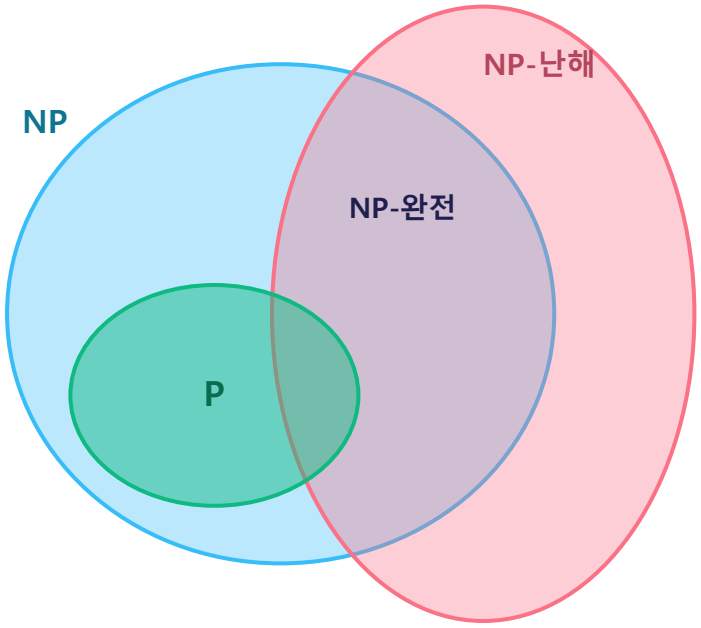
그림으로 쉽게 설명하는 파이썬 알고리즘

CHAPTER 09

NP-완전 문제

NP-Complete Problems

문제의 한계를 아는 것이 곧 해결의 시작이다



P ≠ NP 라고 가정했을 때의 지형

CONTENTS

이번 장의 구조

8장에서 분기한정으로도 감당할 수 없었던 문제들 — 그 정체를 이 장에서 밝힌다.

9.1

현실의 벽

컴퓨터가 "언젠가는" 답을 내놓는다는 사실이
"살아 있는 동안" 답을 본다는 뜻은 아니다.

9.1

§9.1 현실의 벽

이 장이 답하려는 세 가지 질문

"아무리 요령을 피워도 근본적으로 해결이 불가능해 보이는 문제들의 정체는 무엇인가?"

"왜 TSP 같은 문제 앞에서는 겸손해져야 하는가?"

"그 한계를 어떻게 우회할 수 있는가?"

"더 빠른 알고리즘"을 찾는 대신, "빠른 알고리즘이 존재하는가"를 묻는 장이다.

§9.1 현실의 벽

"계산 가능하다"는 것의 의미

컴퓨터 과학에서 어떤 문제가 계산 가능하다(computable)고 말할 때는, 단순히 답을 낼 수 있다는 것 이상의 엄밀한 정의가 필요하다.

앨런 튜링

계산의 본질을 정의한 튜링 머신의 창시자다. 1936년의 논문 한 편이 컴퓨터 과학의 출발점이 되었다.

유한성 (finiteness)

알고리즘은 반드시 언젠가 종료되어야 한다. 끝나지 않는 절차는 알고리즘이 아니다.

계산 불가능

무한 루프에 빠지는 절차는 아무리 정교해도 알고리즘의 자격을 얻지 못한다.

"답이 있다"와 "답을 구할 수 있다"는 전혀 다른 진술이다

수학적으로 해가 존재한다는 것과, 그 해를 유한한 절차로 찾아낼 수 있다는 것은 별개의 문제다. 이 구분에서 계산 이론이 시작된다.

여기에 "얼마나 오래 걸리는가"라는 질문이 하나 더 붙으면 이 장의 주제가 된다.

튜링 머신 (Turing machine)

개념

무한히 긴 테이프에 0과 1을 쓰고 읽으며, 정해진 규칙에 따라 상태를 바꾸는 아주 단순한 모델이다.

왜 중요한가

현대의 슈퍼컴퓨터를 포함한 모든 컴퓨터의 논리적 조상이다.

무한 테이프 · 한 칸씩 좌우로 이동하며 상태를 바꾼다

계산 가능 = 튜링 머신이 유한한 단계를 거쳐 반드시 멈추고 답을 내놓을 수 있다는 뜻

주의

"언젠가 끝난다"는 사실이 "살아 있는 동안 답을 볼 수 있다"는 것까지 보장하지는 않는다.

계산 가능성은 "가능한가"의 문제이고, 이 장이 다루는 것은 "현실적인가"의 문제다.

§9.1 현실의 벽

이론적 계산 가능성 vs 현실적 해결 가능성

이론적 계산 가능성

"알고리즘이 존재하고, 유한 시간 안에 반드시 종료된다"

답이 존재함을 보장한다

그러나 그 시간이 얼마인지는 말해 주지 않는다

현실적 해결 가능성

"실제로 합리적인 시간 안에 답을 구할 수 있는가?"

시간이 너무 오래 걸릴 수 있다

35억 년 뒤에 나오는 답은 없는 답과 같다

이 괴리가 바로 NP-완전 문제의 출발점이다

계산 이론은 "풀 수 있다"고 말하는데 현실에서는 풀리지 않는다. 그 간극이 어디서 오는지를 설명하는 것이 이 장의 목표다.

