NP & Code Base

https://youtu.be/4tKNQg9T734

최승주

HSU 한성대학교 HANSUNG UNIVERSITY

CryptoCraft LAB
https://crypto.modoo.at

Contents

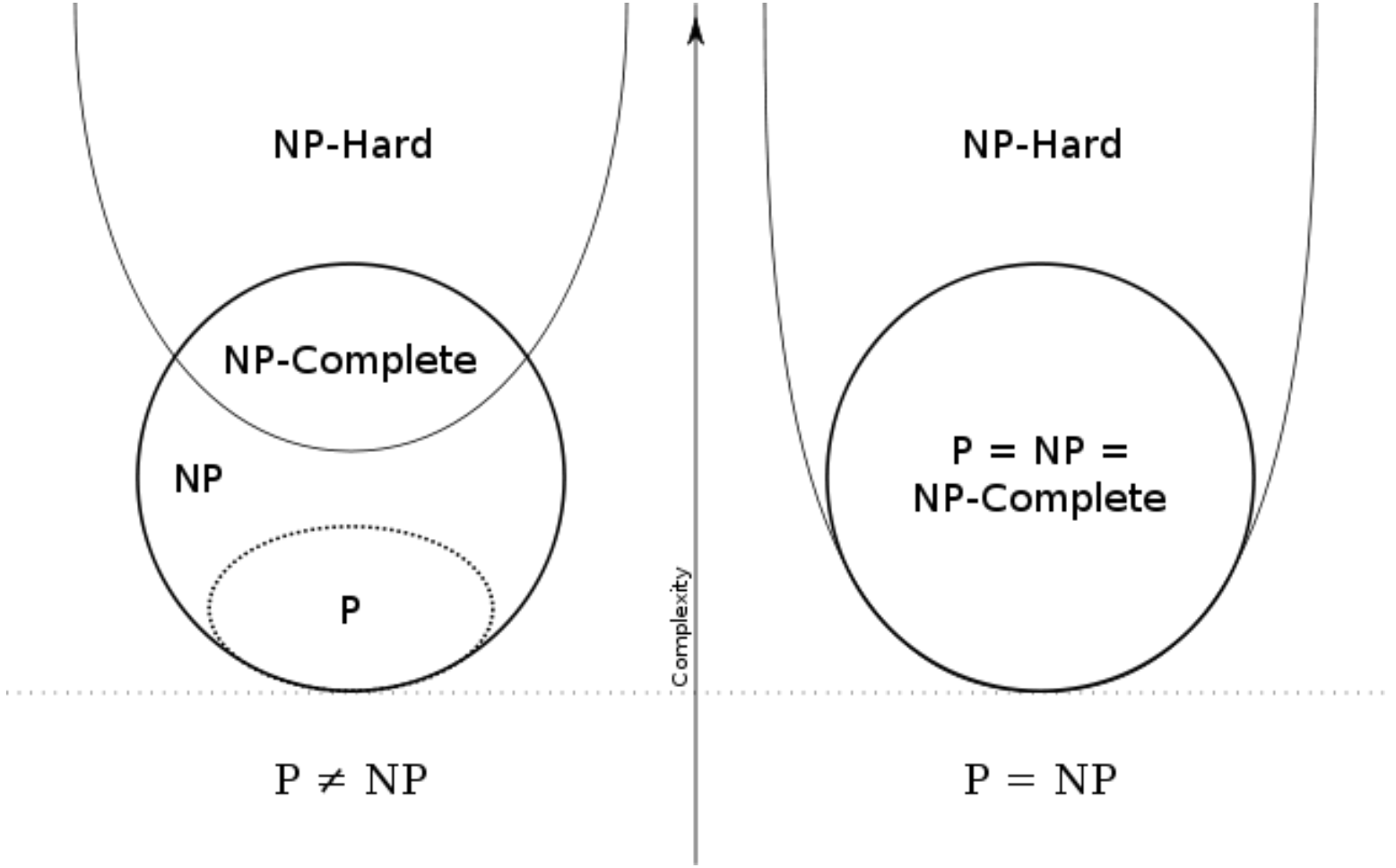
CryptoCraft LAB

1, 2 장의 목표

- NP 완전 문제를 배워보자

  야호

NP 문제



이게 뭔 그림일까
넘모 궁금하다

NP 문제 – Class P

- 어떤 문제에 대해서 Polynomial Time Algorithm이 존재하면

  해당 **문제**는 클래스 P에 속한다.

  - 알고리즘이 클래스 P에 속하는 것이 아닌, 해당 **문제**가 속하는 것

  - Polynomial Time Algorithm

    n^k 형태로 최악의 시간 복잡도가 정의되는 **알고리즘**

    ex) 선택정렬, 삽입정렬, 버블정렬, 퀵 정렬 등등

    n^2의 최악의 시간 복잡도

NP 문제 – Class P

- 어떤 문제에 대해서 Polynomial Time Algorithm이 존재하면
  해당 **문제**는 클래스 P에 속한다.

  - N^k의 형태, 즉 다항시간에 해당 문제에 대한 solution을 찾을 수 있으면 P에 속함
  - n^2, n^3, n^4… 등 다항시간이면 **효율적**(efficient)인 알고리즘으로 본다.

NP 문제 – Class P

- 지금까지의 문제들은 시간이 오래 걸린다 해도 풀리기는 함
- 세상에는 풀지 못하는 문제들이 존재함
  - 풀 수는 있지만, Polynomial Time Algorithm이 개발되지 않은 문제
  - 다항시간에 풀리지 않음

대체 이런 **어려운 문제**는 무엇인가?

NP 문제 – Class NP

- 문제를 해결하는 Non-Deterministic Polynomial Time algorithm이 존재할 시, 해당 문제는 클래스 NP에 속함

  - 문제가 속하는 것이지 알고리즘이 속하는 것은 아님

"어떤 문제에 대해서 Polynomial Time Algorithm이 존재하면 해당 **문제**는 클래스 P에 속한다." - P 클래스 -

NP 문제 – Class NP

- 문제를 해결하는 **Non-Deterministic** Polynomial Time algorithm이 존재할 시, 해당 문제는 클래스 NP에 속함

  - Non-Deterministic: 비 결정적

  비 결정적 다항시간 알고리즘

  - 같은 인풋이 들어가도 다른 아웃풋을 낼 수 있는 알고리즘

NP 문제 – Class NP

- 비 결정적 다항시간 알고리즘

  - 같은 인풋이 들어가도 다른 아웃풋을 낼 수 있는 알고리즘

왜 다항시간인가?

- 어떤 어려운 문제를 푸는 상황
- 다항시간 안에 아웃풋을 만들어 낼 수도 있는데,

  어떠한 경우에는 다항시간 안에 끝나지 않는 경우도 있는 상황

NP 문제 – Class NP

- 비 결정적 다항시간 알고리즘
  - 같은 인풋이 들어가도 다른 아웃풋을 낼 수 있는 알고리즘
  - **비결정적** 다항시간 알고리즘이 존재할 경우 클래스 NP에 속함

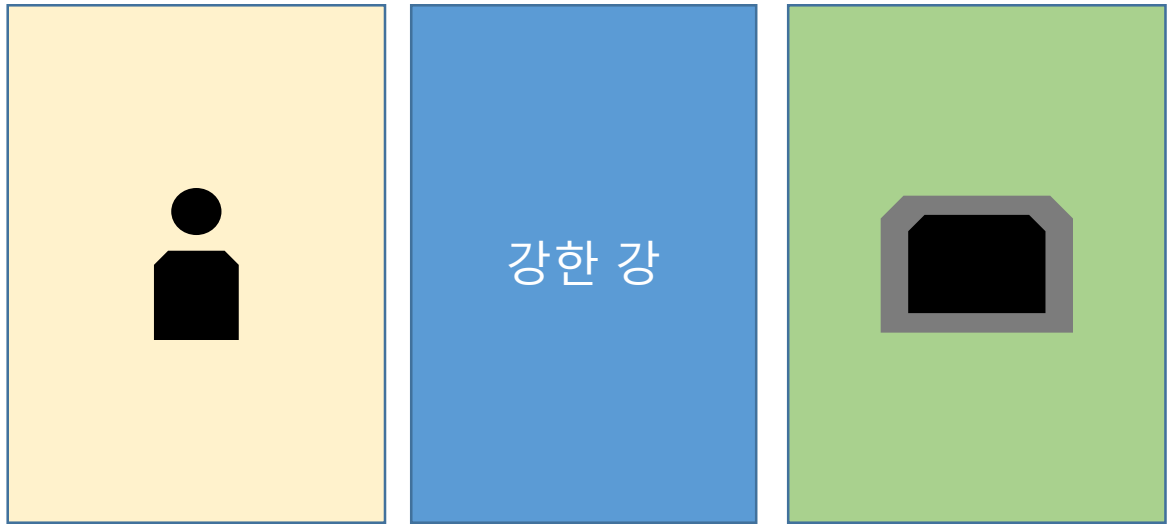
NP 문제 – Class NP

- 비 결정적 다항시간 알고리즘
  - 어떤 certificate가 다항시간(Polynomial Time)에 verify할 수 있으면, 그 문제는 클래스 NP에 속한다.

