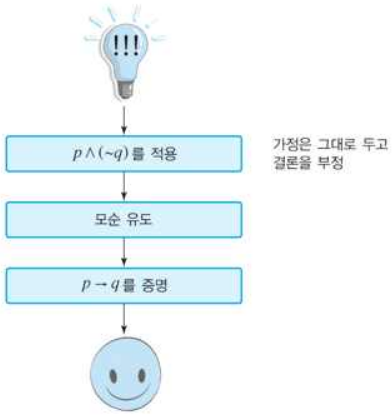
정의 **모순 증명법(귀류법)**

주어진 문제의 명제를 일단 부정해 놓고 논리를 전개하여 그것이 모순됨을 보임으로써 본래의 명제가 사실임을 증명하는 방법

참고 **$p \rightarrow q$의 모순 증명법**

$p \rightarrow q$: 참 $\equiv$ $p \wedge (\sim q)$: 거짓

$p \wedge (\sim q)$: 참이라 가정 $\Rightarrow$ 결과 모순 유도



예제

4. 'a, b가 실수일 때, $a+b-2>0$이면 $a>1$ 또는 $b>1$' 임을 증명해보자.

5. $\sqrt{2}$는 유리수(rational number)가 아님을 증명해 보자.

6. $1+3\sqrt{2}$ 가 무리수임을 증명해보자.

정의 **직접 증명법**
통상 주어진 유용한 정보로부터 추론을 통하여 목적하는 결론에 도달할 수 있도록 유도하는 증명법

참고 **$p \rightarrow q$ 의 직접 증명법**
p: 참이라 가정 $\Rightarrow$ q: 참임을 보임

예제

8. 만약 $6x+9y=7$이라면 x 또는 y가 정수가 아님을 증명해보자.

9. $|a|>|b|$일 때 $a^2>b^2$임을 증명해보자.

7. n이 자연수이고 n이 2가 아닌 소수(prime number)일 경우, n은 반드시 홀수가 됨을 증명해보자.

10. 두 짝수의 합은 항상 짝수가 됨을 증명해보자.

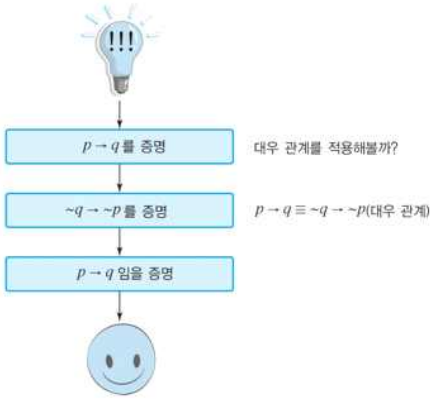
정의 **대우 증명법**

명제가 참이 되는 것을 논리적 동치 관계인 대우를 이용하여 간접적으로 보여주는 증명법

참고 **$p \rightarrow q$의 대우 증명법**

$p \rightarrow q \equiv \sim q \rightarrow \sim p$

$\sim q \rightarrow \sim p$: 참임을 증명



예제

11. x가 짝수이면 x는 2이거나 소수가 아님을 증명해보자.

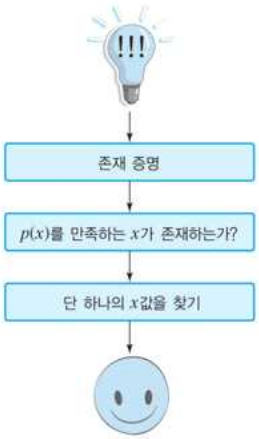
12. 완전수는 소수가 아님을 증명해보자.

13. 모든 정수 n에 대해 n^2이 짝수라고 가정하면 n도 짝수임을 증명해보자.

14. n이 자연수이고 n이 2가 아닌 소수라면 n이 홀수임을 증명해보자.

정의 **존재 증명법**

$p(x)$: 변수 x를 가지는 명제일 때,
$\exists x\, p(x)$를 보이는 증명법



예제

15. $p(x)$가 술어 'x는 정수이고 $x^2 = 289$'일 때 이 식을 만족하는 x가 존재함을 증명해보자.

