2021-2
알고리즘

문자열 매칭(1)

NP-완비

NP-완비의 증명

한남대학교 컴퓨터공학과

문자열 매칭(1)

문자열 매칭

- **입력**
  - **A[1⋯*n*]** : 텍스트 문자열
  - **P[1⋯*m*]** : 패턴 문자열
  - $m << n$

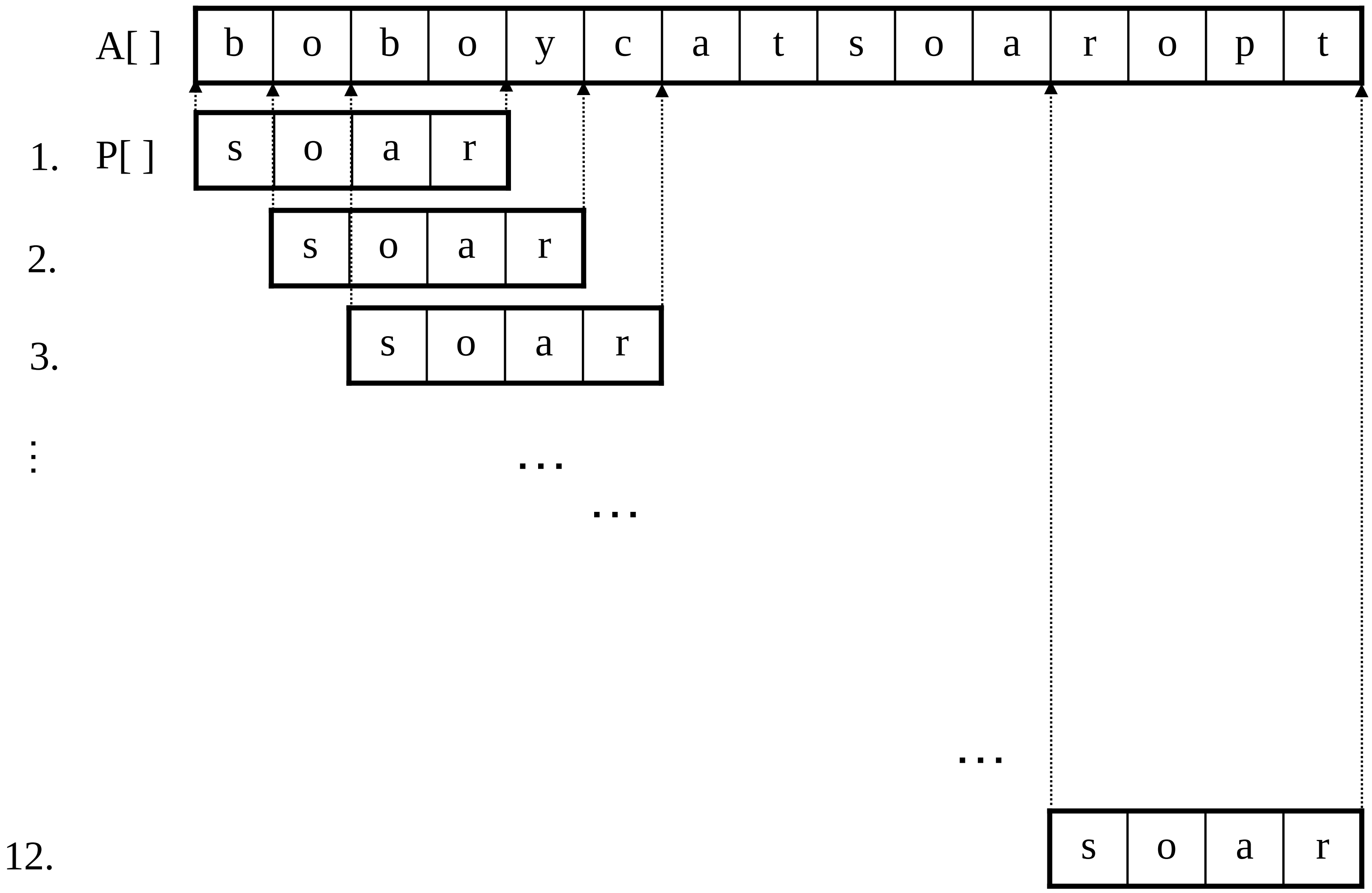
- **출력**
  - 텍스트 문자열 **A[1⋯n]**에 패턴 문자열 **P[1⋯*m*]**이 포함된 위치들

원시적인 매칭

```
naiveMatching(A, P)
{
    ▷ n: 배열 A[ ]의 길이, m: 배열 P[ ]의 길이
    for i ← 1 to n-m+1{
            if (P[1…m] = A[i…i+m-1])
                then  A[i] 자리에서 매칭이 발견되었음을 알린다;
    }
}
```

