

# 문자열 알고리즘

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 = "ABCABDABCABEABC"  
0 1 2 3 4 5 6
- P = "ABCABE" *p가*
- S[6]부터 P가 나타남!

Ctrl+F

찾을거

# 문자열 매칭 알고리즘

String Matching Algorithm

$$O(|S| |P|)$$

- 모든 경우를 다 해보는 알고리즘
- <https://gist.github.com/Baekjoon/7ad611f0c4dfe6f88017>

$$\boxed{|S|} = \text{S의 길이}$$
$$|P| = \text{P의 길이}$$

$$|S| \times |P|$$

KMP

---

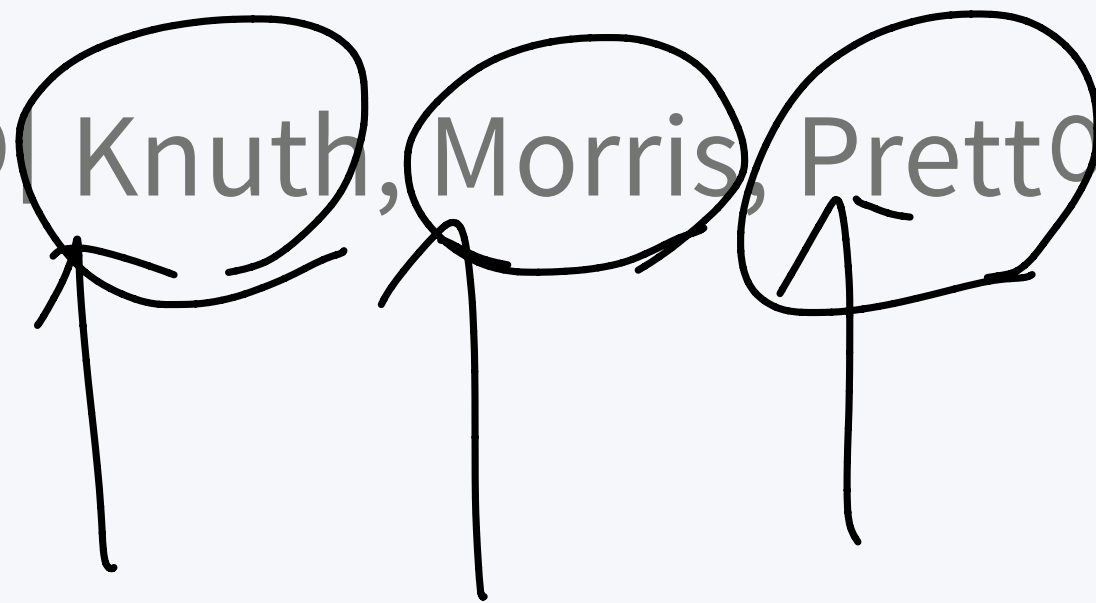
# KMP

## String Matching Algorithm

- KMP는 왜 KMP일까?
- <https://www.acmicpc.net/problem/2902>

KMP

- 만든 사람의 성이 Knuth, Morris, Prett이어서



# KMP

## String Matching Algorithm

- KMP는 pi 배열을 이용해야 한다

• pi[i]