16. a가 0이 아닌 실수이고 b가 실수일 때, 방정식 $ax + b = 0$을 만족시키는 실수 x가 존재함을 증명해보자.

정의 **반례 증명법**

어떤 명제가 참 또는 거짓임을 입증하기가 상당히 어려운 경우, 주어진 명제에서 모순이 되는 간단한 하나의 예를 보이는 증명법

참고 **$\forall x\, p(x)$ 거짓 증명**

$\sim(\forall x\, p(x)) \equiv \exists x(\sim p(x))$
반례 x가 적어도 하나 존재함을 보임

예제

17. 'p가 양의 정수이고 $x = p^2 + 1$이면 x는 소수이다'란 명제가 거짓임을 증명해보자.

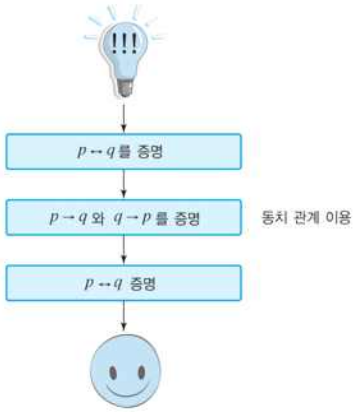
18. 모든 실수 x에 대해 $(x+1)^2 \geq x^2$이 성립하지 않음을 증명해보자.

19. '모든 실수 a와 b에 대하여 $a^2 = b^2$이면 $a = b$이다'가 거짓임을 증명해보자.

정의 **필요충분조건 증명법**

쌍조건문 $p \leftrightarrow q$를 증명하기 위해

$p \to q$와 $q \to p$ 모두 참임을 보이는 증명법



예제

20. 모든 정수 n에 대해, $n-1$이 짝수임과 n이 홀수임이 동치라는 것을 증명해보자.

4.3 프로그램의 입증

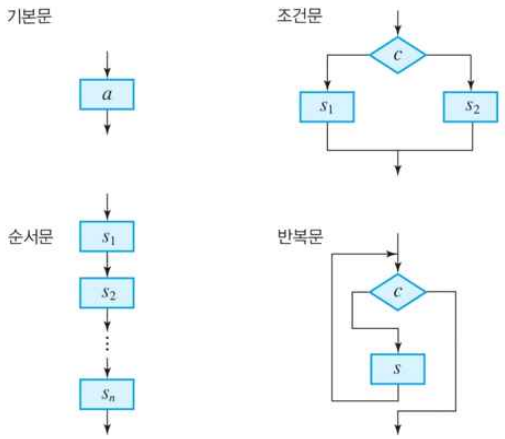
교과목명	이산수학	분반		담당교수	김 외 현
학부(과)		학번		성명	

참고 **프로그램의 입증**

- 증명 못지않게 엄밀한 정확성이 요구되는 컴퓨터 프로그램을 입증하는 것 또한 매우 중요함
- 정확한 프로그램이 되기 위해서는 구문 오류(syntax error)를 포함하지 않아야 함
- 예상치 못하게 끝나서도 안 되며, 주어진 입력에 대해 정확한 결과를 도출해야 함
- 프로그램의 정확성에 대한 입증이 필요함

정의 **프로그램의 제어 구조**

1. 순서문(Sequential statements)
2. 조건문(Conditional statements)
3. 반복문(Repeated statements)
4. 무조건적 이동문(Unconditional transfer statements)



예제

21. 다음과 같이 1부터 n까지의 정수값을 합하는 C언어 프로그램에서 정확성을 입증해보자.

```
void Sum_n()
{
   int i, n, sum = 0;
   scanf("%d", &n);
   for (i = 1; i <= n; i++)
      sum += i;
   printf("%d, %d", n, sum);
}
```

22. n개의 양의 정수 중에서 가장 작은 수를 찾는 다음과 같은 프로그램을 입증해보자.

```
Find_Min (int array[ ], int MIN)
{
   int i ;
   MIN = array[0];
   for (i = 1; i < n; i++)
      if (MIN > array[i]) MIN = array[i];
   return (MIN);
}
```