ERC 여름 워크샵
2012년 7월 27일

NP-Completeness 와 Cook-Levin 정리

정교민, 응용 알고리즘 연구실
Computer Science, KAIST
Joint Appointments in Math and EE depts.

목차

- Algorithm 의 개념
- P, NP, 그리고 NP-complete (NP-싹쓸이)
- SAT problem
- NP-complete 문제의 예
- Cook-Levin 정리의 증명 스케치

Algorithm 의 기본 개념

- An **algorithm** is a rule for solving a problem using finitely many pre-determined instructions.

- 예를 들어, 10진수 두 자연수의 곱을 구하는 문제를 위해 우리는 다음과 같은 알고리즘을 사용한다.

$$23 \times 45$$
$$= 23 \times (40 + 5)$$
$$=(20 + 3) \times 40 + (20 + 3) \times 5$$
$$= 800 + 120 + 100 + 15 = 1035$$

- 다른 예: 주어진 자연수가 소수(prime number)인가?
  - 알고리즘: Eratosthenes 의 채

결정 불가능한 문제

잘 정의된 문제라도, 그 문제를 푸는 알고리즘이 항상 존재하는 것은 아니다. (결정 불가능함 문제; undecidable problem).

1. **Halting problem**

2. **Hilbert's 10th problem :** Does there exist an algorithm to determine whether a given Diophantine equation has an integer solution?

Diophantine equation : a polynomial equation that allows the variables to be integers only

Ex1: $3x + 4y = 1$

Ex2: $x^2 - 5y^2 = 1$

Resolved: Matiyasevich's theorem(1970) implies that there is no such algorithm.

효율적인 알고리즘?

문제: compute the GCD(Greatest Common Divisor) of integers A and B with A > B >= 0.

Euclid Algorithm utilizes the theorem that GCD(A,B)=GCD(B, A-kB) for any integer k.

Euclid Algorithm (A,B)

a. If B = 0 then output GCD(A,B) = A

b. If B > 0 then

let C = A % B (remainder of A divided by B)

Output Euclid Algorithm (B,C)

Example: GCD(120,85)=GCD(85,35)=GCD(35,5)=GCD(5,0)=5

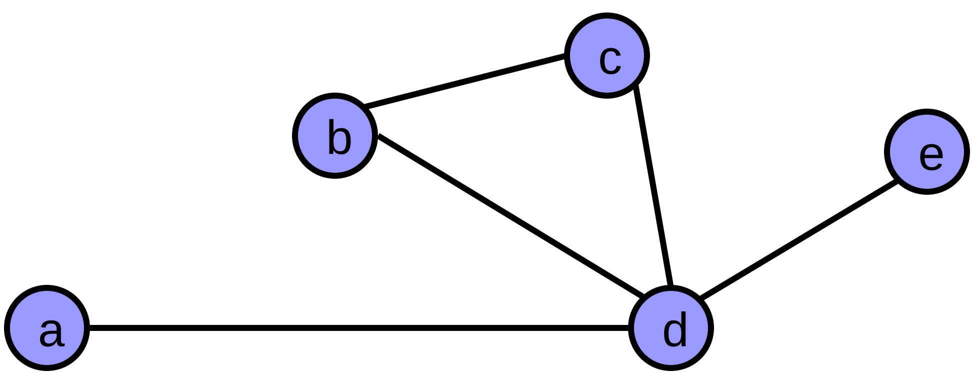
120 = 85*1 + 35, 85 = 35*2 + 15,

35 = 15*2 + 5, 15 = 5*3 + 0

Hamiltonian Path Problem

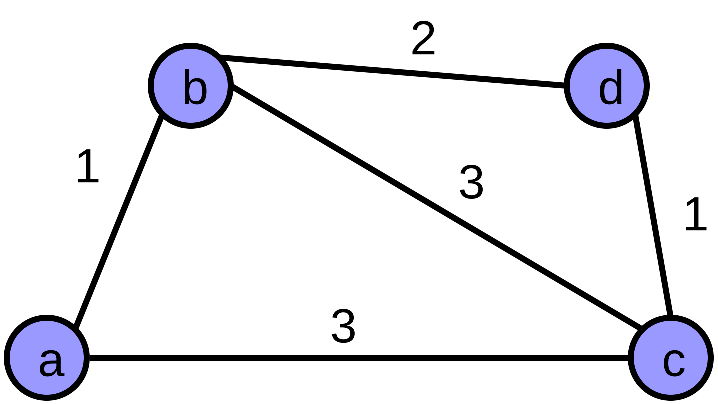
- Given a graph G, a Hamiltonian path is a path which visits each vertex exactly once.
- Hamiltonian Path Problem
  - Decide whether a graph G has a hamiltonian path or not