✓ 수행시간: $O(mn)$

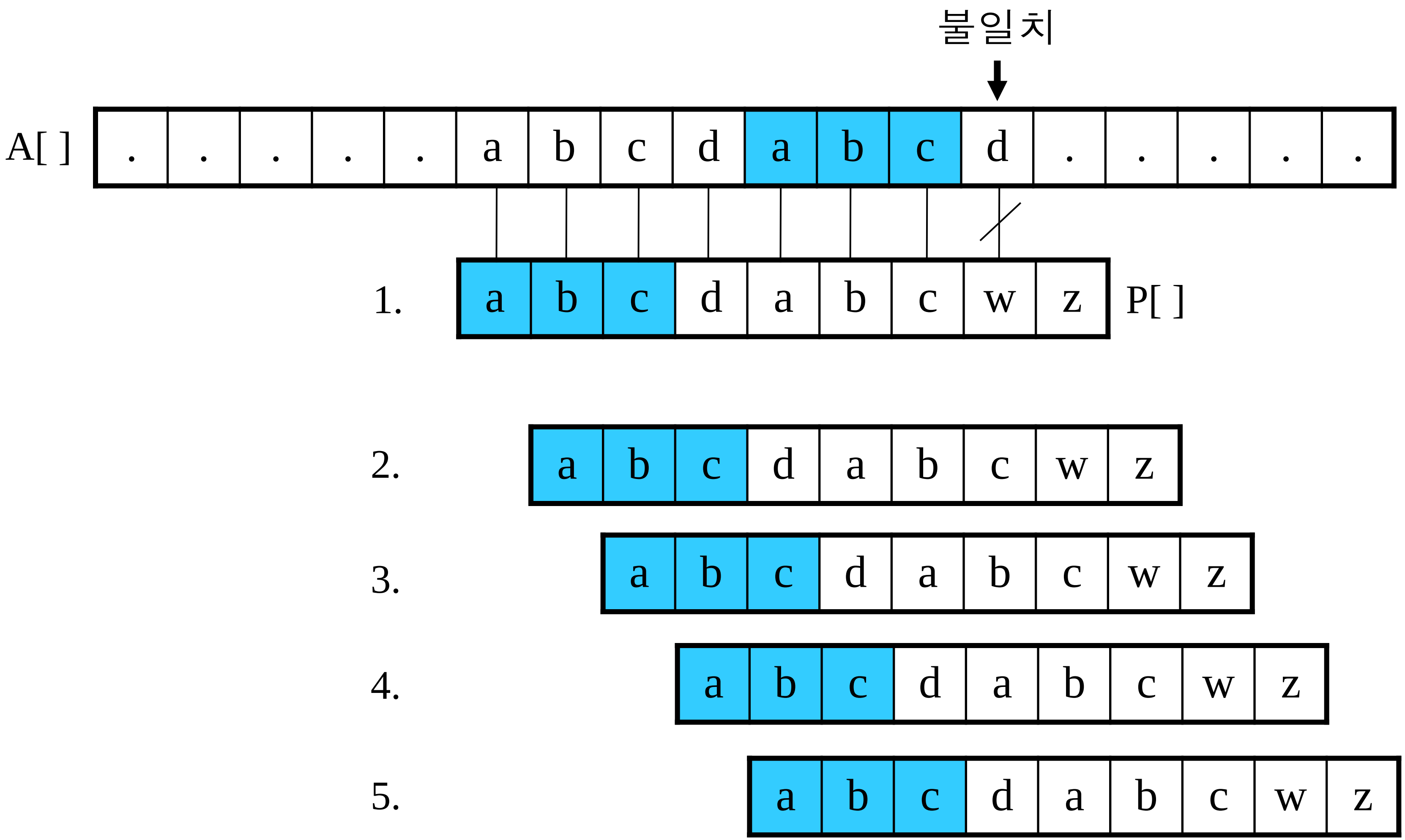
원시적인 매칭의 작동 원리



연습문제

```
def match(A, P):
    # P가 A에 포함되어 있으면 첫 번째 인덱스를,
    # 아니면 -1을 리턴한다.
    return -1
```

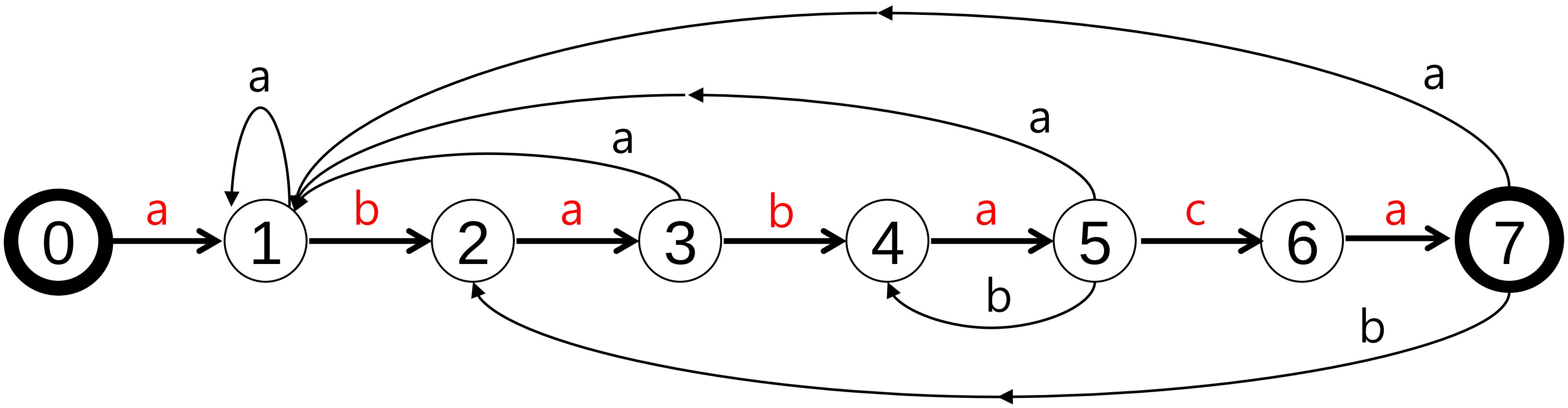
원시적인 매칭이 비효율적인 예



오토마타를 이용한 매칭

- **오토마타 Automata**
  - 문제 해결 절차를 상태state의 전이로 나타낸 것
- **구성 요소: (Q, q_0, A, Σ, δ)**
  - Q : 상태 집합
  - q_0 : 시작 상태
  - A : 목표 상태들의 집합
  - Σ : 입력 알파벳
  - δ : 상태 전이 함수
- **매칭이 진행된 상태들 사이의 관계를 오토마타로 표현**

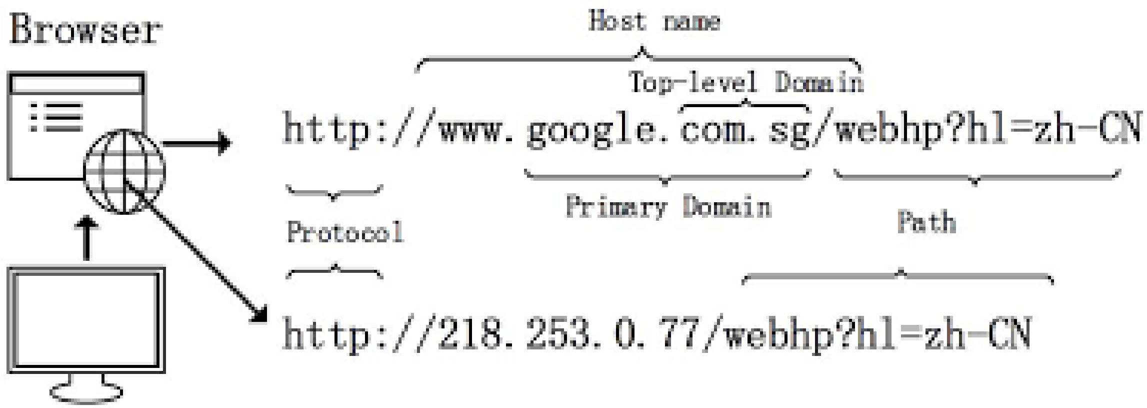
ababaca를 체크하는 오토마타



S: dvganbbactababaababacabababacaagbk…

오토마타를 이용한 매칭이 사용되는 예

- **악성코드(URL) 탐지**



탐지 규칙들

```
rule SMB_Worm_Tool_Generic {
    meta:
        description = "Generic SMB Worm/Malware Signature"
        author = "Florian Roth"
        reference = "http://goo.gl/N3zx1m"
        date = "2015/02/08"
        hash = "db6cae5734e433b195d8fc3252cbe58469e42bf3"
        score = 70
    strings:
        $mz = { 4d 5a }

        $s1 = "%s\\Admin$\\%s.exe" fullword ascii
        $s2 = "SVCHOST.EXE" fullword wide

        $a1 = "LoadLibrary( NTDLL.DLL ) Error:%d" fullword ascii
        $a2 = "\\svchost.exe" fullword ascii
        $a3 = "msvcrt.bat" fullword ascii
        $a4 = "Microsoft@ Windows@ Operating System" fullword wide
```

오토마타의 S/W 구현

상태 \ 입력문자	a	b	c	d	e	…	z
0	1	0	0	0	0	…	0
1	1	2	0	0	0	…	0
2	3	0	0	0	0	…	0
3	1	4	0	0	0	…	0
4	5	0	0	0	0	…	0
5	1	4	6	0	0	…	0
6	7	0	0	0	0	…	0
7	1	2	0	0	0	…	0