§9.1 현실의 벽

다항 시간 $O(n^k)$ vs 지수 시간 $O(2^n)$

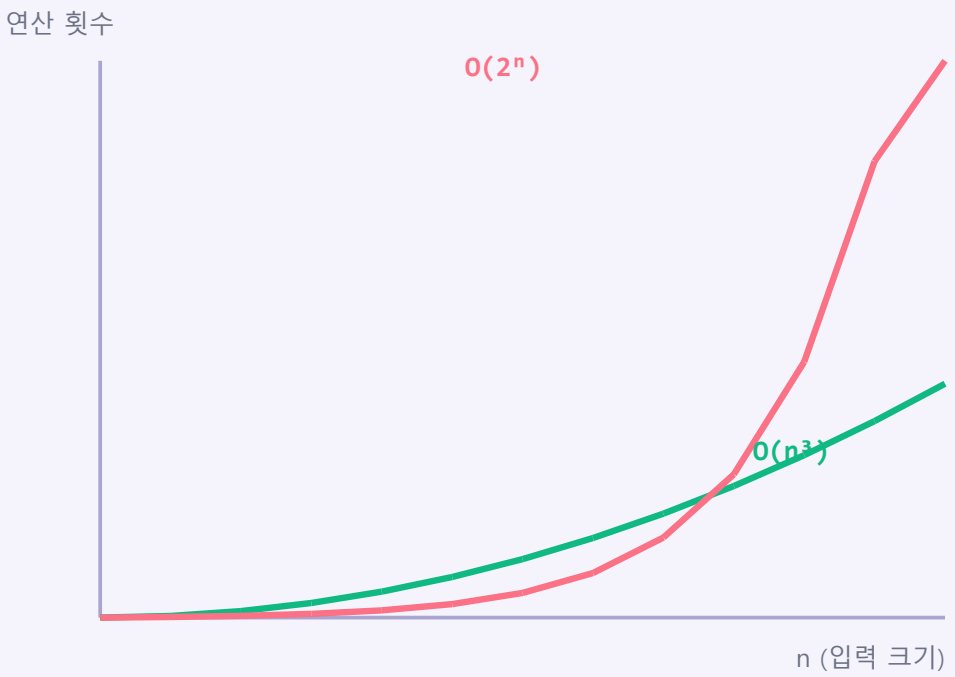
다항 시간 — $O(n)$, $O(n^2)$, $O(n^3)$

n의 거듭제곱 형태다. 입력이 커져도 감당할 수 있는 속도로 늘어난다.

지수 시간 — $O(2^n)$, $O(n!)$

n이 지수 자리에 놓인다. 입력이 조금만 커져도 폭발적으로 증가한다.

n이 조금만 커져도 두 곡선의 차이는 상상을 초월할 만큼 벌어진다.



§9.1 현실의 벽

n = 50 일 때의 실행 시간 비교

구분	알고리즘	연산 횟수	소요 시간
다항 시간	$O(n^3)$	125,000	0.0001초
지수 시간	$O(2^n)$	약 1,125조	약 35.7년

※ 1초에 10^9 연산을 수행한다고 가정

1

n = 50, 다항 시간

0.0001초. 눈 깜짝할 사이에 끝난다.

2

n = 50, 지수 시간

35.7년. 사람의 경력 전체를 걸어야 한다.

3

n = 100, 지수 시간

우주의 나이보다 긴 시간. 사실상 답이 없다.

§9.1 현실의 벽

세상의 모든 문제는 두 종류

쉬운 문제 (tractable)

다항 시간 알고리즘이 존재한다

정렬 — O(n log n)

탐색 — O(log n)

최단 경로 — 다익스트라

최소 신장 트리 — 크루스칼

어려운 문제 (intractable)

다항 시간 알고리즘이 아직 발견되지 않았다

외판원 순회 (TSP)

0/1 배낭 문제

그래프 색칠 문제

빈 포장 문제

"발견되지 않았다"이지 "존재하지 않는다"가 아니다 — 이 미묘한 차이가 P vs NP의 핵심이다.

대표적인 어려운 문제들

외판원 순회 (TSP)	모든 도시를 한 번씩 방문하고 출발지로 돌아오는 최단 경로를 찾는다.
배낭 문제	용량 제한 안에서 총 가치가 최대가 되는 부분집합을 고른다.
빈 포장 문제	물건들을 상자에 나눠 담되 사용하는 상자 수를 최소로 만든다.
그래프 색칠	인접한 정점이 같은 색이 되지 않도록 하는 최소 색의 수를 구한다.
정수 선형 계획법	변수가 정수여야 한다는 조건 아래 선형 함수를 최적화한다.

분야는 제각각이지만, 뒤에서 보게 되듯 이들은 사실 같은 난이도의 문제다.

§9.1 현실의 벽

신도 풀 수 없는 문제 — 정지 문제

정지 문제(halting problem, 앨런 튜링 1936) — 프로그램이 언젠가 멈출지, 아니면 영원히 무한 루프를 돌지를 미리 판별하는 완벽한 프로그램을 만들 수 있을까?

귀류법 증명 — 청개구리 프로그램 Q

① 완벽한 판별기 H가 존재한다고 가정한다.

② H가 "멈춘다"고 답하면 → Q는 일부러 무한 루프에 들어간다.

③ H가 "무한 루프"라고 답하면 → Q는 즉시 멈춘다.

④ Q(Q)를 실행하면 어느 경우에도 모순이 발생한다.

∴ 완벽한 판별기 H는 존재할 수 없다

결정 불가능 (undecidable)

