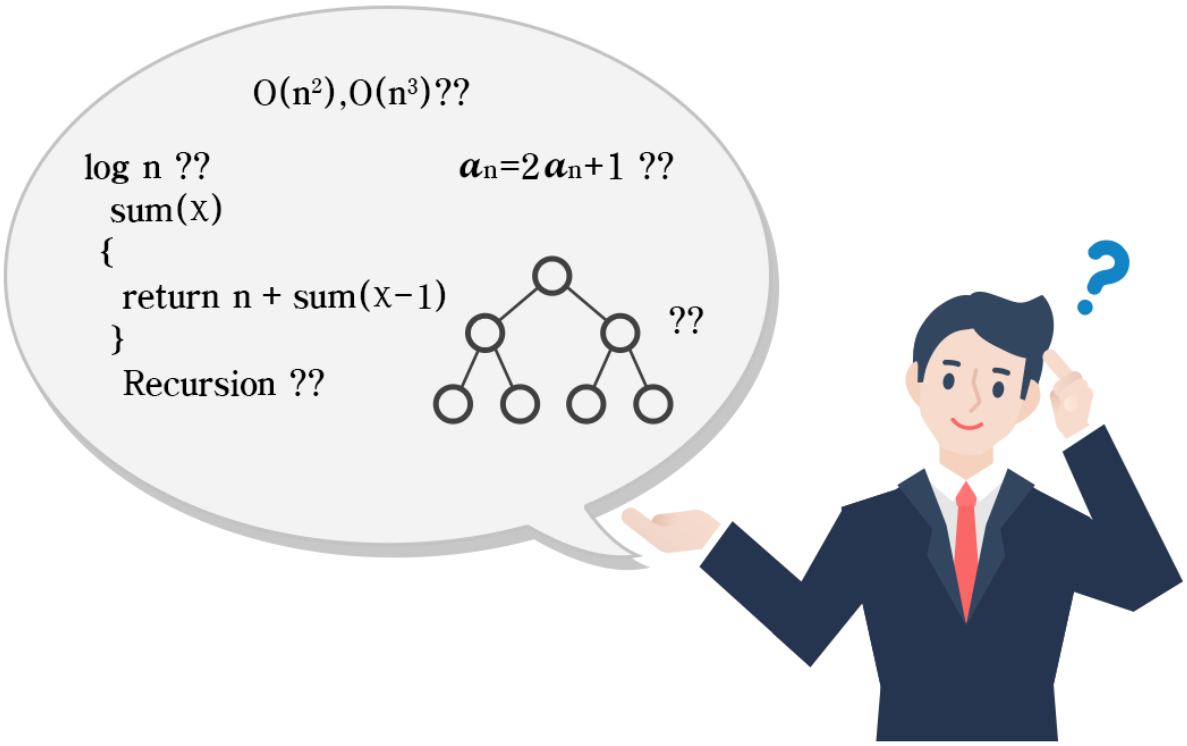
서문



■ 당신이...

- Problem Solving 문제들 봤는데 어떻게 풀어야 하는지 전혀 감이 오지 않는다.
- Dijkstra 알고리즘 아무리 봐도, "될 것 같기는 한데" 이유는 모르겠다.
- 도구나 라이브러리는 잘 쓰는데 프로그램 처음부터 짜려면 막막하다.
- 어떻게 프로그램을 짜면 더 빠른지, 더 느린지 전혀 감이 없다.
- $\log n$을 본 적이 없거나, 본 적이 있는데 무슨 뜻이라고 한마디로 설명하지는 못한다. 혹은, $\log n$의 밑이 10 이나 e라고 알고 있다.
- Problem Solving 자료를 보고 시간을 들여도 발전이 없다.

■ 하나라도 일치하는 상황이 있다면..., 어쩌면...

- Problem Solving을 본격적으로 공부할 준비가 안된 것일 수 있다.

- **"왜 이해가 안되는 것인가?"**

■ 필요한 것은..., 아마도...

- **논리적**으로 **정확**하게 **확인**하는 과정에 대한 연습이 없어서이다.

- 되는 것 같다는 기분이나 "공식을 외우는 것" 말고 정확하게 확인해 본 적이 있는가?

- 프로그램을 짜기 시작하기 전에, 정확한 결과가 나올 것인지, 얼마나 빠르게 돌아갈 것인지 미리 알 수 있는가?

- 확인이 안된 상태에서 프로그램을 짜기 시작하면, 결과가 정확할지, 얼마나 빠를지 예측할 수 없고, 제대로 된 결과가 나오지 않으면 고치는 것이 어렵고 무작정 여러가지를 시도해 볼 수 밖에 없다.

- 정확히 확인하는 훈련이 되어 있지 않으면, 단순 작업 이상의 코드를 작성하기 어렵고, 다른 사람의 코드를 고치는 것도 매우 어렵다.

- 정확하게 확인하는 과정을 수많은 세월 동안 정리해 둔 것이 "증명" 기법이다.

- 증명 기법은 딱딱한 것이 아닌 기발한 아이디어들의 집합이고 "이해하면 재미있는 그림"들과 같다.

- 이 과정에서 쉬운 문제들을 보고 정확하게 확인하는 것을 연습해 보자.

"어떤 전공도 상식선에서 이해되는 분야는 없다."

Computational Thinking

- 기초 논리 & 수학 -

0. 서론 – 프로그래밍과 논리/수학

■ 프로그래밍의 어려운 점 두 가지

- 프로그래밍 언어 문법과 라이브러리 사용

- 논리 (Hard Logic)

■ 문법과 라이브러리

- 많이 알려진 어려운 점

```
#include <iostream>
#include <set>
#include <functional>
using namespace std;

int main(){

    set<int> s;

    pair<set<int>::iterator, bool> pr;
    pr = s.insert(50);      // returns result pair
    s.insert(40);
    s.insert(80);

    if (true == pr.second)
        cout << *pr.first << " Success!" << endl;
    else
        cout << *pr.first << " Failure! " << endl;
……
```

- 위 프로그램이 무엇을 하는 것인지 처음 보는 사람은 알 수 없음

- 능숙해 지기 위해 많은 훈련이 필요하지만, 이 과정의 중요 목표는 **아님**

- 프로그래밍을 최초로 배울 때 약간의 어려움이 있지만 훈련에 비례하여 실력이 느는 경향이 있음

- 일반 상식으로 원래 알고 있는 것이 아니기 때문에 훈련의 필요성에 대해 반감이 없음

■ 논리 (Hard Logic)

- Hard vs. Soft Logic

- 카드 문제

  - 사실: 모든 카드의 한쪽에는 알파벳이, 다른 쪽에는 숫자가 써 있음

  - 주장: 만약 한쪽이 D 이면 반대쪽은 3

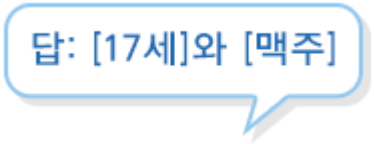
  - 주장이 사실인지 확인하기 위해 다음 카드들 중 반드시 뒤집어 보아야 하는 것은 몇 개이고 어느 것인가?

  - 잠깐 생각해 봅시다.....

답: [D]와 [7]

- [D]를 뒤집어 보아야 한다는 것은 누구나 알아 냄
- [3]을 뒤집어 보아야 한다고 말하는 경우가 많이 있음
- 중요: [3] 뒤에 [D]가 있든 없든 주장이 사실인지 여부에 영향이 없음
- [7]을 뒤집어 볼 필요가 없다고 말하는 경우도 많음
- 중요: [7] 뒤에 [D]가 있으면 주장이 성립하지 않게 됨

- 맥주집 문제
  - 규칙: 20 세 이하인 사람은 맥주를 마실 수 없음
  - 나이 혹은 마시고 있는 것을 표시한 다음 4 명 중 확인이 필요한 사람은 몇 명이고 누구인가?



- 카드 문제와 맥주집 문제의 비교
  - 맥주집 문제가 훨씬 풀기 쉽다
  - 사실, 두 문제는 완전히 같은 문제임. 즉, 논리적 구성은 완전히 동일함
  - 왜 맥주집 문제가 풀기 쉬운가?
  - 논리 구조를 정확히 이해하고 맥주집 문제를 푸는 사람은 카드 문제를 똑같이 풀 수 있음
  - 즉, 맥주집 문제를 풀 때 **논리를 사용한 것이 아니다**!

- Hard vs. Soft Logic
  - 맥주집 문제를 풀 때는 직관을 사용한 것
  - 직관은 논리적인 **느낌**을 주는 것
  - 직관의 장점은 (익숙한 상황에서) 빠르다는 것
  - 직관의 단점은 정확하지 않다는 것 (가끔은 익숙한 상황에서도 틀림)
  - 또 다른 단점은 강한 착각을 일으킨다는 것

- 과자와 버스

  - "너 과자 몇 개 먹었니?" vs. "버스 타려고 하는데 천원 있니?"

  - 두 질문은 같은 표현을 사용하지만, 하나는 정확한 개수를 요구하고, 다른 하나는 천원 이상이 있는지 물어보는 것

- 토플과 복권

  - "합격하려면 토플 500 점 이상 혹은 토익 600 점 이상이 필요" vs. "복권에 당첨되면 자동차 혹은 천만원을 줍니다"

  - 두 말은 같은 표현을 사용하지만 하나는 inclusive or, 다른 하나는 exclusive or

- 일상 생활에서는

  - Soft Logic 이 빠르기 때문에 유용

  - 논리적으로 부정확한 표현을 사용하지만, 어떤 의미인지 모든 사람이 이미 알고 있다는 가정이 존재

- 프로그래밍은 Hard Logic 을 사용

  - 프로그래밍 언어의 표현들이 모두 논리학에서 나온 것

  - 사용되는 수많은 알고리즘들을 이해하기 위해서는 Hard Logic 이 필요

- 오해의 근원

  - **Soft Logic 으로 알고리즘을 이해하려고 하는 것!**

  - 알고리즘 설명을 보고 또 봐도 이해가 안되는 것은 증명을 안 봤기 때문

  - 증명을 봐도 이해가 안되는 것은 직관으로 이해하려고 하기 때문

  - 가끔 직관적으로 이해되는 알고리즘이 있지만 조금만 어려워지면
    직관으로 완전한 이해를 얻는 것은 사실상 불가능

■ 논리 연습

- 문제 1: 다음을 명제식 형태로 쓰고 참인지 거짓인지 판단하시오

① 만약 0이 홀수라면, 미국에서 2080년 월드컵이 열린다.

② 만약 19893827938274839이 Prime Number라면, 2는 짝수이다.

[Solution]

① p : 0 은 홀수이다. (거짓)

q : 미국에서 2080 년 월드컵이 열린다. (알 수 없음)

명제식 : $p \rightarrow q$, p 명제가 거짓이므로, q명제의 참/여부에 상관없이 해당 명제식은 참이다.

② p : 19893827938274839 은 Prime Number 이다. (알 수 없음)

q : 2 는 짝수이다. (참)

명제식 : $p \rightarrow q$, 대우 명제는 $\sim q \rightarrow \sim p$ 인데, ~q 는 '2 가 홀수이다' 가 되어 거짓인 명제가 된다. 따라서 $\sim q$ 명제가 거짓이므로, $\sim p$ 명제의 참/여부에 상관없이 해당 명제식은 참이 된다.
대우 명제식이 참이므로, 본 명제식 또한 참이다.

- 문제 2: p와 q가 명제이고 $p \to q$가 거짓이라고 하자. 다음 명제식의 참 거짓은 어떻게 되는가?

① $\sim p \to q$ ② $p \vee q$ ③ $q \to p$

[Solution]

① $\sim p \to q$

: $p \to q$가 거짓이기 위해선 p 참, q 거짓인 경우이다. 따라서 $\sim p$ 는 거짓이고 q또한 거짓이므로 $\sim p \to q$는 참이다.

② $p \vee q$

: p 참, q 거짓이므로 $p \vee q$는 참이다.

③ $q \to p$

: p 참, q 거짓이므로 $q \to p$은 참이다.

- 문제 3: 다음 명제들의 역, 이, 대우를 쓰시오

① 만약 0이 홀수라면, 미국에서 2080년 월드컵이 열린다.

② 만약 1989382793827483 9이 Prime Number라면, 2는 짝수이다.

[Solution]

① 명제 : 만약 0이 홀수라면, 미국에서 2080년 월드컵이 열린다.
역 : 만약 미국에서 2080년 월드컵이 열린다면, 0이 홀수이다.
이 : 만약 0이 짝수라면, 미국에서 2080년 월드컵이 열리지 않는다.
대우 : 만약 미국에서 2080년 월드컵이 열리지 않는다면, 0은 짝수이다.

② 명제 : 만약 19893827938274839 이 Prime Number 라면, 2 는 짝수이다.
역 : 만약 2 가 짝수이면 19893827938274839 이 Prime Number 이다.
이 : 만약 19893827938274839 이 Prime Number 가 아니라면 2 는 홀수이다.
대우 : 만약 2 가 홀수이면 19893827938274839 이 Prime Number 가 아니다.

