

# KMP

최백준 [choi@startlink.io](mailto:choi@startlink.io)

---

# 문자열 매칭 알고리즘

---

# 문자열 매칭 알고리즘

## String Matching Algorithm

- 문자열 S에서 패턴 P를 찾는 알고리즘
- S에서 가장 먼저 나타나는 P를 찾아야 함

- $S$ 의 길이:  $|S|$   
 $P$  :  $|P|$
- $S = \text{"ABCABDABCABEABC"}$
- $P = \text{"ABCABE"}$
- $|S| \cdot |P|$
- $S[6]$ 부터 P가 나타남!

aaaaaaaaab

aaab

1 2 3 4 5 6 7      1 2 3 4 5 6 7

aaaaaaaaab      aaaaa

aaaaaaabaaaa

# 문자열 매칭 알고리즘

String Matching Algorithm

4

- $O(|S| |P|)$
- 모든 경우를 다 해보는 알고리즘
- <https://gist.github.com/Baekjoon/7ad611f0c4dfe6f88017>

KMP

---

# KMP

## String Matching Algorithm

- KMP는 왜 KMP일까?
- <https://www.acmicpc.net/problem/2902>
- 만든 사람의 성이 Knuth, Morris, Prett이라서

# KMP

## String Matching Algorithm

- KMP는 pi 배열을 이용해야 한다

- $pi[i]$

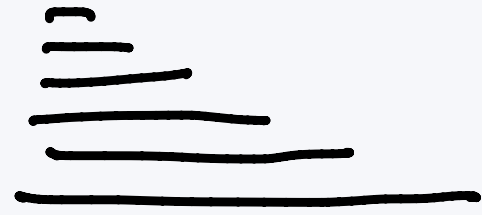
- P의 i까지 부분 문자열에서  $prefix == suffix$ 가 될 수 있는 부분 문자열 중에서 가장 긴 것의 길이
- 이 때, prefix가 i까지 부분 문자열과 같으면 안된다.

$pi[i] = S$ 의 i까지 부분문자열  
suffix == prefix

# Prefix

## String Matching Algorithm

- P = ABCABE



6??

- Prefix
- A
- AB
- ABC
- ABCA
- ABCAB
- ABCABE



# Suffix

## String Matching Algorithm

- $P = \text{ABCABE}$



- Suffix
- E
- BE
- ABE
- CABE
- BCABE
- ABCABE

# KMP

## String Matching Algorithm

- ABCABE 의 pi[]

S = A B C A B E

i	부분 문자열	pi[i]
0	A	0
1	AB	0
2	ABC	0
3	<u>ABCA</u>	1
4	<u>ABCAB</u>	2
5	ABCABE	0

# KMP

## String Matching Algorithm

- ABCABDABCABEABC 의 pi

i	부분 문자열	pi[i]
0	A	0
1	AB	0
2	ABC	0
3	ABCA	1
4	ABCAB	2
5	ABCABD	0
6	ABCABDA	1

# KMP

## String Matching Algorithm

- ABCABDABCABEABC 의 pi

i	부분 문자열	pi[i]
7	ABCABDAB	2
8	ABCABDABC	3
9	ABCABDABCA	4
10	ABCABDABCAB	5
11	ABCABDABCABE	0
12	ABCABDABCABEA	1
13	ABCABDABCABEAB	2
14	ABCABDABCABEABC	3

# KMP

## String Matching Algorithm

- <https://gist.github.com/Baekjoon/aea48836f93633e83920>

# KMP

## String Matching Algorithm

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- $pi[4] = 2$ 라는 것은
- $p[0 \dots 1] == p[3 \dots 4]$  라는 것을 의미
- ABCAB**

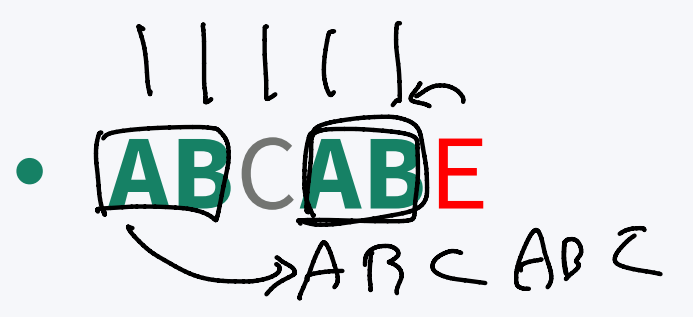
# KMP

## String Matching Algorithm

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- pi[4] = 2라는 것은
- $p[0 \cdots 1] == p[3 \cdots 4]$  라는 것을 의미
- ABCABE에서 ABCABE를 찾을 때
- ABCABE



# KMP

16

## String Matching Algorithm

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- pi[4] = 2라는 것은
- $p[0 \cdots 1] == p[3 \dots 4]$  라는 것을 의미
- ABCABCAE에서 ABCABE를 찾을 때
- ABCABCAE
- ABCABE (앞의 2개를 건너뛰고 비교를 이어가도 된다)
- ABCABE



# KMP

## String Matching Algorithm

- 패턴 ABCABE의  $pi$

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기

[illegible]

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 0, j = 0$ )

[illegible]

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 1, j = 1$ )

[illegible]

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 2, j = 2$ )

[illegible]

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 3, j = 3$ )

[illegible]

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 4, j = 4$ )

[illegible]

# KMP

## String Matching Algorithm

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 5, j = 5$ ) 다르기 때문에  $j = pi[j-1]$ 로 이동한다

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	A	B	C	A	B	D	A	B	C	A	B	C	A	B	E	F
P	A	B	C	A	B	E										

Handwritten annotations: A box around P[0] and P[1] (A, B), a box around P[3] and P[4] (A, B), and a circled '2' with an arrow pointing to P[5] (E).



