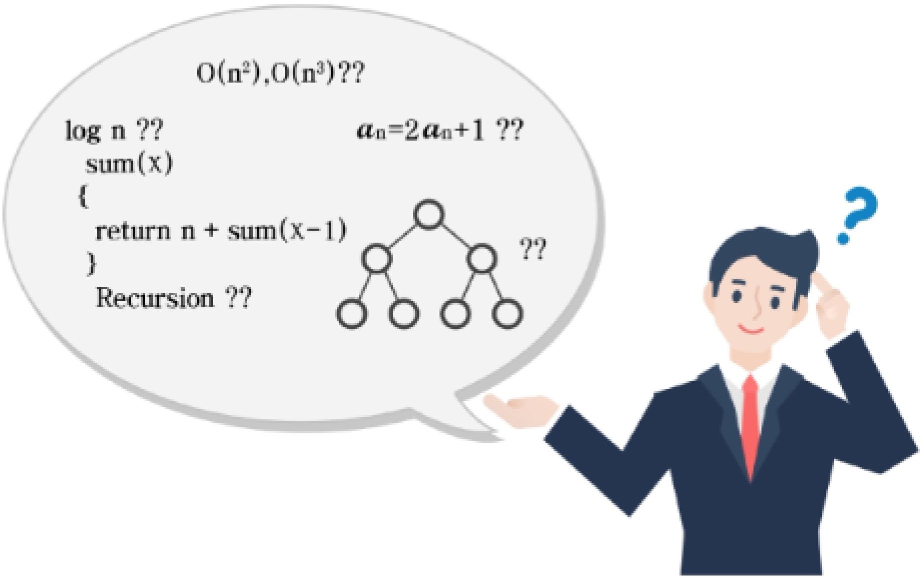
컴퓨팅 사고력

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 Solving 을 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

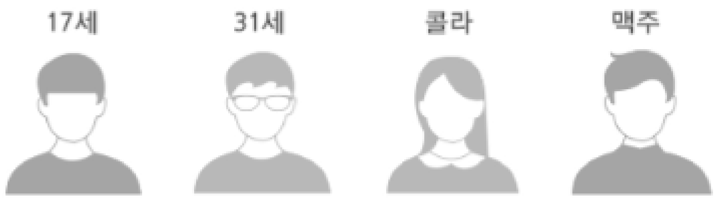
  - 주장이 사실인지 확인하기 위해 다음 카드들 중 반드시 뒤집어 보아야 하는 것은 몇 개이고 어느 것인가?

  - 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?



답: [17세]와 [맥주]

- 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
  - **<u>Soft Logic 으로 알고리즘을 이해하려고 하는 것!</u>**
  - 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 198938279382748390이 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 $p \rightarrow q$를 $p \leftrightarrow q$와 혼동하는 것에서 일어남

- 모든 당구공은 색이 같다는 다음 증명에서 잘못된 것은?
  - 수학적 귀납법: $P(1)$이 참이고, $P(n) \rightarrow P(n+1)$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 1개가 들어있는 집합은 모두 색이 같음
  - $P(n) \rightarrow P(n+1)$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 $P(n) \to P(n+1)$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

- $P(x-1) \rightarrow P(x)$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]" 그리고 "소팅을 시작할 때 {A[1], A[2], ..., A[n]}과 소팅을 마쳤을 때 {A[1], A[2], ..., A[n]}은 집합으로서 같다."

    - 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 $4n^3 + 6n^2 + 11$이 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 $4n^3 + 6n^2 + 11$이 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 $\sim(\sim p \land q) \lor q$

② $(\sim p \lor q) \lor (p \land \sim q)$

- 문제 2: 다음 명제들이 모순명제라는 것을 진리표를 이용해서 보이시오

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

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

- 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 $\sim (p \vee q)$

- 문제 4: 명제식의 변형을 통하여 다음 명제를 간소화하시오

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

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

- 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 $\exists x \in Z, x^2 < x$

- 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...)

- 문제 7: n이 홀수이면 n^2+n은 짝수임을 증명하라.

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

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

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

- 문제 10: n^2이 짝수이면 n은 짝수임을 증명하라.