- 문제 4: 다음 명제식의 진리표를 만드시오

① $p \wedge (q \rightarrow \sim p)$

② $(p \wedge \sim q) \rightarrow r$

[Solution]

① $p \wedge (q \rightarrow \sim p)$

p	q	~p	$(q \rightarrow \sim p)$	$\boldsymbol{p \wedge (q \rightarrow \sim p)}$
T	T	F	F	**F**
T	F	F	T	**T**
F	T	T	T	**F**
F	F	T	T	**F**

② $(p \wedge \sim q) \rightarrow r$

p	q	r	~q	$(p \wedge \sim q)$	$\boldsymbol{(p \wedge \sim q) \rightarrow r}$
T	T	T	F	F	**T**
T	T	F	F	F	**T**
T	F	T	T	T	**T**
T	F	F	T	T	**F**
F	T	T	F	F	**T**
F	T	F	F	F	**T**
F	F	T	T	F	**T**
F	F	F	T	F	**T**

■ 증명

- 증명은 정확한 명제식으로 표현할 수 있는 것이라야 함
- 보통은 정확한 명제식까지 쓰지는 않으나 근본적으로는 명제식으로 바꿀 수 있음
- 증명에 대한 수많은 오해가 $p \to q$를 $p \leftrightarrow q$와 혼동하는 것에서 일어남

- 모든 당구공은 색이 같다는 다음 증명에서 잘못된 것은?
  - 수학적 귀납법: $P(1)$이 참이고, $P(n) \to P(n+1)$이 참이면 $P(n)$은 모든 자연수 n에 대해서 참이다.
  - 모든 자연수 n에 대해 당구공 n개가 들어있는 집합에서 그 집합에 포함된 당구공은 모두 색이 같다는 것을 증명함
  - $P(1)$: 당구공 1 개가 들어있는 집합은 모두 색이 같음
  - $P(n) \to P(n+1)$을 증명하기 위해 $P(n)$이 참이라고 가정
  - 당구공 $n+1$개가 들어 있는 임의의 집합을 생각함
  - 이 집합에서 하나를 빼면 당구공 n개가 있는 집합이 되므로 지금 상황에서 모든 당구공의 색이 같음
  - 방금 뺀 원소를 다시 넣고, 다른 당구공을 빼면 역시 당구공 n개가 있는 집합이 되므로 지금 상황에서도 모든 당구공의 색이 같음
  - 위의 두 상황에서 처음 뺀 당구공과 두번째로 뺀 당구공의 색이 같음을 알수 있으므로 당구공 $n+1$개가 들어 있는 임의의 집합은 색이 같은 것만을 포함함

- 대부분의 사람들이 $P(n)$이 참이라고 가정할 수 없다고 반론함
- 수학적 귀납법에서 필요한 것은 $P(n) \rightarrow P(n+1)$이 참임을 보이는 것 뿐이므로 $P(n)$이 정말로 참일 필요는 없음

- 위 증명에서 실제로 잘못된 것은 다음 부분
  - 위의 두 상황에서 처음 뺀 당구공과 두번째로 뺀 당구공의 색이 같음을 알수 있으므로...

- 처음 뺀 당구공과 두번째로 뺀 당구공의 색이 같다는 것은 공통 부분이 있다는 것인데, 실제로 $n = 1$인 경우, 즉 $n + 1 = 2$인 경우 공통 부분이 없음

- Prime Number 의 개수는 무한히 많다는 다음 증명은 옳은가?

  - Prime Number 의 개수가 유한한 k개라고 가정

  - 모든 Prime Number 를 다 곱하고 1 을 더한 수를 n이라고 하자

  - 이 수 n은 어떤 Prime 으로 나누어도 나머지가 1 이다

  - 그런데 n은 어떤 Prime 보다도 크므로 합성수이다

  - 합성수이지만 어떤 Prime 으로도 나누어지지 않으므로 모순 발생

- 이 증명에 대한 반론으로 몇 개의 Prime 이 더 존재하면 되는 것이 아니냐는 주장이 자주 있음

- 위 증명은 "Prime Number 가 k개 이면 모순이 발생", 즉, "Prime Number 가 k개" → "항상 거짓", 이 명제가 항상 참임을 확인한 것

- 즉, "Prime Number 가 k개"라는 명제가 항상 거짓일 수 밖에 없다!

■ 수학적 귀납법과 증명의 수준

- 수학적 귀납법의 기본형: $P(1)$이 참이고, $P(n) \rightarrow P(n+1)$이 참이면 $P(n)$은 모든 자연수 n에 대해서 참이다.
- 수학적 귀납법의 강한 형태: $P(1)$이 참이고, $P(1) \wedge P(2) \wedge \cdots \wedge P(n) \rightarrow P(n+1)$이 참이면 $P(n)$은 모든 자연수 n에 대해서 참이다.

- 다음 함수가 1 부터 x 까지의 합을 계산함을 증명해 보자

```
int sum(int x)
{
    if (x <= 0) return 0;
    return x + sum(x-1);
}
```

- High-level 증명에서는 1 부터 x 까지 합의 정의 중 하나인 $S(n) = S(n-1) + n$을 그대로 코딩한 것이므로 증명이 된 것이라고 말하는 경우가 많음
- 상세한 증명을 하려면 단순히 "답이 맞는 것이 당연하다"라고 말하는 것으로는 충분하지 않음
  - **증명이 가능한 명제**를 만들어야 함
  - 이 경우 증명이 가능한 명제는 다음과 같음: "sum(x)가 리턴하는 값은 1+2+…+x 의 값과 항상 같다"
  - 이제 수학적 귀납법을 적용할 수 있음
  - $P(1)$이 참이다: "sum(1)이 리턴하는 값은 1 이다"를 증명하면 됨. 실제 코드에 1 을 대입하면 1 을 리턴함을 알 수 있음

- $P(x) \rightarrow P(x+1)$이 참이다: "sum(x-1)이 1+2+…+(x-1)을 리턴하면 sum(x)는 1+2+…+x 를 리턴한다"를 증명하면 됨. 코드를 보면 sum(x)는 x+sum(x-1)의 값을 리턴함. sum(x-1)의 리턴 값은 1+2+…+(x-1)과 같다고 가정했으므로 sum(x)는 1+2+…+(x-1)+x=1+2+…+x 를 리턴함을 확인할 수 있음

- sum(x-1)을 블랙박스로 보는 것이 이해에 도움을 줄 때가 있음

- 소팅의 사례

  - High-level 증명에서는 소팅이 된다는 것을 직관적인 수준에서 설명하는 경우가 많음

  - 상세한 증명을 위해서는 증명이 가능한 명제가 필요

  - 배열 A[1], A[2], …, A[n]을 소팅하는 알고리즘의 정확성을 증명하려고 한다면, 증명이 가능한 명제는 다음과 같을 것임: "A[1] < A[2] < …< A[n]"

  - 버블 소트가 정확함을 어떻게 증명할 지 생각해 봅시다.

> 상세한 증명에 대한 경험이 없는 경우가 많고, 상세한 증명 없이는 확인하거나 이해할 수 없는 문제들이 많으므로 연습 문제들은 상세한 증명을 제시하는 것을 목표로 함

■ 증명 연습

- Trivial Proof: $\forall x, P(x) \rightarrow Q(x)$를 증명하려는데, $Q(x)$가 항상 참인 경우

- 문제 1: 다음 명제를 증명하시오

① 실수 x에 대해, 만약 $x < -1$이면 $x^2 + \frac{1}{4} > 0$이다

② n이 홀수이면 $4n^3 + 6n^2 + 12$는 짝수이다

[Solution]

Proof)

① 실수 x에 대해, 만약 $x < -1$이면 $x^2 + \frac{1}{4} > 0$이다.

$x^2 + \frac{1}{4} > 0, x^2 > -\frac{1}{4}$ 이고, x는 실수이므로 Q(x)는 항상 참이다.

따라서 $\forall x, P(x) \rightarrow Q(x)$ 이다.

② n이 홀수이면 $4n^3 + 6n^2 + 12$는 짝수이다

$4n^3 + 6n^2 + 12 = 2(2n^3 + 3n^2 + 6)$ 이므로 $4n^3 + 6n^2 + 12$는 짝수이다.

그러므로 Q(x)는 항상 참이다.

따라서 $\forall x, P(x) \rightarrow Q(x)$ 이다.

- Vacuous Proof: $\forall x, P(x) \rightarrow Q(x)$를 증명하려는데, $P(x)$가 항상 거짓인 경우

- 문제 2: 다음 명제를 증명하시오

① 실수 x에 대해, 만약 $2x^2 - 4x + 4 < 0$이면 $x > 8$이다

② $4n^3 + 6n^2 + 11$는 짝수이면 n이 홀수이다

[Solution]

Proof)

① 실수 x에 대해, 만약 $2x^2 - 4x + 4 < 0$이면 $x > 8$이다.

$2x^2 - 4x + 4 = 2(x^2 - 2x) + 4 = 2(x-1)^2 + 6 \geq 0$ 이다.

따라서 $2x^2 - 4x + 4 < 0$은 거짓이다.

그렇기 때문에 $P(x)$는 거짓이므로 해당 명제 $\forall x, P(x) \rightarrow Q(x)$는 참이다.

② $4n^3 + 6n^2 + 11$는 짝수이면 n이 홀수이다

$4n^3 + 6n^2 + 11 = 2(2n^3 + 3n^2 + 5) + 1$ 이므로 $4n^3 + 6n^2 + 11$ 은 홀수이다.

그러므로 $P(x)$는 거짓이므로 해당 명제 $\forall x, P(x) \rightarrow Q(x)$는 참이다.

1. 논리와 증명

- 문제 1: 다음 명제들이 항진명제라는 것을 진리표를 이용해서 보이시오

① $\sim(\sim p \wedge q) \vee q$

② $(\sim p \vee q) \vee (p \wedge \sim q)$

[Solution]

① $\sim(\sim p \wedge q) \vee q$

p	q	~p	$(\sim p \wedge q)$	$\sim(\sim p \wedge q)$	$\sim(\sim p \wedge q) \vee q$
T	T	F	F	T	T
T	F	F	F	T	T
F	T	T	T	F	T
F	F	T	F	T	T

② $(\sim p \vee q) \vee (p \wedge \sim q)$

p	q	~p	$(\sim p \vee q)$	~q	$(p \wedge \sim q)$	$(\sim p \vee q) \vee (p \wedge \sim q)$
T	T	F	T	F	F	T
T	F	F	F	T	T	T
F	T	T	T	F	F	T
F	F	T	T	T	F	T

- 문제 2: 다음 명제들이 모순명제라는 것을 진리표를 이용해서 보이시오

① $(\sim p \vee q) \wedge (p \wedge \sim q)$

② $(p \wedge q) \wedge (p \wedge \sim q)$

[Solution]

① $(\sim p \vee q) \wedge (p \wedge \sim q)$

p	q	~p	$(\sim p \vee q)$	~q	$(p \wedge \sim q)$	$(\sim \boldsymbol{p} \vee \boldsymbol{q}) \wedge (\boldsymbol{p} \wedge \sim \boldsymbol{q})$
T	T	F	T	F	F	**F**
T	F	F	F	T	T	**F**
F	T	T	T	F	F	**F**
F	F	T	T	T	F	**F**

② $(p \wedge q) \wedge (p \wedge \sim q)$

p	q	~q	$(p \wedge q)$	$(p \wedge \sim q)$	$(\boldsymbol{p} \wedge \boldsymbol{q}) \wedge (\boldsymbol{p} \wedge \sim \boldsymbol{q})$
T	T	F	T	F	**F**
T	F	T	F	T	**F**
F	T	F	F	F	**F**
F	F	T	F	F	**F**

- 문제 3: 다음 명제의 쌍 들에 대해서 두 명제가 동등한지를 진리표를 이용해 확인하시오

① $p \wedge (p \vee q)$와 p

② $\sim p \vee \sim q$와 $\sim(p \vee q)$

[Solution]

① $p \wedge (p \vee q)$와 p

p	q	$(p \vee q)$	$\boldsymbol{p \wedge (p \vee q)}$
T	T	T	**T**
T	F	T	**T**
F	T	T	**F**
F	F	F	**F**

② $\sim p \vee \sim q$와 $\sim(p \vee q)$

~p	~q	$\boldsymbol{\sim p \vee \sim q}$	p	q	$(p \vee q)$	$\sim(\boldsymbol{p \vee q})$
F	F	**F**	T	T	T	**F**
F	T	**F**	T	F	T	**F**
T	F	**F**	F	T	T	**F**
T	T	**T**	F	F	F	**T**

- 문제 4: 명제식의 변형을 통하여 다음 명제를 간소화하시오

① $(p \wedge \sim q) \vee (p \wedge q)$

② $(p \vee \sim q) \wedge (\sim p \vee \sim q)$

[Solution]

① $(p \wedge \sim q) \vee (p \wedge q) = p \wedge (\sim q \vee q) = p \wedge U = p$ (이때 U는 항진명제이다.)

② $(p \vee \sim q) \wedge (\sim p \vee \sim q) = (\sim q \vee p) \wedge (\sim q \vee \sim p) = \sim q \vee (p \wedge \sim p) = \sim q \vee \emptyset = \sim q$

- 문제 5: 다음 명제들이 참인지 확인하시오. 단, R은 실수의 집합을 의미하고, Z는 정수의 집합을 의미한다.