# KMP

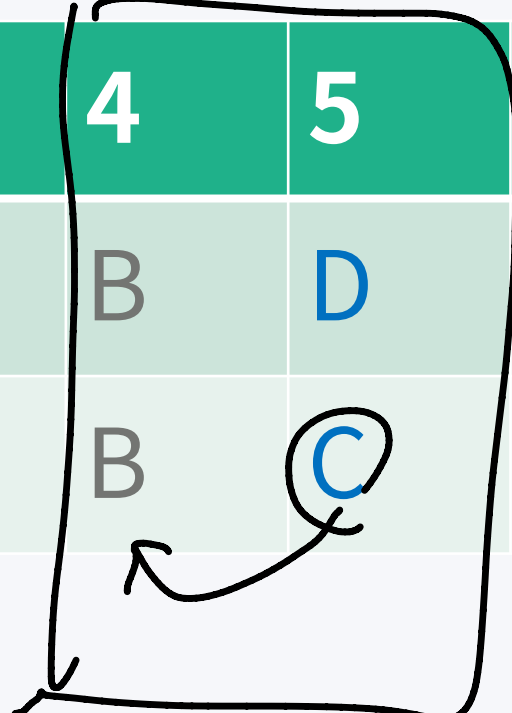
## String Matching Algorithm

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 5, j = 2$ ) 다르기 때문에  $j = pi[j-1]$ 로 이동한다

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	A	B	C	A	B	D	A	B	C	A	B	C	A	B	E	F
P				A	B	C	A	B	E							



# KMP

## String Matching Algorithm

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCEEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 5, j = 0$ ) 다른데,  $j = 0$ 이므로 다음으로 넘어간다

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	A	B	C	A	B	D	A	B	C	A	B	C	A	B	E	F
P						A	B	C	A	B	E					

# KMP

## String Matching Algorithm

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 6, j = 0$ )

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	A	B	C	A	B	D	A	B	C	A	B	C	A	B	E	F
P							A	B	C	A	B	E				

# KMP

## String Matching Algorithm

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 7, j = 1$ )

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	A	B	C	A	B	D	A	B	C	A	B	C	A	B	E	F
P							A	B	C	A	B	E				

# KMP

## String Matching Algorithm

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 8, j = 2$ )

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	A	B	C	A	B	D	A	B	C	A	B	C	A	B	E	F
P							A	B	C	A	B	E				

# KMP

## String Matching Algorithm

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCEEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 9, j = 3$ )

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	A	B	C	A	B	D	A	B	C	A	B	C	A	B	E	F
P							A	B	C	A	B	E				

# KMP

## String Matching Algorithm

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 10, j = 4$ )

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	A	B	C	A	B	D	A	B	C	A	B	C	A	B	E	F
P							A	B	C	A	B	E				

# KMP

## String Matching Algorithm

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 11, j = 5$ ) 다르기 때문에,  $j = pi[j-1]$ 로 이동한다

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	A	B	C	A	B	D	A	B	C	A	B	C	A	B	E	F
P							A	B	C	A	B	E				

Handwritten annotations: A box around S[6:8] (ABD) and P[6:8] (ABC). A box around S[9:11] (AC) and P[9:11] (AB). A box around S[11] (C) and P[11] (E). A box around S[12] (A) and P[12] (A). A box around S[13] (B) and P[13] (B). A box around S[14] (E) and P[14] (E). A box around S[15] (F) and P[15] (F). A vertical line is drawn through the table between indices 8 and 9.



# KMP

## String Matching Algorithm

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 11, j = 2$ )

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	A	B	C	A	B	D	A	B	C	A	B	C	A	B	E	F
P										A	B	C	A	B	E	

# KMP

## String Matching Algorithm

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 12, j = 3$ )

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	A	B	C	A	B	D	A	B	C	A	B	C	A	B	E	F
P										A	B	C	A	B	E	

# KMP

## String Matching Algorithm

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

- 문자열 ABCABDABCABCABEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 13, j = 4$ )

i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	A	B	C	A	B	D	A	B	C	A	B	C	A	B	E	F
P										A	B	C	A	B	E	

# KMP

String Matching Algorithm

AAAA B  
0 1 2 3 4  
2

AAAAA A A A A A A B  
A A A A B  
A A A A B

36

- 패턴 ABCABE의 pi

i	0	1	2	3	4	5
p[i]	A	B	C	A	B	E
pi[i]	0	0	0	1	2	0

$|S| + |P|$

- 문자열 ABCABDABCABCEEF 에서 패턴 ABCABE를 찾아보기
- $S[i] == P[j]$ 를 비교하기 ( $i = 14, j = 5$ ) 찾았다!