NP 문제 – Class NP

- 비 결정적 다항시간 알고리즘

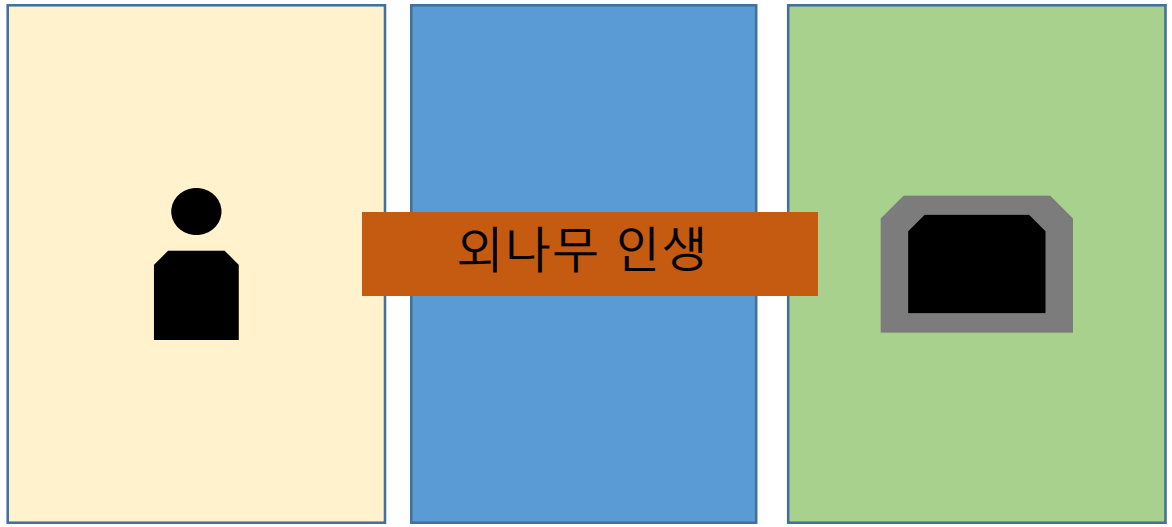
  예시) 재미있는 예시



NP 문제 – Class NP

- 비 결정적 다항시간 알고리즘

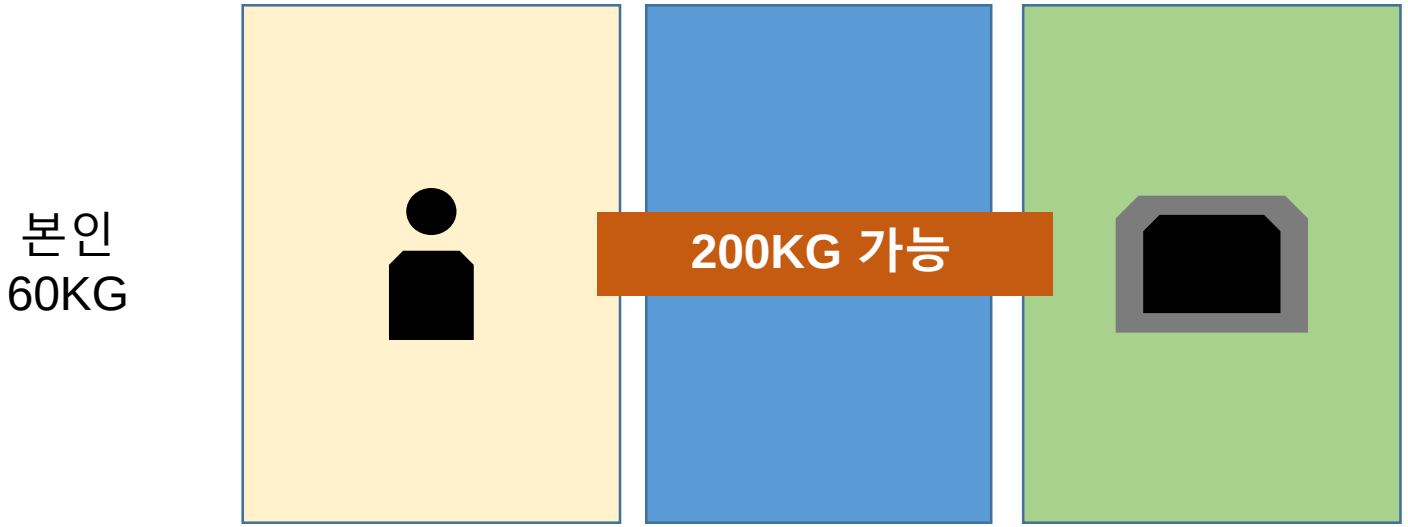
  예시) 재미있는 예시



NP 문제 – Class NP

- 비 결정적 다항시간 알고리즘

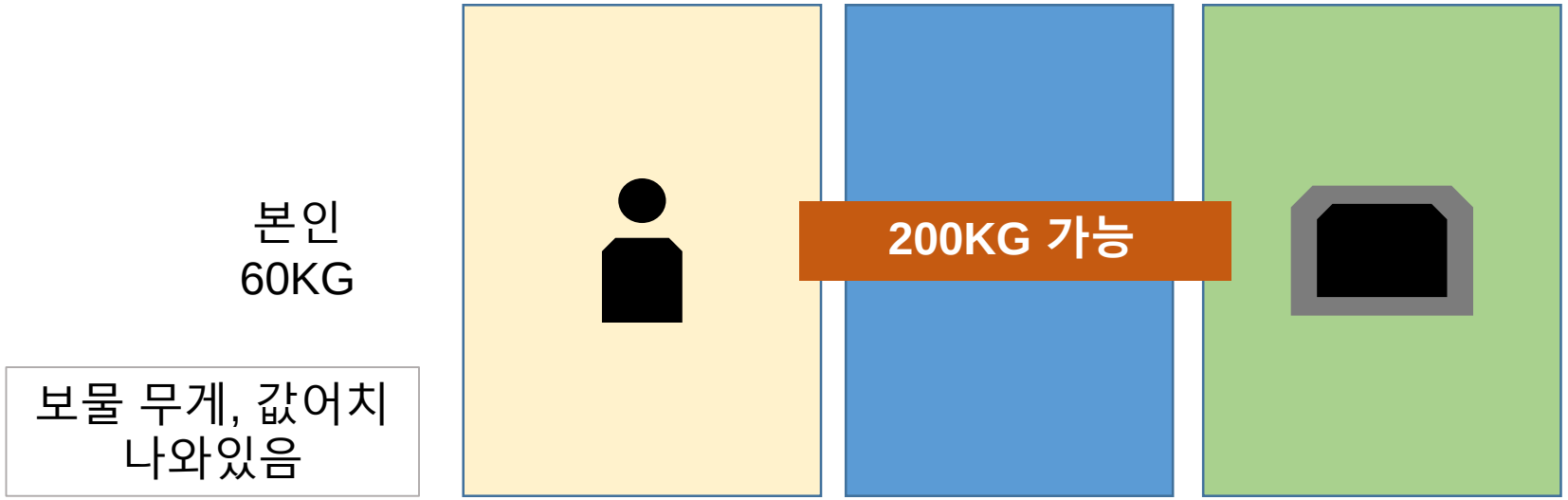
  예시) 재미있는 예시



NP 문제 – Class NP

- 비 결정적 다항시간 알고리즘

  예시) 재미있는 예시

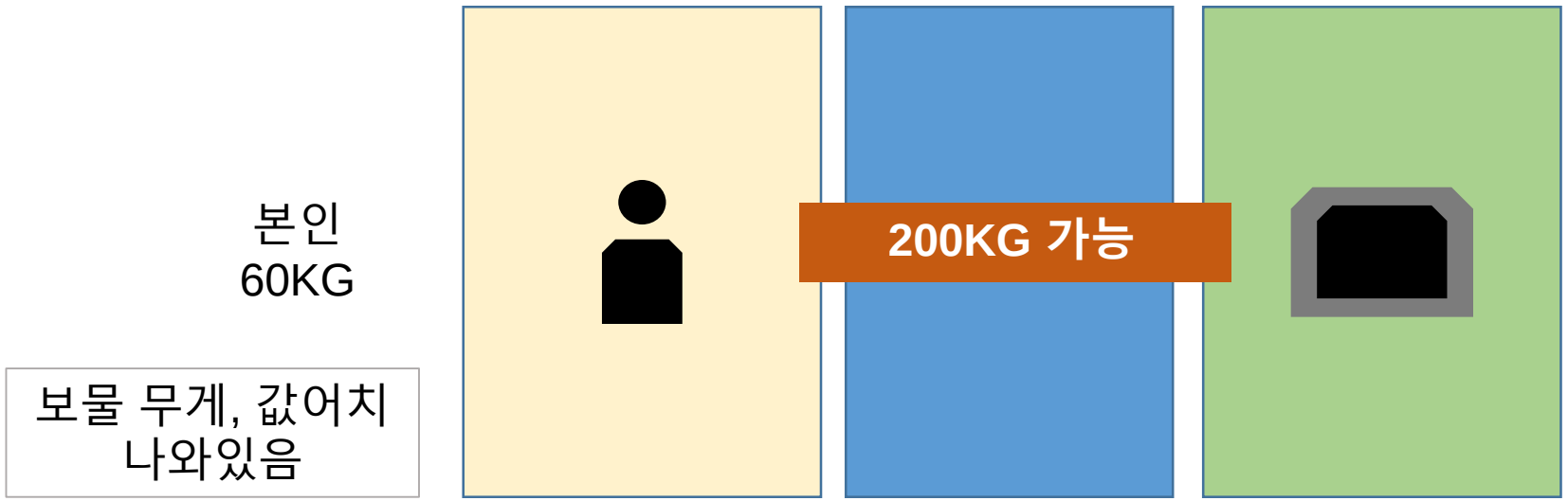


NP 문제 – Class NP

- 비 결정적 다항시간 알고리즘

  예시) 2억 이상, 200kg 넘지 않는 보물의 조합이 있나 없나

Decision Problem

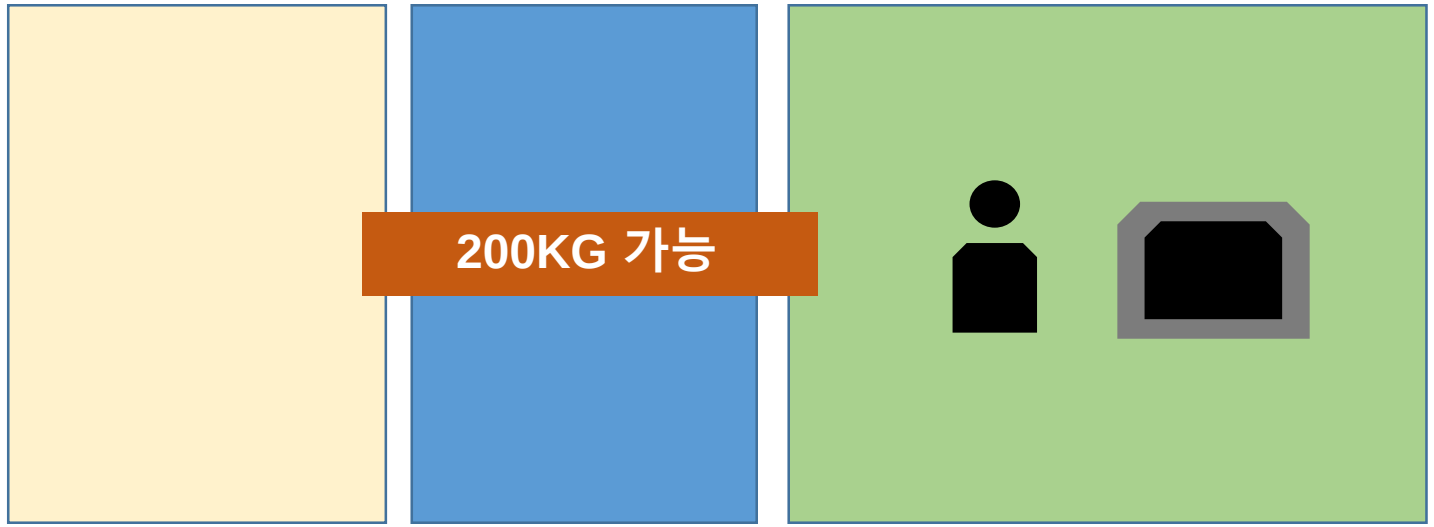


NP 문제 – Class NP

- 비 결정적 다항시간 알고리즘

  예시) 2억 이상, 200kg 넘지 않는 보물의 조합이 있나 없나

Decision Problem



NP 문제 – Class NP

- 비 결정적 다항시간 알고리즘

  예시) 2억 이상, 200kg 넘지 않는 보물의 조합이 있나 없나

  보물이 n개가 있다.

  각 보물에 번호가 있다.

  N비트가 나온다.

  보물을 배낭에 담으면 1, 아니면 0

NP 문제 – Class NP

- 비 결정적 다항시간 알고리즘

  예시) 2억 이상, 200kg 넘지 않는 보물의 조합이 있나 없나

  00001010001…1000 ➔ 2억 원 넘고 200kg 안되나?

  11000010010…1000 ➔ 2억 원 넘고 200kg 안되나?

  01101010000…1000 ➔ 2억 원 넘고 200kg 안되나?

  2^n개의 조합

NP 문제 – Class NP

- 비 결정적 다항시간 알고리즘

  예시) 2억 이상, 200kg 넘지 않는 보물의 조합이 있나 없나

  어떤 사람이 조합을 줌

  00001010001110101101010...10100

  그냥 건너가서 번호대로 담기만 하면 됨

  ➔ 2억이 넘는데 200kg가 넘지가 않음

NP 문제 – Class NP

- 비 결정적 다항시간 알고리즘

  예시) 2억 이상, 200kg 넘지 않는 보물의 조합이 있나 없나

  그냥 건너가서 번호대로 담기만 하면 됨

  ➔ 2억이 넘는데 200kg가 넘지가 않음

  조합 찾았다!!

  다항시간 안에 찾음

NP 문제 – Class NP

- 비 결정적 다항시간 알고리즘

  예시) 2억 이상, 200kg 넘지 않는 보물의 조합이 있나 없나

  다항시간 안에 찾음

  해당 문제에 대해 다항시간(Polynomial Time)에 대답을 하였다.

  “어떤 certificate가 다항시간에 verify할 수 있으면, 그 문제는 클래스 NP에 속한다.”

NP 문제 – Class NP

- 비 결정적 다항시간 알고리즘

  예시) 2억 이상, 200kg 넘지 않는 보물의 조합이 있나 없나

  Certificate를 받고, 다항시간에 확인할 수 있으면 **문제**는 클래스 NP에 속함

  - 만약 해당 certificate가 2억이 넘지 않거나 200kg를 넘어버린다고 해도 충족 시켜야 하는 조합이 없다고 단정지을 수 없다.

  ➔ 아직 모르는 것이다.

NP 문제 – Class NP

- P 클래스와 NP 클래스

  “어떤 문제에 대해서 Polynomial Time Algorithm이 존재하면
  해당 **문제**는 클래스 P에 속한다.” - P 클래스 -

  즉, 클래스 P에 있는 문제들은 모두 클래스 NP에 속한다.
  “어떤 certificate가 다항시간에 verify할 수 있으면, 그 문제는 클래스 NP에 속한다.”

NP 문제 – Class NP

- Polynomial Time Algorithm: n^k 형태로 최악 시간복잡도가 정의되는 알고리즘

- 클래스 P: 특정 문제에 대해서 Polynomial Time Algorithm이 존재하는 문제

- 클래스 NP: 어떤 certificate가 다항시간에 verify할 수 있는 문제

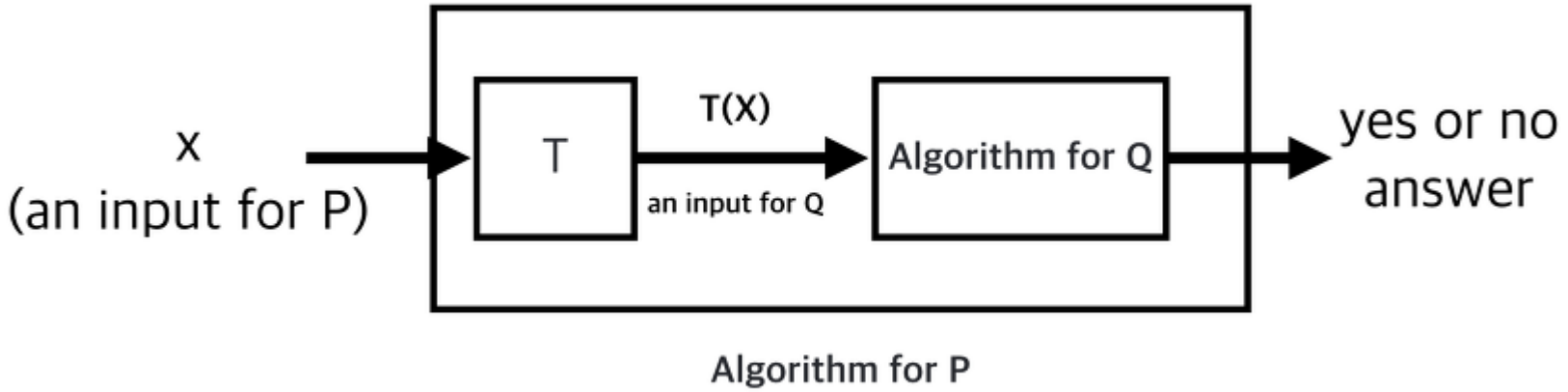
  Non-Deterministic Polynomial Time Algorithm이 존재하는 문제