상태 \ 입력문자	a	b	c	기타
0	1	0	0	0
1	1	2	0	0
2	3	0	0	0
3	1	4	0	0
4	5	0	0	0
5	1	4	6	0
6	7	0	0	0
7	1	2	0	0

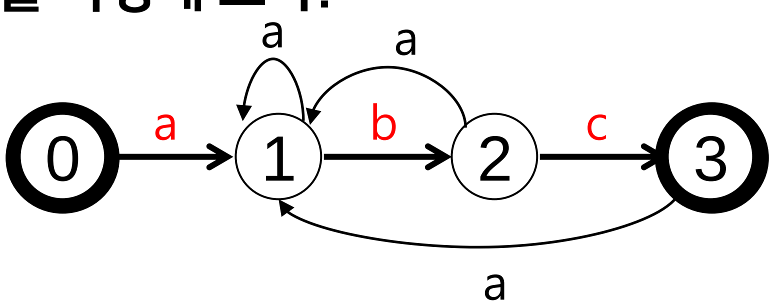
오토마타를 이용해 매칭을 체크하는 알고리즘

FA-Matcher (A, δ , f)
▷ f: 목표 상태
{
    ▷ n: 배열 A[ ]의 길이
    $q \leftarrow 0$;
    for $i \leftarrow 1$ **to** n {
        $q \leftarrow \delta(q, A[i])$;
        if $(q = f)$ **then** A[i-m+1]에서 매칭이 발생했음을 알린다;
    }
}

✓ 총 수행시간: $\Theta(n + |\Sigma|m)$

연습문제(1/2)

- 입력 문자 집합이 {'a', 'b', 'c'}일 때, 패턴 문자열 "abc"를 식별하는 오토마타이다.
- 상태 전이 테이블을 작성해 보자.



상태0 에서 각각 'a', 'b', 'c'가 들어왔을 때 상태 전이

```
tbl = [
    {'a': 0, 'b': 0, 'c': 0},
    {'a': 0, 'b': 0, 'c': 0},
    {'a': 0, 'b': 0, 'c': 0},
    {'a': 0, 'b': 0, 'c': 0}
]
```

연습문제(2/2)

- 앞의 테이블을 사용해서 “abc”가 포함된 인덱스를 출력

```
A = "aaabcababbabcabcccaab"
P = "abc"
state = 0
final = 3

for i in range(len(A)):
```

```
2
10
13
```

문자열 매칭(2)

라빈-카프Rabin-Karp 알고리즘

- **문자열 패턴을 수치로 바꾸어 문자열의 비교를 수치 비교로 대신한다.**

- **수치화**
  - 가능한 문자 집합 ∑의 크기에 따라 진수가 결정된다.
  - 예: ∑ = {a, b, c, d, e}
    - |∑| = 5
    - A, b, c, d, e를 각각 0, 1, 2, 3, 4에 대응시킨다.

    - 문자열 “cad”를 수치화하면
    - $2*5^2+0*5^1+3*5^0 = 28$

라빈-카프Rabin-Karp 알고리즘

- **기본적인 원리만 설명**

a	b	c	d	e	f	g	h	i	j
0	1	2	3	4	5	6	7	8	9

A[] = "abcdhij"
P[] = "cdh" → 237

abc → 012
bcd → (012 - 0 * 100) * 10 + 3 = 123
cdh → (123 – 1 * 100) * 10 + 7 = 237

수치화를 이용한 매칭의 예

P[ ]	e	e	a	a	b

$p = 4*5^4 + 4*5^3 + 0*5^2 + 0*5^1 + 1 = 3001$

A[ ]	a	c	e	b	b	c	e	e	a	a	b	c	e	e	d	b

$a_1=0*5^4+2*5^3+4*5^2+1*5^1+1 = 356$

a	c	e	b	b	c	e	e	a	a	b	c	e	e	d	b

$a_2= 5(a_1\text{-}0*5^4)+2 = 1782$

a	c	e	b	b	c	e	e	a	a	b	c	e	e	d	b

$a_3=5(a_2\text{-}2*5^4)+4 = 2664$

…

a	c	e	b	b	c	e	e	a	a	b	c	e	e	d	b

$a_7=5(a_6\text{-}2*5^4)+1 = 3001$

…

라빈-카프 알고리즘

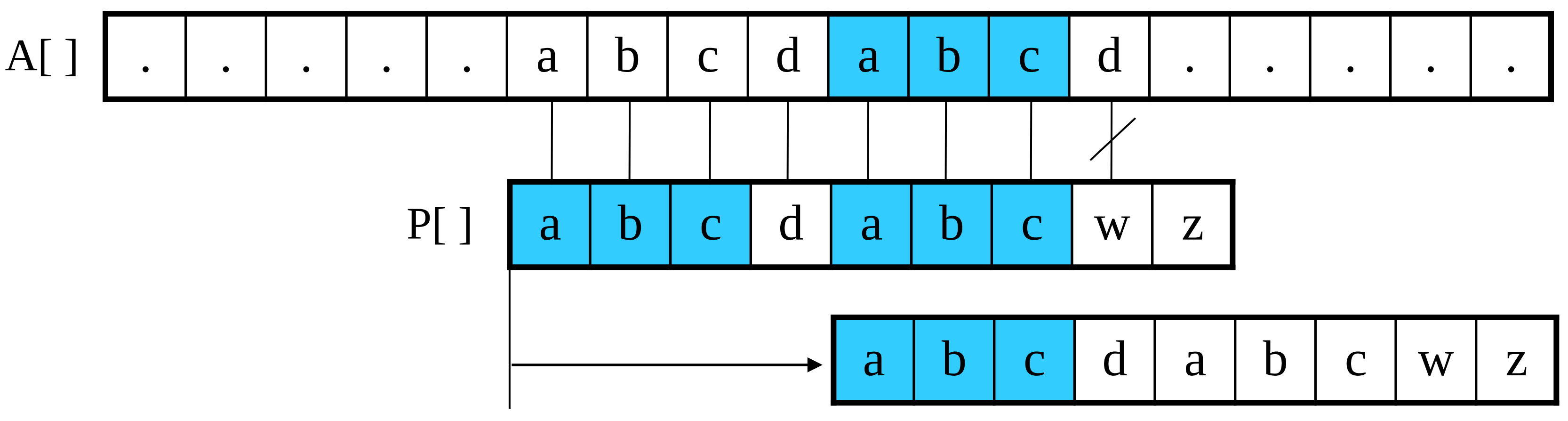
(몇 가지 문제점이 개선되었음.
설명은 생략함)

RabinKarp(A, P, d, q)
{
    ▷ n : 배열 A[ ]의 길이, m : 배열 P[ ]의 길이
    $p \leftarrow 0$; $b_1 \leftarrow 0$;
    for $i \leftarrow 1$ **to** m {   ▷ b_1 계산
        $p \leftarrow (dp + \mathrm{P}[i]) \bmod q$;
        $b_1 \leftarrow (db_1 + \mathrm{A}[i]) \bmod q$;
    }
    $h \leftarrow d^{m\text{-}1} \bmod q$;
    for $i \leftarrow 1$ **to** $n\text{-}m\text{+}1${
        if $(i \neq 1)$ **then** $b_i \leftarrow (d(b_{i\text{-}1} - h\mathrm{A}[i\text{-}1]) + \mathrm{A}[i+m\text{-}1]) \bmod q$;
        if $(p = b_i)$ **then**
            if $(\mathrm{P}[1\ldots m] = \mathrm{A}[i\ldots i+m\text{-}1])$ **then**
                A[i] 자리에서 매칭이 되었음을 알린다;
    }
}