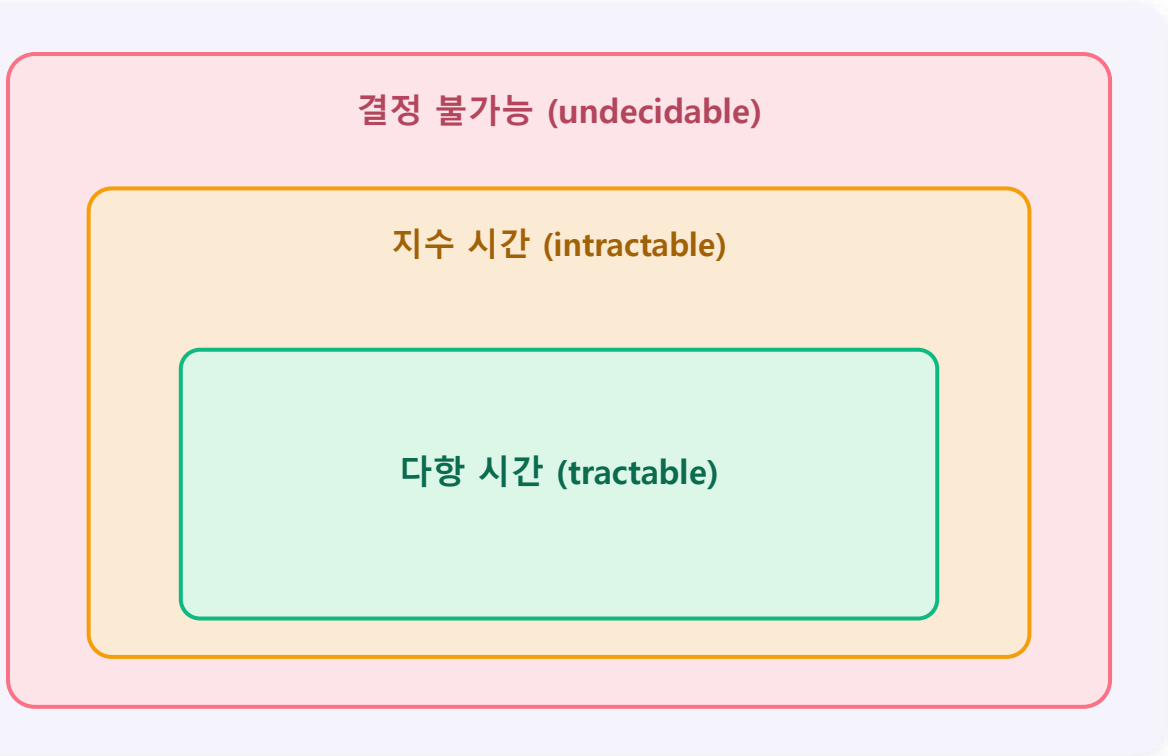
시간이 부족한 것이 아니다. 무한한 시간이 주어져도 논리적으로 해결이 불가능한 문제가 존재한다.

어려움에도 등급이 있다 — 오래 걸리는 것과 아예 불가능한 것은 다른 층위다.

§9.1 현실의 벽

문제의 계층 구조

정지 문제가 알려 준 사실은 하나다 — 문제의 세계는 두 층이 아니라 세 층으로 되어 있다.



다항 시간에 풀리는 문제

정렬, 탐색, 최단 경로 — 현실적으로 해결 가능하다.

지수 시간이 걸리는 문제

TSP, 배낭, 그래프 색칠 — 답은 있지만 기다릴 수 없다.

결정 불가능한 문제

정지 문제 — 무한한 시간을 줘도 논리적으로 불가능하다.

9.2

P와 NP

답을 찾는 일과 답을 채점하는 일은
전혀 다른 난이도의 작업이다.

9.2

결정 문제 (decision problem)

이론에서는 "최적값"을 구하는 대신, Yes / No로만 답하는 문제를 다룬다.

이유 ① 출력 크기 제한

정답 자체가 지수적으로 긴 문제라면, 다항 시간 안에 출력하는 것조차 불가능하다.

이유 ② 환원 가능성

"최단 경로를 구하라"를 "경로가 K보다 짧은가?"로 바꾸면 서로 비교할 수 있는 형태가 된다.

그래프 색칠 "k개 이하의 색으로 칠할 수 있는가?"

TSP "총비용이 K 이하인 순회 경로가 존재하는가?"

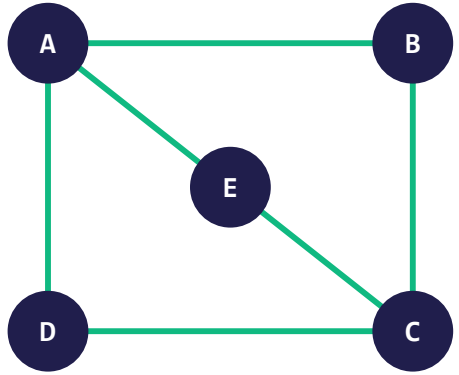
배낭 "가치 합이 V 이상인 조합이 존재하는가?"

오일러 순환 vs 해밀토니안 순환

겉보기에 거의 같아 보이는 두 문제가 완전히 다른 세계에 속한다.

오일러 순환 — 쉽다 (P)

모든 간선을 한 번씩 지나는 경로

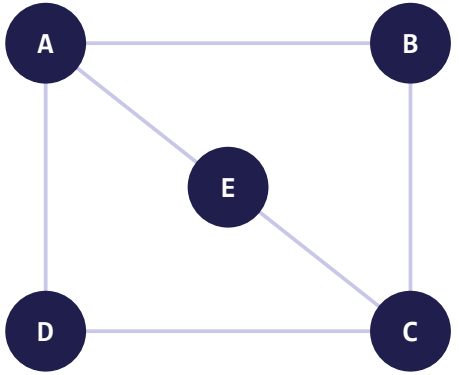


판정 조건
연결 그래프이면서
모든 정점의 차수가 짝수

O(|V| + |E|)

해밀토니안 순환 — 어렵다

모든 정점을 한 번씩 방문하는 경로



판정 조건
알려진 효율적 판정법이
존재하지 않는다

NP-완전

"간선을 한 번씩"과 "정점을 한 번씩" — 단어 하나 차이가 난이도를 가른다

문제가 쉬워 보이는지 어려워 보이는지는 겉모습으로 판단할 수 없다. 그래서 난이도를 재는 엄밀한 도구가 필요하다.

핵심 — 해결은 어렵지만 검증은 쉽다

많은 어려운 문제의 공통점 — 해를 찾는 것은 매우 어렵지만, 주어진 해가 올바른지 확인하는 것은 쉽다.

해를 찾기 — 어렵다

해밀토니안 순환을 직접 찾으려면 사실상 가능한 모든 순서를 시도해 봐야 한다. 정점이 조금만 늘어도 손을 쓸 수 없다.

해를 검증하기 — 쉽다

경로가 주어지면 정점 수가 맞는지, 중복 방문이 없는지, 각 이동이 실제 간선인지만 훑으면 된다 — 다항 시간이다.