- Eulerian path 문제 와 달리 판별이 어려움



Traveling Salesman Problem (TSP)

- Given a graph with non-negative edge distance, find a shortest possible Hamiltonian path.



Polynomial Time Reduction

- Intuitively: If R reduces in polynomial time to Q, Q is "a more general problem than" R

- A problem R can be reduced to another problem Q in polynomial time if
  - Any instance of R can be solved in polynomial time by using a polynomial time oracle for Q.

- Denote $R \leq_p Q$ (Q가 R보다 더 일반적이고 어려운 문제)

- Ex: Hamiltonian Path can be reduced to TSP.

Algorithm 수행이 필요한 주요 문제들

- **결정 문제 (Decision Problem)**
  - **The answer is Yes or No. Ex: Hamiltonian path problem.**
- **최적화 문제 (Optimization Problem)**
  - **Goal is to compute the optimal value, or the labeling**
  - **Ex: TSP problem (optimal value: length of the shortest travel, labeling: the visiting order of the optimal travel)**
  - **Can be reduced to a decision problem by binary search**
- **계산 문제 (Computation Problem)**
  - **Ex: Compute a solution of an equation**
  - **Ex: Compute eigenvalues of a matrix**
  - **Can be reduced to a decision problem by binary search**

Turing Machine

- A Turing Machine (**TM**) is a device which manipulates symbols on an (infinite) tape according to a finite amount of manipulation rules.
  - The tape corresponds to the memory
  - The manipulation rules corresponds to a program
  - Each rule corresponds to each “line” of a program
  - TM has stopping rules, and TM outputs either yes or no when it stops
  - Hence TM solves a decision problem

Turing Machine

- Turing Machine 은 특정 프로그램을 돌리고 있는 컴퓨터

- Church-Turing thesis: every decision problem that is computable by an “algorithm” can be computed by some Turing Machine

- 참고 자료: Introduction to the Theory of Computation, 2nd edition by Michael Sipser, Course Technology

P and NP

- **P** = set of decision problems that can be solved in polynomial step of the input bit size by a TM
- **NP** = set of decision problems for which any *yes* instance has some "proof" that verifies the problem to be yes in polynomial step (ex: Hamiltonian path)
  - 다른 정의: NP = set of decision problems that can be solved in polynomial step by a Nondeterministic TM
- **P** is similar to a problem that a normal people can find its solution easily.
- **NP** is a problem that a normal people can grade whether other person's solution is correct or not easily.
- **P** $\subseteq$ **NP**
- A big question: Does **P ≠ NP**?

NP-Hard and NP-Complete (NP 싹쓸이)

- Definition of NP-Hard and NP-Complete :
  - If all problems $R \in$ **NP** are reducible to Q, then Q is NP-Hard
  - We say Q is NP-Complete (NP 싹쓸이) if Q is NP-Hard and $Q \in$ **NP**
- If $R \leq_p Q$, and R is NP-Complete, and $Q \in$ **NP, then** Q is also NP- Complete
- 참고: PSPACE-Complete, EXPTIME-Complete

Proving NP-Completeness

- Is there at least one NP-complete problem?
  - Cook-Levin Theorem shows that SAT problem is NP-Complete
- 특정 문제 Q 가 NP-Complete 임을 증명하려면?
  - Prove Q $\in$ **NP**
  - Pick a known NP-Complete problem R
  - Reduce R to Q
    - Prove the reduction runs in polynomial time

SAT Problem

- 정의
  - *Boolean variables*: variables that can take the values TRUE(1) or FALSE(0)
  - *Boolean operations*: AND, OR, and NOT
  - *Boolean formula*: an expression involving Boolean variable and Boolean operations

SAT Problem

- 정의
  - satisfiable: if some assignment of 0s and 1s to the variables make the Boolean formula True

- Example of a satisfiable Boolean formula
  - $\varphi=(\neg x \wedge y) \vee (x \wedge \neg z)$
  - x=0, y=1, and z=0

SAT Problem

- Definition (SAT Problem)
  - SAT = {<φ>|φ is a satisfiable Boolean formula}.