NP 문제 – NP Hard

- NP 클래스 안에 있는 모든 문제가 어떤 문제 Q로 reducible하면

  그 문제 Q는 NP-Hard

NP 문제 – NP Hard

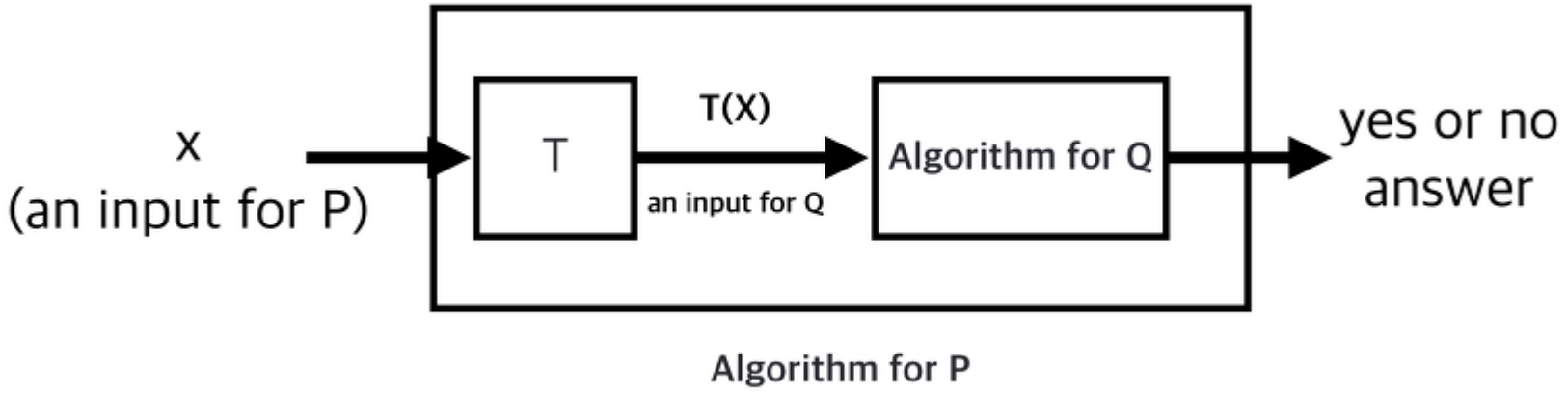


- x는 P의 인풋
- x가 T(Transformation Function)라는 박스를 만나서 아웃풋을 만들어 준다

  ➔ Q의 인풋

Transformation Function: P의 인풋을 Q의 인풋으로 바꾸어주는 역할

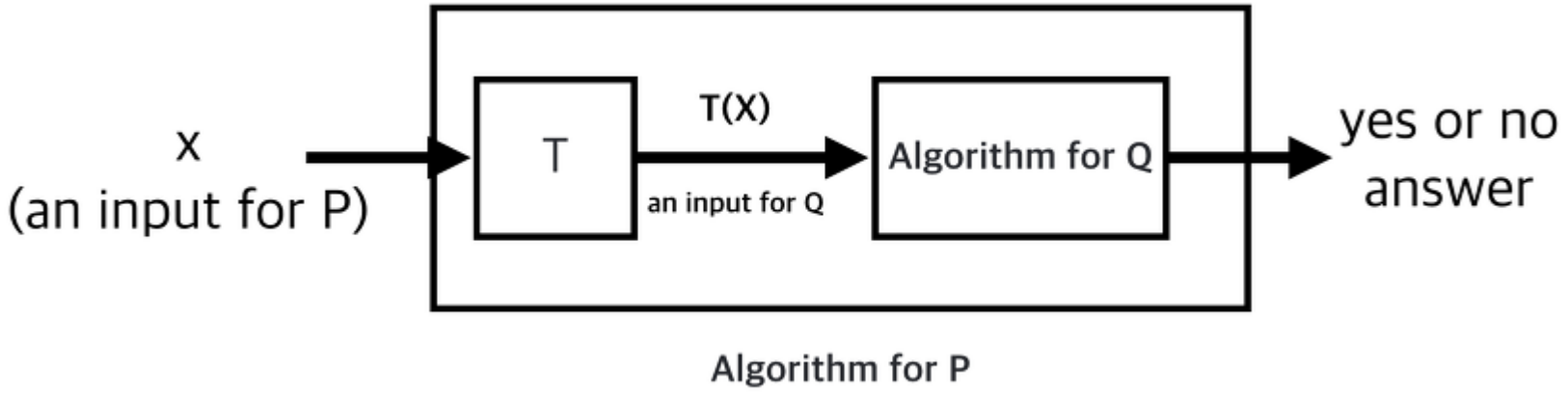
NP 문제 – NP Hard



- Q를 풀 수 있는 알고리즘이 존재한다고 가정
- 원래 P의 인풋인 x를 T를 거쳐 Q를 풀 수 있는 알고리즘에 대입한 결과:

  Yes | No

NP 문제 – NP Hard

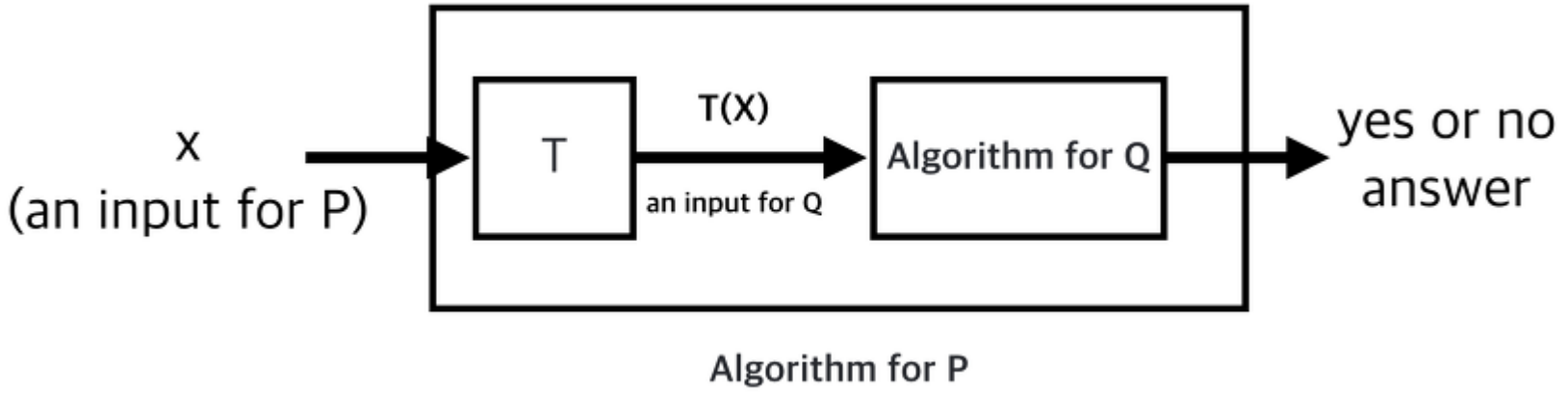


- 원래 P의 인풋인 x를 T를 거쳐 Q를 풀 수 있는 알고리즘에 대입한 결과:

  Yes: 이 상황에서 P에 Yes를 대입하니 Yes

  No: 이 상황에서 P에 No를 대입하니 No

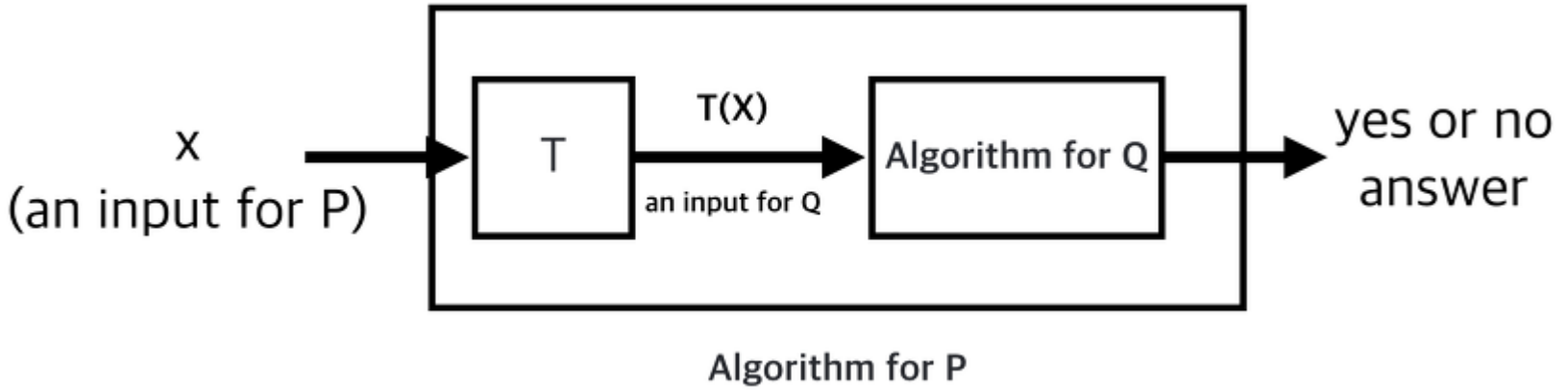
NP 문제 – NP Hard



- Q를 풀 수 있는 알고리즘으로 P를 풀었다.
- Q를 해결하는 알고리즘으로 P도 풀렸으니 Q가 조금 더 어려운 문제

  어렵거나 같은 난이도의 문제

NP 문제 – NP Hard

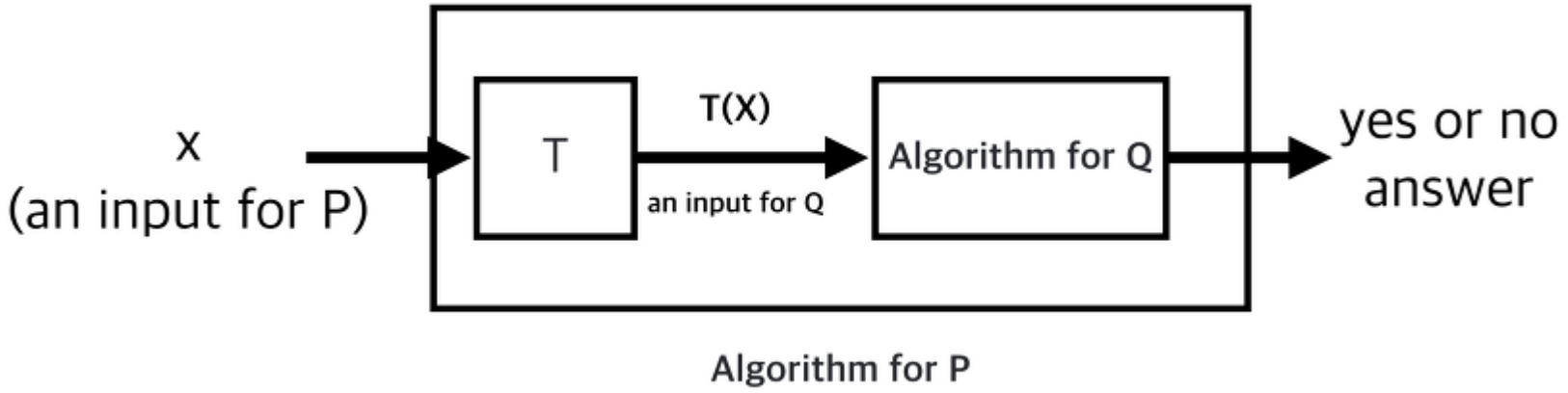


“Problem P is reducible to Q”

뒤에 있는 Q가 P보다 어려운 문제(P <= Q)

“NP 클래스 안에 있는 모든 문제가 어떤 문제 Q로 reducible하면
그 문제 Q는 NP-Hard이다.”

NP 문제 – NP Hard



“NP 클래스 안에 있는 모든 문제가 어떤 문제 Q로 reducible하면

그 문제 Q는 NP-Hard이다.”

- NP클래스 안에 있는 모든 문제들보다 Q가 어렵다.

NP 문제 – NP Hard

또 다른 즐거운 예시)

우리는 지금 모두 학생 - NP 클래스

“어렵다”의 기준은 **나이**

학생이 아닌 나이가 많은 사람 A가 있다.

NP클래스는 아니지만 어려운 사람

즉, NP-Hard한 사람

NP 문제 – NP Hard

또 다른 즐거운 예시)

우리는 지금 모두 학생 - NP 클래스

“어렵다”의 기준은 **나이**

학생이 아닌 나이가 많은 사람 A가 있다.

이런 NP-Hard한 사람이 학생이 되면

NP-Complete가 된다.

NP 문제 – NP Complete

NP-Complete

- NP-Hard이면서 NP클래스 안에 있으면 NP-Complete이다.
- NP 클래스에 속해 있으면서, 어떤 기준에 의해 가장 어려운 문제

NP 문제 – NP Complete

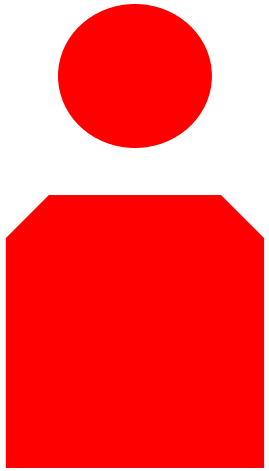
NP-Complete 즐거운 예시)

제일 잘 싸움
잘 싸우니 NP-Hard
NP 클래스 안에 있으니 NP Complete

NP 문제 – NP Complete

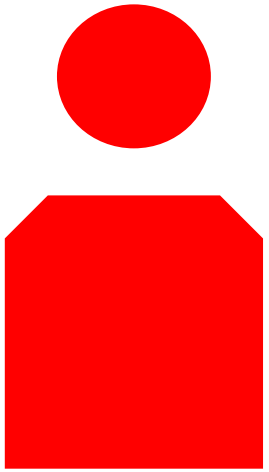
NP-Complete 즐거운 예시)

짱이 되고 싶은 전학생
- 모든 학생과 1대1 싸우는 것보다 제일 쎈 애들과 싸운다.