① $\forall x \in R, x^2 \geq x$

② $\forall x \in Z, x^2 \geq x$

③ $\exists x \in R, x^2 < x$

④ $\exists x \in Z, x^2 < x$

[Solution]

① $\forall x \in R, x^2 \geq x$

$x = \frac{1}{2}$일 때, $x^2 \geq x$ 을 만족하지 않는다. (반례)

따라서 ① 명제는 거짓이다.

② $\forall x \in Z, x^2 \geq x$

해당 명제를 만족하지 않는 x의 범위는 $0 < x < 1$ 인데, 해당 범위엔 정수 값이 존재하지 않는다. 다시 말해 모든 정수 x에 대해 부등식 $x^2 \geq x$ 을 만족한다고 할 수 있다. 따라서 ② 명제는 참이다.

③ $\exists x \in R, x^2 < x$

$x = \frac{1}{2}$일 때, $x^2 < x$ 을 만족한다. (어떤 x 존재)

따라서 ③ 명제는 참이다.

④ $\exists x \in Z, x^2 < x$

$x^2 < x, x^2 - x < 0, x(x-1) < 0$ 이므로, 위 부등식의 해는 $0 < x < 1$ 가 된다.

이때, 해당 부등식 조건을 만족하는 어떤 정수도 존재하지 않는다.

따라서 ④ 명제는 거짓이다.

- 문제 6: (직접 증명) n이 짝수이면 $3n+5$는 홀수임을 증명하라.

(힌트: $n=2k$로 두고 $3n+5$가 $2\left(\textit{어떤 정수}\right)+1$ 형태로 표현될 수 있는지...)

[Solution]

Proof)

n = 2k 일 때, 3n + 5 = 3*(2k) + 5 = 6k + 5 = 6k + 4 + 1 = 2*(3k + 2) + 1

그러므로 3n + 5 는 홀수이다.

- 문제 7: n이 홀수이면 n^2+n은 짝수임을 증명하라.

[Solution]

Proof)

n = 2k + 1 일 때, $n^2+n=(2k+1)^2+(2k+1)=4k^2+4k+1+2+1=4k^2+4k+4=2(2k^2+2k+2)$

그러므로 n^2+n은 짝수이다.

- 문제 8: m이 짝수이고 n이 홀수이면 $2m + 3n$은 홀수임을 증명하라

[Solution]

Proof)

m = 2k, n = 2l+1 일 때, 2m + 3n = 2*(2k) + 3*(2l+1) = 4k + 6l + 3 = 2(2k + 3l + 1) + 1

그러므로 2m+3n 은 홀수이다.

- 문제 9: (대우를 증명) 자연수 n에 대해, $n^2 + 5$가 홀수이면 n은 짝수임을 증명하라

(힌트: 명제 대신, n이 홀수이면 $n^2 + 5$은 짝수임을 증명한다)

[Solution]

Proof)

문제 9 명제의 대우명제는, 힌트처럼 'n이 홀수이면 $n^2 + 5$은 짝수이다.'와 같다.

n = 2k + 1 일 때, $n^2 + 5 = (2k + 1)^2 + 5 = 4k^2 + 4k + 6 = 2(2k^2 + 2k + 3)$

그러므로 n이 홀수이면 $n^2 + 5$은 짝수이다.

대우 명제가 참이므로, 본 명제 또한 참이다.

- 문제 10: n^2이 짝수이면 n은 짝수임을 증명하라.

[Solution]

Proof)

주어진 명제의 대우명제는, 'n 이 홀수이면 n^2이 홀수이다.'와 같다.

$n = 2k + 1$ 일 때, $n^2 = (2k+1)^2 = 4k^2 + 4k + 1 = 2(2k^2 + 2k) + 1$

그러므로 n 이 홀수이면 n^2이 홀수이다.

대우 명제가 참이므로, 본 명제 또한 참이다.

- 문제 11: (경우를 나누어 증명) 자연수 n에 대해 $n^2 + 5n + 3$은 항상 홀수임을 증명하라.

(힌트: n이 짝수인 경우와 홀수인 경우를 따로 증명한다)

[Solution]

Proof)

[n이 짝수인 경우]

$n = 2k$, $n^2 + 5n + 3 = (2k)^2 + 5(2k) + 3 = 4k^2 + 10k + 3 = 2(2k^2 + 5k + 1) + 1$

그러므로, n이 짝수인 경우 $n^2 + 5n + 3$은 항상 홀수이다.

[n이 홀수인 경우]

$n = 2k+1$, $n^2 + 5n + 3 = (2k+1)^2 + 5(2k+1) + 3 = 4k^2 + 14k + 9 = 2(2k^2 + 7k + 4) + 1$

그러므로, n이 홀수인 경우 $n^2 + 5n + 3$은 항상 홀수이다.

n이 짝수인 경우와 홀수인 경우 모두 $n^2 + 5n + 3$은 항상 홀수이므로,

자연수 n에 대해 $n^2 + 5n + 3$은 항상 홀수이다.

- 문제 12: n^2이 3 의 배수이면 n은 3 의 배수임을 증명하라.

[Solution]

Proof)

주어진 명제의 대우 명제는 다음과 같다.

n이 3 의 배수가 아니면 n^2은 3 의 배수가 아니다.

$n = 3k + 1$ 일 때, $n^2 = (3k + 1)^2 = 9k^2 + 6k + 1 = 3(3k^2 + 2k) + 1$ 이므로 3 의 배수가 아니다.

$n = 3k + 2$ 일 때, $n^2 = (3k + 2)^2 = 9k^2 + 12k + 4 = 3(3k^2 + 4k + 1) + 1$ 이므로 3 의 배수가 아니다.

따라서, n이 3 의 배수가 아니면 n^2은 3 의 배수가 아니다.

대우 명제가 참이므로, 본 명제 또한 참이다.

- 문제 13: n이 홀수이면 n^2을 8 로 나눈 나머지는 1 임을 증명하라

(힌트: n을 4 로 나눈 나머지가 1 인 경우와 3 인 경우로 나누어 보자)

[Solution]

Proof)

n이 홀수일 때, 다음과 같이 두 가지 경우가 존재한다.

$n = 4k + 1, n = 4k + 3$

위 두 가지 케이스에 대해 n^2을 8 로 나눈 나머지를 구해보자.

$n = 4k + 1, n^2 = (4k + 1)^2 = 16k^2 + 8k + 1 = 8(2k^2 + k) + 1$

$n = 4k + 3, n^2 = (4k + 3)^2 = 16k^2 + 24k + 9 = 8(2k^2 + 3k + 1) + 1$

따라서, 두 경우 모두 n^2을 8 로 나눈 나머지는 1 이 된다. 따라서 n이 홀수이면 n^2을 8 로 나눈 나머지는 1 이 된다.

- 문제 14: 어떤 자연수를 제곱하여도 그 결과를 3 으로 나눈 나머지는 2 가 아님을 증명하라.

[Solution]

Proof)

어떤 자연수 n 을 3k, 3k+1, 3k+2 의 경우로 나눠서 생각해보자.

n = 3k 일 때, $n^2 = (3k)^2 = 9k^2 = 3(3k^2)$ 이므로, 3 으로 나눈 나머지는 0 이 된다.

n = 3k+1 일 때, $n^2 = (3k+1)^2 = 9k^2 + 6k + 1 = 3(3k^2 + 2k) + 1$ 이므로, 3 으로 나누었을 때 나머지는 1 이다.

n = 3k+2 일 때, $n^2 = (3k+2)^2 = 9k^2 + 12k + 4 = 3(3k^2 + 4k + 1) + 1$ 이므로, 3 으로 나누었을 때 나머지는 1 이다.

따라서, 어떤 자연수를 제곱하여도 그 결과를 3 으로 나눈 나머지는 2 가 아니다.

- 문제 15: (귀류법) 유리수와 무리수의 합은 무리수임을 증명하라.

(힌트: 어떤 유리수와 어떤 무리수의 합이 유리수가 된다고 가정하고 모순을 이끌어 낼 수 있는가?)

[Solution]

Proof)

어떤 유리수와 어떤 무리수의 합이 유리수가 된다고 가정하자.

유리수 a, 무리수 b 가 있고 a 와 b 의 합은 유리수 c 가 된다고 하자.

a + b = c, b = c – a 가 되고, 이 때 c – a 값인 b 는 유리수의 성질에 의해 유리수여야만 한다. (가정에 모순)

따라서 b 가 무리수라는 가정에 모순되므로, 유리수와 무리수의 합은 무리수임을 증명할 수 있다.

- 문제 16: $\sqrt{2}$는 무리수임을 증명하라.

(힌트: 유리수가 된다는 것은 기약분수로 표현이 된다는 것이다)

[Solution]

Proof)

$\sqrt{2} = \frac{b}{a}(a, b$는 서로소인 정수), 즉 유리수라고 가정하자.

양 변에 a 를 곱하고 제곱하게 되면 $2a^2 = b^2$이 되어 b^2은 2 의 배수가 된다.

b^2이 2 의 배수이므로 b 또한 2 의 배수이다.

이때, b 가 2 의 배수이므로 a^2 및 a 도 2 의 배수가 된다.

이는 a,b 는 서로소라는 조건에 모순이 되므로, $\sqrt{2}$는 무리수이다.

- 문제 17: $\log_2 5$는 무리수임을 증명하라.

[Solution]

Proof)

$\log_2 5 = \frac{b}{a}$(a, b 는 서로소인 정수), 즉 유리수라고 가정하자.

$2^{\frac{b}{a}} = 5$이고, 양변을 a 제곱하게 되면 $2^b = 5^a$이 된다.

하지만, 해당 수식을 만족시키는 자연수 a, b 는 존재하지 않는다.

따라서, 유리수라는 가정에 모순되므로 $\log_2 5$는 무리수임을 증명할 수 있다.

- 문제 18: (수학적 귀납법) $1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$임을 증명하라.

[Solution]

Proof)

n=1 일 때, 1 = 1(1+1)/2 => 성립

n=k 일 때 성립한다고 가정하자.

$1+2+3+\cdots+k = \frac{k(k+1)}{2} \cdots (1)$

n=k+1 일 때,

$1+2+3+\cdots+k+k+1 = \frac{(k+1)(k+2)}{2}$

위 식을 (1)을 이용하여 다시 써보면

$\frac{k(k+1)}{2} + k + 1$ 이고, 통분하면 $\frac{\{k(k+1)+2(k+1)\}}{2} = \frac{k^2+3k+2}{2}$ 가 되며

$\frac{(k+1)(k+2)}{2} = \frac{k^2+3k+2}{2}$이 된다.

그러므로 $1 + 2 + 3 + \cdots + n = \frac{n(n+1)}{2}$이 성립한다.

- 문제 19: $1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$임을 증명하라.

[Solution]

Proof)

n=1 일 때, 1 = 1(1+1)(2+1)/6 => 성립

n=k 일 때 성립한다고 가정하자.

$1^2 + 2^2 + 3^2 + \cdots + k^2 = \frac{k(k+1)(2k+1)}{6} \cdots (1)$

n=k+1 일 때,

$1^2 + 2^2 + 3^2 + \cdots + (k+1)^2 = \frac{(k+1)(k+2)(2(k+1)+1)}{6}$

위 식을 (1)을 이용하여 다시 써보면

$\frac{k(k+1)(2k+1)}{6} + (k+1)^2$ 이고, 통분하면 $\frac{2k^3+3k^2+k+6k^2+12k+6}{6} = \frac{2k^3+9k^2+13k+6}{6}$ 가 되며

$\frac{(k+1)(k+2)(2(k+1)+1)}{6} = \frac{2k^3+9k^2+13k+6}{6}$이 된다.

그러므로 $1^2 + 2^2 + 3^2 + \cdots + n^2 = \frac{n(n+1)(2n+1)}{6}$이 성립한다.

- 문제 20: $r \neq 1$일 때 $\sum_{i=0}^{n} r^i = \frac{r^{n+1}-1}{r-1}$임을 증명하라

[Solution]

Proof)

$S_n = r^0 + r^1 + r^2 + \cdots + r^{n-1}$ … ①

양변에 r 을 곱하면 $rS_n = r^1 + r^2 + r^3 + \cdots + r^n$ … ②

①-②에서 $S_n - rS_n = 1 - r^n$, 즉 $(1-r)S_n = 1 - r^n$이므로

$S_n = \frac{1-r^n}{1-r} = \frac{r^n-1}{r-1}$ 이다.

$\sum_{i=0}^{n} r^i = S_n + r^n = \frac{r^n-1}{r-1} + r^n = \frac{r^n-1+r^{n+1}-r^n}{r-1} = \frac{r^{n+1}-1}{r-1}$ 이 성립한다.

- 문제 21: 2 이상의 모든 자연수 n에 대해 $n^3 - n$은 6 으로 나누어 떨어짐을 증명하라.

[Solution]

Proof)

n=1 일 때, 1-1=0, 6 으로 나누어 떨어지므로 => 성립

n=k 일 때 성립한다고 가정하자.

$k^3 - k = 6m$ 식이 성립하게 된다.

n=k+1 일 때,

$(k+1)^3 - (k+1) = k^3 + 3k^2 + 3k + 1 - k - 1 = k^3 + 3k^2 + 2k = k(k^2 + 3k + 2) = k(k+1)(k+2)$ 이고, 연속한 세 수의 곱은 6 의 배수이기 때문에,

$(k+1)^3 - (k+1) = 6m'$ 라고 할 수 있다.

