논리적 사고를 기르는

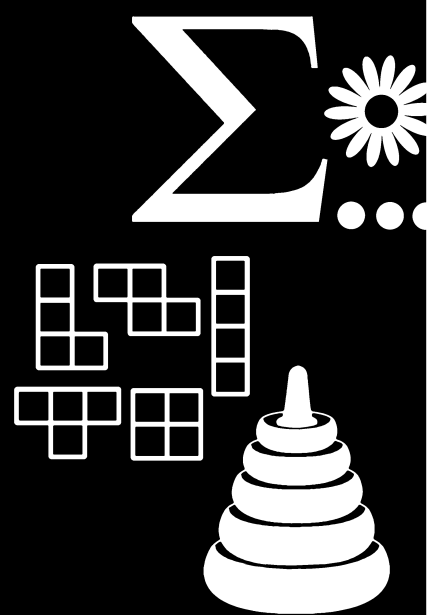
# 알고리즘 수업

# 논리적 사고를 기르는 알고리즘 수업

## 알고리즘보다 먼저 공부하는 알고리즘 사고 방식

**전자책 1쇄 발행** 2024년 2월 8일 **지은이** 롤랜드 백하우스 **옮긴이** 김준원, 윤교준 **펴낸이** 한기성 **펴낸곳** (주)도서출판인사이트
**편집** 나수지 **등록번호** 제2002-000049호 **등록일자** 2002년 2월 19일 **주소** 서울특별시 마포구 연남로5길 19-5 **전화** 02-322-5143 **팩스** 02-3143-5579 **블로그** https://blog.insightbook.co.kr **이메일** insight@insightbook.co.kr **ISBN** 978-89-6626-437-7

프로그래밍 인사이트



# 논리적 사고를 기르는
# 알고리즘 수업

롤랜드 백하우스 지음 | 김준원 • 윤교준 옮김

인사이트

# 차례

## 16장 관계, 그래프, 경로 대수 351

# 옮긴이의 말

《논리적 사고를 기르는 알고리즘 수업》은 노팅엄 대학교의 강의를 기반으로 하며, 알고리즘 설계와 관련된 문제 해결 기술에 중점을 두고 있습니다. 이 책의 전반부(~11장)는 실제 문제를 해결하는 과정에서 이론을 전달해, 자연스럽게 문제 해결에 필요한 논리적 사고를 기르고 알고리즘적 접근법을 익힐 수 있도록 구성되어 있습니다. 후반부(12장~)에서는 수학의 언어를 이용해, 알고리즘을 공부하는 데 필요한 기호논리와 수리논리를 다룹니다.

이 책의 내용과 구조는 컴퓨터 과학의 거장 에츠허르 데이크스트라의 철학으로부터 큰 영향을 받았습니다. 저자 롤랜드 백하우스는 전통적인 수학적 관행에 도전하며, 독특한 표기법과 방법론을 채택하여 이 철학을 깊이 있게 전달합니다. 역자로서 이러한 저자의 시도를 존중하여, 이 책을 한국어로 옮길 때 그의 철학을 최대한 명확하고 이해하기 쉽게 전달하고자 노력했습니다. 표기법은 독특하지만, 철학이 담겨 있습니다. '이걸 이런 방법으로도 표기할 수 있구나!' 하는 열린 마음으로 읽어 주세요.

이 책은 다른 알고리즘 책과는 꽤 다릅니다. 프로그래밍 언어로 작성된 코드가 한 줄도 등장하지 않습니다. 하물며, 알고리즘 문제를 푸는 데 사용하는 여러 방법론조차 구체적으로 설명하는 일이 없습니다. 이 책에서 시종일관 강조하는 것은 불변량, 귀납법 등 알고리즘 문제 풀이를 관통하는 사고 방식입니다. 사람들에게 알고리즘 문제 풀이를 가르칠 때, "문제는 많이 풀어 봤는데, 비슷한 다른 문제들을 보면 전혀 감이 잡히지 않는다."라는 고민을 자주 듣습니다. 문제를 풀거나, 풀이를 알게 된 다음에는, 다시 생각해 보면서 문제에서 풀이로 향하는 '논리적 사고'를 구조화해야 합니다. 단순히 하나의 사례일 뿐인 특정한 문제의 풀이를 외우는 것으로 끝난다면, 홀로 남겨진 지식이 될 뿐입니다.