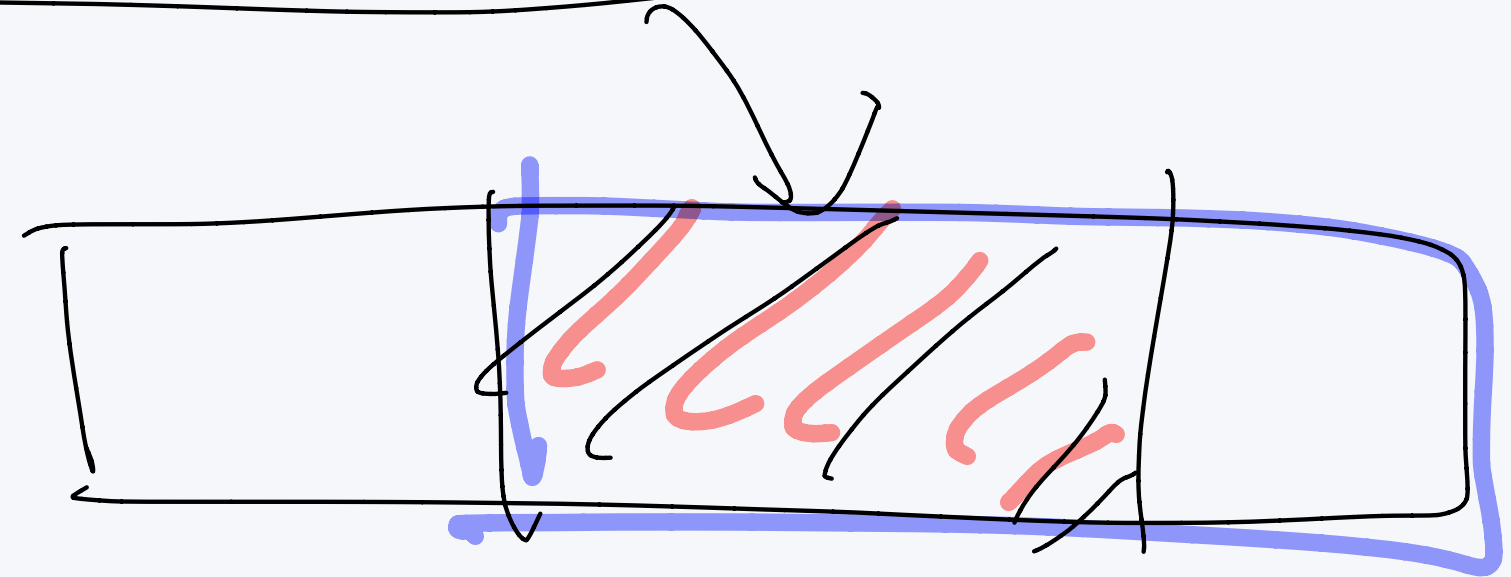
i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
S	A	B	C	A	B	D	A	B	C	A	B	C	A	B	E	F
P										A	B	C	A	B	E	