따라서 모든 자연수 n에 대해 $n^3 - n$은 6 으로 나누어 떨어진다는 명제를 증명할 수 있다.

- 문제 22: 2 이상의 모든 자연수 n에 대해 $\sqrt{n} < \frac{1}{\sqrt{1}} + \frac{1}{\sqrt{2}} + \cdots + \frac{1}{\sqrt{n}}$임을 증명하라.

[Solution]

Proof)

n = k 일 때 $\sqrt{k} < \frac{1}{\sqrt{1}} + \frac{1}{\sqrt{2}} + \cdots + \frac{1}{\sqrt{k}}$ 이 성립한다고 가정하자. $\cdots(1)$

n = k+1 일 때, $\sqrt{k+1} < \frac{1}{\sqrt{1}} + \frac{1}{\sqrt{2}} + \cdots + \frac{1}{\sqrt{k}} + \frac{1}{\sqrt{k+1}}$가 성립함을 보이면 된다.

(1)에 의해, $\sqrt{k} < \frac{1}{\sqrt{1}} + \frac{1}{\sqrt{2}} + \cdots + \frac{1}{\sqrt{k}}$ 이 성립하므로

$\sqrt{k+1} < \sqrt{k} + \frac{1}{\sqrt{k+1}}$ 가 성립함을 보이면 $\sqrt{k+1} < \frac{1}{\sqrt{1}} + \frac{1}{\sqrt{2}} + \cdots + \frac{1}{\sqrt{k}} + \frac{1}{\sqrt{k+1}}$ 또한 성립할 것이다.

양 변에 $\sqrt{k+1}$을 곱하면, $k+1 < \sqrt{k(k+1)} + 1$이고, $k < \sqrt{k(k+1)}$가 되어, 위 부등식은 항상 성립함을 알 수 있다.

따라서 $\sqrt{k+1} < \frac{1}{\sqrt{1}} + \frac{1}{\sqrt{2}} + \cdots + \frac{1}{\sqrt{k}} + \frac{1}{\sqrt{k+1}}$ 또한 성립한다.

그러므로 2 이상의 모든 자연수 n에 대해 $\sqrt{n} < \frac{1}{\sqrt{1}} + \frac{1}{\sqrt{2}} + \cdots + \frac{1}{\sqrt{n}}$임을 증명할 수 있다.

- 문제 23: $n \times n$ 체스판이 있다. 시작 시점에 일부 칸 들이 감염되어 있다. 매 초마다 감염이 증가할 수 있다. 규칙은 다음과 같다. 어떤 감염되지 않은 칸은 상하나 좌우로 인접한 네개의 칸들 중 2 개 이상이 감염된 상태일 때 감염된다. 이 규칙에 따라 모든 칸들을 감염시키기 위해서는 초기에 n개 이상의 칸들이 감염되어 있어야 함을 증명하라.

(힌트: 금방 떠오르는 것은 답이 아닐 가능성이 많다.)

[Solution]

Proof)

한 칸이 감염되기 위해선 상하나 좌우로 인접한 네 개의 칸들 중 2 개 이상이 감염된 상태여야 한다. 그렇기 때문에, 주위에 2 개, 3 개, 4 개의 칸들이 감염되어 있을 경우 해당 칸이 감염될 수 있고 그렇게 될 경우 각각 주변에 2 개, 1 개 0 개의 칸들을 감염시킬 수 있게 된다.

이때, Segment 를 '감염된 칸과 감염되지 않은 칸의 경계인 길이가 1 인 부분'이라고 정의한다면, segment 의 수는 일정하거나 감소하여 증가할 수 없다는 것이다.

따라서, 모든 칸이 감염되었다고 생각할 경우 이때의 segment 수는 $n \times n$ 체스판의 네 모퉁이가 되어, $4n$개가 되고 따라서 초기에는 반드시 $4n$개 segment 들이 존재해야만 한다. 그러므로 초기에 $\frac{4n}{4} = n$개 이상의 칸들이 감염되어 있어야만 위 규칙에 따라 모든 칸들을 감염시킬 수 있다.

2. 수와 표현

■ 약간의 설명

- 컴퓨터는 0/1 을 표현할 수 있는 비트들을 모아 수를 표현
- k개의 비트를 사용하면 0부터 $2^k - 1$까지 표현 가능
- 사실, 꼭 저 범위인 것은 아님. 약속하는 방식에 따라 다르지만, 어떤 경우든 최대 2^k 가지의 값을 표현하는 것이 가능
- 10 진수로 k자리를 쓰면 0부터 $10^k - 1$까지 표현이 가능한 것과 완전히 동일한 과정

- 어떤 값 n을 표현하기 위해서는 몇 개의 비트가 필요할까?
- $2^k - 1 \geq n$이 성립해야 함 -> 즉, $2^k \geq n + 1$
- 같은 의미로, $k \geq \log(n+1)$ -> 약 $\log n$ 비트가 필요
- $x = \log n$과 $2^x = n$은 같은 말

- 위의 식을 잘 보면, $\log n$이란

  (가) 2 의 몇 승이 n이 되느냐의 답

  (나) n을 표현하는 데 몇 비트가 필요한가의 답

  (다) 1 로 시작해서 계속 두 배를 할 때 몇 번 하면 n이 되느냐의 답

  (라) n을 2 로 계속 나눌 때 몇 번 나누면 거의 1 이 되느냐에 대한 답

- $x = \log n$일 때 x와 n을 비교하면 x가 더 작고, n이 커질수록 엄청나게 달라진다

- 100 자리로 표현할 수 있는 10 진수 값은 읽을 수도 없을 정도로 큰 값이다

- 컴퓨터 분야에서 로그의 밑은 항상 2

- 32 비트 컴퓨터의 주소 공간은 2^{32} = 약 40 억개 주소

- $n + \left(\frac{n}{2} + \frac{n}{2}\right) + \left(\frac{n}{4} + \frac{n}{4} + \frac{n}{4} + \frac{n}{4}\right) + \left(\frac{n}{8} + \frac{n}{8} + \cdots\right) + \cdots (1 + 1 + \cdots) = n\log n$ (Why?)

- $n + \frac{n}{2} + \frac{n}{4} + \cdots + 1 \cong 2n$

- 위 두 식의 항의 개수는 $\log n$개 (Why?)

■ 문제들

- 문제 1: 2 진수 표현에서 $\log n$ 비트로 표현할 수 있는 숫자 범위는?

[Solution]

n비트로 표현할 수 있는 숫자는 2^n가지의 값이 있다.

$\log_2 n$ 비트로 표현할 수 있는 숫자는 n가지의 값이 있다.

- 문제 2: 스무고개가 이상적으로 진행된다고 할 때, 맞출 수 있는 답의 종류는 몇 가지인가?

[Solution]

$2^{20} = 1048576$ 가지

- 문제 3: n이 충분히 큰 값일 때 다음 중 어느 값이 더 큰가? 각 쌍에 대해 비교하고 그 이유를 작성하시오.

[Solution]

① $2n$ (<) n^2

$2 < n$ 일 때 $2n < n^2$ 이므로, 2보다 큰 모든 n에 대해 부등호가 항상 성립한다.

② $2^{\frac{n}{2}}$ (<) $\sqrt{3^n}$

$n > 0$ 일 때 $3^{\frac{n}{2}} > 2^{\frac{n}{2}}$ 이 항상 성립하므로, 0보다 큰 모든 n에 대해 부등호가 항상 성립한다.

③ $2^{n \log n}$ (>) $n!$

n > 1일 때, $n^n > n!$ 이 항상 성립한다. 따라서 1보다 큰 모든 n에 대해 부등호가 항상 성립한다.

④ $\log 2^{2n}$ (<) $n\sqrt{n}$

n > 4 일 때 $2n < n^{\frac{3}{2}}$ 이 항상 성립한다. 따라서 4보다 큰 모든 n에 대해 부등호가 항상 성립한다.

- 문제 4: $x = \log_a yz$일 때 x를 2 를 밑으로 하는 로그들로 표현하시오. 단, 로그 함수의 인자는 모두 문자 하나여야 한다.

[Solution]

$$x = \frac{\log_2 y + \log_2 z}{\log_2 a}$$

- 문제 5: 다음 함수들의 역함수를 구하시오

[Solution]

① $f(x) = \log(x-3) - 5$

$10^{f(x)+5} = x - 3$

$x = 10^{f(x)+5} + 3$

$f^{-1}(x) = 10^{x+5} + 3$

② $f(x) = 3\log(x+3) + 1$

$10^{\frac{f(x)-1}{3}} = x + 3$

$x = 10^{\frac{f(x)-1}{3}} - 3$

$f^{-1}(x) = 10^{\frac{x-1}{3}} - 3$

③ $f(x) = 2 \times 3^x - 1$

$x = \log_3(f(x)+1) - \log_3 2$

$f^{-1}(x) = \log_3(x+1) - \log_3 2$

3. 집합과 조합론

■ 집합과 조합론에 대한 약간의 설명

- 두 집합 A와 B에 대해 A가 B의 부분집합임을 증명한다는 것은 A의 임의의 원소가 B에 포함됨을 보이는 것과 같다.

- 예를 들어 모든 4 의 배수는 2 의 배수라는 것을 증명하려면, $4k = 2(2k)$임을 보이면 되는 것이다.

- 두 집합 A와 B가 같다는 것을 증명하기 위해서는 A가 B의 부분집합이고 B가 A의 부분집합임을 증명하면 된다.

- 다음 두 집합이 같다는 것을 상세히 증명해 보자.

$A = \{x | x = 2k + 1, k$는 자연수$\}$, $B = \{x | x = 4k + 1$ 혹은 $x = 4k + 3, k$는 자연수$\}$

- A가 B의 부분집합이다:

  A에 포함되는 임의의 원소 x를 가정.
  $x = 2k + 1$임.
  k가 짝수$(= 2t)$인 경우와 홀수$(= 2t + 1)$인 경우로 나눔.
  짝수인 경우 $x = 2k + 1 = 2(2t) + 1 = 4t + 1$로서, x는 B에 포함됨.
  홀수인 경우 $x = 2k + 1 = 2(2t + 1) + 1 = 4t + 3$로서, x는 B에 포함됨.
  모든 가능한 경우에 x는 B에 포함됨.

- B가 A의 부분집합이다:

  B에 포함되는 임의의 원소 x를 가정.
  $x = 4k + 1$인 경우, $x = 4k + 1 = 2(2k) + 1$로서 x는 A에 포함됨. $x = 4k + 3$인 경우, $x = 4k + 3 = 2(2k + 1) + 1$로서 x는 A에 포함됨.
  모든 가능한 경우에 x는 B에 포함됨.

- 위 두 가지 증명에서 집합 A와 B는 같다.

- 조합론은 경우의 수를 따지는 문제들을 보통 말한다
- 조합은 개수는 C 를 이용하여 표현하기도 하지만 $\binom{5}{2} = 10$과 같은 괄호 표현을 더 많이 쓴다.

■ 연습 문제들

- 문제 1: $\binom{n}{k} + \binom{n}{k-1} = \binom{n+1}{k}$임을 증명하라

[Solution]

Proof)

$$\binom{n}{k} + \binom{n}{k-1} = \frac{n!}{(n-k)!\,k!} + \frac{n!}{(n-(k-1))!\,(k-1)!}$$

$$= n!\left(\frac{1}{(n-k)!\,k!} + \frac{1}{(n-(k-1))!\,(k-1)!}\right)$$

$$= n!\left(\frac{n-(k-1)}{(n-(k-1))!\,k!} + \frac{k}{(n-(k-1))!\,k!}\right)$$

$$= n!\left(\frac{n+1}{(n-(k-1))!\,k!}\right)$$

$$= \frac{(n+1)!}{(n+1-k)!\,k!}$$

$$= \binom{n+1}{k}$$

- 문제 2: 수학적 귀납법으로 $(x+y)^n = \sum_{k=0}^{n} \binom{n}{k} x^{n-k} y^k$ 임을 증명하라

[Solution]

Proof)

$n = 1$ 일 때,

$$(\text{좌변}) = (x+y)^1 = \sum_{k=0}^{1} \binom{1}{k} x^{1-k} y^k = (\text{우변})$$

이므로 성립한다.