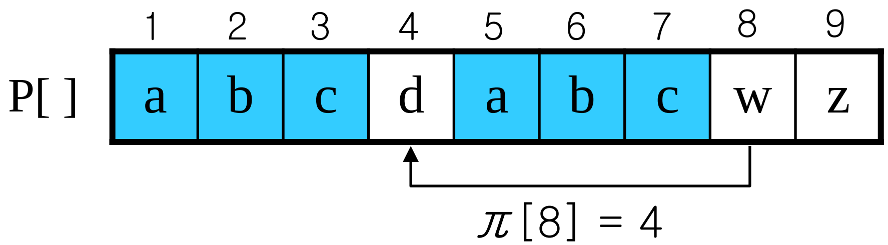
✓ 평균 수행시간: $\Theta(n)$

KMP Knuth-Morris-Pratt 알고리즘

- **오토마타를 이용한 매칭과 동기가 유사**
- **공통점**
  - 매칭에 실패했을 때 돌아갈 상태를 준비해둔다
  - 오토마타를 이용한 매칭보다 준비 작업이 단순하다

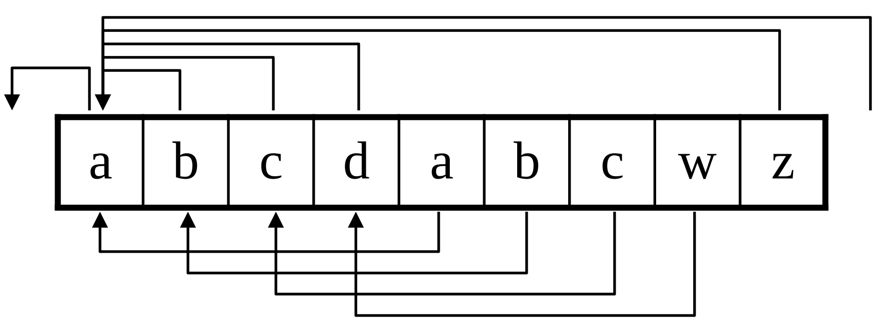


매칭이 실패했을 때 돌아갈 곳 준비 작업



텍스트에서 abcdabc까지는 매치되고, w에서 실패한 상황
패턴의 맨 앞의 abc와 실패 직전의 abc는 동일함을 이용할 수 있다
실패한 텍스트 문자와 P[4]를 비교한다

	1	2	3	4	5	6	7	8	9	10
π[ ]	0	1	1	1	1	2	3	4	1	1



패턴의 각 위치에 대해
매칭에 실패했을 때
돌아갈 곳을 준비해 둔다

KMP 알고리즘

```
KMP(A[ ], P[ ])
{
    preprocessing(P);
    i ← 1;  ▷ 본문 문자열 포인터
    j ← 1;  ▷ 패턴 문자열 포인터
    ▷ n: 배열 A[ ]의 길이, m: 배열 P[ ]의 길이
    while (i ≤ n) {
        if (j = 0 or A[i] = P[j])
                then { i++; j++; }
                else j ← π [j];
        if (j = m+1) then {
                A[i-m]에서 매치되었음을 알림;
                j ← π [j];
        }
    }
}
```

✓수행시간: $\Theta(n)$

준비 작업

```
preprocessing(P)
{
    i ← 1;  ▷ 본문 문자열 포인터
    k ← 1;  ▷ 패턴 문자열 포인터
    while (j ≤ m) {
        if (k = 0 or A[j] = P[k])
            then { j++;  k++; π [j] ← k; }
            else  k ← π [k];
        if (j = m+1) then {
            A[i-m]에서 매치되었음을 알림;
            j ← π [j];
        }
    }
```

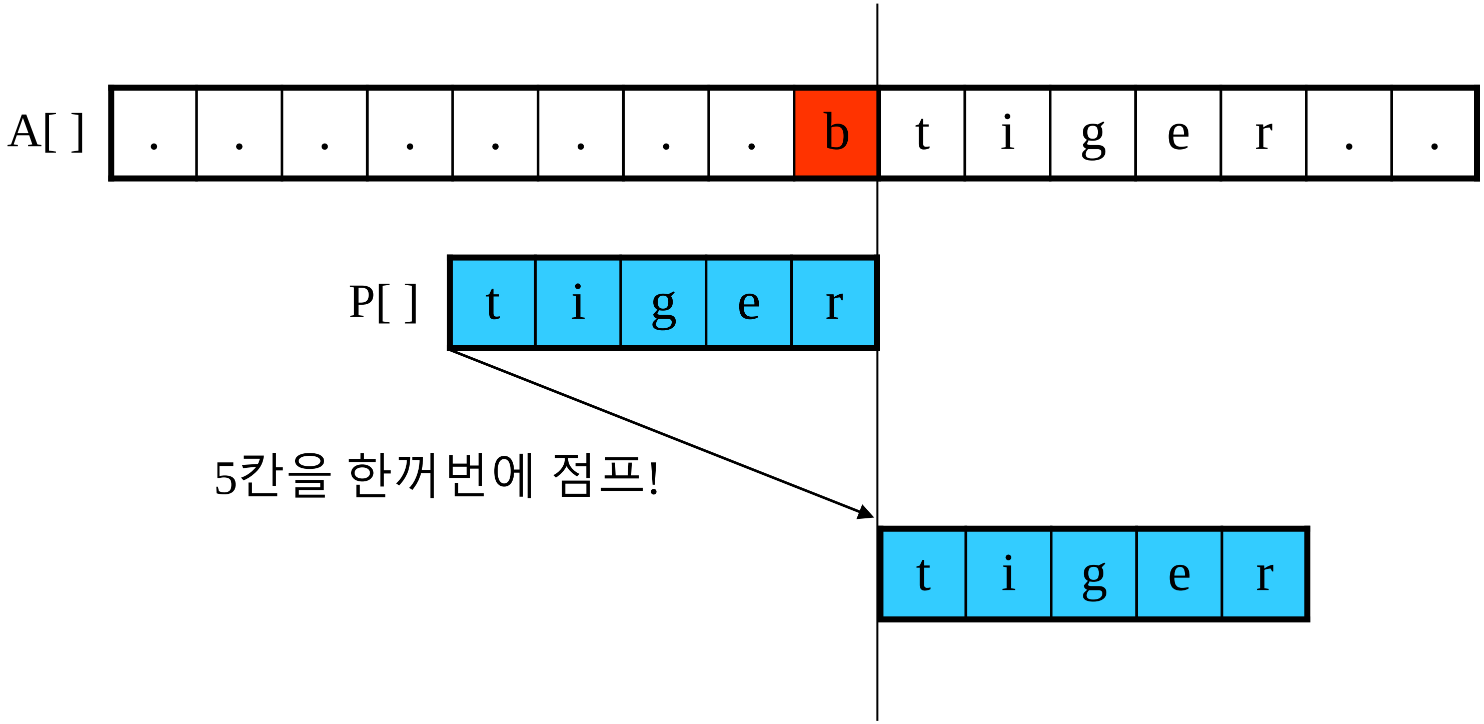
✓수행시간: $\Theta(m)$

보이어-무어Boyer-Moore 알고리즘

- **앞의 매칭 알고리즘들의 공통점**
  - 텍스트 문자열의 문자를 적어도 한번씩 훑는다.
  - 따라서 최선의 경우에도 $\Omega(n)$
- **보이어-무어 알고리즘은 텍스트 문자를 다 보지 않아도 된다.**
  - 발상의 전환: 패턴의 오른쪽부터 비교한다.