저자는 이 책에서 '논리적 사고'를 기르기 위한 각종 도구를 알려 주기 위해 다분히 노력하고 있습니다. 추상적이고 뜬구름 잡는 이야기라고 생각될지라도 저자의

의도를 따라간다면 분명히 알고리즘 문제 해결의 깊은 철학을 얻어 갈 수 있을 것이라 믿습니다. 흔히 알고리즘을 단순히 암기해야 할 학문으로 보고, 문제 해결 능력은 타고난 재능이라고 여깁니다. 이 책을 통해 이러한 오해를 바로잡고, 알고리즘 문제 해결의 본질을 이해함으로써 더 큰 성장을 이루기를 희망합니다.

저희는 수년간 알고리즘 대회를 준비하고 공부하며, 관련된 일에 종사해 왔습니다. 그러기에 저자가 강조하는 문제 해결 능력의 중요성에 깊이 공감하고 있습니다. 이 책이 여러분에게 알고리즘 문제 해결의 깊은 철학을 전달하고, 학문적 여정에 도움이 되기를 바랍니다. 부족한 저희에게 번역을 제안해 주시고, 부끄러운 초벌 번역 안을 출판할 수 있을 정도로 다듬고 편집, 조판해 주신 인사이트 출판사에 감사드립니다. 여러분도 이 책을 읽으며 문제 해결의 즐거움을 경험해 보세요!

김준원, 윤교준

# 머리말

현대의 발전된 세상에서 우리의 생활 방식과 생계, 심지어 삶 자체는 50년 전에는 상상하기 어려웠을 정도로 컴퓨터 기술에 크게 의존하게 되었다. 그리고 이러한 발전의 근간은 우리의 문제 해결 능력의 본질적인 변화에 있다. 많은 경우, 고난도 문제의 해결책은 컴퓨터라는 어리석은 기계에서 실행될 프로그램으로 구현되어야 한다. 해결하려는 문제의 크기와 복잡성뿐만 아니라, 해결책을 공식화하는 데 요구되는 정밀도와 정확성에도 혁명적인 변화가 일어났다. 알고리즘 문제 해결은 우리가 직면한 새로운 도전에 대처하기 위해 필요한 기술에 관한 것이다.

이 책은 2003년 9월, 노팅엄 대학교에서 처음 도입한 1학년 1학기 교과목을 기반으로 한다. 처음에는 선택 과목이었지만, 2006년에는 모든 1학년 컴퓨터 과학 및 소프트웨어 공학 학생들의 필수 수강 과목이 되었으며, 이는 지금까지도 유지되고 있다.[1] 중국과 말레이시아, 영국에 있는 대학 캠퍼스 세 곳에서 이 교과목을 채택했다.

이 책의 목표는 좋은 문제 해결 기술을 심어 주는 것이며, 특히 알고리즘 설계가 필요한 문제를 다루는 기술을 강조한다. 접근 방식은 문제 중심적이다. 이 책의 주요 부분인 1부는 알고리즘 문제 해결의 원칙을 체계적으로 소개하는 여러 예제로 구성되어 있다. 문제 중심적인 접근법은 학생들의 도전 성향을 자연스럽게 자극할 수 있기에 매우 중요하다. 그러나 이론 없는 예제는 무의미하므로 2부는 예제를 뒷받침하는 수학에 관하여 다루었다. 본문에 등장하는 상자는 특정 예제와 관련된 수학 지식을 알려 준다.