NP 문제 – NP Complete

NP-Complete 즐거운 예시)



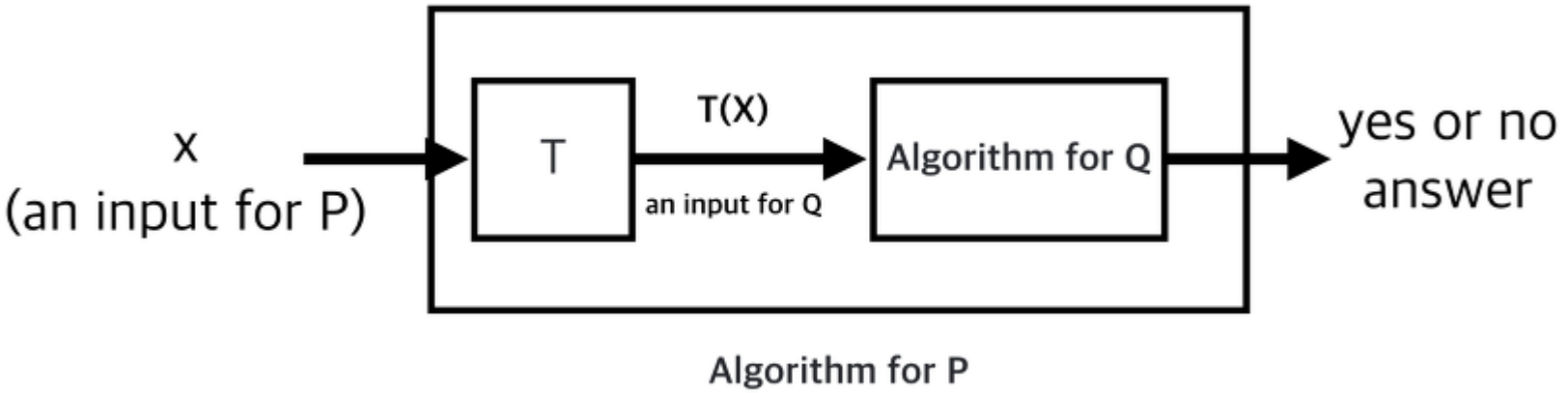
전학생이 싸워서 이겼다고 가정
- 전학생은 NP 클래스의 NP-Hard가 된다.
- 진 학생들은 지기는 했지만 여전히 쌘 NP-Hard

NP 문제 – NP Complete

NP-Complete 즐거운 예시)

- 전학생이 1등을 해서 기존 1등이 부정이 되는 것은 아니다.
- 기존 1등 및 새로 1등한 전학생도 모두 NP-Complete

NP 문제 – NP Complete



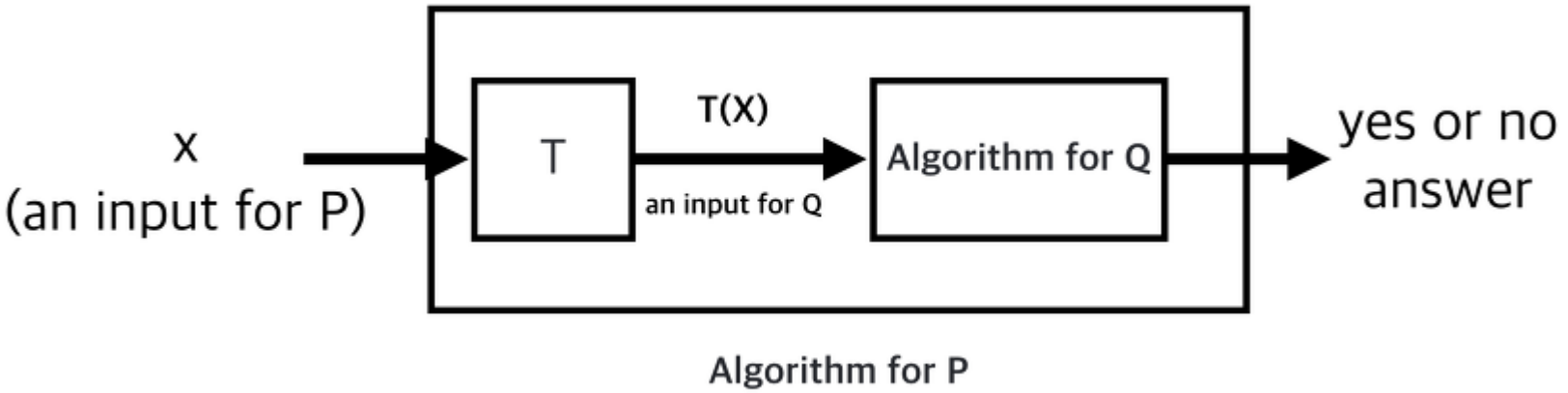
증명할 것: 어떤 문제 P가 Q로 reducible하다(Q가 더 어렵거나 같다).

“NP클래스 안에 있는 모든 문제가 어떤 문제Q로 reducible하면, 그 문제 Q는 NP-Hard이다.”

P의 인풋을 T에 넣어서 Q의 인풋으로 바꿔 푼 결과를 P에 넣었더니 전부 성립

Q는 NP-Hard

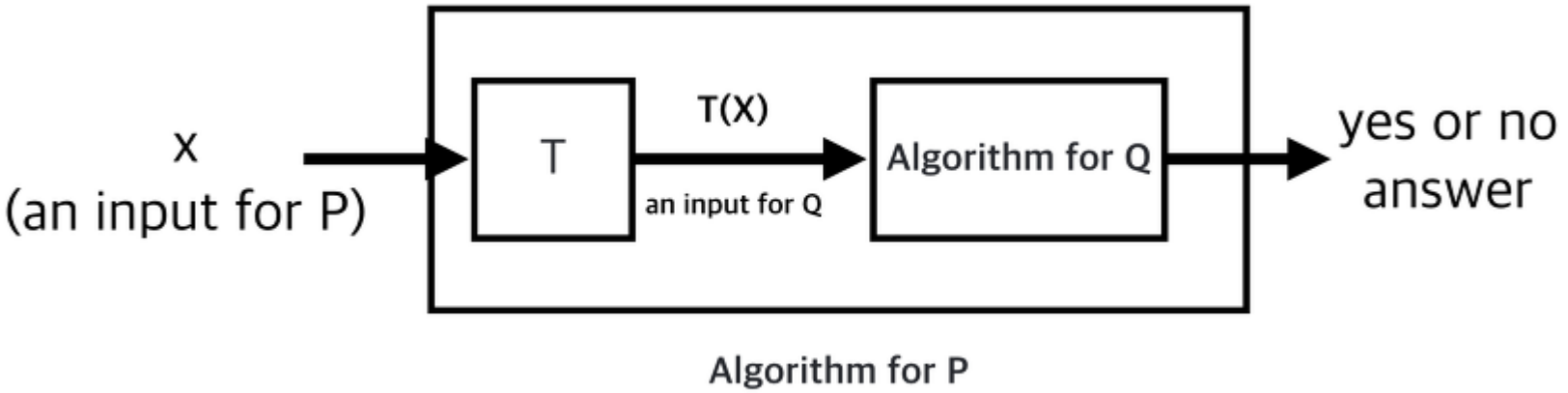
NP 문제 – NP Complete



Q는 NP-Hard

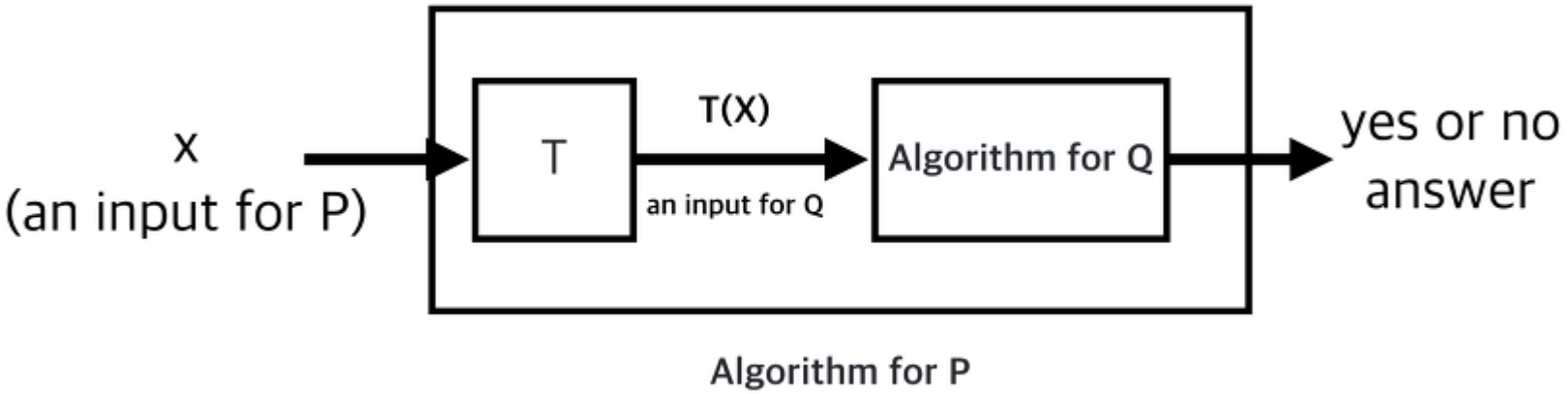
NP 클래스 인지 구별: 어떤 certificate가 다항시간에 verify되면 그 문제는 NP 클래스 이다.

NP 문제 – NP Complete



- T에서 다항시간이 걸렸다고 가정
- Algorithm for Q도 다항시간이 걸렸다고 가정
- 전체 박스가 다항시간에 풀림
- P를 다항시간에 풀게 되는 것

NP 문제 – NP Complete



- 만약 문제 P를 NP클래스 안에 있는 어떤 문제보다 어려운 문제, NP-Complete로 골랐는데, 다항시간에 풀리는 경우
- NP-Complete 문제들은 이때까지 알려진 다항시간 알고리즘이 없다.

NP 문제 – NP Complete

- 만약 문제 P를 NP클래스 안에 있는 어떤 문제보다 어려운 문제, NP-Complete로 골랐는데, 다항시간에 풀리는 경우
- NP 클래스 안에 있는 어떤 문제보다 어려운 문제가 풀린다.

  ➔ NP 클래스 안에 있는 모든 문제를 다항시간에 풀 수 있다.

- **모든 과정이 다항시간에 이루어진다면,** NP가 P에 속하게 된다.

  “어떤 문제에 대해서 Polynomial Time Algorithm이 존재하면 해당 **문제**는 클래스 P에 속한다.” - P 클래스 -

NP 문제 – NP Complete

- 만약 문제 P를 NP클래스 안에 있는 어떤 문제보다 어려운 문제, NP-Complete로 골랐는데, 다항시간에 풀리는 경우
- NP 클래스 안에 있는 어떤 문제보다 어려운 문제가 풀린다.

  ➔ NP 클래스 안에 있는 모든 문제를 다항시간에 풀 수 있다.

- **모든 과정이 다항시간에 이루어진다면,** NP가 P에 속하게 된다.

  P = NP 성립

  모든 NP 문제가 P인가 증명하면 된다.

NP 문제 – NP Complete

- 만약 문제 P를 NP클래스 안에 있는 어떤 문제보다 어려운 문제, NP-Complete로 골랐는데, 다항시간에 풀리는 경우
- NP 클래스 안에 있는 어떤 문제보다 어려운 문제가 풀린다.

  ➔ NP 클래스 안에 있는 모든 문제를 다항시간에 풀 수 있다.

- **모든 과정이 다항시간에 이루어진다면,** NP가 P에 속하게 된다.