Motivation

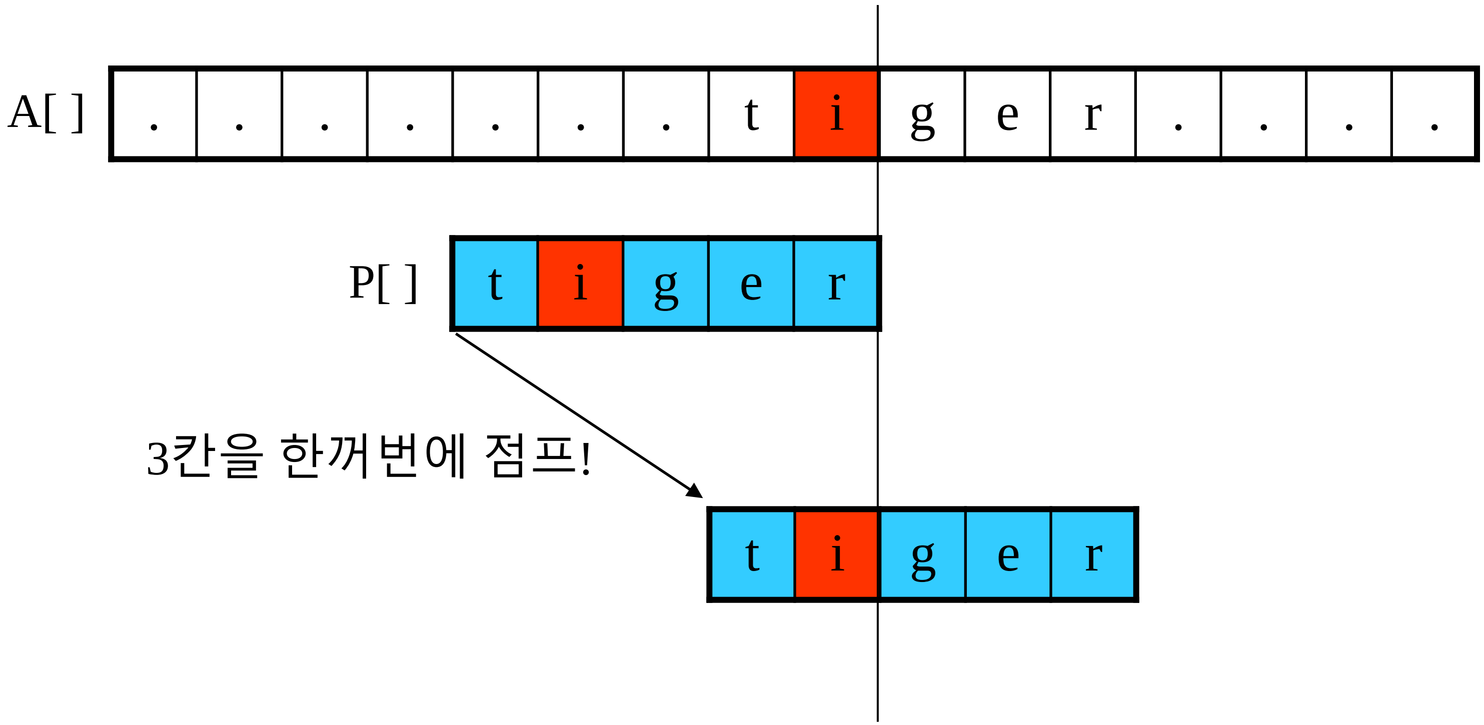
상황: 텍스트의 b와 패턴의 r을 비교하여 실패했다



✓ 관찰: 패턴에 문자 b가 없으므로

패턴이 텍스트의 b를 통째로 뛰어넘을 수 있다

상황: 텍스트의 i와 패턴의 r을 비교하여 실패했다



✓ 관찰: 패턴에서 i가 r의 3번째 왼쪽에 나타나므로 패턴이 3칸을 통째로 움직일 수 있다

점프 정보 준비

패턴 “tiger”에 대한 점프 정보

오른쪽 끝문자	t	i	g	e	r	기타
jump	4	3	2	1	5	5

패턴 “rational”에 대한 점프 정보

오른쪽 끝문자	r	a	t	i	o	n	a	l	기타
jump	7	6	5	4	3	2	1	8	8

오른쪽 끝문자	r	t	i	o	n	a	l	기타
jump	7	5	4	3	2	1	8	8

NP-완비

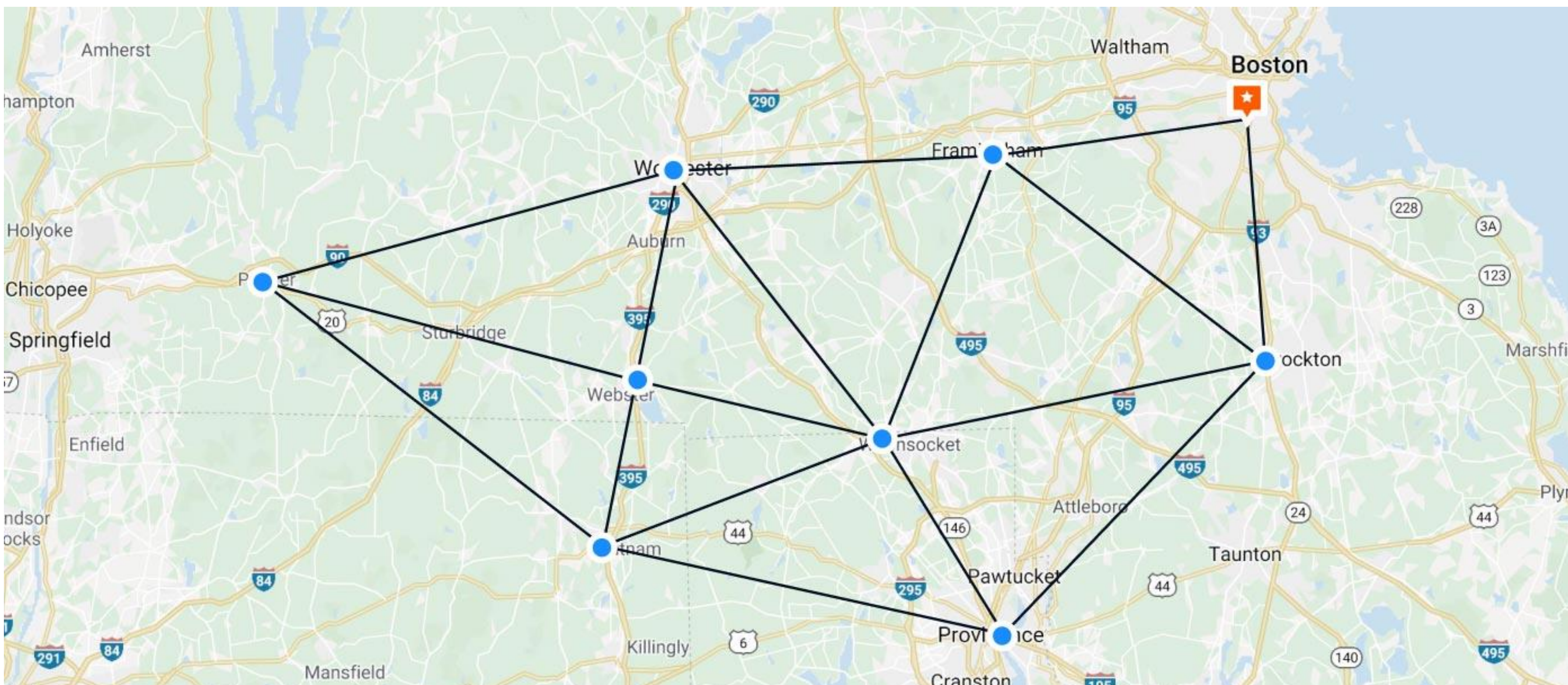
수행 시간의 현실성

- **다항식시간 Polynomial-time**
  - 입력의 크기 n의 다항식으로 표시되는 시간
  - 예: $3n^k + 5n^{k-1} + \cdots$

- **비다항식 시간 Non-polynomial-time**
  - 지수 시간. 예) 2^n
  - 계승시간. 예) **n!**

TSP (외판원 문제)

- **TSP(Traveling Salesman Problem)**
  - 각 도시를 한 번씩 방문하고 출발점으로 돌아오는 최단 경로



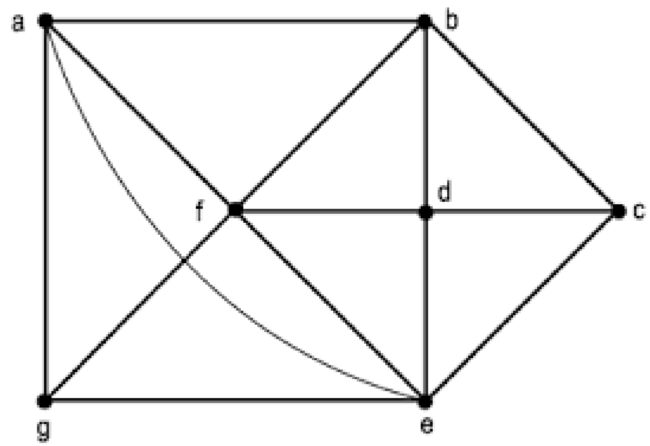
TSP (외판원 문제)

- **TSP(Traveling Salesman Problem)**
  - 입력: 가중치 있는 완전 그래프
  - 출력: 모든 도시들을 단 한 번만 방문하고 원래 시작점으로 돌아오는 최소 비용의 이동 순서
  - 현실적인 시간(다항식 시간)에 해를 구할 수 없는 대표적인 문제

TSP (외판원 문제)

- **해밀턴 경로 Hamiltonian Path**
  - 모든 정점을 단 한번씩만 방문하는 경로

- **해밀턴 싸이클 Hamiltonian Cycle**
  - 모든 정점을 단 한번씩만 방문하고 시작점으로 돌아오는 경로



- **TSP**