이 책에서 사용하는 표현 방식은 여러 측면에서 전통적인 수학적 관행에서 벗어나 있다. 일례로, 불리언 대수와 논리를 다루는 방법은 표준적이지 않고, 사용하는 표기법 또한 그러하다. 그러나 이러한 이탈은 적어도 20년 동안 잘 이해되어 온 알

1 (편집자) 이 과목은 2013년에 그가 은퇴하면서 폐강되었다.

고리즘 문제 해결의 발전에 기반하고 있다. (수학 발전에서 20년은 아주 짧은 시간이지만, 현대 컴퓨터 시대에서는 비교적 긴 기간이다.) 아직 자료에 대한 확신이 없거나 익숙하지 않은 교육자는 개방적인 마음가짐으로 접근하기를 부탁드린다.

이 책의 대부분은 수업에서 이미 테스트를 거쳤지만 모두 그런 것은 아니다. 또한 모든 내용이 학부 1학년 1학기 수준인 것도 아니다. 저자는 3년 동안 수학적 기법(이 책의 2부) 과목과 알고리즘 문제 해결(이 책의 1부) 과목을 같은 학기에 가르쳐 보기도 했지만, 더 이상 그렇게 하지 않는다. 또한 책의 두 부분 모두 한 해 동안 다루기 어려운 양의 자료를 담고 있다. 불리언 대수와 같은 일부 주제는 고등학교 수준에서 가르칠 수 있지만 (뒷부분에서 다루는) 일부 주제는 나중으로 미루는 편이 좋을 것이다.

예제가 아주 많으면 선택의 폭이 넓어진다는 장점이 있다. 저자는 학생들이 적극적으로 문제 해결에 참여하도록 하는 수업 과제를 토대로 평가하며, 매년 수업 내용을 바꾸고 있다. 모든 연습문제의 모범 풀이는 학생들에게 피드백 목적으로 제공했으며, 이 책의 끝에 포함되어 있다. 그러나 수업에서 '프로젝트'라고 칭하는 일부 연습문제의 풀이는 생략했고, 몇몇 프로젝트는 아예 책에 싣지 않았다. (프로젝트의 모범 풀이는 표절의 위험 없이 재사용하기 위하여 학생들에게 인쇄물로 제공한다.)

이와 같은 주제를 가르칠 때, 학생들이 해결하기에는 어렵지만 이해는 할 수 있을 만한 수준의 예제를 사용하는 것이 매우 중요하다. 이 책에 수록한 문제 대부분은 독자가 수학적 지식 없이도 무차별 대입 방식으로 해결할 수 있지만 지식을 활용하면 더 효과적으로 해결할 수 있다. 이 책의 숨겨진 목표는 독자를 '수학 하는 과정'에 참여시키는 것이다. 즉, 수학과 직접적인 관련이 없어 보여도 문제의 해결 방법을 모델링하고 계산하는 수학적 과정을 체험하도록 하는 것이다.

여기에 제시된 내용은 에츠허르 데이크스트라(1930-2002)의 글에 매우 큰 영향을 받았다. 데이크스트라는 알고리즘 설계에 기여한 것으로 유명하다. 또한 그는 경력 전반에 걸쳐 수학적 방법을 명확하게 표현하고 이를 통해 수학적 방법을 개선하는 데 많은 노력을 기울였다. 개인적으로 데이크스트라가 컴퓨터 과학 분야에 기여한 것보다 수학적 방법의 개선에 기여한 바가 더 크다고 본다. 가끔 직접적인 언급 없이 데이크스트라의 표현을 사용하기도 했는데, 독자는 이를 옛날 속담처럼 (데이크스트라의 권위를 신경 쓰지 말고) 받아들이기를 바란다. 그중 하나는 데이크스트라가 수학을 '효과적인 추론의 예술'로 정의한 것이다(12장 참조). 저자는 그

의 문제 해결 능력을 따라갈 수 없겠지만, 그의 수학적 방법에 관해 계속해서 연구하려는 노력을 통해 사람들이 좋고 나쁜 문제 해결 능력에 대해 더 비판적으로 생각하고, 명확하게 설명할 수 있게 도움을 주고자 한다.