# KMP

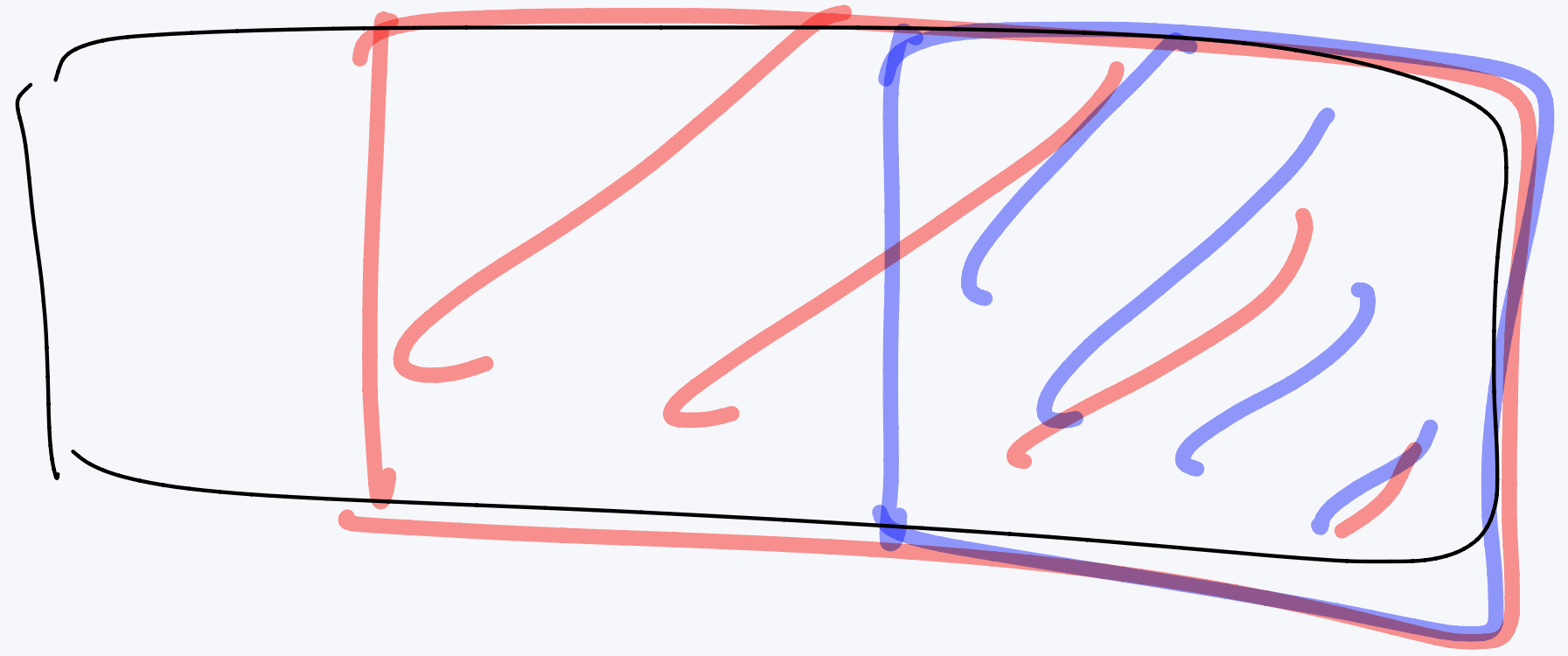
String Matching Algorithm

- pi 배열 구하는데 걸리는 시간  $O(M)$
- 문자열 매칭하는데 걸리는 시간  $O(N+M)$

Sei 부분 문자열



Sei Suffix의 prefix



# 찾기

38

<https://www.acmicpc.net/problem/1786>

- 문자열 T가 주어졌을 때 P가 몇 번 등장하는지 구하는 문제

# 찾기

<https://www.acmicpc.net/problem/1786>

- <https://gist.github.com/Baekjoon/8d3e2a1b93e59879011f>

# 광고

aaba

<https://www.acmicpc.net/problem/1305>

- 광고판의 크기: L
- 광고 문구의 길이: N
- 광고 문구: aaba
- 광고판의 크기: 5
- 인 경우
- aabaa -> abaaa -> baaab -> aaaba -> ...

aabaa aaba aabaa aaba

5

$L \leq 1000$

aabaa

aabaa

aaba

aab

abaaa

baaab

aaaba

↑  
aaaba

1aab (4)

aaa

aa

a

$L - P[i][L]$

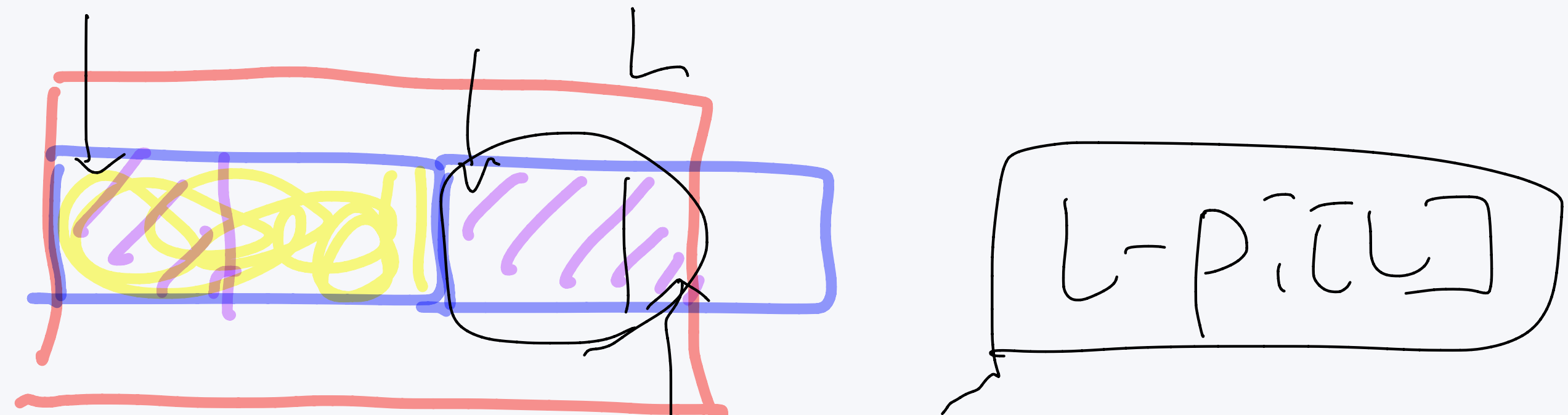
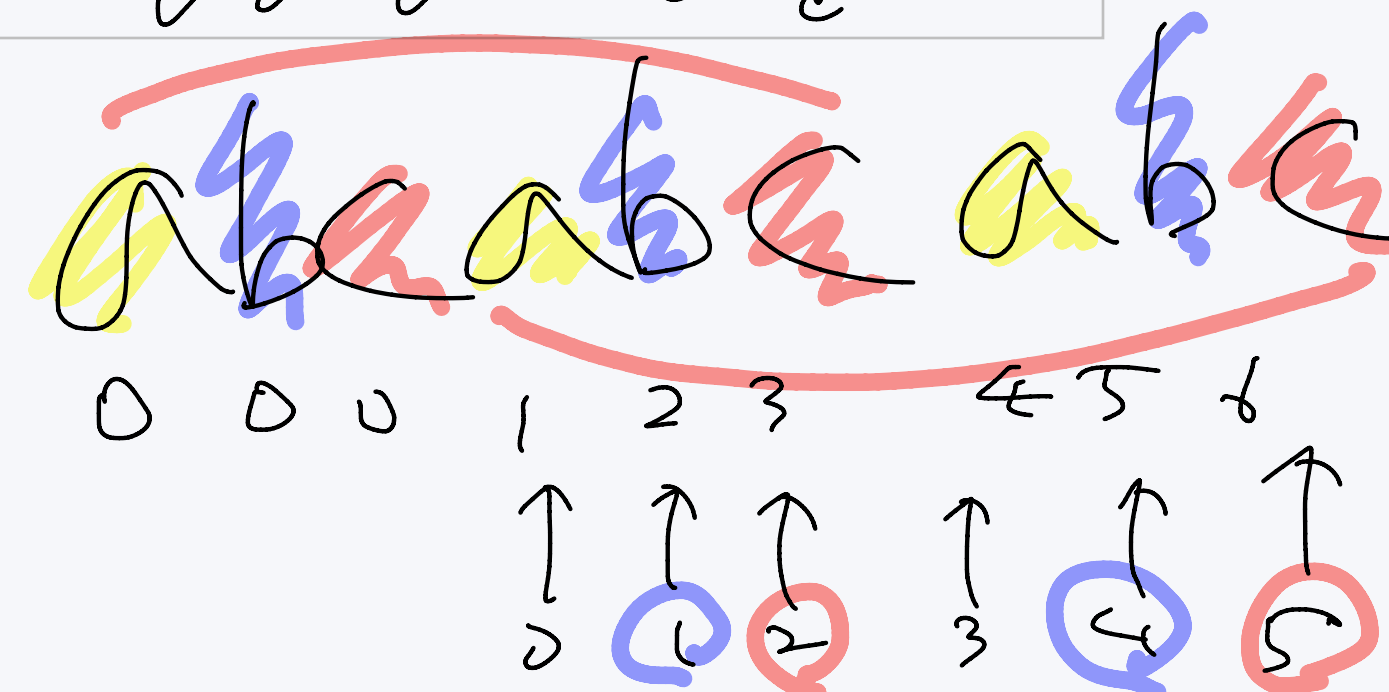
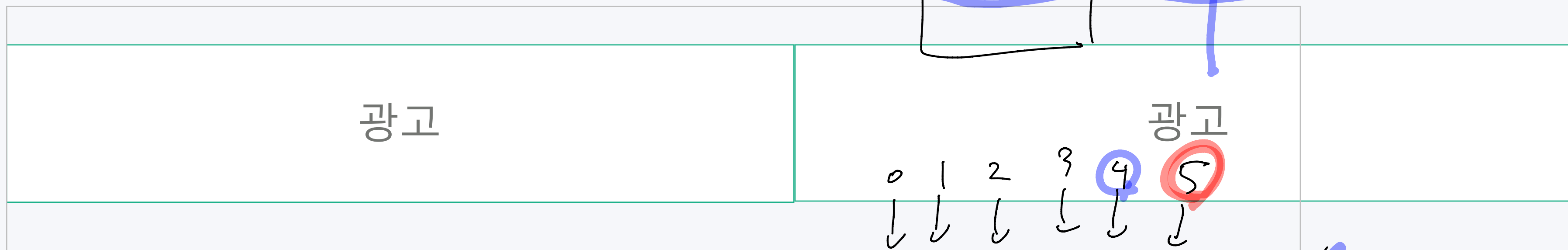