  - 가장 짧은 해밀턴 싸이클을 구하는 문제

문제의 종류

- **Yes/No 문제**
  - 예: 그래프 G에서 길이가 k 이하인 해밀토니안 경로가 존재하는가?
- **최적화 문제**
  - 예: 그래프 G에서 길이가 가장 짧은 해밀토니안 경로는 얼마인가?

✓ 두 문제는 동전의 앞뒷면

NP-완비NP-Complete 이론

- **문제를 현실적인 시간에 풀 수 있는가?**
- **에 관한 이론**

- **Yes/No 의 대답을 요구하는 문제에 국한**
  – 그렇지만 최적화 문제와 밀접한 관계를 가지고 있다.

- **거대한 군을 이룸**
  – 이 중 한 문제만 현실적인 시간에 풀면 다른 모든 것도 저절로 풀리는 논리적 연결관계를 가지고 있다.

현재까지의 연구결과

- **어떤 문제가 NP-완비임이 확인되면…**

- **→ 이 문제를 현실적인 시간에 풀 수 없다?**
- - 아직 증명되지 않음

- **→ 이 문제를 현실적인 시간에 풀 수 없다?**
- - 지금까지의 연구결과로는 아직 없다.
- - **이 문제를 현실적인 시간에 풀 수 없다고 강하게 추정된다. (O)**

NP-완비에 관한 비유

상사가 아주 어려운 문제를 해결하라고 지시했다.

(b)

NP-완비에 관한 비유

NP-완비 이론의 상황을 비유적으로 보여줌

P와 NP

- **P 문제: Polynomial**
  - 다항식 시간에 Yes 또는 No 대답을 할 수 있으면 P

- **NP 문제: Nondeterministic Polynomial**
  - **Non-Polynomial의 준말이 아님!**

  - Yes 대답이 나오는 해를 제공했을 때,
  - 이것이 **Yes 대답을 내는 해라는 사실을 다항식 시간에 확인**해 줄 수 있으면 NP

- **어떤 문제가 NP임을 보이는 것은 대부분 아주 쉽다.**
  - NP-완비 증명에서 형식적으로 확인하고 넘어가는 정도

문제의 종류

풀 수 없는 문제들

(Unsolvable)

(Undecidable)

정지 문제

힐버트의 10번째 문제

…

풀 수 있는 문제들

(Solvable)

(Decidable)

Presburger 산술

…

NP-완비 문제들

여기에 속할 것이라고 강력히 추정!