- Given a Boolean formula, is it satisfiable?

- Cook-Levin Theorem
  - SAT∈NPC

Corollary: 3SAT∈NPC

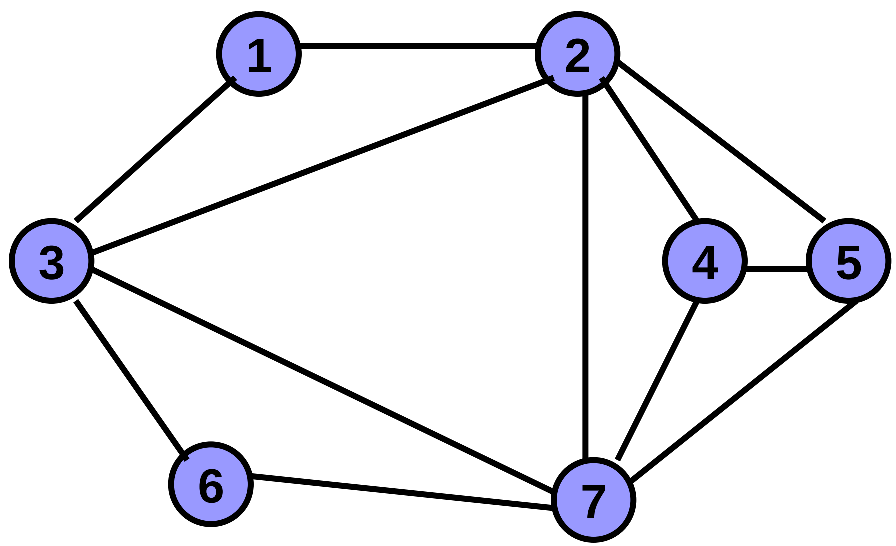
- Any Boolean formula can be expressed as a CNF (Conjunctive Normal Form)
  - Ex: $(\neg x \vee z) \wedge (x \vee y \vee w) \wedge (y \vee \neg z \vee \neg x)$
- CNFs can be converted into CNFs with three literals per clause.
- Examples
  - $(x_1 \vee x_2) \equiv (x_1 \vee x_2 \vee x_2)$
  - $(x_1 \vee x_2 \vee x_3 \vee x_4) \equiv (x_1 \vee x_2 \vee z) \wedge (\neg z \vee x_3 \vee x_4)$
  - $(x_1 \vee x_2 \vee x_3 \vee x_4 \vee x_5) \equiv$
    $(x_1 \vee x_2 \vee z_1) \wedge (\neg z_1 \vee x_3 \vee z_2) \wedge (\neg z_2 \vee x_4 \vee x_5)$

Corollary: 3SAT $\in$ NPC

- Hence, a SAT problem can be converted into a equivalent 3SAT problem (in polynomial time).
- That is, SAT $\leq_P$ 3SAT.
- Since SAT $\in$ NPC, and 3SAT $\in$ NP, we have 3SAT $\in$ NPC.

Example of NP-complete problem: Clique

- CLIQUE = { <G,k> | G is a graph with a clique of size k }
- A clique is a subset of vertices that are all connected.
- Easy: CLIQUE ∈ NP.