$n = t\ (t \geq 1)$ 일 때, 등식이 성립한다고 가정하면,.

$$(x+y)^t = \sum_{k=0}^{t} \binom{t}{k} x^{t-k} y^k$$

$n = t+1$ 일 때,

$$(x+y)^{t+1} = (x+y)^t (x+y) = \left(\sum_{k=0}^{t} \binom{t}{k} x^{t-k} y^k \right) (x+y)$$

$$= \sum_{k=0}^{t} \binom{t}{k} x^{t-k+1} y^k + \sum_{k=0}^{t} \binom{t}{k} x^{t-k} y^{k+1}$$

$$= \left(\binom{t}{0} x^{t+1} + \sum_{k=1}^{t} \binom{t}{k} x^{t-k+1} y^k \right) + \left(\binom{t}{t} y^{t+1} + \sum_{k=0}^{t-1} \binom{t}{k} x^{t-k} y^{k+1} \right)$$

여기서

$$\sum_{k=0}^{t-1} \binom{t}{k} x^{t-k} y^{k+1} = \sum_{k=1}^{t} \binom{t}{k-1} x^{t-(k-1)} y^k \text{ 이고,}$$

이항 계수 성질에 의하여,

$$\binom{t}{k}+\binom{t}{k-1}=\binom{t+1}{k}, \qquad \binom{t}{0}=\binom{t+1}{0}, \qquad \binom{t}{t}=\binom{t+1}{t+1}$$

이므로

$$(x+y)^{t+1}=\binom{t+1}{0}x^{t+1}+\binom{t+1}{k+1}y^{t+1}+\sum_{k=1}^{t}\binom{t}{k}x^{t-k+1}y^{k}+\sum_{k=1}^{t}\binom{t}{k-1}x^{t-k-1}y^{k}$$

$$=\binom{t+1}{0}x^{t+1}+\binom{t+1}{k+1}y^{t+1}+\sum_{k=1}^{t}\left[\binom{t}{k}+\binom{t}{k-1}\right]x^{t-k+1}y^{k}$$

$$=\binom{t+1}{0}x^{t+1}+\binom{t+1}{k+1}y^{t+1}+\sum_{k=1}^{t}\binom{t+1}{k}x^{t-k+1}y^{k}$$

$$=\sum_{k=0}^{t+1}\binom{t+1}{k}x^{t-k+1}y^{k}$$

즉, $n=t+1$ 일 때도 주어진 등식이 성립함을 알 수 있다.

따라서 주어진 등식은 모든 자연수 n 에 대하여 성립한다.

- 문제 3: 위의 결과를 이용해서 n개의 원소를 가진 집합의 가능한 부분집합의 종류는 2^n개임을 증명하라

[Solution]

Proof)

부분 집합의 원소의 개수를 총 n개라고 했을 때,

원소의 개수가 0 개인 부분집합의 수 : $\binom{n}{0}$

원소의 개수가 1 개인 부분집합의 수 : $\binom{n}{1}$

원소의 개수가 2 개인 부분집합의 수 : $\binom{n}{2}$

…

원소의 개수가 n개인 부분집합의 수 : $\binom{n}{n}$

원소의 개수가 n개 일 때, 총 부분집합의 수 S는 $\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{n}$ 이다.

$S = \binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{n} = \sum_{k=0}^{n} \binom{n}{k} 1^{n-k} 1^k = (1+1)^n = 2^n$

이므로 n개의 원소를 가진 집합의 가능한 부분집합의 종류는 2^n개이다.

- 문제 4: 귀류법을 이용하여 $(A-B)\cap(B-A)=\emptyset$임을 증명하라

[Solution]

Proof)

$p \in (A-B)$ 이고 $p \in (B-A)$ 인 원소 p가 존재한다고 가정하자. …①

1. $p \in \mathrm{A}$, $p \notin \mathrm{B}$ 일 경우

$(B-A)$는 집합 B의 부분 집합이다. $p \notin \mathrm{B}$ 이므로 $p \notin (B-A)$ 이다. 이는 가정 ①의 모순이다.

2. $p \notin \mathrm{A}$, $p \in \mathrm{B}$ 일 경우

$(A-B)$는 집합 A의 부분 집합이다. $p \notin \mathrm{A}$ 이므로 $p \notin (A-B)$ 이다. 이는 가정 ①의 모순이다.

3. $p \notin \mathrm{A}$, $p \notin \mathrm{B}$ 일 경우

$(A-B)$와 $(B-A)$는 집합 A, B의 부분 집합이다. $p \notin \mathrm{A}$, $p \notin \mathrm{B}$ 이므로 $p \notin (A-B)$ 이고 $p \notin (B-A)$이다. 이는 가정 ①의 모순이다.

4. $p \in \mathrm{A}$, $p \in \mathrm{B}$ 일 경우

$p \in A \cap B$ 이므로, $p \notin (A-B)$이고 $p \notin (B-A)$ 이다. 이는 가정 ①의 모순이다.

따라서 $p \in (A-B)$ 이고 $p \in (B-A)$ 인 원소 p가 존재한다는 가정은 모순이므로, $p \in (A-B)$ 이고 $p \in (B-A)$ 인 원소 p는 존재하지 않는다. 그러므로 $(A-B)\cap(B-A)=\emptyset$ 이다.

- 문제 5: 두 집합이 다르다는 것은 다음 명제와 동치임을 증명하라. 증명에는 앞에서 설명한 내용과 기본 논리만을 사용해야 한다.

$$\exists x(x \in A \wedge x \notin B) \vee (x \in B \wedge x \notin A)$$

[Solution]

Proof)

임의의 두 집합 A, B가 같다는 것은

$$\begin{aligned} A = B &\Leftrightarrow \forall x : (x \in A \Leftrightarrow x \in B) \\ &\Leftrightarrow \forall x : (x \in A \rightarrow x \in B \Leftrightarrow x \in B \rightarrow x \in A) \\ &\Leftrightarrow \forall x : (x \in A \rightarrow x \in B) \wedge (x \in B \rightarrow x \in A) \end{aligned}$$

이다.

$$\begin{aligned} A \neq B &\Leftrightarrow \sim \big(\forall x : (x \in A \rightarrow x \in B) \wedge (x \in B \rightarrow x \in A)\big) \\ &\Leftrightarrow \exists x : (x \in A \wedge x \notin B) \vee (x \in B \wedge x \notin A) \end{aligned}$$

- 문제 6: 다음이 사실임을 증명하라

$$(A \cup B) \cap (A \cap B)^c = (A - B) \cup (B - A)$$

[Solution]

Proof)

$$(A \cup B) \cap (A \cap B)^c = \{\, p : (p \in A \vee p \in B) \wedge \neg(p \in A \wedge p \in B) \,\}$$

$$(A - B) \cup (B - A) = \{\, p : (p \in A \wedge p \notin B) \vee (p \notin A \vee p \in B) \,\}$$

$$(p \in A \vee p \in B) \wedge \neg(p \in A \wedge p \in B)$$

$$\Leftrightarrow (p \in A \wedge p \notin B) \vee (p \notin A \vee p \in B)$$

를 보이면 된다.

$$\equiv p \in (A \cup B) \cap (A \cap B)^c$$

$$\equiv (p \in A \vee p \in B) \wedge \neg(p \in A \wedge p \in B)$$

$$\equiv (p \in A \vee p \in B) \wedge (p \notin A \vee p \notin B)$$

$$\equiv \Big((p \in A \wedge (p \notin A \vee p \notin B)) \vee (p \in B \wedge (p \notin A \vee p \notin B)) \Big)$$

$$\equiv ((p \in A \wedge p \notin B) \vee (p \notin A \wedge p \in B)$$

$$\equiv p \in (A - B) \cup (B - A)$$

임을 알 수 있다.

또는

1. $p \in \mathrm{A}$, $p \notin \mathrm{B}$ 일 경우
2. $p \notin \mathrm{A}$, $p \in \mathrm{B}$ 일 경우
3. $p \notin \mathrm{A}$, $p \notin \mathrm{B}$ 일 경우

로 가정하여, 증명할 수 있다.

이는 문제 4 번과 유사하게 증명해 나가면 된다.

- 문제 7: $A \oplus B$는 두 집합의 합집합에서 교집합을 뺀 것을 말한다. 다음 식이 항상 성립함을 증명하라.

$$(A \oplus B) \oplus B = A$$

[Solution]

Proof)

이는 쉽게 밴 다이어그램으로 확인할 수 있다.

또 다른 방법으로는 $\forall x[\, x \in (A \oplus B) \oplus B \leftrightarrow x \in A]$ 를 보이는 것이다.

1. $x \in \mathrm{A}, x \notin \mathrm{B}$일 경우

   $x \in \mathrm{A}$이고 $x \notin \mathrm{B}$ 이므로, $x \in (A \cup B)$, $x \notin (A \cap B)$이다.

   따라서 $x \in (A \oplus B)$이다.

   그리고 $x \in (A \oplus B)$이고 $x \notin \mathrm{B}$ 이므로, $x \in (A \oplus B) \cup B$, $x \notin (A \oplus B) \cap B$이다.

   $(A \oplus B) \oplus B = (A \oplus B) \cup B - (A \oplus B) \cap B$ 이므로, $x \in (A \oplus B) \oplus B$ 이다.

   그러므로 $x \in (A \oplus B) \oplus B \longrightarrow x \in A$이다..

2. $x \in \mathrm{A}, x \in \mathrm{B}$일 경우

   $x \in \mathrm{A}$이고 $x \in \mathrm{B}$ 이므로, $x \in (A \cup B)$, $x \in (A \cap B)$이다.

   따라서 $(A \oplus B) = (A \cup B) - (A \cap B)$ 이므로, $x \notin (A \oplus B)$이다.

   그리고 $x \notin (A \oplus B)$이고 $x \in \mathrm{B}$ 이므로, $x \in (A \oplus B) \cup B$, $x \notin (A \oplus B) \cap B$이며,

   $(A \oplus B) \oplus B = (A \oplus B) \cup B - (A \oplus B) \cap B$ 이므로, $x \in (A \oplus B) \oplus B$ 이다.

그러므로 $x \in (A \oplus B) \oplus B \longrightarrow x \in A$이다.

$x \in (A \oplus B) \oplus B \longleftarrow x \in A$는 어떠한 경우라도 성립한다.

그러므로 $x \in (A \oplus B) \oplus B \leftrightarrow x \in A$ 이 성립함을 알 수 있고, $(A \oplus B) \oplus B = A$이다.

- 문제 8: 8×8 체스 판에 말 두개를 놓으려고 한다. 아무 곳에나 놓아도 되지만 한 칸에 두개가 들어가지는 못한다. 가능한 방법은 모두 몇가지인가?

[Solution]

8×8 체스 판 중에서 두 곳을 고르는 조합을 구하는 문제이므로,

$$\binom{64}{2} = 2{,}016$$

모두 2,016개가 존재한다.

- 문제 9: n개의 원소를 가진 집합의 가능한 부분집합의 종류는 2^n개임을 조합론을 이용해 증명하라.

[Solution]

n개의 원소를 가진 집합 S를 $S = \{ a_1, a_2, a_3, \dots , a_n \}$ 이라 하자.
그러면 임의의 원소 a_k ($a_k \in S$)가 부분 집합으로 포함될 경우는
2가지(포함된다, 포함되지 않는다)이므로, 전체 부분집합의 종류는 2^n이 된다.

- 문제 10: 비밀번호를 0 부터 9 까지의 숫자만 가지고 만든다고 하자. 4 개 이상 6 개 이하의 숫자를 쓸 수 있다고 할 때 가능한 비밀번호의 가지수는 얼마인가?

[Solution]

가능한 비밀번호는 숫자가 4 개, 5 개, 6 개가 들어갈 경우이다.

숫자가 4 개 사용될 경우 : $10 \times 9 \times 8 \times 7 = 5{,}040$

숫자가 5 개 사용될 경우 : $10 \times 9 \times 8 \times 7 \times 6 = 30{,}240$

숫자가 6 개 사용될 경우 : $10 \times 9 \times 8 \times 7 \times 6 \times 5 = 151{,}200$

따라서 총 가능한 비밀번호의 가지 수는 $186{,}480$개 이다.

- 문제 11: 원소가 m개인 집합에서 원소가 n개인 집합으로 가는 단사함수의 개수는 몇가지인가?

[Solution]

두 가지 경우로 나눠서 생각할 수 있다.

1. $m \leq n$일 경우

n개의 집합에서 m개를 선택하고, m개의 원소를 순열로 곱한 값이 총 개수이다.

$$\binom{n}{m} \times m!$$

2. $m > n$일 경우

단사 함수는 공역의 각 원소가 정의역의 원소 중 최대 한 원소만을 만족해야 하므로, 성립되지 않는다.

- 문제 12: 52 개의 카드를 이용해서 만들 수 있는 5 개 카드의 조합은 몇가지인가?

[Solution]

52 개의 카드에서 5 개 카드를 선택하는 조합의 경우이므로,
총 $\binom{52}{5} = 2{,}598{,}960$ 가지 있다.

- 문제 13: 52 개의 카드를 이용해서 만들 수 있는 5 개 카드 조합 중 같은 무늬의 카드가 정확히 3 개인 경우는 몇가지인가?

[Solution]

무늬의 종류는 4 가지이고, 이 중 하나를 선택하는 경우는 $\binom{4}{1}$ 가지이다.
그리고 같은 무늬일 때, 서로 다른 숫자를 뽑는 경우는 13 개 중 3 개를 뽑는 경우 $\binom{13}{3}$ 이다.

나머지 2 개의 카드는 무늬가 달라야 하므로, 나머지 3 개에서 2 개를 뽑는 $\binom{3}{2}$ 가지이고, 수는 서로 상관없으므로 $\binom{13}{1} \times \binom{13}{1}$이다.

따라서 이들을 모두 곱하면,
$\left(\binom{4}{1} \times \binom{13}{3}\right) \times \left(\binom{3}{2} \times \binom{13}{1} \times \binom{13}{1}\right) = 580{,}008$ 가지가 된다.

- 문제 14: $x+y+z=100$의 자연수 해는 몇가지인가?

[Solution]

100개의 공을 칸막이 2 개를 포함하여 x,y,z 칸에 넣는 것이랑 같다. 이때 x,y,z가 자연수 해가 되어야 하므로, 공을 미리 하나씩 넣어 최소한 1 보다 같거나 큰 수로 만든다. 그러면 97 개의 공을 x,y,z칸에 넣으면 된다.

$$\frac{(97+2)!}{97!\,2!} = 4{,}851$$

총 4,851가지의 자연수 해 쌍이 존재한다.