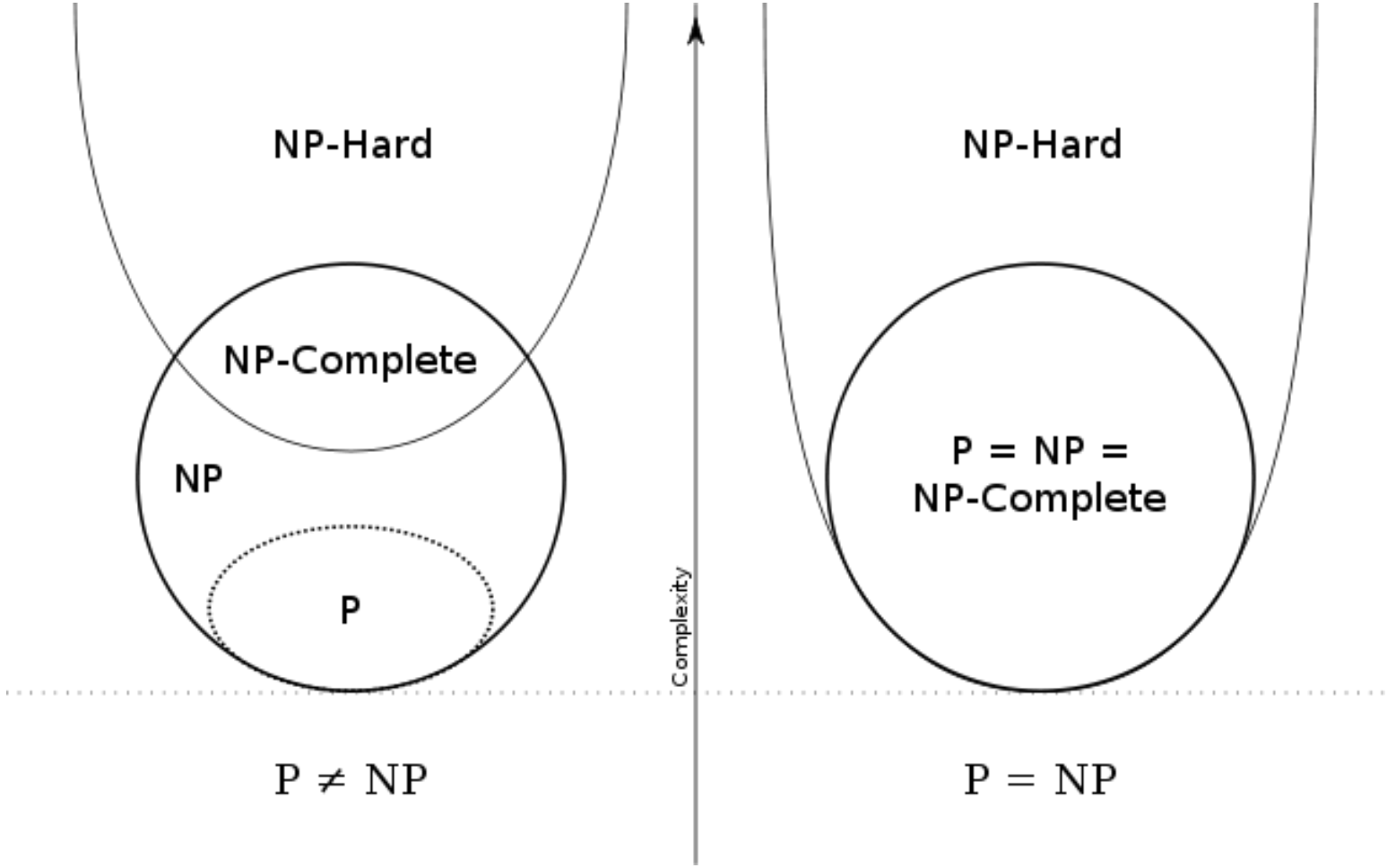
  P = NP 성립

  NP-Complete 문제들은 이때까지 알려진 다항시간 알고리즘이 없다.

NP 문제 – NP Complete

- 현재까지 알려진 NP-Complete 문제들이 많음
- NP-Complete 문제 하나라도 다항시간 알고리즘을 만들어내면
  모든 NP-Complete 문제들이 다항시간에 풀린다는 것을 증명
  (서로 더 어렵거나 같은 난이도의 관계이기 때문)

NP 문제 – NP Complete



양자 컴퓨터

- NP-Complete 문제들은 양자 컴퓨터로도 풀리지 않을 것으로 예상

  만약 풀리는 것이 가능하다면 양자 내성 암호라는 것은 불가능

LEDAkem & LEDApkc

- NIST에서 2 round를 진행하고 있는 code base 양자 내성 암호 제안
- 키를 암호화 하는 LEDAkem과 공개키 LEDApkc로 나눠서 제출
- 두 제안을 통합하여 LEDAcrypt

LEDAkem & LEDApkc

- LEDAkem
  - 선형 오류 정정 코드 사용 및 code based 키 암호화 메커니즘
  - 임시 키 KEM 사용, Niederreiter 암호 시스템

- LEDApkc
  - 선형 오류 정정 코드 사용 및 code based 공개키 암호 시스템
  - 공개키 사용, McEliece 암호 시스템

LEDAkem & LEDApkc 목표

- 보안

  - 모든 공격에 대항 → 공격 복잡성에서 파생된 매개 변수 사용

  - NP-Complete와 연관된 문제 → 신드롬 디코딩

- 효율성

  - 작은 키 쌍 사이즈 → 순환 행렬 H와 G 사용

  - 효율적인 인코딩 및 디코딩 → 순환 행렬 H와 G 그리고 LDPC 코드

    LDPC(저밀도 패리티 체크)

LEDAkem & LEDApkc

- 순환 행렬(Block Circulant) 방식 사용

$$A = \begin{pmatrix} a_0 & a_1 & a_2 \\ a_2 & a_0 & a_1 \\ a_1 & a_2 & a_0 \end{pmatrix} = a_0 + a_0x + a_0x^2 \bmod x^3 + 1$$

- n * n의 행렬이 첫 행만을 가지고도 표현이 가능

  - 키 사이즈 절약 $O(n^2) \rightarrow O(n)$

- mod $x^n + 1$ 형식의 산술과 동일

LEDAkem & LEDApkc

- 순환 행렬 패리티와 선형 부호 행렬 이용
- 신드롬 디코딩 방식으로 Quasi-Cyclic(준 순환) 구조를 깰 수 없음

  - QC 구조를 깨는 문제는 NP-Complete

Cyclic & LDPC

- Cyclic

  - Code C = {000, 101, 011, 110}

$$G = \begin{pmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{pmatrix}$$

$C_1 = 101100$ $C_2 = 010110$ $C_3 = 0010111$

$C_1 + C_2 = 1110010$ $C_1 + C_3 = 1001011$ $C_2 + C_3 = 0111001$

$C_1 + C_2 + C_3 = 1100101$

Cyclic & LDPC

- Cyclic

$C_1 = 101100$ $C_2 = 010110$ $C_3 = 0010111$

$C_1 + C_2 = 1110010$ $C_1 + C_3 = 1001011$ $C_2 + C_3 = 0111001$

$C_1 + C_2 + C_3 = 1100101$

$c_1 \rightarrow c_2$, $c_2 \rightarrow c_3$, $c_3 \rightarrow c_1 + c_3$

$c_1 + c_2 \rightarrow c_2 + c_3$, $c_1 + c_3 \rightarrow c_1 + c_2 + c_3$, $c_2 + c_3 \rightarrow c_1$

$c_1 + c_2 + c_3 \rightarrow c_1 + c_2$

Cyclic & LDPC

- Cyclic

  - Code C = {000, 101, 011, 110}

  - Code A = {0000, 1001, 0110, 1111} 는 완전한 Cyclic은 아니지만 Cyclic Code로 취급해준다.

    원소 전부가 Cyclic은 아니지만 몇 개는 Cyclic인 경우

  - 이러한 Cyclic Code는 polynomial 형식으로 나타내짐

Cyclic & LDPC

- LDPC

  Low Density Parity Check

  Bit Protection

  ex)

  100011 100011

  같은 메시지를 2번 보내 확인을 할 수 있다.

Cyclic & LDPC

- LDPC

  Bit Protection
  ex) 중복 전송

  ?00011

  ?00011

  ?00011

  - 같은 비트에 문제가 생기면 찾을 수 없음
  - 코드 수치 효율성이 떨어짐

Cyclic & LDPC

- LDPC

  Bit Protection
  ex)

  ?00011

  ?00011

  ?00011

  - 같은 비트에 문제가 생기면 찾을 수 없음
  - 코드 수치(Code Rate) 효율성이 떨어짐

Cyclic & LDPC

- LDPC

  Code Rate

$$\frac{\#\text{전체 메시지 bit}}{\#\text{전송한 전체 bit}} = \frac{100011}{100011\ 100011} = \frac{6}{12} = 50\%$$

Cyclic & LDPC

- LDPC

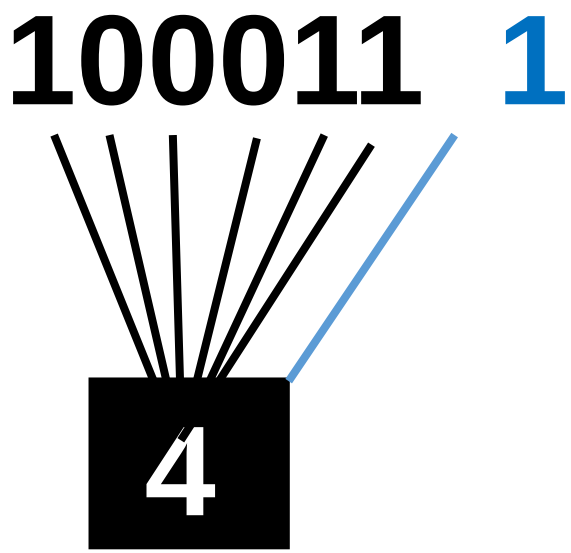
  - Code Rate 저하하면 Protection 증가
  - Code Rate가 낮으면 낮을 수록 비용이 많이 든다.
    보내는 bit의 양이 많아지기 때문

  - Code Rate와 Probability of Failure을 잘 생각해 봐야 한다.
    메시지는 충분히 보호가 되면서도 효율적인(인코딩과 디코딩이 빠른) 방법이 필요

Cyclic & LDPC

- LDPC

  - 1의 개수를 확인해 전체 bit의 1의 개수가 짝수가 되게 만든다: **Parity Bit**



Cyclic & LDPC

- LDPC

  - 1의 개수를 확인해 전체 bit의 1의 개수가 짝수가 되게 만든다: **Parity Bit**

  - 에러 확인: 1의 개수가 짝수인지 확인

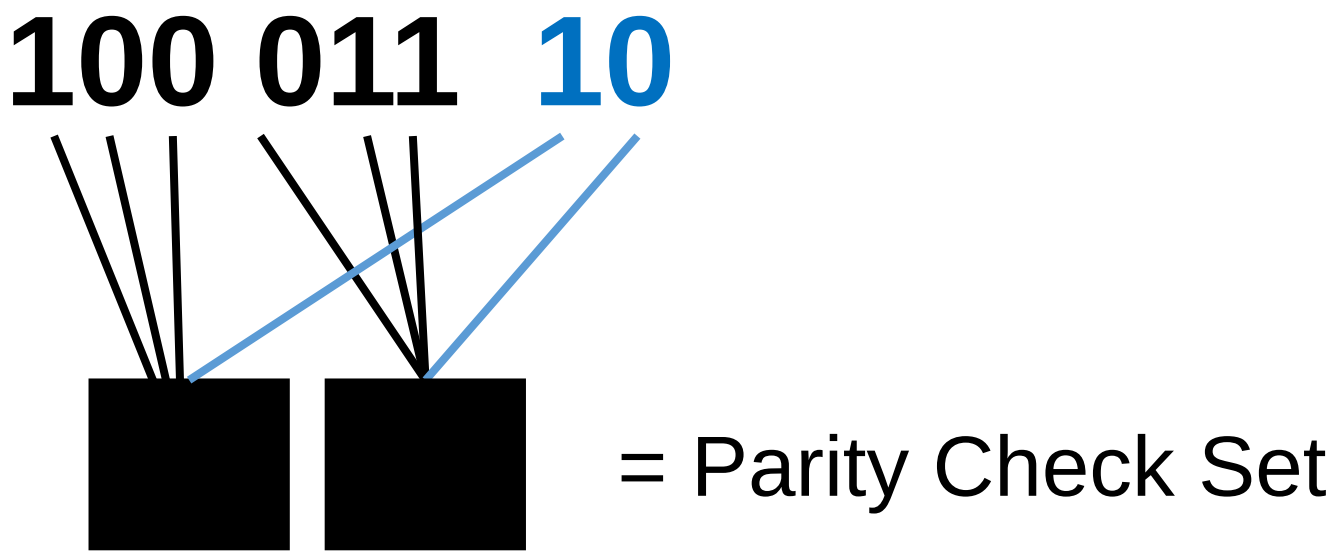
  100011
  ---------- = 86%
  1000111

  - 1개의 Parity Bit으로는 1개의 에러만 확인 가능

Cyclic & LDPC

- LDPC

  - 에러의 검출을 높이기 위해 전체 메시지를 2개로 나누어 2개의 패리티 비트 사용



Cyclic & LDPC

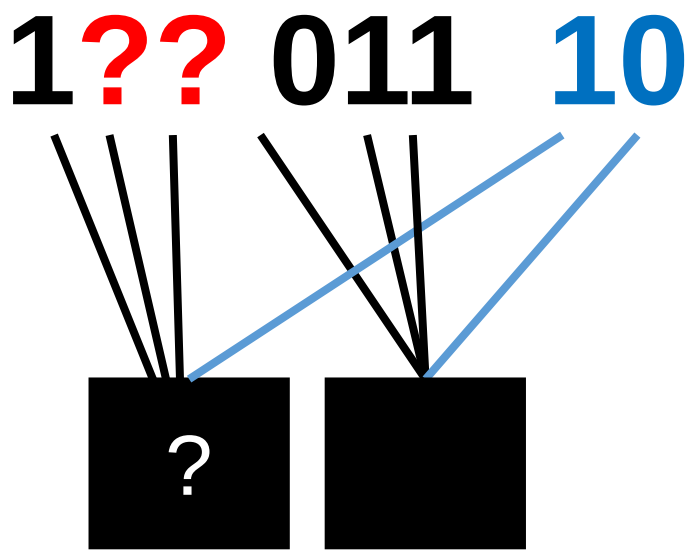
- LDPC

  - 패리티 비트를 많이 쓰면 에러 수정 가능 비트의 개수가 증가
  - Code rate는 감소

Cyclic & LDPC

- LDPC

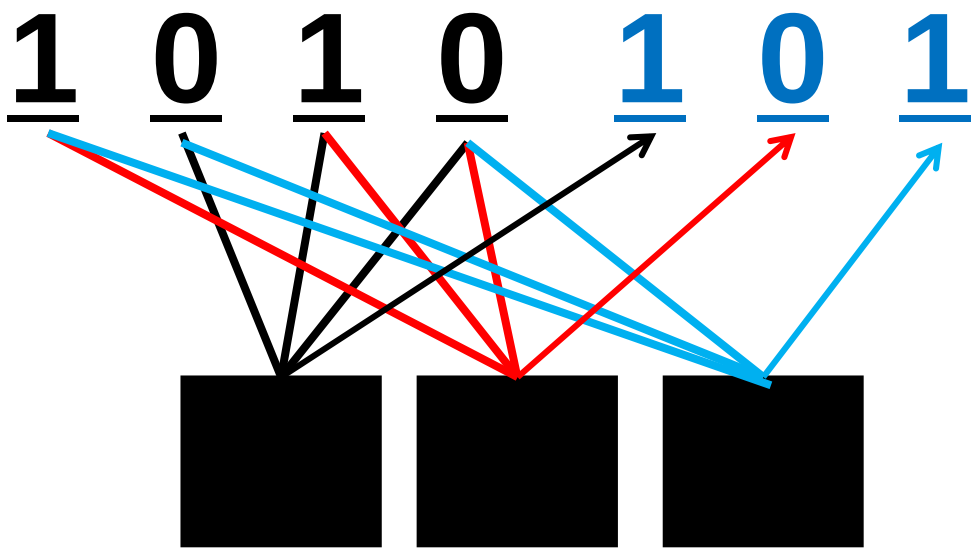
  - 여전히 같은 세트에서 오류가 2개 이상 발생하면 오류 찾기 불가능