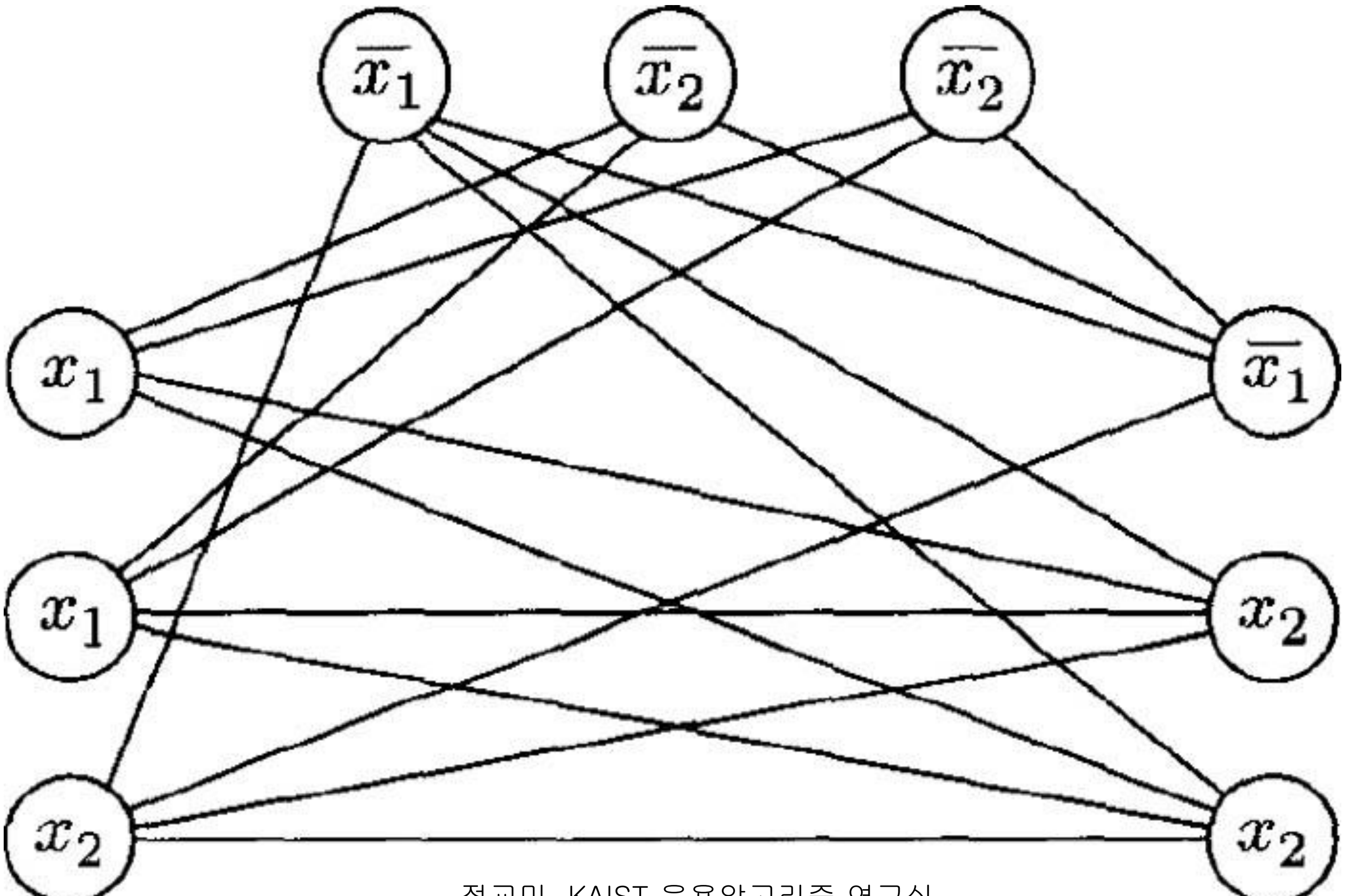


Reduction of 3-SAT to Clique

- Pick an instance of 3-SAT, Φ, with *k* clauses
- Make a vertex for each literal
- Connect each vertex to the literals in other clauses that are not the negation
- Any k-clique in this graph corresponds to a satisfying assignment

An Example

$$\phi = (x_1 \vee x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2} \vee \overline{x_2}) \wedge (\overline{x_1} \vee x_2 \vee x_2)$$