# 광고

<https://www.acmicpc.net/problem/1305>

- 어느 순간 광고 문구가 주어졌을 때
- 가능한 광고의 길이 중 가장 짧은 것 구하기
- 가능한 광고

aab

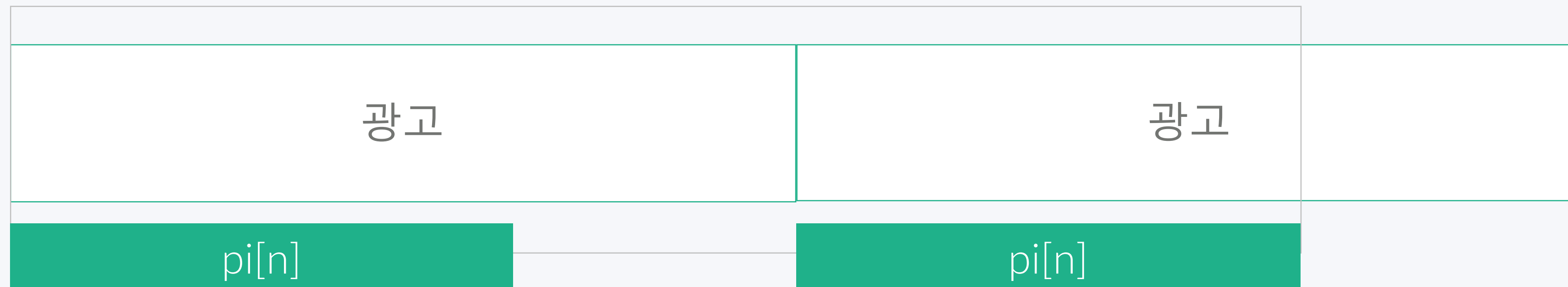


# 광고

42

<https://www.acmicpc.net/problem/1305>

- 어느 순간 광고 문구가 주어졌을 때
- 가능한 광고의 길이 중 가장 짧은 것 구하기
- 가능한 광고



# 광고

3779

$N / (N - P, [N])$

abcabcabcabc

43

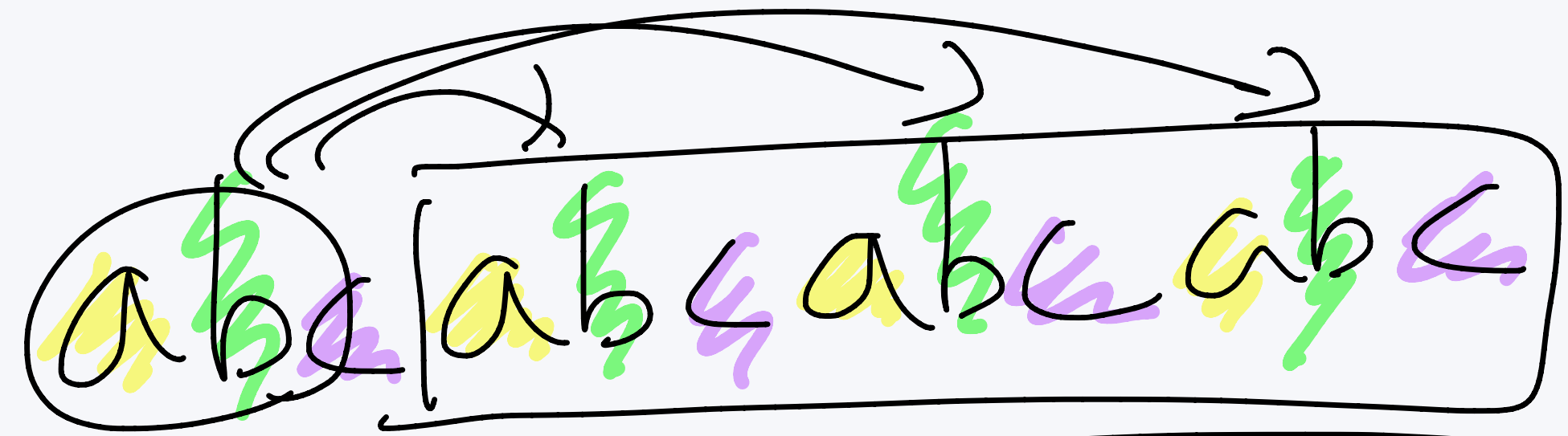
<https://www.acmicpc.net/problem/1305>

- 정답은  $N - pi[N]$  이 된다.

① (abcabcabcabc)