이 책을 준비하는 과정에서 저자는 동료들로부터 아주 많은 도움을 받았고 모두에게 감사의 말씀을 전한다. 특히 세세한 비평과 개선 제안을 해 주신 Diethard Michaelis와 David Gries에게 감사드린다. 이 교과목을 강의하는 데에 도움을 주신 분들에게도 특별한 감사를 표한다. Siang Yew Chong과 John Woodward는 각각 수년 동안 말레이시아와 중국에서 이 교과목을 강의해 주셨다. João Ferreira는 저자의 안식년 동안 영국 노팅엄에서 강의를 해 주셨고, 그 이후로도 Wei Chen와 Alexandra Mendes와 함께 수업과 과제 평가 및 시험 채점에서 저자를 돕고 있다. 비전통적이고 틀에 얽매이지 않은 교육 방법에 대한 도전을 매우 긍정적으로 받아 주신 모든 분들에게 매우 감사하게 생각한다. 출판사 John Wiley & Sons와, 특히 Georgia King과 Jonathan Shipley의 인내심과 협조에 감사를 표하며, 또한 10명의 (대부분 익명의) 평론가들에게 의견과 피드백을 주신 데 감사드린다. 마지막으로, 이 강의를 성공적으로 수강한 학생들에게 감사를 표한다. 저자가 받은 최고의 피드백은 "수업은 매우 어려웠지만, 도전하는 게 즐거웠습니다."이었다. 이렇게 말해 준 학생의 이름은 모르지만, 얼굴은 아직까지도 기억에 남는다.

롤랜드 백하우스
2011.3.

논리적 사고를 기르는 알고리즘 수업

제1부

# 알고리즘 문제 해결

ALGORITHMIC PROBLEM SOLVING

# 1장

Algorithmic Problem Solving

# 들어가며

1964년에 출간된 영어 사전 《Concise Oxford Dictionary(간결한 옥스퍼드 사전)》 5판에서, '알고리즘(Algorithm)'은 없는 단어였다. 그러나 이제 '알고리즘'은 누구나 자주 접하는 단어가 되었다. 인터넷의 부적절한 사용을 감지하는 알고리즘, 자동차의 안전을 개선하는 알고리즘, 사용자의 동작을 인식하고 응답하는 알고리즘 등. 모든 것이 전산화된 현대 사회에서 알고리즘이라는 단어의 뜻을 모르는 사람은 이제 거의 없다.

이는 알고리즘이 최근에 새로이 생겨난 개념이라는 뜻이 아니다. 오히려 알고리즘은 수천 년 동안 사용되어 왔다. 예를 들어, 알고리즘은 건설과 측량에 사용되어 왔다. 산의 양쪽에 터널을 뚫어 중간에서 만나게 하는 데 알고리즘을 사용했다. 지도를 만드는 사람들은 직접 산에 오르지 않고 산의 높이를 확인하기 위해 알고리즘을 사용했다. 그러나 컴퓨터 시대가 오기 전, 사람들은 이러한 작업에 알고리즘이 사용된다는 사실을 중요하게 여기지 않았다.

## 1.1 알고리즘

알고리즘은 차례대로 실행되는 몇 개의 명령으로 이루어진, 올바르게 정의된 과정이다. 과거에는 알고리즘을 실행하는 주체가 대부분 인간이었기 때문에, 알고리즘은 크게 강조되지 않았다. 인간은 똑똑하기 때문에 불완전하거나 엄밀하지 않은 명령도 잘 해낼 수 있다. 그러나 현대에 들어서 알고리즘의 실행은 더욱 자동화되었

고, 보통 컴퓨터가 실행한다.[1] 컴퓨터는 '멍청한' 기계이기 때문에, 알고리즘을 이루는 명령들은 엄밀해야 하고, 매우 구체적으로 정의되어야 한다. 이 점은 문제 해결을 까다롭게 하는 새로운 도전 과제가 되었으며, 문제를 푼다는 것이 무엇인가에 대한 우리의 생각을 뒤바꿨다. 컴퓨터 시대는 우리 삶의 방식뿐 아니라 우리가 생각하는 방식에도 대변혁을 일으켰다.