이 비대칭성이 바로 NP 클래스의 핵심이다

"찾기는 어렵고 채점은 쉽다"는 성질을 가진 문제들을 한데 묶은 것이 NP다. 다음 두 슬라이드에서 P와 NP를 각각 정의한다.

P 클래스 (polynomial class)

일반 컴퓨터가 다항 시간 안에 해를 직접 찾아낼 수 있는 문제들의 집합 — "코드를 짜서 돌리면 답이 금방 나오는 착한 문제들"

정렬 문제	병합 정렬, 퀵 정렬	`O(n log n)`
최단 경로	다익스트라 알고리즘 (5장)	`O(E log V)`
최소 신장 트리	크루스칼, 프림 (5장)	`O(E log E)`
탐색	이진 탐색	`O(log n)`

지금까지 여덟 장에 걸쳐 배운 알고리즘은 대부분 이 P 클래스 안에 있다.

NP 클래스 (nondeterministic polynomial)

NP는 "Not Polynomial"이 아니다 — NP = Nondeterministic Polynomial

이론적 정의

비결정론적 튜링 머신이 다항 시간 안에 풀 수 있는 문제들의 집합이다.

직관적 정의

누군가 답을 주면, 그 답이 정답인지 다항 시간 안에 검산할 수 있는 문제들의 집합이다.

"답을 찾는 것은 어려울 수 있지만, 답안지 채점은 쉬운 문제"

검증자(verifier)의 존재가 NP의 정의다

해를 알려 주는 증거(certificate)를 함께 받았을 때 그것이 옳은지 다항 시간에 판정할 수 있다면, 그 문제는 NP에 속한다. 해를 어떻게 찾았는지는 묻지 않는다.

NP 이해하기 — 스도쿠 비유

풀기 — 어렵다

빈칸이 수두룩한 거대한 스도쿠를 풀라고 하면 매우 오랜 시간이 걸리거나, 끝내 풀지 못할 수도 있다.

5		3
	?	?
?		9

빈칸을 채우는 경우의 수가 폭발한다

검산 — 쉽다

누군가 답안지를 주면? 각 줄과 각 칸에 숫자가 겹치지 않는지만 확인하면 끝난다.

5	1	3
7	2	4
6	8	9

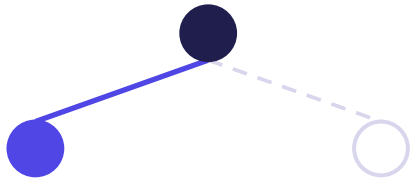
한 번 훑으면 순식간에 검산 가능

푸는 사람과 채점하는 사람이 쓰는 시간이 이렇게 다르다는 것 — 그것이 NP의 전부다.

비결정론적 튜링 머신이란?

결정론적 컴퓨터 (현실)

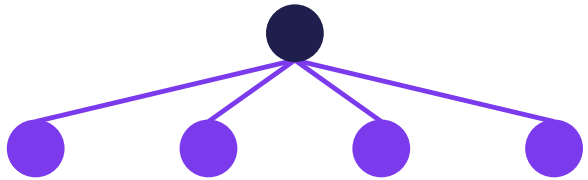
갈림길에서 한쪽을 가보고, 막히면 다시 돌아와서 다른 쪽을 시도한다 — 순차적 탐색이다.



한 번에 한 갈래씩

비결정론적 컴퓨터 (이론)

갈림길에서 몸을 복제해 모든 길을 동시에 탐색한다 — 마법처럼 정답을 찍는 능력이다.



모든 갈래를 동시에

NP = 이 마법의 컴퓨터가 다항 시간에 풀 수 있는 문제

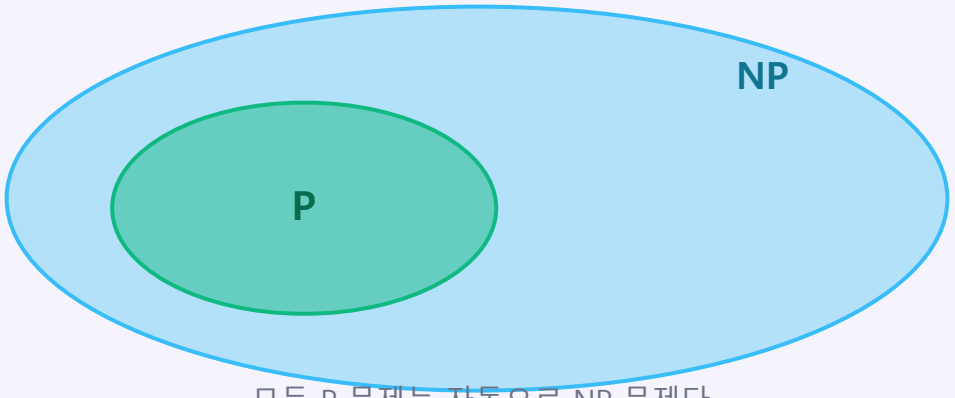
달리 말하면, 정답을 추측(guess)한 뒤 그것을 검증(verify)하는 데 다항 시간이면 충분한 문제다. 두 정의는 완전히 같은 것을 가리킨다.

비결정론적 컴퓨터는 현실에 존재하지 않는다 — 어디까지나 정의를 위한 도구다.

§9.2 P와 NP

P ⊆ NP — 풀 수 있으면 확인도 할 수 있다

내가 1분 만에 풀 수 있는 수학 문제라면, 친구의 답안지도 직접 풀어서 비교하면 그만이다. 검산 역시 금방 끝난다.



모든 P 문제는 자동으로 NP 문제다

왜 포함 관계가 성립하는가

다항 시간에 직접 풀 수 있다면, 증거를 무시하고 그냥 풀어서 답을 대조하면 된다. 검증도 다항 시간이다.

그 역은 어떤가

"모든 NP 문제가 사실은 P 문제인가?" — 이 질문에는 아직 아무도 답하지 못했다. 반세기 넘게 열려 있는 문제다.