- 문제 15: (포함 배제 원리) 5 개의 원소를 가진 집합에서 3 개의 원소를 가진 집합으로 가는 전사함수는 몇가지가 있는가?

[Solution]

5 개의 원소를 가진 집합에서 3 개의 원소로 가진 집합으로 가는 총 경우의 수는 $3^5(=243)$이다.

1. 3 개의 원소 중 2 개의 원소로만 가능 경우 : $\binom{3}{2} \times 2^5(=96)$
   3 개의 원소 중 2 개의 원소로 갈 때, 1 개의 원소로만 갈 경우 : $\binom{3}{2} \times 2\,(=6)$

2. 3 개의 원소 중 1 개의 원소로만 가능 경우 : $\binom{3}{1} \times 1^5(=3)$

따라서 전사함수로 가는 총 경우의 수는 $243-96+6-3=150$가지이다.

- 문제 16: 52 개 카드에서 5 개 카드 조합을 만들 때, 숫자가 같은 카드가 한 쌍도 없는 경우는 몇가지인가?

[Solution]

52 개의 카드에서 5 개 카드를 선택하는 총 경우의 수는 $\binom{52}{5}\,(=2{,}598{,}960)$이다.

1. 2 장의 카드의 숫자가 같은 경우

   (1) 2 2 1 1 1

   13 개의 수에서 하나를 선택하고, 4 가지색 중에서 2 가지 색을 선택하다. 나머지 3 장의 카드는 숫자가 달라야 하므로 12 개의 수에서 3 개를 선택할 수 있고, 각 카드는 숫자가 다르므로 4 가지색을 모두 가질 수 있다.

$$\binom{13}{1}\binom{4}{2}\binom{12}{3}\binom{4}{1}^{3} = 1{,}098{,}240$$

   (2) 2 2 1

   13 개의 수에서 하나를 선택하고, 4 가지색 중에서 2 가지 색을 선택하다. 나머지 3 장의 카드는 2 장이 같고 한 장이 달라야 하므로 12 개의 수에서 2 개를 선택할 수 있고, 각 카드는 숫자가 다르므로 4 가지색을 모두 가질 수 있다.

$$\binom{13}{1}\binom{4}{2}\binom{12}{1}\binom{4}{2}\binom{11}{1}\binom{4}{1} = 247{,}104$$

   (2) 2 3

   13 개의 수에서 하나를 선택하고, 4 가지색 중에서 2 가지 색을 선택하다. 나머지 3 장의 카드는 숫자가 달라야 하므로 12 개의 수에서 3 개를 선택할 수 있고, 각 카드는 숫자가 다르므로 4 가지색을 모두 가질 수 있다.

$$\binom{13}{1}\binom{4}{2}\binom{12}{1}\binom{4}{3} = 3{,}744$$

2. 3 장의 카드의 숫자가 같은 경우(3 1 1)

   : 13 개의 수에서 하나를 선택하고, 4 가지색 중에서 3 가지 색을 선택하다. 나머지 2 장의 카드는 숫자가 달라야 하므로 12 개의 수에서 2 개를 선택할

수 있고, 각 카드는 숫자가 다르므로 4 가지색을 모두 가질 수 있다.

$$\binom{13}{1}\binom{4}{3}\binom{12}{2}\binom{4}{1}^2 = 54{,}912$$

3. 4 장의 카드의 숫자가 같은 경우(4 1)

: 13 개의 수에서 하나를 선택하고, 4 가지색 중에서 4 가지 색을 선택하다. 나머지 1 장의 카드는 숫자가 달라야 하므로 12 개의 수에서 1 개를 선택할 수 있고, 하나 카드는 4 가지색을 모두 가질 수 있다.

$$\binom{13}{1}\binom{4}{4}\binom{12}{1}\binom{4}{1}^1 = 624$$

따라서 52 개의 카드에서 5 개 카드를 조합할 때, 숫자가 같은 카드가 한 쌍도 없는 경우는 $2{,}598{,}960 - 1{,}098{,}240 - 247{,}104 - 3{,}744 - 54{,}912 - 624 = 1{,}194{,}336$개 이다.

- 문제 17: n개의 원소를 가진 배열에서 연속된 구간을 잡으려고 한다. 잡을 수 있는 가능한 구간은 몇가지인가? 단 구간의 크기는 1 이상이다.

[Solution]

1. 처음과 끝을 같게 잡는 경우 : n
2. 처음과 끝을 다르게 잡는 경우 : $\binom{n}{2}$

총 가능한 구간은 $n + \binom{n}{2}$ 이다.

4. 기초 수식

■ 약간의 설명

- 알고리즘의 시간 복잡도를 표현할 수 있는 다양한 수식들이 존재한다.

- 풀이법을 익혀 두어야 알고리즘의 시간 복잡도를 계산할 수 있고,
  알고리즘이 시간이 얼마나 걸릴 지 예측할 수 있다.

■ 연습 문제들: 다음 재귀식들을 O() notation 수준으로 풀어라.

- 문제 1: $T(n) = T(n-1) + 1$

[Solution]

$$T(n) = T(n-1) + 1$$
$$= T(n-2) + 1 + 1$$
$$= T(1) + 1 + \cdots + 1$$

$$T(n) = \boldsymbol{O(n)}$$

- 문제 2: $T(n) = T(n-1) + n$

[Solution]

$$T(n) = T(n-1) + n$$
$$= T(n-2) + (n-1) + n$$
$$= 1 + \cdots (n-1) + n$$
$$= \frac{n(n-1)}{2}$$
$$\therefore T(n) = \boldsymbol{O(n^2)}$$

- 문제 3: $T(n) = T(n-1) + \log n$

[Solution]

$$T(n) = \mathrm{T}(1) + \log n + \log n - 1 + \log n - 2 + \cdots + \log 2$$
$$\leq \mathrm{T}(1) + \log n + \log n + \log n + \cdots + \log n$$
$$\leq \mathrm{T}(1) + n \log n$$
$$\therefore T(n) = \boldsymbol{O(n \log n)}$$

- 문제 4: $T(n) = T\left(\frac{n}{2}\right) + 1$

[Solution]

$$T(n) = 1 + T\left(\frac{n}{2}\right)$$

$$= 1 + \left(1 + T\left(\frac{n}{4}\right)\right)$$

$$= 1 + 1 + 1 + \cdots + 1 \quad (\log n \text{ 번 반복})$$

$$\therefore T(n) = \boldsymbol{O(\log n)}$$

- 문제 5: $T(n) = T\left(\frac{n}{2}\right) + n$

[Solution]

$$T(n) = n + T\left(\frac{n}{2}\right)$$

$$= n + \left(\frac{n}{2} + T\left(\frac{n}{4}\right)\right)$$

$$= n + \frac{n}{2} + \frac{n}{2^2} + \cdots + \frac{n}{2^{\log_2 n}}$$

$$= n\frac{1 - \frac{1}{2}^{log_2 n}}{1 - \frac{1}{2}}$$

$$= 2n - 2$$

$$\therefore T(n) = \boldsymbol{O(n)}$$

- 문제 6: $T(n) = 2T\left(\frac{n}{2}\right) + n$

[Solution]

$$T(n) = n + 2T\left(\frac{n}{2}\right)$$

$$= n + 2\left(\frac{n}{2} + 2 \cdot T\left(\frac{n}{4}\right)\right)$$

$$= n + 2 \cdot \frac{n}{2} + 2^2 \cdot \frac{n}{2^2} + \cdots + 2^{\log_2 n} \cdot \frac{n}{2^{\log_2 n}}$$

$$= \mathrm{n}\log_2 n$$

$$\therefore T(n) = \boldsymbol{O(\mathbf{n}\log \boldsymbol{n})}$$

- 문제 7: $T(n) = 3T\left(\frac{n}{2}\right) + n$

[Solution]

$$T(n) = n + 3T\left(\frac{n}{2}\right)$$

$$= n + 3\left(\frac{n}{2} + 3 \cdot T\left(\frac{n}{4}\right)\right)$$

$$= n + 3 \cdot \frac{n}{2} + 3^2 \cdot \frac{n}{2^2} + \cdots + 3^{\log_2 n} \cdot \frac{n}{2^{\log_2 n}}$$

$$= n\frac{\frac{3}{2}^{log_2 n} - 1}{\frac{3}{2} - 1}$$

$$= 2 \cdot \mathrm{n} \cdot n^{\log_{2\frac{3}{2}}} - 2 \cdot \mathrm{n}$$

$$\therefore T(n) = \boldsymbol{O(n^{\log_2 3})}$$

- 문제 8: $T(n) = T(n-1) + \frac{1}{n}$

[Solution]

$$T(n) = T(n-1) + \frac{1}{n}$$

$$= T(n-2) + \frac{1}{n-1} + \frac{1}{n}$$

$$\cdots$$

$$= T(1) + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \frac{1}{5} + \frac{1}{6} + \frac{1}{7} \cdots \frac{1}{n-2} + \frac{1}{n-1} + \frac{1}{n}$$

$$< T(1) + (\frac{1}{2} + \frac{1}{2}) + (\frac{1}{4} + \frac{1}{4} + \frac{1}{4} + \frac{1}{4}) \cdots (\frac{1}{n} + \cdots + \frac{1}{n} + \frac{1}{n})$$

$$= T(1) + 1 + 1 + \cdots + 1 = T(1) + \boldsymbol{O(\log n)}$$

$$\therefore T(n) = \boldsymbol{O(\log n)}$$

- 문제 9: $T(n) = T(n/2) + T(n/4) + T(n/6) + T(n/12) + 1$

[Solution]

$T(n) \leq n$으로 추측해보자. ***(Guess and Induction)***

$$T(n) = T(n/2) + T(n/4) + T(n/6) + T(n/12) + 1$$

$$\leq \frac{n}{2} + \frac{n}{4} + \frac{n}{6} + \frac{n}{12} + \mathbf{1}$$

$$\leq n + 1$$

$$\therefore T(n) = \boldsymbol{O(n)}$$

- 문제 10: $T(n) = \sqrt{n} \cdot T(\sqrt{n}) + n$

[Solution]

$$T(n) = n + n^{\frac{1}{2}} \cdot T\left(n^{\frac{1}{2}}\right)$$

$$= \mathrm{n} + (\mathrm{n}^{\frac{1}{2}} \cdot \mathrm{n}^{\frac{1}{2}} + \mathrm{n}^{\frac{1}{4}} \cdot \mathrm{T}\left(\mathrm{n}^{\frac{1}{4}}\right))$$

$$= n + n + \cdots + \mathrm{n}$$

n 이 몇번 나오는지 세기 위해 함수의 호출인자를 살펴보면, $n^{\frac{1}{2^k}}$임을 알 수 있다.(k 는 함수의 호출 횟수)

$n = 2^{\log_2 n}$이므로,

$$n^{\frac{1}{2^k}} = 2^{\frac{(\log_2 n)}{2^k}}$$

$n^{\frac{1}{2^k}} = 2$에서 함수가 끝난다고 할 때,

$$2^{\frac{(\log_2 n)}{2^k}} = 2$$

$$\frac{(\log_2 n)}{2^k} = 1$$

$$\log_2 n = 2^k, \mathrm{k} = \log_2(\log_2 n)$$

$\therefore T(n) = \boldsymbol{O}(n\log_2(\log_2 n))$

5. 재귀

■ 약간의 설명

- 재귀란 자기 자신을 호출하는 함수, 그럼 끝날 수가 있는가?