## 1.2 알고리즘 문제 해결

인간은 알고리즘을 실행하는 데 꽤 능하다. 예를 들어 어린이들은 일찍이 123456을 78로 나눈 몫과 나머지 같은 것을 계산하기 위해 큰 수를 나눗셈하는 방법을 배우고, 곧 능숙해진다. 그러나 인간은 자주 실수를 한다. 컴퓨터는 알고리즘이 엄밀하고 올바르게 정의되었을 때 우리보다 알고리즘을 더 잘 실행할 수 있다. 반복적인 계산을 수행하는 데 컴퓨터를 사용하는 것은 매우 효과적이다.

알고리즘을 표현하는 것은 실행하는 것과 다른 문제이다. 알고리즘을 표현하는 연습을 하는 사람은 거의 없고, 인간이 알고리즘을 표현하는 데에 능하다고 하기는 힘들다. 하지만 알고리즘을 표현할 능력이 없는 컴퓨터와 달리, 인간은 창의적이다. 소위 '지능적인' 시스템은 컴퓨터가 실행할 수 있도록 알고리즘을 표현하는 인간의 능력에 의존하고 있다. 알고리즘 문제 해결은 여러 가지 문제를 해결하는 알고리즘을 표현하는 능력에 관한 것이다. 알고리즘 문제 해결 능력을 키우는 것은 컴퓨터 시대의 주요 과제이다.

이 책은 입문서이고, 여기서 다루는 문제들은 아주 작은 '토이' 문제들이다. 아래 문제는 전형적인 토이 문제 중 하나로 '손전등' 문제나 'U2' 문제라는 이름으로 불리곤 한다. (지금은 너무 잘 알려진 문제이지만) 과거 주요 소프트웨어 기업에서 입사 면접으로 쓰인 문제로 알려져 있다.

> 4명의 사람이 다리를 건너려고 한다. 밖은 어두워서, 다리를 건너려면 횃불을 들고 건너야 한다. 횃불은 하나뿐이다. 다리는 동시에 두 명까지만 건널 수 있다. 사람들이 다리를 건너는 데 서로 다른 시간이 걸린다. 두 명이 같이 건널 때는 둘 중 느린 사람이 건너는 데 드는 시간만큼 걸린다. 첫 번째 사람은 다리를 건너는 데 1분, 두 번

1 오래 전 '컴퓨터(computer)'는 계산을 하는 사람을 뜻했기 때문에, 엄밀히 '디지털 전자 계산기(electronic digital computer)'라 부르는 것이 맞겠다. 하지만 여기서는 '디지털 전자 계산기'가 '컴퓨터'의 유일한 의미라고 생각하기로 하자.

째 사람은 2분, 세 번째 사람은 5분, 네 번째 사람은 10분이 걸린다. 횃불은 다리를 왔다 갔다 할 때 항상 들고 다녀야 한다.

4명이 모두 다리를 건너는 데 17분이면 충분함을 보여라.

이 문제에 대한 해답을 제시하려면, 4명이 모두 다리를 건너는 데 필요한 명령들을 나열하면 된다. 명령은 일반적으로 '사람 x와 y가 함께 다리를 건넌다'나 '사람 z가 다리를 건넌다'와 같은 꼴이 될 것이다. 만약 명령을 모두 실행하는 데 드는 시간이 (최대) 17분이라면 정답이 된다.