② (abcabc)<sup>2</sup>

③ (abc)<sup>④</sup>

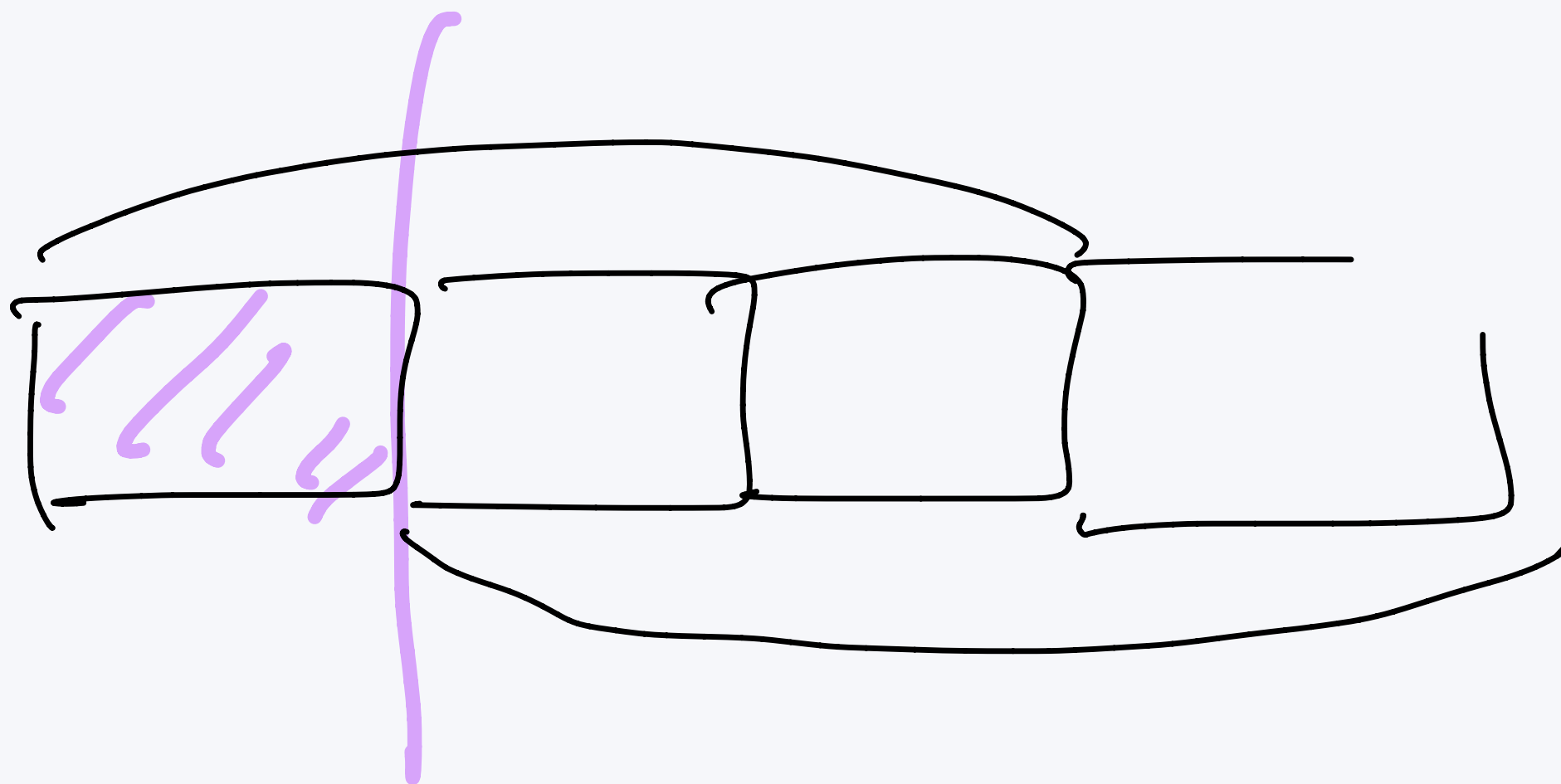


0	1	2	3	4	5	6	7	8	9
a	b	c	a	b	c	a	b	c	a
b	c	a	b	c	a	b	c	a	b