Proof of Cook-Levin Theorem

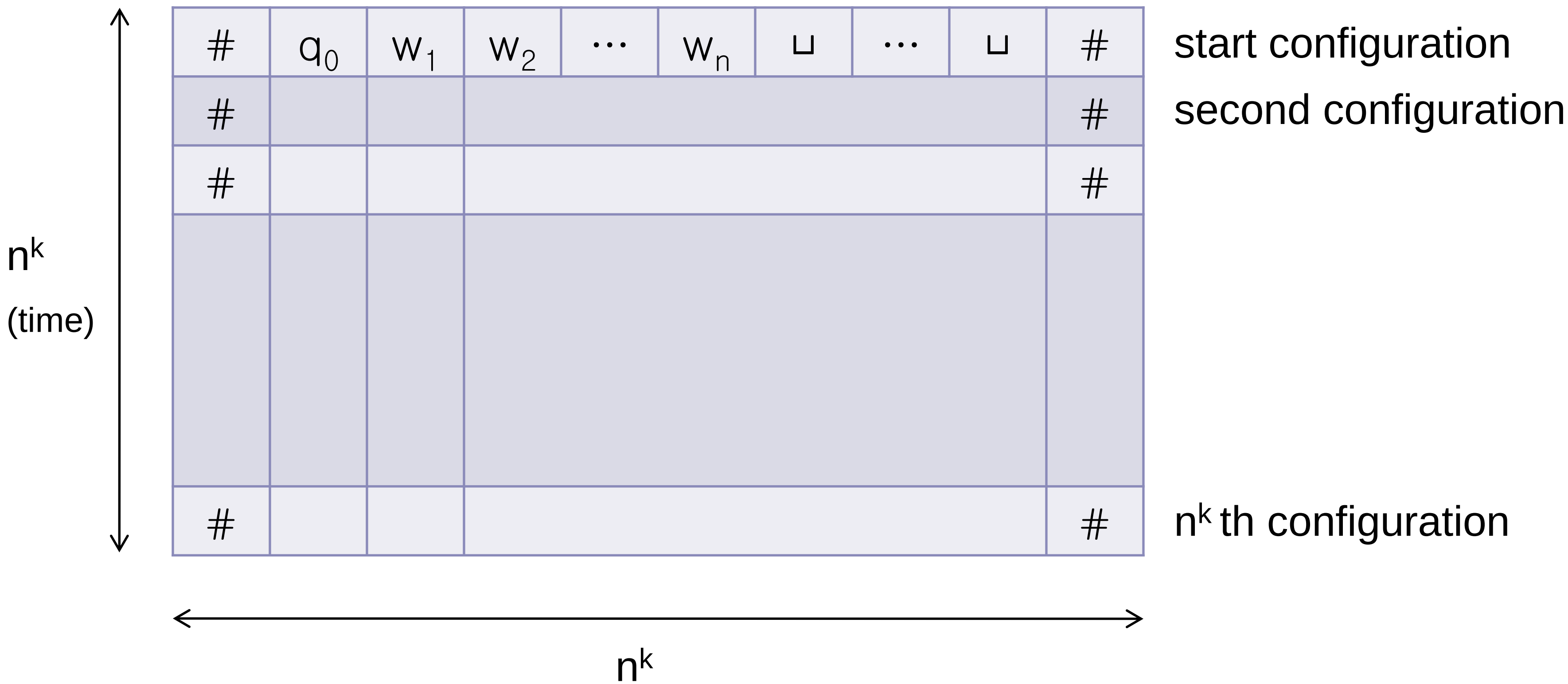
- What to prove is …
  - SAT∈NP (clear)
  - ∀A∈NP, $A \leq_P SAT$
    - The proof shows that for each problem A∈NP and a given input w to A, it is possible to produce a Boolean formula F (depending on A and w) in polynomial time of |w| so that F is satisfiable if and only if w is a yes instance of A.

Proof of Cook-Levin Theorem

- Proof
  - A: a Problem
  - w: an input
  - N: NP Turing machine that decides A
    - Assume that N decides whether $w \in A$ in n^k steps, for some constant k.
  - NP Turing machine : consists of states, tape alphabet, move rule, accept rule, and reject rule.

Proof of Cook-Levin Theorem

- Consider $n^k \times n^k$-cell **tableau of tape change history** for input w. (k is some constant)



Proof of Cook-Levin Theorem

- cell[i,j]: the cell located on the ith row and the jth column.

- variables of the Boolean formula: $x_{i,j,s}$.
- $x_{i,j,s}$: true if cell[i,j] is the symbol s.

Proof of Cook-Levin Theorem

- The tableau, without any restriction, may contain many invalid series of configurations.
  - E.g. cells containing multiple symbols, not starting with the input w, neighbor configurations not corresponding the transition rules, not resulting in the accept state, and etc.
- Produce a Boolean formula which
  - Forces the tableau to be valid according to the state change rules of the Nondeterministic Turing machine
  - And at least one of the configuration results in the accept state.