현실적인 시간내에 풀 수 없는 문제들

최소 신장 트리 문제

최단 거리 문제

…

현실적인 시간내에 풀 수 있는 문제들

NP-완비/하드

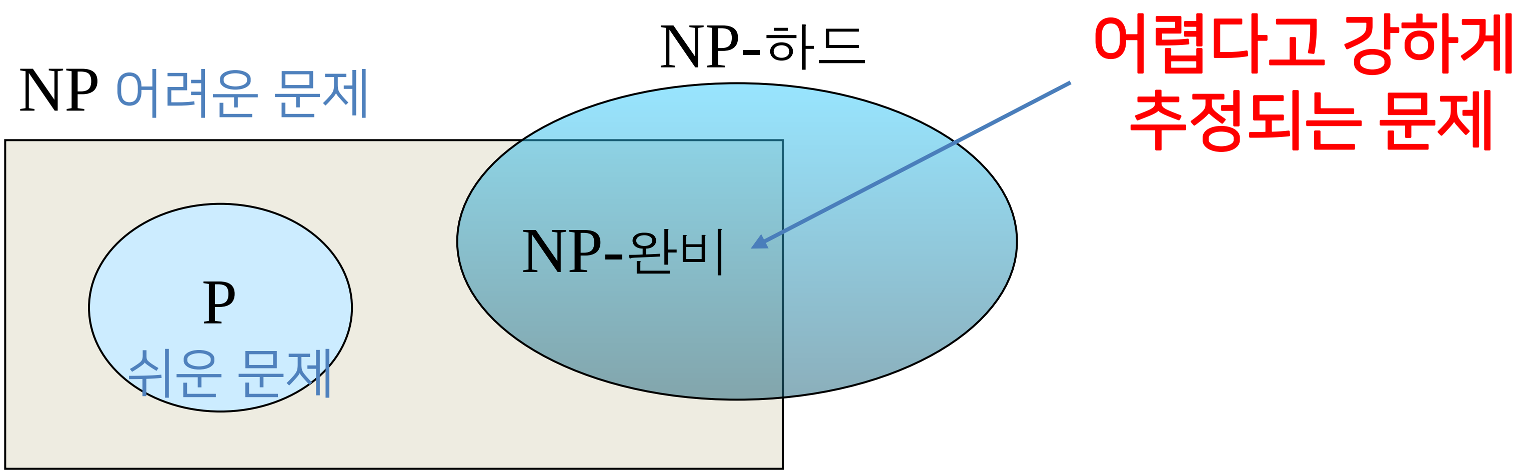
- **다음 성질을 만족하면 문제 L은 NP-하드이다.**
  - 모든 NP 문제가 L로 다항식 시간에 변환 가능하다.
- **다음의 두 성질을 만족하면 문제 L은 NP-완비이다.**
  1) L은 NP이다.
  2) L은 NP-하드이다.

✓ NP-완비는 NP-하드의 일부. NP-완비인 문제를 NP-하드라고 불러도 맞다.

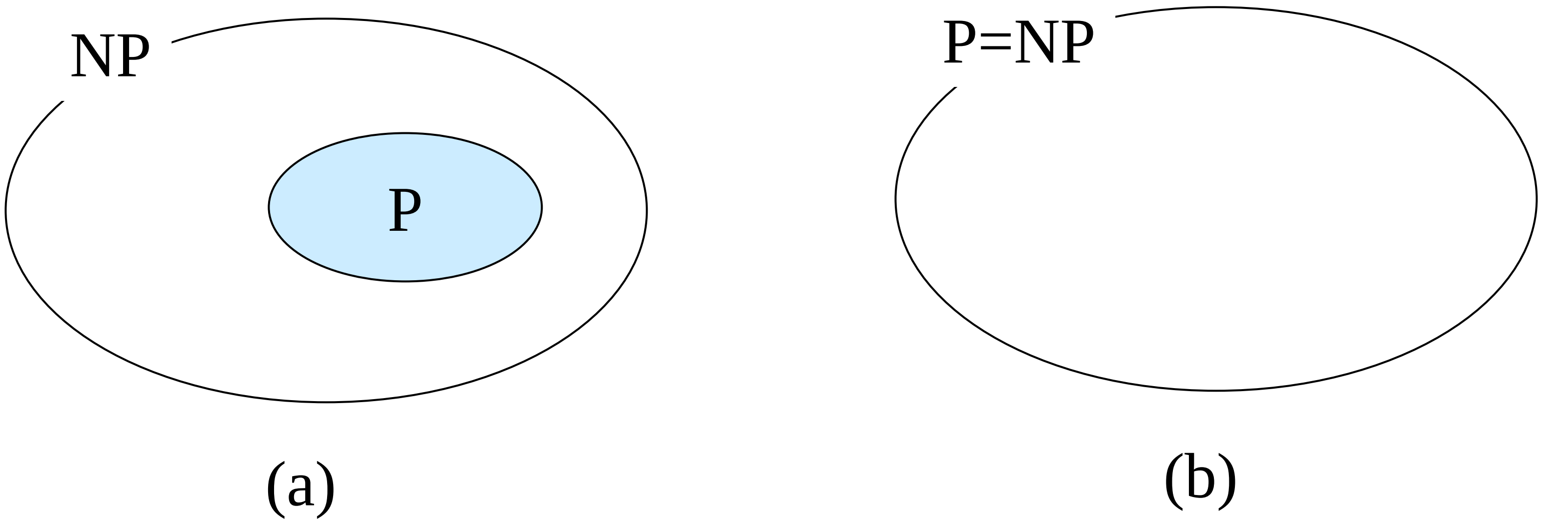
✓ NP-완비의 성질 1)은 대부분 자명하므로 핵심에 집중하기 위해 NP-하드에 초점을 맞추자.

NP와 NP-완비, NP-하드의 관계

NP이고 NP-하드이면 NP-완비이다.



P와 NP의 포함 관계



✓ 위 (a)인지 (b)인지는 아직 밝혀지지 않음.
현대 수학의 7대 난제.
백만불의 상금이 걸려 있다.

NP-완비의 증명

다항식 시간 변환

문제1: 정수 $x = x_1x_2\cdots x_n$ 은 3의 배수인가?

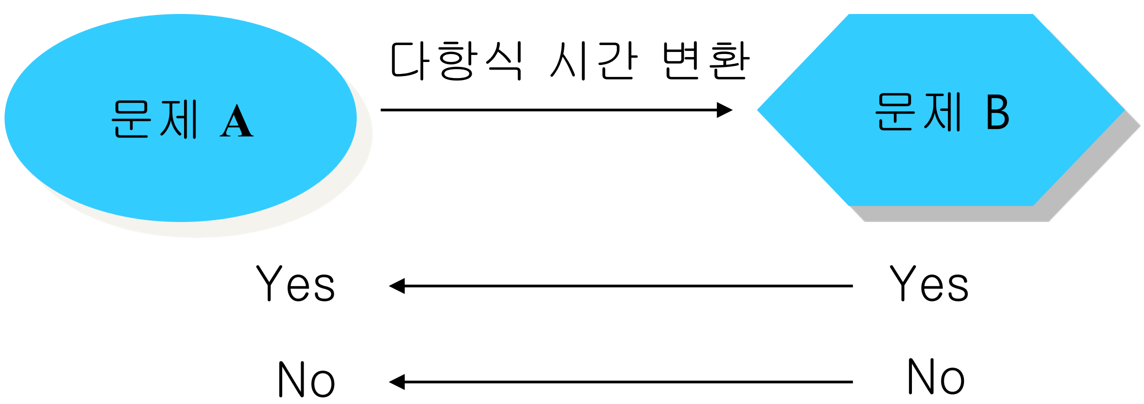
문제2: $x_1+x_2+\ldots+x_n$은 3의 배수인가?

- ✓위 두 문제의 대답은 같다.
  - ➢Yes/No 대답이 일치한다.
- ✓문제 2가 쉬우면, 문제 1도 쉽다.

쉽다 = 현실적인 시간에 풀 수 있다.

다항식 시간 변환

– 문제 B는 쉽다 .
– 문제 A는 Yes/No 대답이 일치하는 문제 B로 쉽게 변형된다.



✓ 문제 A도 쉬운가?