Cyclic & LDPC

- LDPC

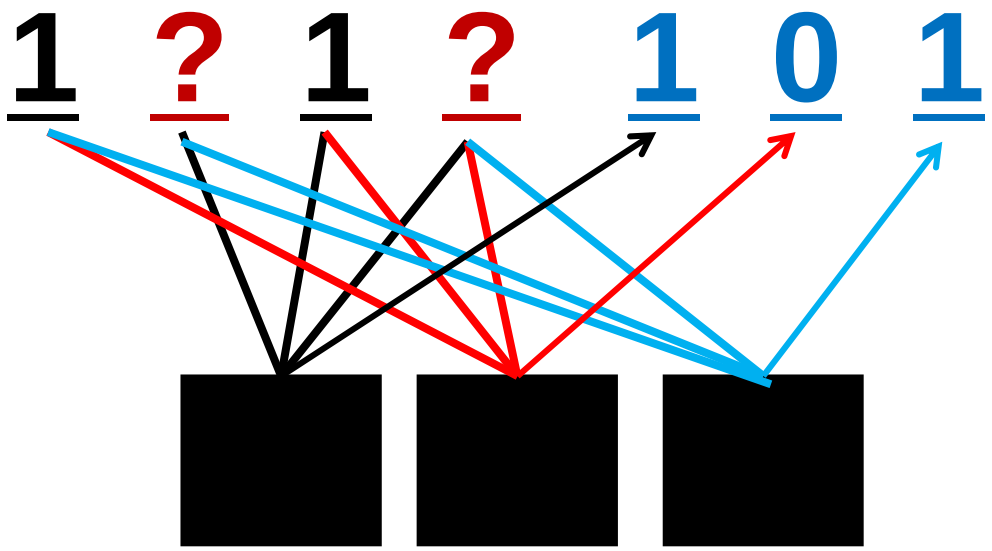
  - 이러한 한계를 보안하기 위해 겹치는(overlapping) 패리티 비트 사용 제안



Cyclic & LDPC

- LDPC

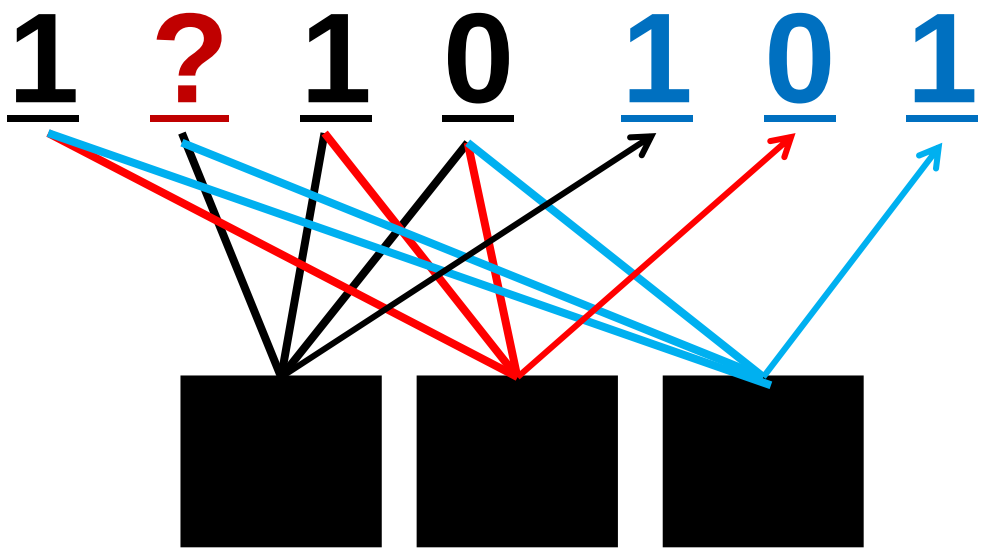
  - 이러한 한계를 보안하기 위해 겹치는(overlapping) 패리티 비트 사용 제안



Cyclic & LDPC

- LDPC

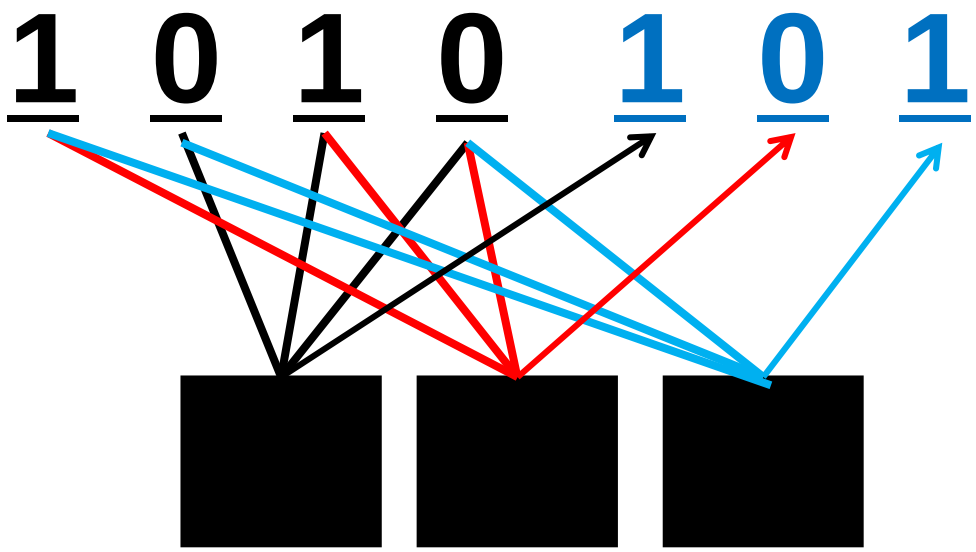
  - 이러한 한계를 보안하기 위해 겹치는(overlapping) 패리티 비트 사용 제안



Cyclic & LDPC

- LDPC

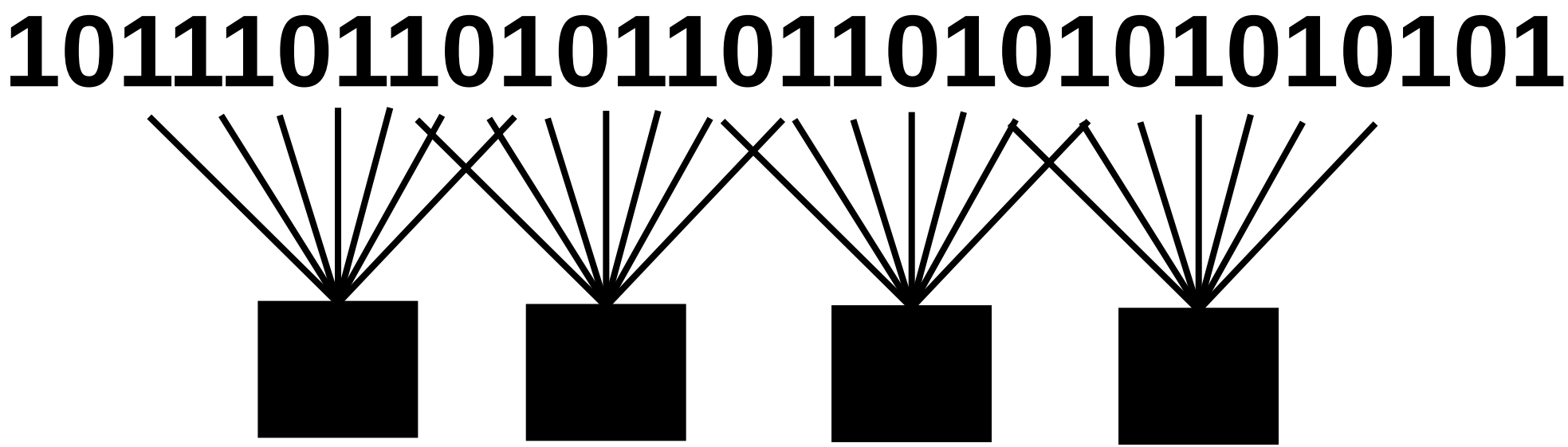
  - 한번에 에러를 검출하는 것이 아닌 set를 이용해 하나씩 찾는 방식을 사용



Cyclic & LDPC

- LDPC

  - 이러한 원리를 이용한 많은 패리티 세트 구조가 연구 되었음

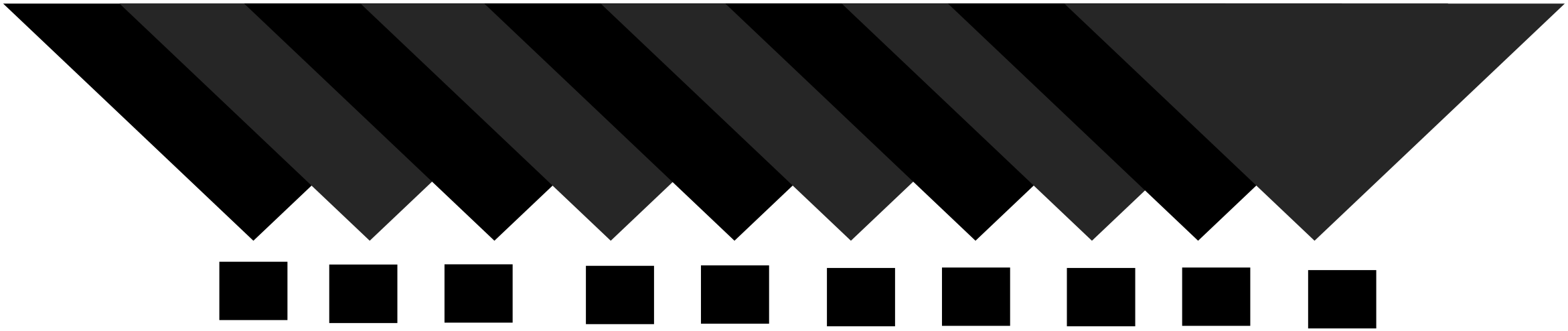


Cyclic & LDPC

- LDPC

  - 그러나 이러한 구조를 매우 큰 길이의 메시지에 적용을 하면 문제가 발생

..10111011010110110101010101011011101101011011010101010110111011010110010110110 0..

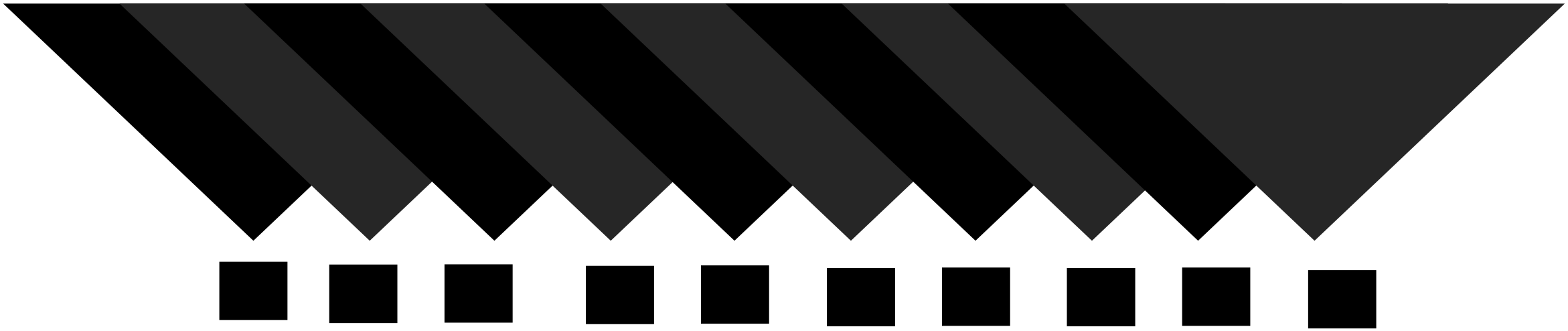


Cyclic & LDPC

- LDPC

  - 하나의 set가 담당해야 하는 메시지가 커지고 담당 면적에 비해 에러는 적게 발생

..10111011010110110101010101011011101101011011010101010110111011010110010110110 0..