알려진 것과 알려지지 않은 것을 구분해 두자

P ⊆ NP 는 증명된 사실이다. P = NP 인지 P ≠ NP 인지는 증명되지 않았다. 대부분의 연구자는 후자를 믿지만, 그것은 믿음이지 증명이 아니다.

P vs NP 문제

P = NP ?

"검산이 쉬우면, 답을 찾는 것도 쉬운가?"

만약 P = NP 라면

TSP, 소인수분해, 배낭 문제가 정렬처럼 순식간에 풀린다.

신약 개발과 물류 최적화에는 산업혁명이 오지만, 현대 암호 체계는 하루아침에 무너진다.

만약 P ≠ NP 라면 (현재의 추측)

검산은 쉽지만 푸는 것은 어려운 문제가 근본적으로 존재한다.

답답한 결론처럼 보이지만, 바로 그 어려움이 암호학의 기반을 지탱한다.

이 질문은 §9.7에서 다시, 밀레니엄 난제라는 이름으로 만나게 된다.

9.3

환원 (Reduction)

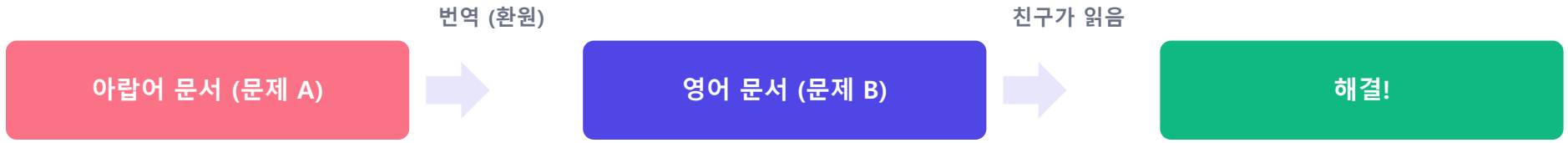
낯선 문제를 익숙한 문제의 형태로 바꾸면,
난이도를 서로 비교할 수 있게 된다.

환원 (reduction)이란?

하나의 문제를 다른 문제로 변환하여 푸는 과정 — "낯선 문제를 익숙한 문제의 형태로 바꾸는 기술"

비유 — 번역기

아랍어 문서를 읽어야 하는데 나는 한국어밖에 모른다. 그런데 영어를 읽는 친구가 있고, 아랍어를 영어로 바꿔 주는 번역기가 있다면?



환원이 성립하면 두 문제의 난이도가 연결된다

B를 푸는 방법이 있다면 A도 풀 수 있다. 뒤집어 말하면, A가 어렵다는 것을 알고 있을 때 B도 최소한 그만큼 어렵다는 뜻이 된다.

다항식 환원 $A \leq_p B$

$A \leq_p B$

"문제 A는 문제 B로 다항식 환원 가능하다" = B를 풀면 A도 풀린다

① 입력 변환 A의 입력을 B의 입력 형식으로 바꾼다. 이 변환 자체가 다항 시간이어야 한다.

② 실행 B를 푸는 알고리즘에 변환된 입력을 넣고 실행한다.

③ 출력 해석 B가 내놓은 답을 A의 답으로 다시 해석한다.

변환 비용이 다항 시간이어야 한다는 조건이 핵심이다

변환에 지수 시간이 든다면 아무리 B가 쉬워도 전체가 빨라지지 않는다.

난이도의 상대성

A ≤p B 라면 → "B는 A보다 적어도 같거나 더 어렵다"

비유 — 일차방정식 vs 이차방정식

이차방정식 계산기가 있다면 a = 0을 넣는 것만으로 일차방정식도 풀 수 있다. 일차방정식은 이차방정식의 특수한 경우이기 때문이다.

일차방정식 ≤p 이차방정식 → 이차방정식이 적어도 그만큼은 어렵다

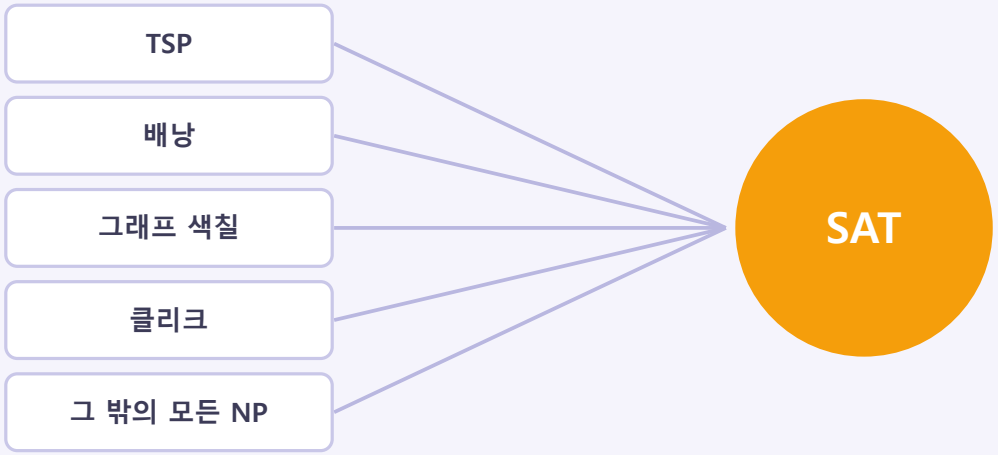
이 논리가 NP-완전성 증명의 핵심 도구다

이미 어렵다고 알려진 문제를 새로운 문제로 환원할 수 있다면, 그 새 문제도 최소한 그만큼 어렵다는 결론이 나온다. 새 문제를 직접 분석할 필요 없이 난이도를 물려받는 것이다.

방향에 주의하자 — "어려운 문제 → 새 문제"로 환원해야 새 문제가 어렵다는 결론이 나온다.