- 함수는 입력이 있으며, 자기 자신의 입력과 동일한 입력으로 자기 자신을 호출하면 당연히 끝나지 않음

- 하지만, 다른 입력으로 호출하면 끝날 수 있음

```
int abc(int x)     // 이 함수는 안 끝남
{
    return abc(x);
}

int sum(int x)     // 이 함수는 끝남. 결과 값은?
{
    if (x <= 0) return 0;
    return x + sum(x-1);
}
```

- 함수란 어떤 문제를 해결하는 방법을 코딩한 것

- 함수가 어떤 문제의 단 한 케이스만을 해결하는 것이 아님

- 제대로 코딩 된 것이라면 해결하는 문제의 **모든 케이스들을** 해결해야 함

- 수학적 귀납법 증명 사용 가능

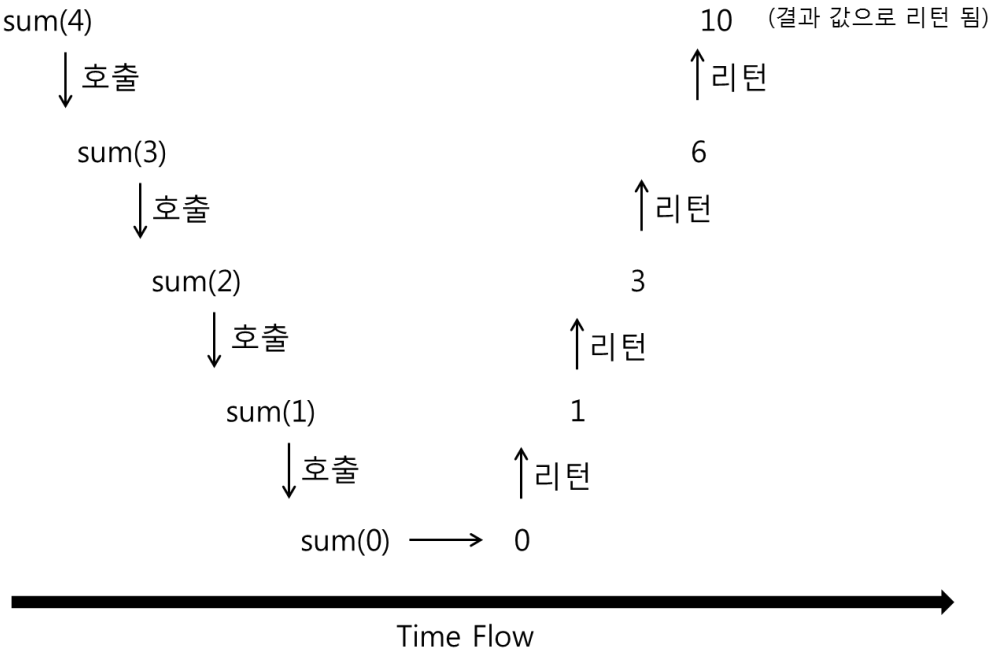
  (가) n이 0 일 때 문제를 풀 수 있음

  (나) $n-1$에서 문제를 풀 수 있으면 n에서도 문제를 풀 수 있다

- 위 두 가지가 사실이면 모든 가능한 n에 대해 문제를 풀 수 있다는 것이 사실

- 위 박스의 함수 sum()을 보면 정확히 위의 두 가지를 코딩한 것임
- 따라서, sum() 함수는 문제를 해결한다는 것을 증명할 수 있음
- 방금 보인 증명은 high-level 증명이며, 상세한 증명은 과정의 첫날에 본 적이 있음

- 어떤 미스터리한 이유로 문제가 해결되는 것은 아니고, 실제로 프로그램을 돌리면 필요한 계산이 다 일어남. 순차적인 코드에서 일어나는 계산과 완전히 동일. 다만 표현하는 방법이 달라진 것



- 다르게 생각하는 방법: 어떤 문제를 해결하려다 **부분 문제**를 만났는데, 원래 해결하려던 입력 케이스와 **동일한 문제**에 속하지만 "크기가 더 작은" 입력 케이스를 해결하는 것이 그 부분 문제였다!
- 즉, 부분 문제가 동일한 문제인 경우!

■ 연습 문제들: 다음 문제들을 푸는 재귀 알고리즘을 수도코드로 작성하고, 정확성 증명 및 시간 복잡도 계산을 수행하라

- 문제 1: 피보나치 수열: $F(n) = F(n-1) + F(n-2),\ F(1) = F(2) = 1$

[Solution]

```
fibo(n){
        if(n <= 2 ) return 1
        return fibo(n-1) + fibo(n-2)
}
```

Proof)

가정 : fibo(n)을 호출하면 F(n)이 return 됨을 알 수 있다.

(1) $n = 1$, n=2 일 때,

fibo(1) = 1 이고 fibo(2)=1 인데, 이는 F(1) = 1, F(2) = 1 이므로 위의 소스코드는 성립한다.

(2) $n = k$일 때,

fibo(1) … fibo(k-1)이 F(1) .. F(k-1)과 같다고 가정하자. 이때 F(k) = F(k-1)+ F(k-2) 이므로, fibo(k) = fibo(k-1)+fibo(k-2) = F(k)가 성립한다.

따라서 fibo(n)을 호출하면 F(n)이 return 됨을 알 수 있다.

Time Complexity)

$$T(n) = T(n-1) + T(n-2) + 1$$
$$< 2\,T(n-1) + 1$$
$$T(n) = O(2^n)$$

- 문제 2: Merge Sort, 크기 n인 배열을 입력으로 받아,

배열을 절반으로 두개로 나눈 후,

각 작은 배열을 재귀적으로 정렬하고,

그 결과를 Merge 한다.

[Solution]

```
mergesort(array, buffer, start, end){
        if(start == end) return
        mid = (start+end)/2
        i0 = start
        i1 = mid+1
        mergesort(array, buffer, start, mid)
        mergesort(array, buffer, mid+1, end)

        for(i from start to end){
                if(i1 > end || i0 <= mid && array[i0] > array[i1])
                        buffer[i] = array[i0++]
                else
                        buffer[i] = array[i1++]
        }
        copy buffer to array
}
```

Proof)

mergesort 함수는 주어진 배열을 소팅한다.

- k = start – end 라고 할 때, k=1 및 k=2 일 때 Merge Sort 가 배열을 소팅함을 알 수 있다.
- k=1 , k=2, … ,k=n 에서 Merge Sort 가 배열을 소팅한다고 가정하면 k=n+1 일 때 Merge Sort 가 배열을 소팅함을 코드를 확인해서 증명할 수 있다. (Merge 알고리즘의 정확성을 사용함)
- 따라서 모든 범위 k 에서 Merge Sort 가 성립함을 알 수 있다.

$$T(n) = n + 2T\left(\frac{n}{2}\right)$$

$$= n + 2\left(\frac{n}{2} + 2 \cdot T\left(\frac{n}{4}\right)\right)$$

$$= n + 2 \cdot \frac{n}{2} + 2^2 \cdot \frac{n}{2^2} + \cdots + 2^{\log_2 n} \cdot \frac{n}{2^{\log_2 n}}$$

$$= \mathrm{n} \log_2 n$$

$$\therefore T(n) = \boldsymbol{O}(\mathbf{n}\log \boldsymbol{n})$$

- 문제 3: 다음 소팅 알고리즘이 실제로 소팅에 항상 성공한다는 것을 증명하라.

```
Stupid (A[0..n-1])
{
    if n=2 and A[0] > A[1]
        then swap A[0] and A[1]
    else
        m = ceiling(2n/3)
        Stupid(A[0..m-1])
        Stupid(A[n-m..n-1])
        Stupid(A[0..m-1])

}
```

[Solution]

Proof)

배열 A[0..n-1]을 [0,n-m-1], [n-m,m-1], [m,n-1]의 3 구간으로 나눌 수 있다. 각각의 구간을 A,B,C 구간이라고 할 때, 위의 소팅 방법은

.. ①[A B], ②[B C], ③[A B]의 순서로 소팅이 됨을 알 수 있다.

기저 조건에 대해 증명하고, 원소의 개수가 2 부터 n-1 까지의 소팅이 성립한다고 가정할 때 개수가 n 일때의 소팅이 성립함을 보이는 귀납적 방법으로 위의 stupid 소팅법을 증명할 수 있다.

첫째,

```
if n=2 and A[0] > A[1]
    then swap A[0] and A[1]
```

위의 구문에 의해, n=2 일 때는 A[0]과 A[1]이 소팅이 됨을 알 수 있다.

둘째,

어떤 원소 x 는 최종 소팅의 결과에서 C 구간에 있을 원소라고 하자. 배열 전체에서 x 보다 작은 원소의 개수를 a 라고 하고, [A B]구간에서 x 보다 작은 원소의 개수를 b 라고 하자. a-b 는 C 구간의 원소의 개수보다 클 수 없으므로, a-b <= 1/3n 이다.

원소의 개수 n=2 부터 n-1 까지 소팅이 된다고 가정할 때, 소팅 후 C 의 위치에 있어야 하는 어떤 원소 x 가 ①연산 후 A 에 위치하는 경우에만 소팅이 재대로 되지 않고, 그러한 x 가 없는 경우 항상 소팅이 재대로 된다는 것을 경우를 따져보면 원소 n 개에 대한 소팅이 성립함을 알 수 있다.

원소 x 가 소팅 후 C 의 위치에 있어야 하기 때문에, 배열 전체에서 x 보다 작은 원소의 개수 a 는, (a) >= 2/3*n 임을 알 수 있다.

③연산 이후에 C 의 위치에 있기 위해서는 ①연산 이후에 B 의 위치에 있어야 하는데, ①연산 이후에 B 의 위치가 아닌 A 의 위치에 있다고 가정한다면, 이 원소 x 는 C 의 위치로 갈 수 없다고 할 수 있다.

이 때 입력 상태의 [A B] 범위에서 x 보다 작은 원소의 개수 b 는 (b) < 1/3 * n 이다. (왜냐하면 소팅 이후 A 에 있었기 때문에) 따라서 (a) - (b) > 1/3*n 인데, 앞서 증명한 (a) - (b) <= 1/3 * n 와 모순이 되므로, 그러한 x 는 없다는 것을 알 수 있다.

- 문제 4: 위의 소팅 알고리즘에서 수행하는 Swap 의 횟수는 최대 몇번인가?

[Solution]

$$T(n) = 1 + 3T\left(\frac{n}{\frac{3}{2}}\right)$$

$$= 1 + 3 + 3^2 \cdot T\left(\frac{n}{(\frac{3}{2})^2}\right)$$

$$= 1 + 3 + 3^2 + \cdots + 3^{\log_{\frac{3}{2}} n}$$

$$= \frac{3^{\log_{\frac{3}{2}} n} - 1}{3 - 1}$$

$$= \frac{1}{2} \cdot 3^{\log_{\frac{3}{2}} n} - \frac{1}{2}$$

$$\therefore T(n) = O\left(n^{\log_{\frac{3}{2}} 3}\right) \approx O(n^{2.7})$$

최대 $n^{2.7}$번 Swap 할 수 있다.

- 문제 5: 어떤 배열 A[1..n]에 (음수 포함) 정수 값이 증가하는 순서로 저장되어 있다. A[i]=i 가 되는 인덱스 i 가 존재하는 지 찾는 알고리즘을 수도코드 수준으로 작성하고 정확성 증명 및 시간 복잡도 계산을 수행하라. 동일한 문제이지만, 저장된 값이 자연수로 제한되면 어떻게 풀 수 있는가?

[Solution]

Pseudo Code)

저장된 값이 정수일 경우

```
BinarySearch(array,left,right){
      A[n];
      BinarySearch(array,left, right)
}
main(){
       BinarySearch(A[i]-i,1,n)
}
```

Proof)

- A[i]-i 는 감소하지 않는 수열이므로, 정렬되어있다..
- Binary search 의 정확성을 가정하자. 탐색하고자 하는 배열을 A[i]-i 라고 두고 찾고자 하는 값을 0 이라고 한다면, Binary search 를 통해 원하는 값이 있는지 찾을 수 있다.

Proof of binary search)

- 배열 A[i]는 오름차순의 순서대로 정렬되어있다고 가정한다.
- x 는 찾고자 하는 값이다.

1-1 A[left] < x 이고 A[right] > x 를 만족하면, [left,right]의 범위에서 A[i]=x 가 있는지 확인 할 수 있다.

1-2 A[left] > x 이고 A[right] > x 일 경우 [left,right]의 범위에서 답이 없다.

1-3 A[left] < x 이고 A[right] < x 일 경우 [left,right]의 범위에서 답이 없다.

- $\text{middle} = \frac{\text{left}+\text{right}}{2}$ 라고 하면,

2-1 A[middle] < x 일 경우, [left,middle]의 범위에서는 답이 없으므로(1-3), (middle,right]의 범위에서 답이 있는지 확인하는 것은 [left,right]에서 답이 있는지 확인하는 것과 같다.

2-2 A[middle] > x 일 경우, [middle,right]의 범위에서는 답이 없으므로(1-2), [left,middle)의 범위에서 답이 있는지 확인하는 것은 [left,right]에서 답이 있는지 확인하는 것과 같다.

2-3 A[middle] = x 일 경우, 답이다.

- 모든 k=left-right 에 대해,
  1. k=1 일 때 [left,right]의 범위에서 답인지 아닌지 알 수 있다.**(2-3)**
  2. k=1 .. k=n-1 의 범위에서 답이 있는지 없는지 알 수 있으면 k=n 에서 답이 있는지 없는지 알 수 있다. **(2-1,2-2)**

- 귀납적 방법에 의해, binary search 가 성립함을 알 수 있다.

- 문제 6: 루트 있는 트리를 입력으로 받아 아래와 같이 출력하는 알고리즘을 작성하라. 트리의 각 노드에는 1,000 미만의 자연수가 저장되어 있다. 트리의 노드 연결 관계는 다음과 같이 표현해야 한다. 아래 출력에서 루트에는 자식이 3 개 있고 그 자식들 중 하나는 더 이상 자식이 없는 것임을 알 수 있을 것이다.

```
[030]--+--[054]-----[001]
       +--[002]
       L--[045]-----[123]
```

[Solution]

Pseudo Code)

```
tree[][];
int current_line;
make_tree(int current_tree, int depth){
    if( child_maxnumber == 0){
            print "₩n"
            return 0
    }
    for(current_child in child_maxnumber){
            if( current_child != start)
                    for(indent_index in depth)
                            print "₩t"
            if( child_maxnumber == 1)
                    print "---"
            else if( current_child == child_maxnumber)
                    print "--L"
            else
                    print "--+"
            print "--[" + tree[current_tree][current_child] + "]";
            make_tree(current_child, depth+1)
    }
}

main(){
    scan tree[][], root
    make_tree(root,0)
}
```

- 문제 7: (어려움) 무한한 크기의 물통이 3 개 있다. 초기에 각 물통에는 자연수 리터 만큼의 물이 들어 있다. 가능한 작업은 두개의 물통을 잡아서 그 중 많거나 같은 양의 물이 들어 있는 곳에서 작은 쪽으로 물을 부어서 작은 쪽의 물의 양을 **두배**로 만드는 것이다. 즉, 4 리터, 3 리터를 잡았다면 1 리터, 6 리터가 될 것이다. 입력으로 초기 물의 양을 받아서 한 물통에 들어 있는 물의 양을 0 리터로 만들고 싶다. (실행 시간이 많이 걸려도 좋으니) 그렇게 만드는 과정을 계산하는 알고리즘을 작성하라.