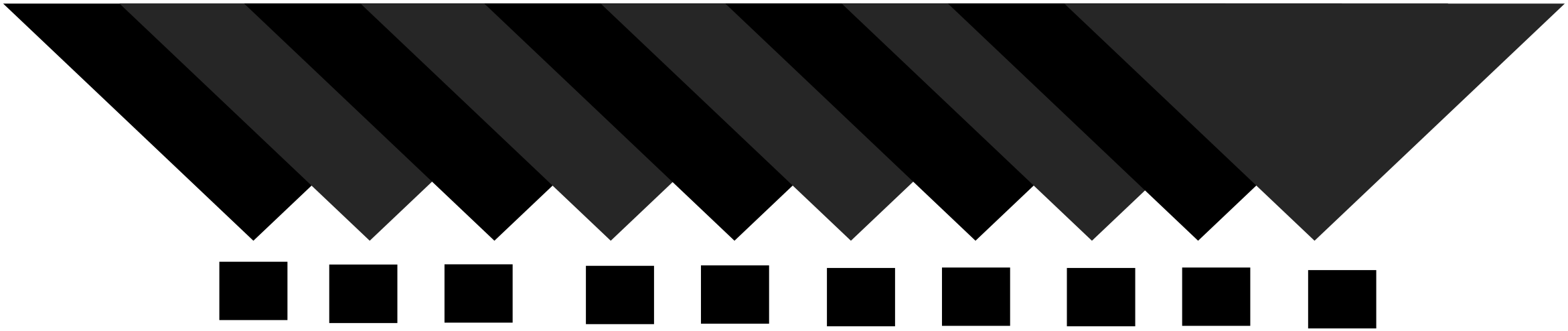


Cyclic & LDPC

- LDPC

  - 패리티 확인을 하는데 너무 복잡한 연산이 소모가 된다.

..10111011010110110101010101011011101101011011010101010110111011010110010110110 0..



Cyclic & LDPC

- LDPC

  - Overlapping 방식을 사용
  - 빠르게 연산을 할 수 있어야 함
  - 매우 긴 메시지에 적용을 해야 함
  - 어떠한 오류든 찾아내야 함
  - 적당한 Code rate를 유지해야 함

Cyclic & LDPC

- LDPC

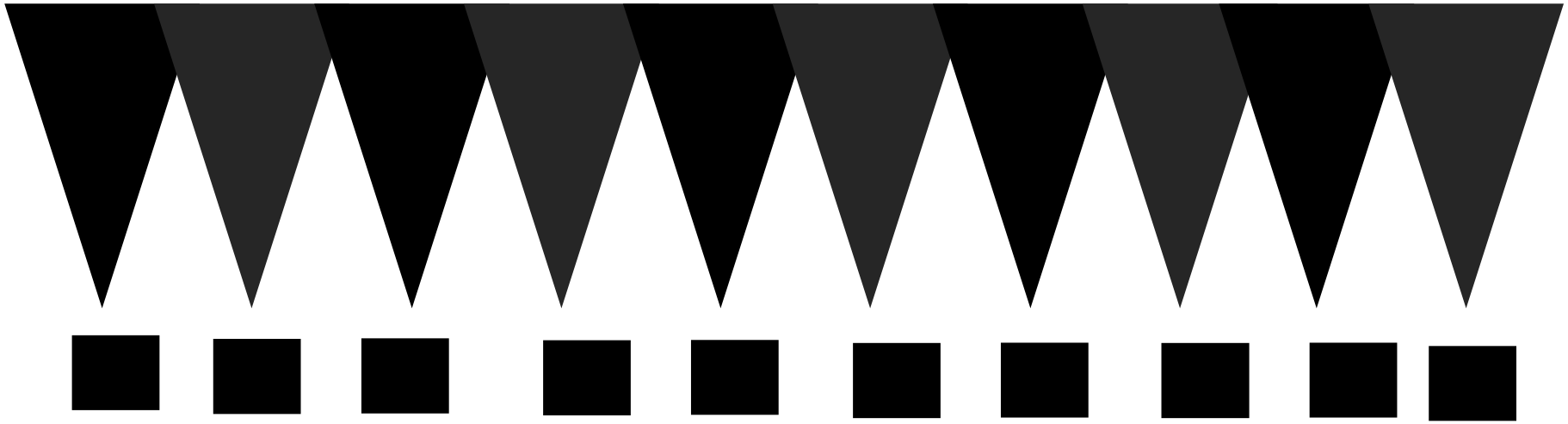
  - 1963 Robert G Gallager가 이러한 문제를 해결하기 위한 구조 제안
  - 현재 패리티 비트 확인 모델은 Gallager의 방식을 따름

Cyclic & LDPC

- LDPC

  - 세트가 담당하는 비트 수를 줄인다; 하나의 세트에 존재할 오류의 수를 줄인다.
  - 비트를 확인하는데 걸리는 연산의 시간을 줄임

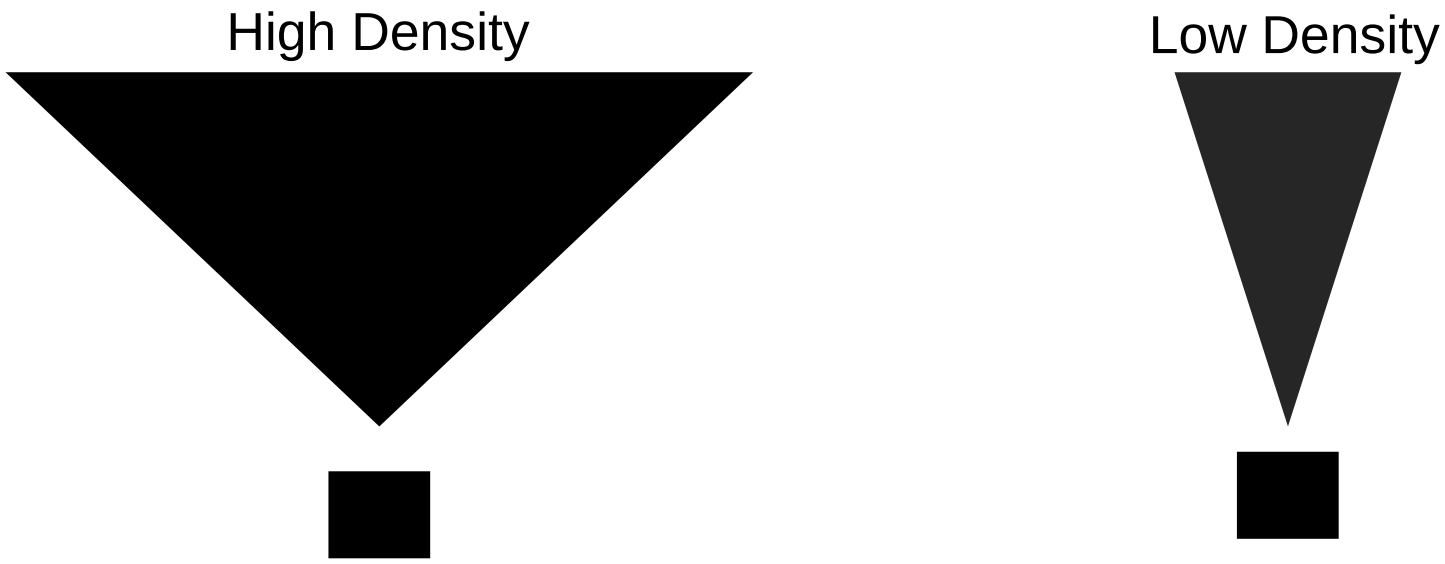
..10110110101010101011011101101011011010101010110111011010110011100..



Cyclic & LDPC

- LDPC

  - 세트가 담당하는 비트 수를 줄인다; 하나의 세트에 존재할 오류의 수를 줄인다.
  - 담당하는 밀도(Density)를 줄인다: **Low Density Parity Check Code**

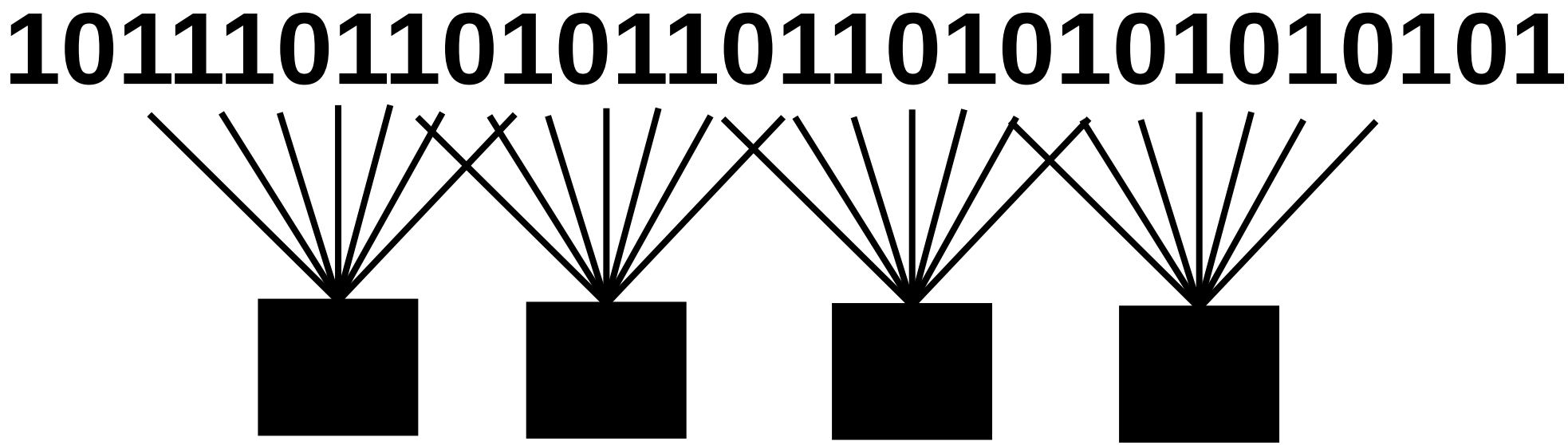


Cyclic & LDPC

- LDPC

  - 이런 세트의 구조는 어떠한 모양을 해야 하는가

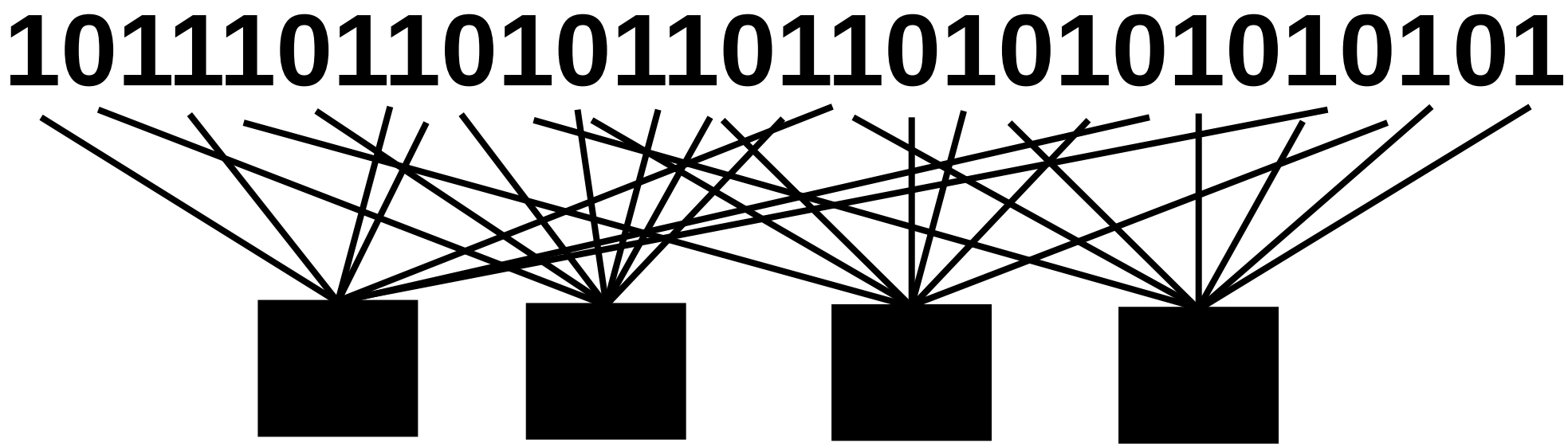
  - Peter Alliss: 큰 메시지에서 구조의 모양은 의미가 없다.