알고리즘은 주로 이보다 일반적이다. 보통 알고리즘은 **입력**을 받는다. 각 입력에 대해, 알고리즘은 **출력**을 계산해 내야 한다. 입력과 출력은 서로 **입력-출력 관계**라고 불리는 관계를 가진다. 다리 건너기 문제의 경우, 알고리즘은 각 사람이 다리를 건너는 데 걸리는 시간을 나타내는 4개의 정수를 입력으로 받고, 4명이 모두 다리를 건너는 데 드는 최소 시간을 출력한다. 예를 들어 입력이 1, 3, 19, 20이라면 출력은 30이 되고, 입력이 1, 4, 5, 6이라면 출력은 17이 될 것이다. 입력 값은 알고리즘의 **매개변수**라고 한다. (다리 건너기 문제를 푸는 알고리즘은 3.5절에서 유도한다. 문제를 체계적으로 다루는 법을 보여 주기 위해, 풀이는 면접관과 면접자의 가상 대화 형태로 서술했다.)

두 번째 예제는 9장에 있는 닭 쫓기 문제이다. 이 문제는 1914년에 유명한 퍼즐 제작자 샘 로이드(Sam Loyd)가 만들었다. 요약하면, 이 문제는 체스판 위에서 하는 (체스보다 훨씬) 간단한 규칙의 게임이다. 이 게임은 인터넷으로 플레이할 수 있으며, 매우 쉽게 승리할 수 있다. 대부분의 플레이어는 조금만 해 보면 승리할 수 있을 것이다. 하지만 임의의 크기의 체스판에서 최소한의 움직임만으로 게임에서 승리하는 알고리즘을 표현할 수 있는 사람은 거의 없을 것이다. 그는 최소의 움직임으로 게임에서 승리하는 방법을 묻는 문제를 냈지만, 그의 풀이는 알고리즘이 아니라 단지 횟수만을 제시한다. 현대에는 알고리즘을 실행하는 컴퓨터에 대한 의존이 커져, 문제 해결에 대한 개념이 바뀌었다. 과거의 문제 해결이 단순히 문제에 대한 답을 제시하는 것이었다면, 현대의 문제 해결은 특정한 종류의 문제를 모두 해결할 수 있는 알고리즘을 표현하는 것이다. 알고리즘 문제 해결은 이를 다룬다.

알고리즘을 표현해야 한다는 것은 확실히 문제 해결을 어렵게 한다. 문제를 해결하는 절차를 엄밀하고 명확하게 표현해야만 하기 때문이다. 문제가 더 일반적일수록 더 어려워진다. (예를 들어, 다리 건너기 문제는 다리를 건너는 사람의 수가 정

해져 있지 않으면 더 어려워진다.) 그렇지만, 풀이를 더욱 잘 이해할 수 있다는 장점이 있다. 알고리즘을 표현하는 과정은 **알고리즘이 왜 옳은가**에 대한 완전한 이해를 필요로 한다.

## 1.3 개요

효율적으로 문제를 풀 때 중요한 것은 효율적으로 생각하고 표현하는 것이다. 불필요한 세부 사항과 복잡도를 피해야 한다. 컴퓨터 프로그램의 크기가 전례 없이 커지고 있기 때문에, 복잡도에 대한 이해는 컴퓨터 시대에 특히 중요해지고 있다. 일반적인 컴퓨터 프로그램은 수백, 수천, 심지어 수백만 줄의 코드로 이루어져 있다. 단 하나의 오류로도 시스템 전체가 무너질 수 있는 디지털 컴퓨터의 까다로운 특성 때문에, 알고리즘을 잘 설계하는 것은 우리의 문제 해결 능력에서 매우 중요한 부분을 차지한다.