쿡-레빈 정리 (Cook–Levin theorem)

1971년 스티븐 쿡 — 모든 NP 문제는 SAT 문제로 다항 시간 안에 환원할 수 있다.



모든 NP 문제가 SAT 하나로 모인다

SAT 하나만 풀면

모든 NP 문제가 도미노처럼 한꺼번에 해결된다.

난이도의 거미줄

흩어져 보이던 난제들이 사실은 서로 연결되어 있다는 것을 처음으로 보인 정리다. SAT는 모든 NP-완전 문제의 조상이 된다.

기준점이 하나 생기면서, 이후의 증명은 "이미 NP-완전인 문제를 새 문제로 환원한다"는 훨씬 쉬운 작업이 되었다.

충족 가능성 문제 (SAT)

논리 변수 x_1, x_2, … 로 구성된 논리식을 True로 만드는 변수 조합이 존재하는가?

$$(x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_3) \wedge (x_2 \vee \neg x_3) \wedge \dots$$

찾기 — 어렵다

변수가 n개면 조합은 2^n가지다. n = 100이면 경우의 수가 우주의 원자 수를 넘어선다. 완전탐색 말고는 알려진 방법이 없다.

검산 — 쉽다

누군가 변수 값을 알려 주면 식에 대입해 참·거짓을 즉시 확인할 수 있다. 절 개수에 비례하는 시간이면 충분하다.

SAT는 NP에 속하면서 동시에 모든 NP 문제가 환원되는 문제 — 곧 최초의 NP-완전 문제다.

9.4

NP-완전과 NP-난해

NP 안에서 가장 어려운 문제들과,
NP 밖에서 더 어려운 문제들.

NP-난해 (NP-Hard)

"적어도 모든 NP 문제만큼 어렵거나, 그보다 더 어려운 문제" — 모든 NP 문제 L이 H로 다항 시간 안에 환원될 수 있어야 한다.

NP-난해 ≠ NP에 속하는 문제

NP-난해 문제가 반드시 NP에 속할 필요는 없다. "적어도 그만큼 어렵다"는 조건만 요구할 뿐, 검산이 쉬워야 한다는 조건은 없기 때문이다.

정지 문제 해결 자체가 불가능하므로 NP에 속하지 않는다. 그러나 NP-난해이기는 하다.

TSP 최적화 "이것이 진짜 최단 경로인가"를 검산하는 것도 어렵다. 따라서 NP에 속하지 않지만 NP-난해다.

NP-난해는 "난이도의 하한"을 말하는 개념이다 — 위쪽 천장은 정해 주지 않는다.

NP-완전 (NP-Complete)

NP-완전 = NP ∩ NP-Hard

NP 문제 중 가장 어려운 "끝판왕"들의 집합

조건 ① NP에 속한다

답안지가 주어지면 그것이 옳은지 다항 시간에 검산할 수 있어야 한다.

조건 ② NP-난해이다

모든 NP 문제가 이 문제로 다항 시간 안에 환원될 수 있어야 한다.

두 조건을 모두 만족해야 NP-완전이다

①만 만족하면 그냥 NP, ②만 만족하면 NP-난해다. 둘의 교집합에 있는 문제들이 NP 안에서 가장 어려운 자리를 차지한다.

NP-완전의 의의 — 만능 열쇠

1

수천 개의 NP-완전 문제

분야도 형태도 제각각인 문제들이 모두 이 집합 안에 들어 있다.

2

단 하나만 다항 시간에 풀면

환원의 사슬을 타고 나머지 전부가 함께 풀린다.

3

도미노처럼 한꺼번에

모든 NP 문제가 동시에 P로 내려앉는다. 곧 P = NP가 증명된다.