c	a	b	c	a	b	c	a	b	c

(12 - 9)

$(N - P, [N])$

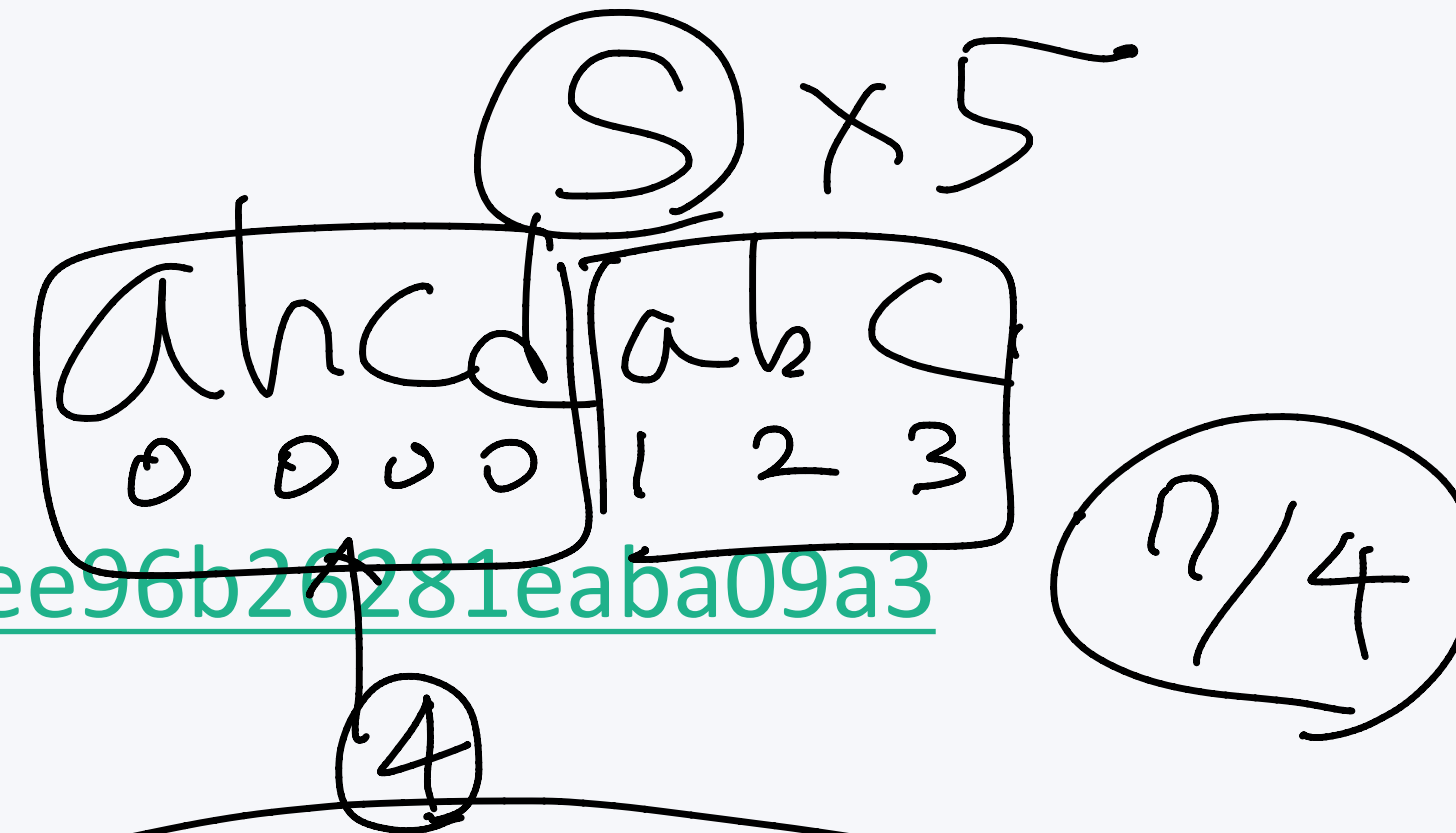
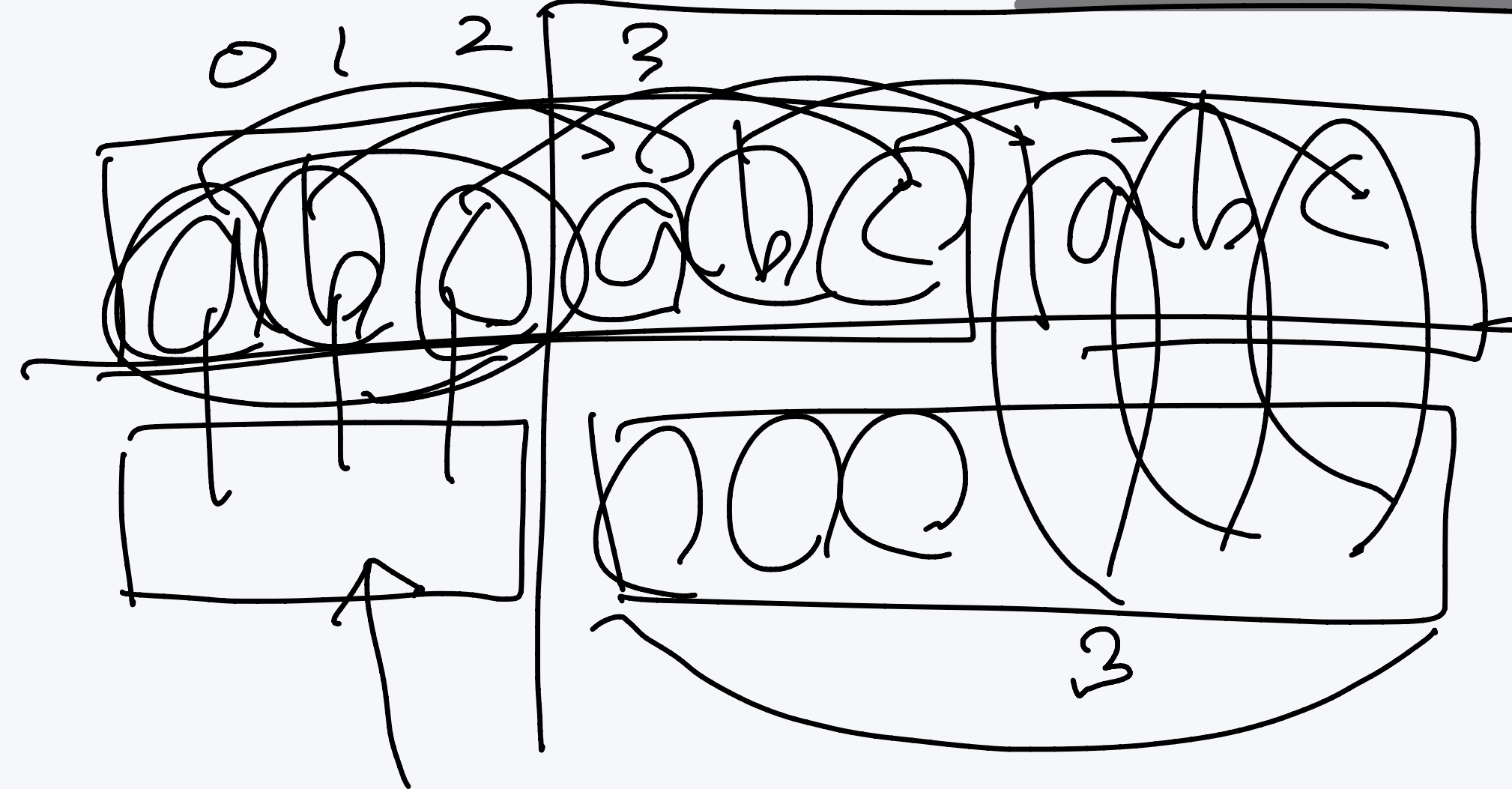
문자의 길이