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

(힌트: n이 짝수인 경우와 홀수인 경우를 따로 증명한다)

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

- 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)

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

- 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)

- 문제 17: $\log_2 5$는 무리수임을 증명하라.

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

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

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

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

- 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.

- 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.)

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: 2진수 표현에서 logn 비트로 표현할 수 있는 숫자 범위는?

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

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

① $2n$ ( ) n^2

② $2^{\frac{n}{2}}$ ( ) $\sqrt{3^n}$

③ $2^{n \log n}$ ( ) $n!$

④ $\log 2^{2n}$ ( ) $n\sqrt{n}$

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

- 문제 5: 다음 함수들의 역함수를 구하시오

① $f(x) = \log(x-3) - 5$

② $f(x) = 3\log(x+3) + 1$

③ $f(x) = 2 \times 3^x - 1$

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
- 조합의 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

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

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

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

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

$$\exists x(x \in A \land x \notin B) \lor (x \in B \land x \notin A)$$

- 문제 6: 다음이 사실임을 증명하라

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

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

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

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

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

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

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

- 문제 12: 52 개의 트럼프 카드 한 세트를 이용해서 만들 수 있는 5 개 카드의 조합은 몇가지인가?

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

- 문제 14: $x + y + z = 100$의 자연수 해는 몇가지인가?

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

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

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

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]

$$\begin{aligned} T(n) &= T(n-1) + 1 \\ &= T(n-2) + 1 + 1 \\ &= T(1) + 1 + \cdots + 1 \end{aligned}$$

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

- 문제 2: $T(n) = T(n-1) + n$

- 문제 3: $T(n) = T(n-1) + \log n$

[Solution]

$$
\begin{aligned}
T(n) &= \mathrm{T}(1) + \log n + \log n - 1 + \log n - 2 + \cdots + \log 2 \\
&\le \mathrm{T}(1) + \log n + \log n + \log n + \cdots + \log n \\
&\le \mathrm{T}(1) + n \log n
\end{aligned}
$$

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

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

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

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

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

- 문제 8: $T(n) = T(n-1) + \frac{1}{n}$

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

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

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

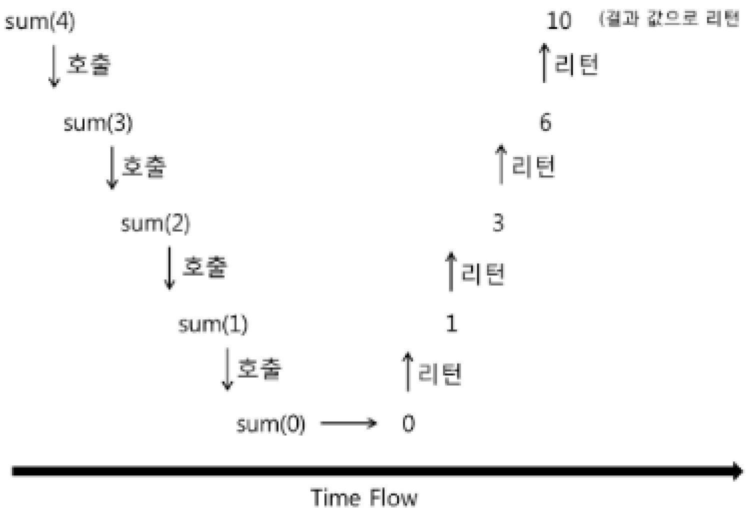
- 위 두 가지가 사실이면 모든 가능한 n에 대해 문제를 풀 수 있다는 것이 사실

- 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 코드로 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)

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

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

  배열을 절반으로 두개로 나눈 후,

  각 작은 배열을 재귀적으로 정렬하고,

  그 결과를 Merge 한다.

- 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])

}
```

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

- 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?

- 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]
```

- 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.

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)

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

  (힌트: 작은 값부터 순서대로 계산한다)

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

- 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.

- 문제 4: (약간 어려움) 배열에 정수(음수 포함)들이 저장되어 있다. 연속인 구간들 중 그 합이 가장 큰 구간을 찾는 알고리즘을 작성하라. 단, 구간의 크기는 1 이상이 허용된다.

- 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.

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]
```