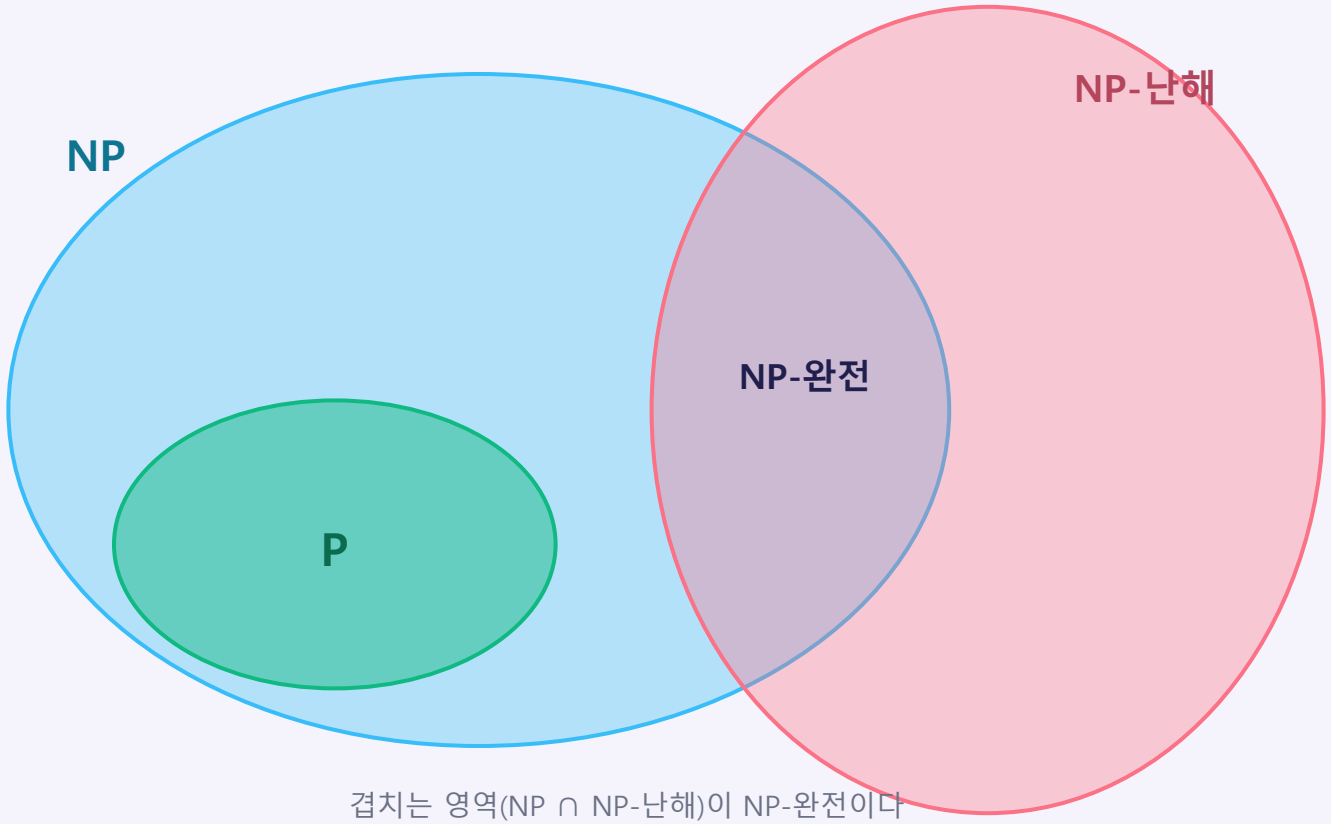
하지만 —

수많은 천재가 수십 년간 매달렸지만, 단 하나의 효율적인 알고리즘도 찾아내지 못했다. 반대로 "불가능하다"는 증명도 나오지 않았다.

실무자에게 이 사실이 주는 의미

내가 만난 문제가 NP-완전임을 보였다면, 그것은 "내 실력이 부족하다"가 아니라 "누구도 못 했다"는 뜻이다. 그때부터는 최적해가 아니라 근사해로 방향을 바꾸는 것이 옳다.

벤 다이어그램 (P ≠ NP 를 가정할 때)



겹치는 영역(NP ∩ NP-난해)이 NP-완전이다

P

풀기도 검산도 쉬운 문제

NP

검산이 쉬운 문제 전체

NP-완전

NP 안에서 가장 어려운 문제

NP-난해

NP보다 어렵거나 같은 문제

P, NP, NP-완전, NP-난해 한눈에

클래스	의미	대표 예
P	풀기도 쉽고 검산도 쉬운 문제	정렬, 검색, 최단 경로
NP	답안지가 있으면 채점이 쉬운 문제	SAT, TSP(결정 문제 버전)
NP-난해	모든 NP 문제보다 어렵거나 같은 문제	정지 문제, TSP(최적화 버전)
NP-완전	NP ∩ NP-난해 — NP 중 최고 난이도	SAT, TSP(결정), 0/1 배낭

같은 문제라도 어떻게 묻느냐에 따라 클래스가 달라진다

TSP를 "비용 K 이하인 경로가 있는가"로 물으면 NP-완전이고, "최단 경로를 구하라"로 물으면 NP-난해다. 검산의 난이도가 달라지기 때문이다.

실무에서는 "이 문제가 NP-완전인가"만 확인하면 충분한 경우가 대부분이다.

9.5

NP-완전 문제의 대표 선수들

9.5

분야는 제각각이지만 난이도는 하나다.
The Zoo of NP-Complete Problems

대표 선수 ① SAT — 충족 가능성 문제

모든 NP-완전 문제의 조상 — 1971년, 최초로 NP-완전임이 증명된 문제다. 실무에서는 SAT 솔버가 놀랄 만큼 잘 동작하는데, 실제 입력이 최악의 경우가 아니기 때문이다.

논리식을 참으로 만드는 변수 조합이 존재하는가?

이론적 기준점 모든 NP 문제를 SAT로 환원할 수 있다. 이후의 NP-완전 증명은 모두 여기서 출발한다.

경우의 수 변수 n개면 2^n가지 조합. 완전탐색 외에 알려진 방법이 없다.

검증 답안지가 주어지면 대입 한 번으로 즉시 확인된다.

응용 회로 설계, AI 추론, 소프트웨어 검증, 암호학 전반에 쓰인다.

대표 선수 ② TSP — 외판원 순회 문제

결정 문제 버전 — "총비용이 K 이하인 순회 경로가 존재하는가?"

검산 — 쉽다

경로가 주어지면 모든 도시를 한 번씩 지났는지 확인하고 비용 합이 K 이하인지만 더해 보면 된다.

풀기 — 극도로 어렵다

가능한 경로의 수가 n!에 이른다. n = 50이면 완전탐색은 물론 8장의 분기한정으로도 감당할 수 없다.

5장·8장에서 계속 만나 온 그 문제다

5장에서는 그리디로 근사해를 구했고, 8장에서는 분기한정으로 최적해를 구했지만 도시 수에 한계가 있었다. 그 한계의 정체가 바로 NP-완전이다.

대표 선수 ③ 0/1 배낭 채우기 문제

결정 문제 버전 — "배낭 용량을 넘지 않으면서 가치 합이 목표치 이상인 조합이 존재하는가?"

물건을 쪼갤 수 없다는 조건이 핵심이다. 넣거나(1) 말거나(0), 둘 중 하나뿐이다.

NP에 속한다 담을 물건 목록이 주어지면 무게 합과 가치 합을 더해 조건을 확인하는 것은 쉽다.

NP-완전이다 최적 조합을 직접 찾는 데는 지수 시간이 걸린다.

실용적 접근 6장에서 본 동적 계획법은 O(nW)로 동작한다. 다만 W가 커지면 함께 커지는 의사 다항 시간이다.

DP가 있다고 P인 것은 아니다 — O(nW)의 W는 입력의 "값"이지 "길이"가 아니기 때문이다.

§9.5 대표 선수들

대표 선수 ④~⑥ 그래프 문제들

클리크 (clique)

k명이 모두 서로 친구인 모임이 존재하는가? 그래프에서 서로 전부 연결된 정점 k개를 찾는 문제다.