Proof of Cook-Levin Theorem

- One cell can contain exactly one symbol among a state, a tape alphabet, and #. ➔ (φ_{cell})
- The first configuration should correspond to input w. ➔ (φ_{start})
- A configuration is derivable from the immediately previous configuration according to the transition rule of the Nondeterministic Turing machine. ➔ (φ_{move})
- There must exist a cell containing the accept state. ➔ (φ_{accept})

- $\varphi = \varphi_{cell} \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}$

Proof of Cook-Levin Theorem

- $\varphi = \varphi_{cell} \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}$

$$\phi_{cell} = \bigwedge_{1 \le i,j \le n^k} \left[\left(\bigvee_{s \in C} x_{i,j,s} \right) \wedge \left(\bigwedge_{\substack{s,t \in C \\ s \ne t}} \left(\overline{x_{i,j,s}} \vee \overline{x_{i,j,t}} \right) \right) \right]$$

Each cell contain at least one symbol.

Each cell contain at most one symbol.

Proof of Cook-Levin Theorem

- $\varphi = \varphi_{cell} \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}$

$$\phi_{start} = x_{1,1,\#} \wedge x_{1,2,q_0} \wedge x_{1,3,w_1} \wedge x_{1,4,w_2} \wedge \ldots \wedge x_{1,n+2,w_n} \wedge x_{1,n+3,\sqcup} \wedge \ldots \wedge x_{1,n^k-1,\sqcup} \wedge x_{1,n^k,\#}$$

Each cell of the first row has a symbol corresponding to the start configuration with input w.

Proof of Cook-Levin Theorem

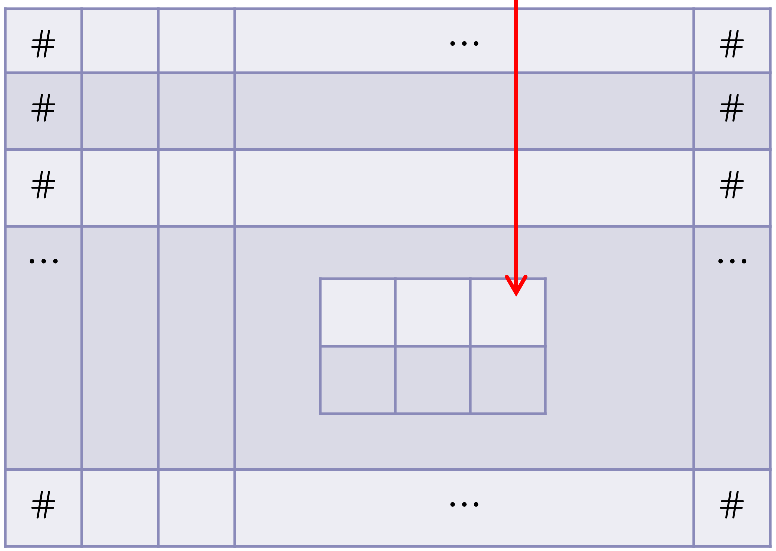
- $\varphi = \varphi_{cell} \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}$

$$\phi_{accept} = \bigvee_{1 \le i,j \le n^k} x_{i,j,q_{accept}}$$

At least one cell is the accept state.

Proof of Cook-Levin Theorem

- $\varphi = \varphi_{cell} \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}$
- φ_{move} checks whether every 2×3 window of the tableau is legal according to the transition rule of the Nondeterministic Turing machine.



Proof of Cook-Levin Theorem

- $\varphi = \varphi_{cell} \wedge \varphi_{start} \wedge \varphi_{move} \wedge \varphi_{accept}$

- Now, we have obtained φ as we wished.
- I.e. φ is satisfiable, iff $w \in A$.
- Also, size of φ is polynomial in n. Hence it is a polynomial time reduction.
- Therefore, SAT is NP-complete.

참고 자료

- Cook-Levin 정리의 자세한 증명은 Introduction to the Theory of Computation, Michael Sipser, Course Technology, 2nd edition 의 Theorem 7.37 에 있습니다.

감사합니다.