Cyclic & LDPC

- LDPC

  - Peter Alliss: 큰 메시지에서 구조의 모양은 의미가 없다: **무작위 담당**
  - 구조를 형성하는데 고민을 했던 것이 사라짐



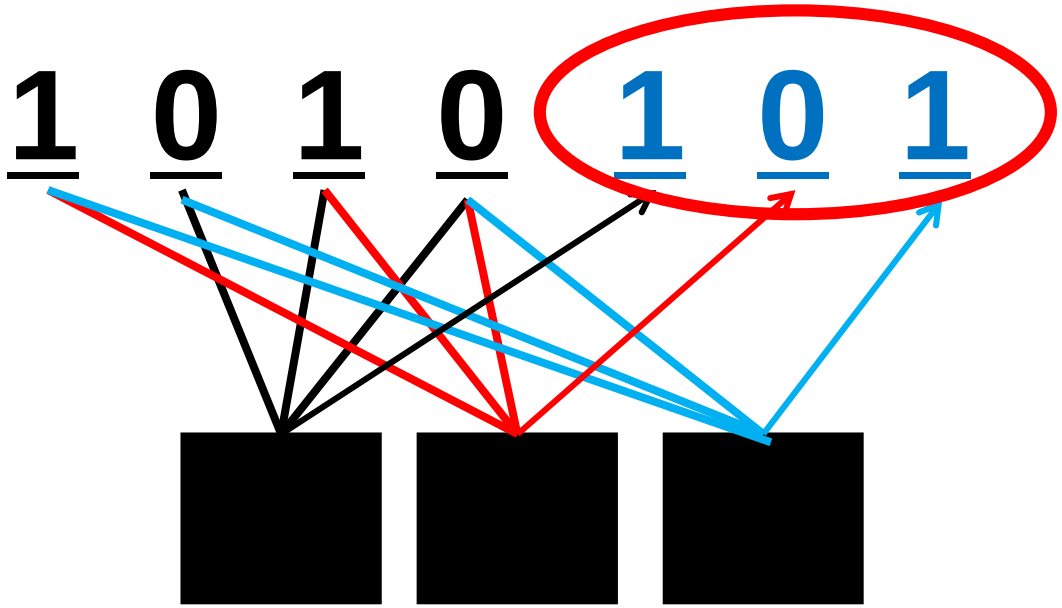
Cyclic & LDPC

- LDPC
  - 작은 밀도의 Overlapping 방식
  - 무작위의 구조

  - 단순한 연산 가능
  - 메시지 길이 상관 없음
  - 문제 해결 하기 좋음

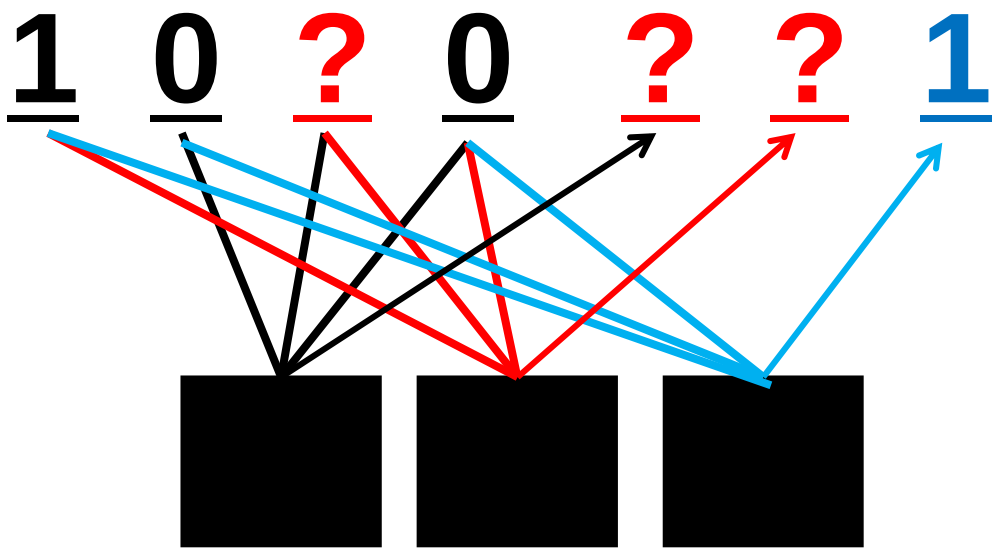
Cyclic & LDPC

- LDPC
  - 아래와 같은 구조에서 패리티 비트는 보호가 되지 않는다.



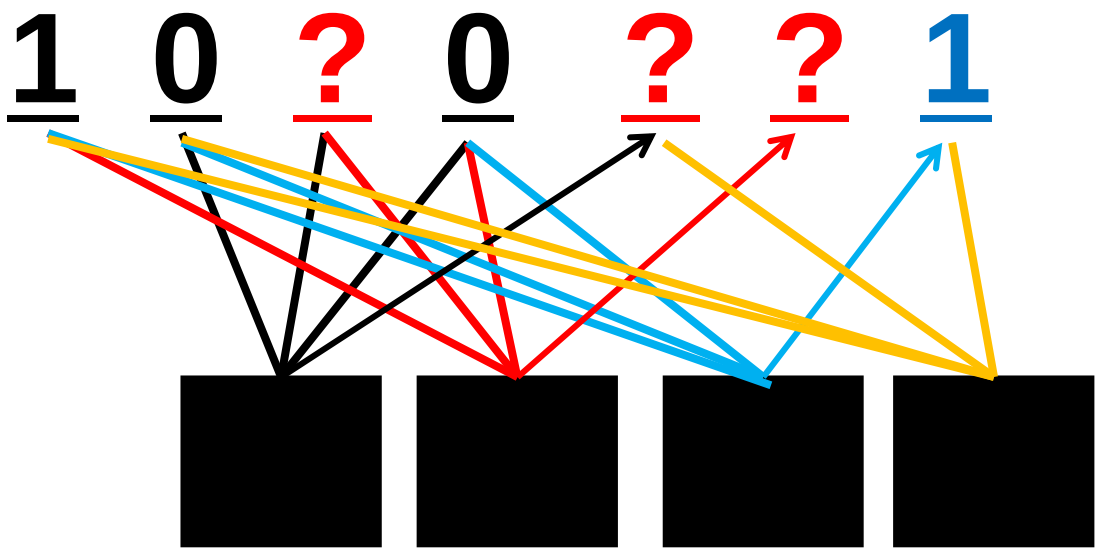
Cyclic & LDPC

- LDPC
  - 아래와 같은 구조에서 패리티 비트는 보호가 되지 않는다.



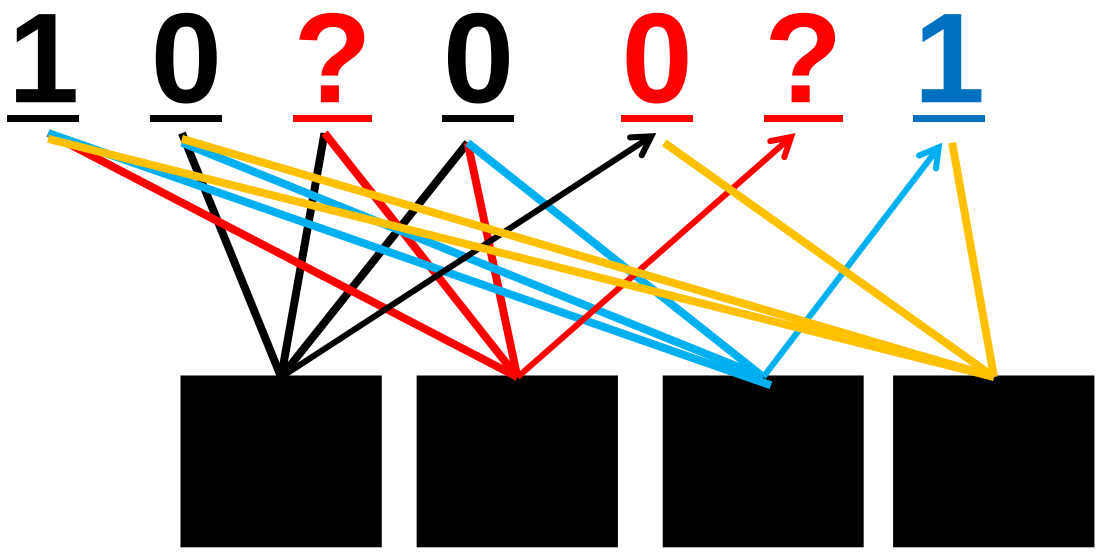
Cyclic & LDPC

- LDPC
  - 메시지를 보호하는 방법처럼 패리티 비트도 보호를 한다.



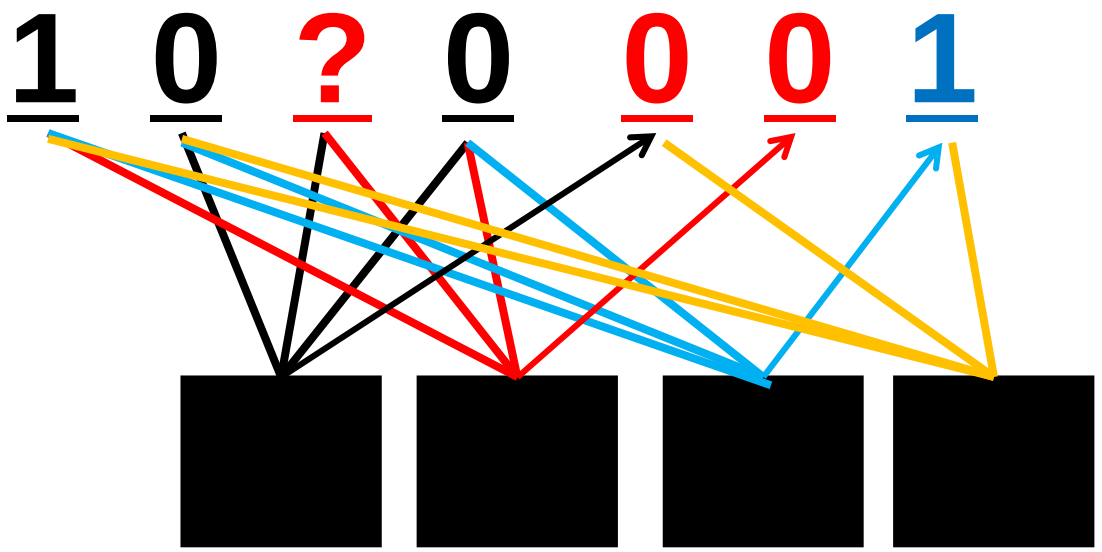
Cyclic & LDPC

- LDPC
  - 메시지를 보호하는 방법처럼 패리티 비트도 보호를 한다.



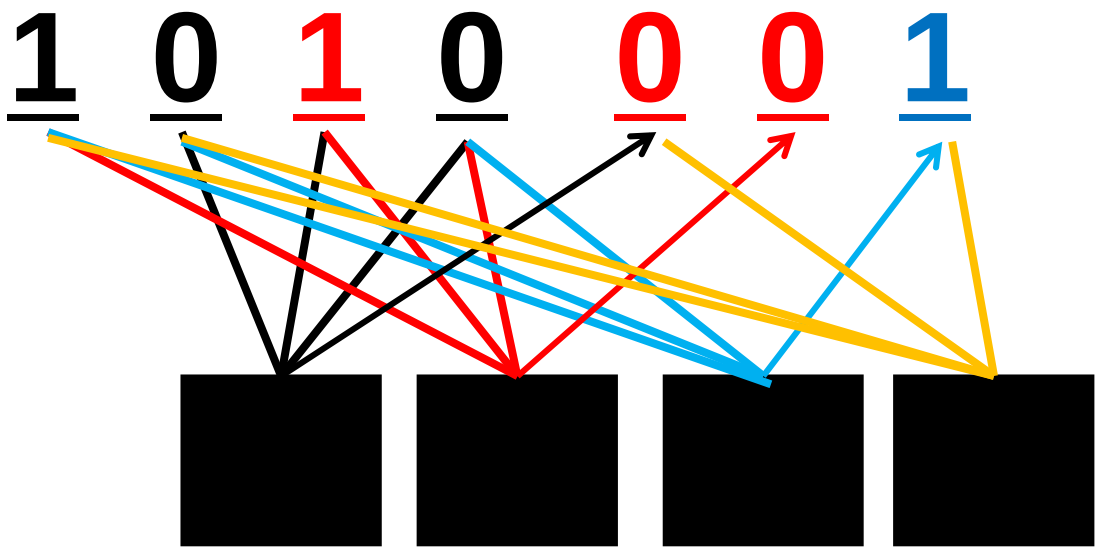
Cyclic & LDPC

- LDPC
  - 메시지를 보호하는 방법처럼 패리티 비트도 보호를 한다.



Cyclic & LDPC

- LDPC
  - 메시지를 보호하는 방법처럼 패리티 비트도 보호를 한다.



Cyclic & LDPC

- LDPC
  - 작은 밀도의 Overlapping 방식
  - 무작위의 구조
  - 패리티 비트도 보호

  - 단순한 연산 가능
  - 메시지 길이 상관 없음
  - 문제 해결 하기 좋음

NIST Round 2

- Round 1

  Lattice 26

  Code 19

  etc 19

  64

NIST Round 2

- Round 1

Lattice	26
Code	19
etc	19
---	---
	64

- Round 2

Lattice	11
Code	8
Muli	4
Iso	1
Hash	1
ZK	1
---	---
	26

NIST Round 2 공개키 암호 및 키 생성 알고리즘

- **Code**
  - BIKE
  - Classic McEliece
  - HQC
  - LEDAcrypt
  - NTS-KEM
  - ROLLO
  - RQC
  - SABER
- **Lattice**
  - CRYSTALS-KYBER
  - FrodoKEM
  - LAC
  - NewHope
  - NTRU
  - NTRU Prime
  - Round5
  - Three Bears
- **Isogency**
  - SIKE

NIST Round 2 전자 서명

- **Multivariate**
  - GeMSS
  - LUOV
  - MQDSS
  - Rainbow
- **Lattice**
  - CRYSTALS-Dilithium
  - FALCON
  - qTESLA
- **Isogency**
  - SIKE
- **Zero Knowledge**
  - Picnic
- **Hash**
  - SPHINCS+

Q & A

감사합니다