[Solution]

Pseudo Code)

```
bottle[3]
reduce(bottle[3]){
    sort, bottle
    if bottle[2] is zero, return
    p = bottle[1]/bottle[2]
    for (index in (1 << index) <= p){
            if (p & (1 << index) )
                    bottle[1] -= bottle[2], bottle[2] *= 2
            else
                    bottle[0] -= bottle[2], bottle[2] *= 2
    }
    reduce(bottle[3])
}

main(){
    reduce(bottle[3])
}
```

Algorithm)

- 물통 3 개를 각각 1,2,3 번이라고 하고, 각각의 물통의 양을 a,b,c(a≤b≤c)라고 한다.

- b=q*a+r, 0≤r<a 를 만족하는 q 가 있을 때, q 의 2 진수를 $q_k q_{k-1} \ldots q_0$ 라고 한다.

- round i=0 to k 까지 loop
  if(q_i=0) 3 번 물통에서 1 번 물통으로 물을 붓는다.
  (1 번을 두배, 3 번은 1 번만큼 감소)
  else 2 번 물통에서 1 번 물통으로 물을 붓는다.
  (1 번을 두배, 2 번은 1 번의 양만큼 감소)

- 모든 round 이후에 남아있는 물의 양을 계산해 보면,
  1 번물통의 양 : 2^{k+1} a
  2 번물통의 양 : r
  3 번물통의 양 : 양수 (∵b≤c)

- r 이 0 이 될 때까지 위 과정을 반복한다.

- ***r<a 이므로 2 번 물통의 양은 round 이전의 가장 적은 물통의 양보다 항상 적다. 따라서 위의 과정을 반복하면 언젠가 물통에 들어있는 물의 양은 0 리터가 된다.***

6. 동적 프로그래밍

■ 약간의 설명

- 간단하게 설명하면 재귀 함수에서 동일한 입력의 함수 호출이 반복적으로 일어날 때 그 결과 값을 저장해 두고 불러 쓰는 것이다. (Memoization)
- 최초 입력에서 파생되는 모든 가능한 입력에 대한 답을 모두 저장할 수 있는 메모리가 있어야 한다.
- 단순히 재귀에서 저장된 값을 찾아보는 것으로도 가능하지만, 결과 값을 순서를 정해서 계산할 수도 있다. (Dynamic Programming)

■ 연습 문제들: 다음 문제들을 푸는 동적 프로그래밍 알고리즘을 수도코드로 작성하고, 정확성 증명 및 시간 복잡도 계산을 수행하라

- 문제 1: Memoization 피보나치 수열:

$$F(n) = F(n-1) + F(n-2),\ F(1) = F(2) = 1$$

(힌트: 계산되는 값이 n 가지 밖에 없으므로 이 값들을 저장할 수 있는 배열을 만들어 두고 재귀 호출에 들어가기 전에 값이 있는 지 확인하는 방법)

[Solution]

Pseudo Code)

```
Fibonacci(n)
{
    if n == 0 or n == 1
        then return n

    if memoization[n] != null
        then return memoization[n]

    memoization[n] = Fibonacci(n-1) + Fibonacci(n-2)

    return memoization[n]
}
```

Correctness)

피보나치 수는 아래와 같은 점화식으로 정의되어 있다.

$$F_n := \begin{cases} 0 & if\ n = 0 \\ 1 & if\ n = 1 \\ F_{n-1} + F_{n-2} & if\ n > 1 \end{cases}$$

위의 코드에서 n이 0 또는 1 일 경우, Base case 로 처리가 된다. n의 값이 0 또는 1 이외의 값이 들어올 경우 $\text{Fibonacci}(n-1), \text{Fibonacci}(n-2)$의 호출을 통해 그 값을 계산하고 이를 저장하는 것을 확인할 수 있다.

Time complexity)

$O(n)$

- 문제 2: Dynamic Programming 피보나치 수열: $F(n) = F(n-1) + F(n-2)$

(힌트: 작은 값부터 순서대로 계산한다)

[Solution]

Pseudo Code)

```
Fibonacci(n)
{
    F[0] ← 0
    F[1] ← 1

    for i ← 2,  i ≤ n,  i ← i + 1
        F[i] = F[i-1] + F[i-2]

    return F[n]
}
```

Correctness)

모든 자연수 n에 대하여 $F(n)$이 성립하는 것을 증명하기 위해, 수학적 귀납법을 이용한다.

(3) $n = 1$일 때,

$F(1) = 1$ 이므로 성립한다.

(4) $n = k$일 때,

$F(k)$가 성립한다고 가정하자.

즉 $F(k) = F(k-1) + F(k-2)$ 는 성립한다고 가정하자.

양변에 $F(k-1)$ 을 더하면,

$$\begin{aligned} F(k) + F(k-1) &= F(k-1) + F(k-2) + F(k-1) \\ &= F(k-1) + F(k) \\ &= F(k+1) \end{aligned}$$

이므로 모든 자연수 n에 대하여 성립함을 알 수 있다.

Time complexity)

$O(n)$

질문: 실제로 실행시키면 세 버전 중 어느 것이 가장 빠를 것으로 예상되는가?

[Solution]

```
Fibonacci(n)
{
    F[0] ← 0
    F[1] ← 1

    for i ← 2,  i ≤ n,  i ← i + 1
        F[i] = F[i-1] + F[i-2]

    return F[n]
}
```

위와 같은 방법이 가장 빠를 것이라 예상되며, 그 이유는 재귀 함수의 호출 시간, 재귀 함수의 호출 횟수 등에서 차이가 날 것이기 때문이다.

- 문제 3: 행렬 곱하기, n개의 행렬을 곱하려고 한다. 크기가 $a \times b$인 행렬과 크기가 $b \times c$인 행렬을 곱하는 데 드는 계산량은 $a \times b \times c$라고 한다. n개의 행렬들을 곱하는데 필요한 계산량을 최소화 하는 순서를 찾는 알고리즘을 작성하라. 행렬들의 크기는 다르고, 입력으로 주어진다고 가정하라. 물론 곱하기가 가능한 크기들만 주어진다.

[Solution]

n개의 행렬을 곱하는 데 드는 계산량은 행렬의 곱을 구하기 위해 수행된 곱셈 수이다. 이는 행렬 곱셈의 결합 법칙을 성질을 잘 이용해야 한다.

예를 들어,

각각 행렬의 차원이 $(100 \times 1), (1 \times 100), (100 \times 1), (1 \times 100)$로 다른 M_1, M_2, M_3, M_4 가 있다고 가정하자.

① $M_1(M_2(M_3M_4)) = 30{,}000$

② $M_1((M_2M_3)M_4) = 10{,}200$

③ $(M_1(M_2M_3))M_4 = 10{,}200$

④ $((M_1M_2)M_3)M_4 = 30{,}000$

⑤ $(M_1M_2)(M_3M_4) = 102{,}000$

위와 같이 곱셈의 순서와 방법에 따라서 연산의 횟수가 변화할 수 있다.
즉, 두 개의 행렬이 하나의 새로운 행렬도 대체되면 다시 n-1 개의 행렬을 곱하는 문제로 변환된다.
또한 중복되어 계산되는 부분은 Memoization 기법을 통해 줄일 수 있다.

따라서 점화식

$$C(i,j) = min\ \{C(i,k) + C(k+1,j) + D(i-1) \times D(k) \times D(j)\},\ \ i \le k \le j-1\ )$$

로 정의할 수 있다.

- 문제 4: (약간 어려움) 배열에 정수(음수 포함)들이 저장되어 있다. 연속인 구간들 중 그 합이 가장 큰 구간을 찾는 알고리즘을 작성하라.

[Solution]

아래와 같이 배열에 임의의 정수들이 포함되어 있다고 가정하자.

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

각 연속인 구간에서 그 합이 가장 큰 구간을 찾기 위해서는 현재 위치 바로 이전까지의 구간들의 합의 크기에 따라 달라지게 된다.

배열의 인덱스를 1부터 n까지라 하고 배열 i위치에 적힌 정수를 A[i]라고 하였을 때, 임의의 위치 i위치에서의 최댓값 C[i]는

$$C[1] = A[1] \quad , \quad i = 1$$

$$C[i] = max\ \{\, C[i-1] + A[i],\ A[i] \,\}\ ,\ (2 \le i \le n)$$

로 정의할 수 있다.

위의 C[i]는 아래와 같이

5	6	2	4	3	-2	-2	8	5	11

로 채워지게 될 것이다.

- 문제 5: (어려움) 배열에 정수(음수 포함)들이 저장되어 있다. 배열의 일부 값들을 골라서 배열에 있는 순서대로 보면 증가하는 순서가 될 수 있다. 이러한 것들 중 가장 긴 것을 찾는 알고리즘을 작성하라.

[Solution]

이 문제는 DP 의 대표적인 문제로 LCS(Longest increasing subsequence, 최장 증가 수열) 문제이다. 단순히 예를 들면

10	20	40	30	70	50	60

이라는 배열이 있었을 때, 여기서 가장 긴 것을 찾아보면

10	**20**	40	**30**	70	**50**	**60**

(10, 20, 30, 50, 60) 순서 길이 5 가 가장 긴 것을 알 수 있다.
(10, 20, 40, 50, 60) 또한 가능하다.

간단히 말해서, 앞에서부터 뒤로 숫자를 선택해 나갈 때, 증가하는 순서로 **최장 증가 수열**을 찾는 것이다. 감소하는 순서로 찾게 되면 최장 감소 수열이 될 것이다.

각 위치에서 자신을 포함한 최대 증가 수열의 길이를 저장하는 점화식을 세우면 쉽게 해결할 수 있다. A[i]를 배열에 저장된 수, D[i]를 i를 포함하는 최대 증가 수열의 길이라고 했을 때,

$$D[i] = max\ \{ D[k], (1 \le k \le i-1 \}, \quad if\ A[k] \le A[i]$$

가 되고, $D[i], (1 \le i \le k)$ 중에 가장 큰 값이 정답이 된다.

위의 예의 D[i]의 값들은 아래와 같이 저장이 된다.

1	2	3	3	4	4	5

7. 조합론 프로그래밍 과제

- 과제 1: 52 장의 카드에서 만들 수 있는 페어가 정확히 하나만 있는 5 장 조합을 모두 출력하는 프로그램을 작성하라. 출력이 너무 많으면 카드 수를 줄일 수 있다.

- 과제 2: $x + y + z = 100$의 자연수 해를 모두 출력하는 프로그램을 작성하라

- 과제 3: m개의 원소를 가진 집합에서 n개의 원소를 가진 집합으로 가는 전사함수의 개수를 출력하는 프로그램을 작성하라. m과 n의 값을 바꾸어 보면서 값이 너무 커지지 않는 입력의 범위가 어느 정도인지 확인해 보라

- 과제 4: m개의 원소를 가진 집합에서 n개의 원소를 가진 집합으로 가는 전사함수를 모두 출력하는 프로그램을 작성하라. 출력을 어떻게 하는 것이 적절할 지 생각해 보아야 한다.

8. 기초 알고리즘 프로그래밍 과제

- 과제 1: 피보나치 수열을 계산하는 3 가지 방법을 모두 작성해 보고 실행시간을 비교하라. 결과 값이 빨리 커지는 것에 주의하라.

- 과제 2: n개의 행렬을 곱하려고 한다. 크기가 $a \times b$인 행렬과 크기가 $b \times c$인 행렬을 곱하는 데 드는 계산량은 $a \times b \times c$라고 한다. n개의 행렬들을 곱하는데 필요한 계산량을 최소화 하는 순서를 찾는 알고리즘을 작성하라. 행렬들의 크기는 다르고, 입력으로 주어진다고 가정하라. 물론 곱하기가 가능한 크기들만 주어진다.

- 과제 3: 배열에 정수(음수 포함)들이 저장되어 있다. 연속인 구간들 중 그 합이 가장 큰 구간을 찾는 프로그램을 작성하라.

- 과제 4: (어려움) 배열에 정수(음수 포함)들이 저장되어 있다. 배열의 일부 값들을 골라서 배열에 있는 순서대로 보면 증가하는 순서가 될 수 있다. 이러한 것들 중 가장 긴 것을 찾는 프로그램을 작성하라.

- 과제 5: 루트 있는 트리를 입력으로 받아 아래와 같이 출력하는 프로그램을 작성하라. 트리의 각 노드에는 1,000 미만의 자연수가 저장되어 있다. 트리의 노드 연결 관계는 다음과 같이 표현해야 한다. 아래 출력에서 루트에는 자식이 3 개 있고 그 자식들 중 하나는 더 이상 자식이 없는 것임을 알 수 있을 것이다.

```
[030]--+--[054]-----[001]
       +--[002]
       L--[045]-----[123]
```