이 책은 예제들을 함께 풀어 보면서 새로운 기술과 통찰을 전달하는 것을 목표로 한다. 수학적 계산의 중요성을 보여 주는 것을 목표로 하지만, 다루는 예제들은 수학적이지 않다. 오히려, 닭 쫓기나 다리 건너기 문제처럼 수학적 지식이 없는 사람들도 기초 지식만으로 이해할 수 있다. 이 책은 또한 도전하기 위한 책이다. 대부분의 문제는 특히 잘 훈련되지 않은 사람들에게는 꽤 어려울 것이다.

이 책은 두 부분으로 이루어져 있다. 앞부분은 도전할 만한 소규모의 문제로 구성되어 있다. 뒷부분은 이 문제들을 이어 붙여서 풀이를 얻어내는 수학적 방법을 다룬다. 책의 앞부분에서는 문제와 관련된 수학적 기술이 팁으로 제시될 것이다.

여기 등장하는 많은 문제들은 재미 삼아 푸는 수학 문제로 잘 알려져 있지만, 우리는 새로운 분야의 수학인 '알고리즘'에 집중한다. 문제들은 알고리즘 이론에 체계적으로 입문할 수 있게 하는 순서대로 제시되어 있다. 이미 수학에서 잘 정립된 분야이지만 새로운 관점을 얻을 수 있을 것이다.

책은 '불변성'을 다루는 2장에서부터 본격적으로 시작한다. 이는 자명하지 않은 모든 알고리즘의 중심이 되는 용어이다. 알고리즘은 몇 개의 서로 다른 종류의 '문장'의 조합으로 표현되어 있다. 문장의 종류와 문장을 조합하는 법(대입 구문, 순차적 분석, 조건 분기, 그리고 수학적 귀납법과 반복문 등)이 예제와 함께 하나씩 소개될 것이다. 그렇기 때문에, 독립적으로 읽을 수 있는 5장을 제외하고 2장에서 7장까지는 순서대로 읽는 것을 추천한다. 이후의 장들은 앞서 소개된 법칙들을 어려

운 문제에 적용하는 장들로서, 원하는 순서대로 읽을 수 있다.

예제를 통해 접근하면 예제 자체가 책의 주제라고 생각할 수도 있겠다. 그러나 이 책의 가장 큰 목표는 알고리즘 문제 해결에 핵심적인 기술을 전달하는 것이다. 어떻게 문제를 분석하는지, 수학적으로 모델링하는지, 그리고 어떻게 알고리즘적인 해답을 계산해 내는지. 그렇기에 이 책은 일차적으로 지식이 아니라 방법에 대한 책이다. 흥미가 생겼다면, 그리고 도전을 받아들일 준비가 되었다면 계속해서 읽어 나가도록 하자!

## 1.4 참고 자료

1960년대까지 '알고리즘'이라는 단어가 유명한 사전에 등장하지 않았다는 관찰은 도널드 커누스(Donald Knuth)[Knu68]가 하였다. (사실 그의 관찰은 1950년대에 이루어졌다. '알고리즘'은 1980년대가 되어서야 유명한 사전에 실리기 시작했다.) 도널드 커누스는 알고리즘과 계산 수학에 관한 매우 영향력 있고 해박한 책들을 집필했다. 계산 수학은 추후 공부해 보기를 강력히 권장하는 분야이다.

저자는 '알고리즘 문제 해결'이라는 단어를 처음 접한 자료이기도 한 [Lev03]에서 처음 다리 건너기 문제를 보았다. [MM08]은 이 책과 유사하게 문제 기반으로 알고리즘에 접근한(하지만 알고리즘에 대해서는 이 책과 달리 암묵적으로만 강조한) 기초 서적이다.

# 2장

Algorithmic Problem Solving

# 불변량

> '불변'은 '변하지 않음'을 의미하는 한자어다. 어떤 변환의 불변량이란 그 변환을 아무리 적용해도 변하지 않는, 항상 일정한 값이나 특징을 말한다. 비슷한 용어로는 '상수'와 '패턴'이 있다.
>
> 문제에 숨겨진 불변량을 찾아내는 것은 문제 해결 기법 중에서 가장 중요하다고 해도 과언이 아니다. 이 장에서는 불변의 개념을 소개하고, 이를 적용할 수 있는 다양한 예제를 다룬다.