정점 커버 (vertex cover)

k개의 정점만으로 모든 간선을 커버할 수 있는가? 감시 카메라 배치 같은 문제로 나타난다.

해밀턴 경로

모든 정점을 정확히 한 번씩 방문하는 경로가 존재하는가? TSP의 뿌리가 되는 문제다.

이 셋은 서로 환원 가능하다 — 하나를 풀면 나머지도 풀린다.

§9.5 대표 선수들

NP-완전 문제들의 공통점

1

겉모습은 제각각

논리식, 지도, 배낭, 그래프 — 분야가 전혀 다르지만 환원으로 보면 동일한 난이도다.

2

검산은 쉽다

답안지가 주어지면 다항 시간에 채점할 수 있다. 그러나 해를 찾는 효율적 알고리즘은 아직 없다.

3

하나 풀면 전부 풀린다

환원의 사슬로 묶여 있어 어느 하나가 무너지면 도미노처럼 전부 무너진다.

현업에서 이런 문제를 만나면?

완벽한 최적해를 찾겠다는 목표를 내려놓고, 근사해를 찾는 전략으로 전환하는 것이 옳다. 이것은 포기가 아니라 문제의 성질에 맞춘 선택이다.

NP-완전임을 아는 것 자체가 실무적 가치가 있다

"조금만 더 고민하면 다항 시간 해법이 나올 것 같다"는 헛된 시도에 시간을 쓰지 않게 해 주기 때문이다.

9.6

NP-완전 문제 대처법

9.6

완벽함을 포기하고 실용성을 얻는 지혜.

§9.6 NP-완전 대처법

최적해를 포기하고 근사해를 찾자

쿠팡 배송 트럭은 계산이 끝나기를 기다려 주지 않으며, 트레이딩 알고리즘은 0.1초 안에 판단해야 한다.

근사 해 (approximation solution)

"가장 짧은 경로" 대신 "꽤 짧은 경로"를 찾는다.
"최고의 배치" 대신 "나쁘지 않은 배치"로 만족한다.

100점 답안을 100년 뒤에 < 90점 답안을 10분 안에

"완벽한 답"이 목표가 아니라 "쓸 만한 답을 제때"가 목표가 된다

문제의 난이도가 바뀌지 않는다면, 바꿀 수 있는 것은 우리가 요구하는 품질뿐이다.

§9.6 NP-완전 대처법

근사 알고리즘 (approximation algorithm)

"최악의 경우라도 최적해의 k배를 넘지 않는다" — 수학적으로 보장되는 품질. k를 근사 비율(approximation ratio)이라 한다.

예시 — TSP에 대한 MST 기반 근사

① 최소 신장 트리를 만든다 — 크루스칼이나 프림으로, P 클래스에서 끝난다.
② 그 트리를 따라 순회하는 경로를 만들면, 비용이 최적 TSP 비용의 2배를 넘지 않는다는 것이 수학적으로 증명된다.

근사 비용 ≤ 2 × 최적 비용

"아무리 재수가 없어도 최적보다 2배 이상 돌아가지는 않는다"

보장이 있다는 점이 휴리스틱과 근사 알고리즘을 가르는 결정적 차이다.

§9.6 NP-완전 대처법

휴리스틱과 메타휴리스틱

수학적 보장 없이 실전 경험과 직관에 의존하는 방법 — 대신 훨씬 넓은 문제에 적용된다.

모의 담금질 (simulated annealing)

금속을 천천히 식히는 물리 현상에서 착안했다.

처음에는 나쁜 해도 받아들여 국소 최적에서 빠져나오고, 나중에는 좋은 해만 골라 전역 최적으로 수렴한다.

응용 — 반도체 레이아웃 설계

유전 알고리즘 (genetic algorithm)

생물의 진화 과정을 모방했다.

무작위 해를 여러 개 만들어 교배시키고 돌연변이를 섞은 뒤, 우수한 개체만 살아남기를 반복한다.

응용 — 게임 AI, 함수 최적화

품질 보장이 없으므로, 결과를 반드시 실측으로 검증해야 한다.

9.7

P vs NP 문제

상금 100만 달러가 걸린 밀레니엄 난제.
인류 지성의 최전선.

9.7

§9.7 P vs NP 문제

밀레니엄 난제 — P vs NP

2000년 클레이 수학연구소가 선정한 7대 밀레니엄 난제 중 하나 · 상금 100만 달러

"답을 확인하는 것이 쉬운 모든 문제는, 답을 찾는 것도 쉬운가?"

만약 P = NP 라면?

모든 암호 체계가 붕괴한다 · AI 특이점이 도래한다 · 수학 정리의 자동 증명이 가능해진다

현재의 결론

대다수 연구자는 P ≠ NP 라고 본다 · 그러나 수학적 증명은 아직 없다 · 여전히 열린 문제로 남아 있다

반세기 넘게 열려 있는 질문이자, 컴퓨터 과학이 가진 가장 큰 미해결 문제다.

SUMMARY

9장 정리

다항 시간 vs 지수 시간

풀 수 있는 문제와 풀 수 없는 문제를 가르는 실질적인 경계선이다.

P 클래스

다항 시간 안에 해를 직접 찾을 수 있는 문제들. 지금까지 배운 알고리즘 대부분이 여기 있다.

NP 클래스

다항 시간 안에 검산할 수 있는 문제들. P ⊆ NP 는 증명된 사실이다.

환원 (reduction)

문제 A를 B로 변환해 난이도를 비교한다. NP-완전성 증명의 유일한 도구다.

NP-완전과 NP-난해

NP-완전 = NP ∩ NP-난해. 하나만 다항 시간에 풀리면 전부 풀린다.

대처법과 P vs NP

근사 알고리즘과 휴리스틱으로 실용적으로 해결한다. 그리고 P vs NP는 여전히 열려 있다.

CHAPTER 09

Q & A

"문제의 한계를 아는 것이 곧 해결의 시작이다"
풀리지 않는 이유를 아는 것도 하나의 답입니다.