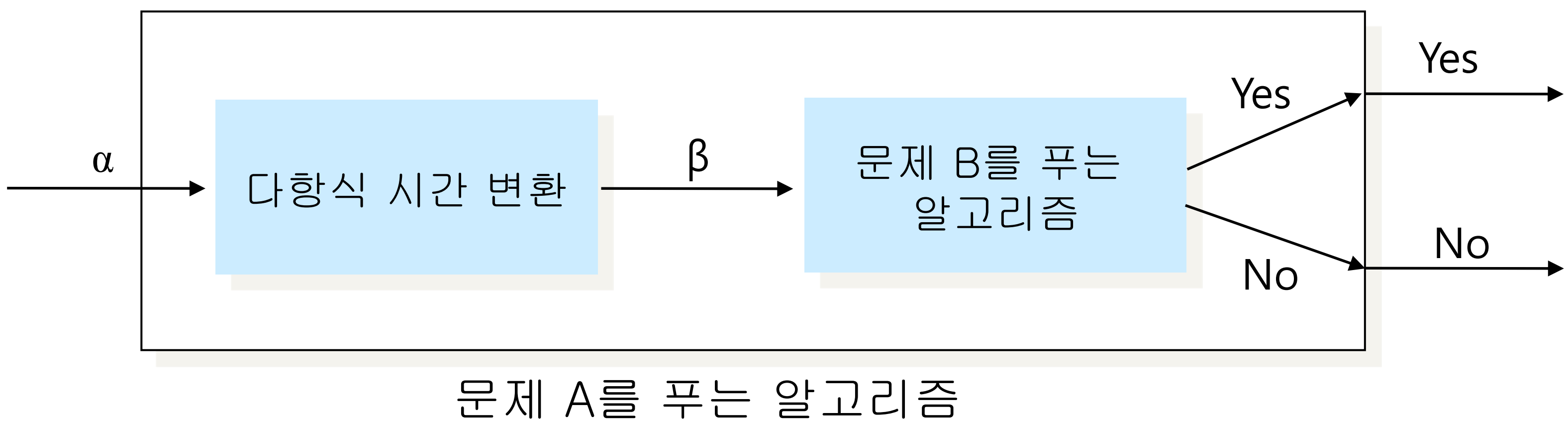
다항식 시간 변환

- **문제 A의 사례 α를 문제 B의 사례 β로 바꾸되 아래 성질을 만족하면 다항식 시간 변환이라 하고, 이를 α ≤ β 로 표기한다.**

① 변환은 다항식 시간에 이루어진다.

② 두 사례의 답은 일치한다.

다항식 시간 변환



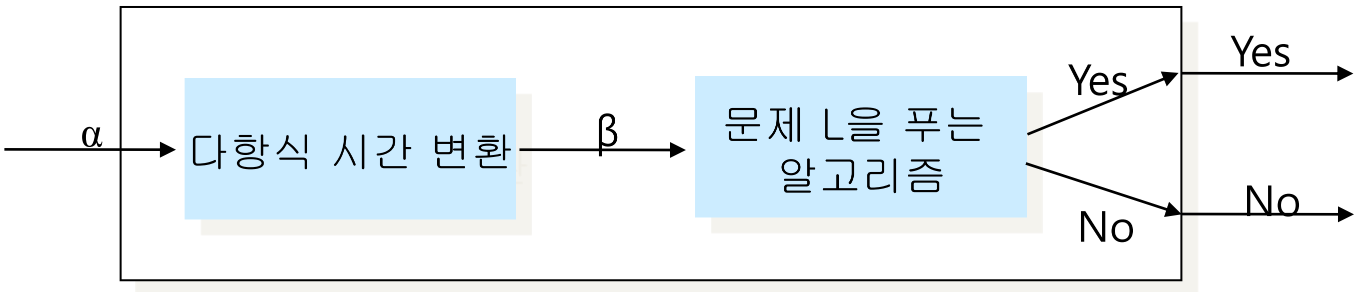
1. 문제 A를 다항식 시간에 문제 B로 변환한다.
2. 변환된 문제 B를 푼다.
3. 문제 B의 대답이 Yes이면 Yes, No이면 No를 리턴한다.

✓문제 B가 쉬운 문제라면 문제 A도 쉬운 문제

정리 1

문제 L이 NP이고 NP-하드이면 NP-완비

- **문제 L이 다음의 성질을 만족해도 NP-하드이다.**
  - 알려진 임의의 NP- 하드 문제 A로부터 문제 L로 다항식 시간에 변환 가능하다.



문제 A를 푸는 알고리즘

✓ 만일 문제 L을 쉽게 풀 수 있다면, 문제 A도 쉽게 풀 수 있다.

➔ 그러므로 모든 NP 문제를 쉽게 풀 수 있다.

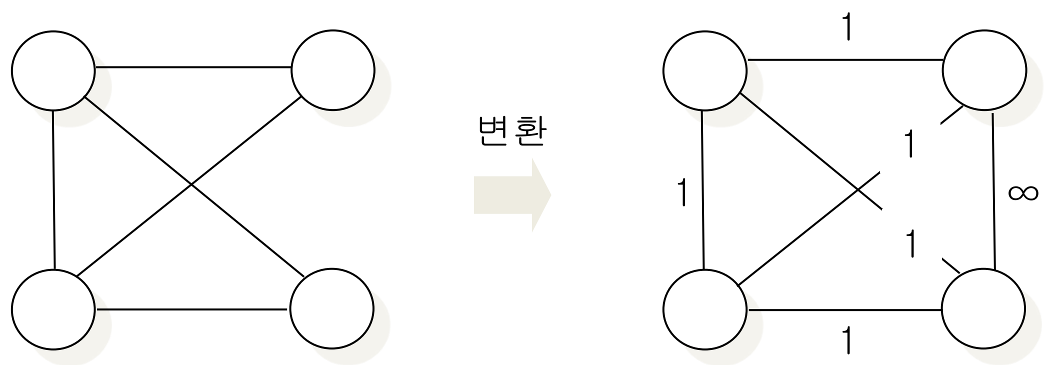
NP-하드 증명의 예

- **해밀토니안 싸이클 문제가 NP-하드임은 알고 있다 가정**
- **이를 이용해서 TSP 문제가 NP-하드임을 보일 수 있다**

- 해밀토니안 싸이클
  - 그래프의 모든 정점을 단 한번씩 방문하고 돌아오는 경로

- 해밀토니안 싸이클 문제
  - 주어진 그래프에서 해밀토니안 싸이클이 존재하는가?

- 해밀토니안 싸이클 문제의 사례 A를 아래와 같이 TSP 문제의 사례 B로 다항식 시간에 변환한다



해밀토니안 싸이클 문제의 사례　　TSP 문제의 사례

사례 A가 해밀토니안 싸이클을 갖는다.

⇔ 사례 B가 길이 4 이하인 해밀토니안 싸이클을 갖는다

(그래프의 크기가 n이면 4 대신 n)

➢ 그러므로 TSP는 NP-하드이다.

정점 수와 일치

NP 이론의 유용성

- **어떤 문제가 NP-완비/하드임이 확인되면**
  ⇨ 쉬운 알고리즘을 찾으려는 헛된 노력은 일단 중지한다.
  ⇨ 주어진 시간 예산 내에서 최대한 좋은 해를 찾는 알고리즘 (휴리스틱) 개발에 집중한다.

Remind: 때로는 어떤 것이 불가능하다는 사실이 유용할 때도 있다.
-- 레오나드 레빈