불변이 무엇인지 알아보기 전에, 이 장에서 다룰 문제들을 먼저 풀어 보자. 이 중에는 쉽게 풀 수 있는 문제도 있고, 많이 어렵거나 혼자서는 풀 수 없는 문제도 있을 것이다. 만약 어떤 문제에서 막혔다면 주저 없이 다음 문제로 넘어가자. 그러나 풀이를 읽기 전에 모든 문제를 한 번씩은 시도해 보는 것이 좋다.

첫 번째 문제에서는 불변량이 문제 해결에 어떻게 활용되는지 자세하게 논의한다. 그 과정에서 컴퓨터 프로그래밍과 관련된 기본 기술인 대입문의 사용법과 대입적으로 사고하는 방법을 소개한다. 문제 뒤에는 유사한 방법으로 해결할 수 있는 연습문제가 있으니 꼭 풀어 보자.

두 번째 문제에서는 앞에서 다룬 문제 해결 기법을 더 발전시킨다. 이후, 좋은 접근 방법과 나쁜 접근 방법에 대하여 논의한다. 세 번째 문제는 꽤 쉽지만 새로운 개념을 포함하며, 이 개념에 대하여 자세하게 다룬다. 여기까지 각 문제의 풀이를 충분히 이해하면서 잘 따라왔다면 남은 문제들을 스스로 충분히 해결할 수 있을 것이다. 이 책에서는 한 문제의 풀이 과정을 설명하고, 뒤이어 여러분이 직접 해결할 수

있도록 몇 개의 추가 문제를 제공한다. 이러한 학습 과정은 문제가 어려워질수록 더 많이 반복된다. 그러나 너무 걱정할 필요는 없다. 앞에서 다룬 문제 해결 기법을 적용해서 추가 문제들을 쉽게 풀고 있는 자신을 발견할 수 있을 것이다.

1. 초콜릿

   격자무늬로 금이 가 있는 직사각형 모양의 초콜릿이 있다. 하나의 큰 초콜릿을 여러 개의 작은 정사각형 조각으로 쪼개려고 한다. 초콜릿을 쪼갠다는 것은 하나의 초콜릿 조각을 직선의 금을 따라 자른다는 것이다. (즉, 한 번 쪼갤 때마다 하나의 조각은 둘로 나뉘게 된다.)

   그림 2.1은 다섯 조각으로 쪼개진 $4 \times 3$ 크기의 초콜릿을 나타낸다. 굵은 선은 초콜릿을 어떻게 쪼갰는지 보여 준다.

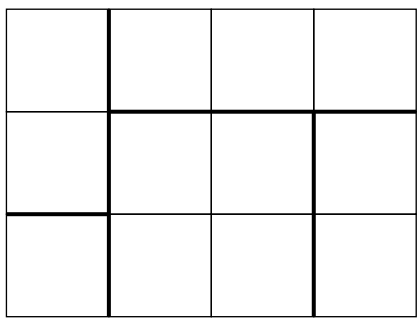


그림 2.1 초콜릿 문제

> 하나의 직사각형 초콜릿을 작은 정사각형 조각들로 완전히 나누기 위해서는 초콜릿을 총 몇 번 쪼개야 하는가?

2. 빈 상자

   책상 위에 크기가 큰 빈 상자 11개가 놓여 있다. 몇 개의 상자를 골라, 각각의 상자 안에 중간 크기의 빈 상자 8개를 넣는다. 다시, 중간 크기의 상자 몇 개를 골라, 각각의 상자 안에 작은 크기의 빈 상자 8개를 넣는다.

> 이 과정을 모두 마친 후, 안이 비어 있는 상자의 개수를 세었더니 총 102개였다. 상자의 총 개수는 몇 개인가?