# 광고

<https://www.acmicpc.net/problem/1305>

- <https://gist.github.com/Baekjoon/14ee96b26281eaba09a3>



$S S S S S$   
 $a b$

$$S = 6$$

44

$$S \times 5 + S$$
$$S + S + S$$

# Cubeditor

<https://www.acmicpc.net/problem/1701>

- 소문자 5,000개 이하로 구성된 문자열 S가 주어진다
- S의 부분 문자열은 연속된 일부분
- 두 번 이상 등장하는 부분 문자열 중에서 가장 긴 것의 길이?

# Cubeditor

46

<https://www.acmicpc.net/problem/1701>

- 모든 부분 문자열은
- 어딘가에서 시작해서 어딘가에서 끝난다

	부분 문자열	
--	--------	--

# Cubeditor

<https://www.acmicpc.net/problem/1701>

- 모든 부분 문자열은
- 어딘가에서 시작해서 어딘가에서 끝난다
- 어떤 suffix의 prefix 이다.
- 원래 문자열에서 KMP를 돌리면, 모든 pi에는
- 가장 처음부터 얼마만큼이 같은지 저장되어 있다

	부분 문자열	
--	--------	--

# Cubeditor

48

<https://www.acmicpc.net/problem/1701>

- 따라서, 모든 부분 문자열  $S[i..N]$ 에 대해서  $p_i$ 를 구하고, 그 중의 최대값을 구하면 된다.

	부분 문자열	
--	--------	--



# Cubeditor

<https://www.acmicpc.net/problem/1701>

- <https://gist.github.com/Baekjoon/aa688a990d1acb00fd8d>