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

# Prefix

## String Matching Algorithm

6

- P = ABCABE

ABCABE

- Prefix

- A

- AB

- ABC

- ABCA

- ABCAB

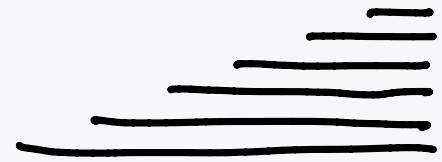
- ABCABE



# Suffix

## String Matching Algorithm

- $P = \text{ABCABE}$



- Suffix

- E

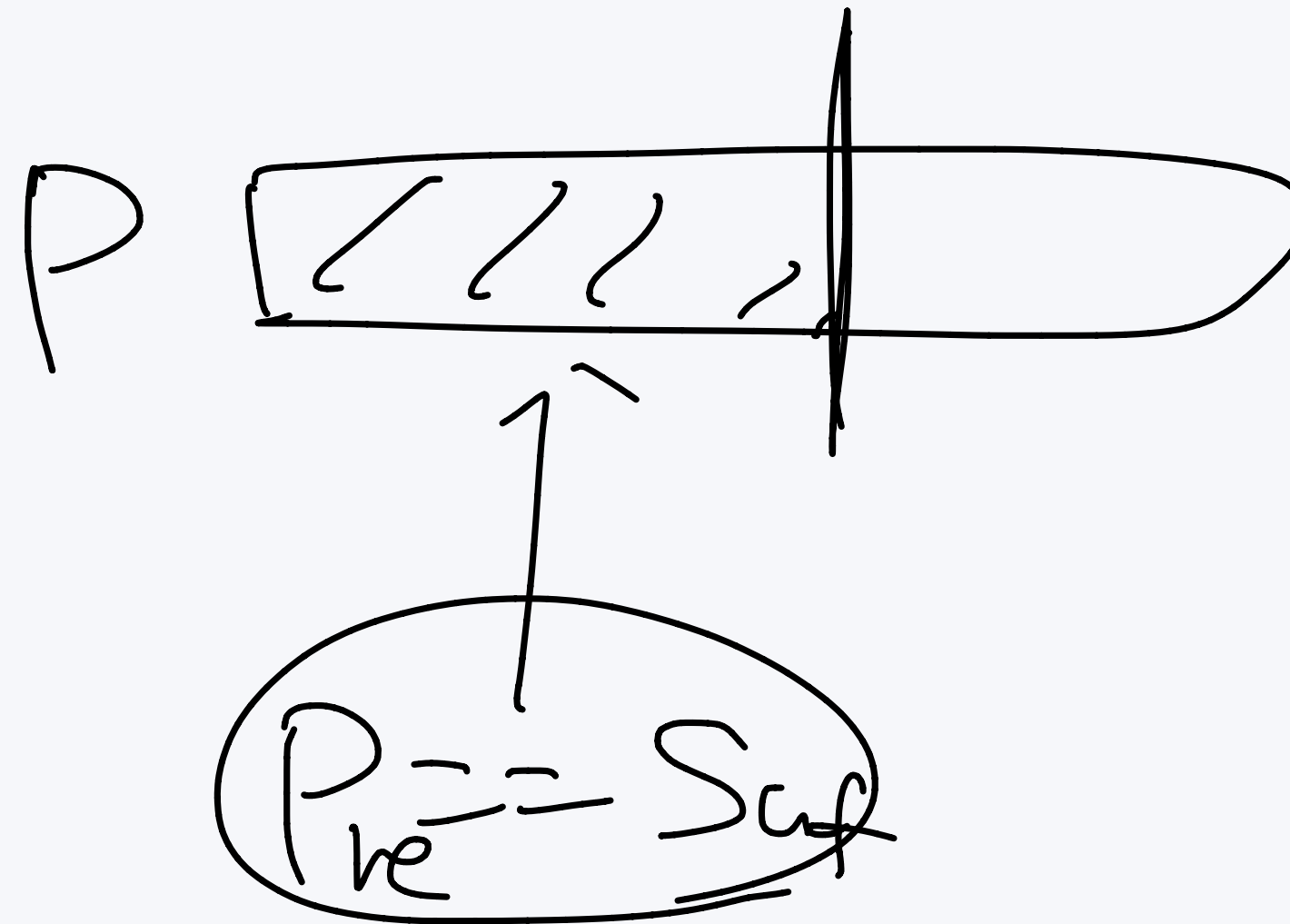
- BE

- ABE

- CABE

- BCABE

- ABCABE



# KMP

## String Matching Algorithm

- ABCABE 의 pi[]

pi[]      7월

i	부분 문자열	pi[i]
0	A	0
1	AB      A, B	0
2	ABC      A, C AB      CA	0
3	ABCA      A, B ABC      CAB	1
4	ABCAB      A, B ABCA      BAB	2
5	ABCABE	0

p

# KMP

String Matching Algorithm

- ABCABDABCABEABC 의 pi

$P$  모든 부분문자열  $|P|$   
 $|P| \times |P|$

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	ABCABD <b>AB</b>	2
8	ABCABD <b>ABC</b>	3
9	ABCABD <b>ABCA</b>	4
10	ABCABD <b>ABCAB</b>	5
11	ABCABDABCABE	0
12	ABCABDABCABE <b>A</b>	1
13	ABCABDABCABE <b>AB</b>	2
14	ABCABDABCABE <b>ABC</b>	3

# KMP

# String Matching Algorithm

[illegible]

# KMP

## String Matching Algorithm

i	0	1	2	3	4	5	6	7	8	9
p[i]	A	B	A	C	A	B	A	B	A	C
pi[i]	0									

A

# KMP

## String Matching Algorithm

i	0	1	2	3	4	5	6	7	8	9
p[i]	A	B	A	C	A	B	A	B	A	C
pi[i]	0	0								

AB

# KMP

## String Matching Algorithm

16

i	0	1	2	3	4	5	6	7	8	9
p[i]	A	B	A	C	A	B	A	B	A	C
pi[i]	0	0	1							

**A**BA**A**

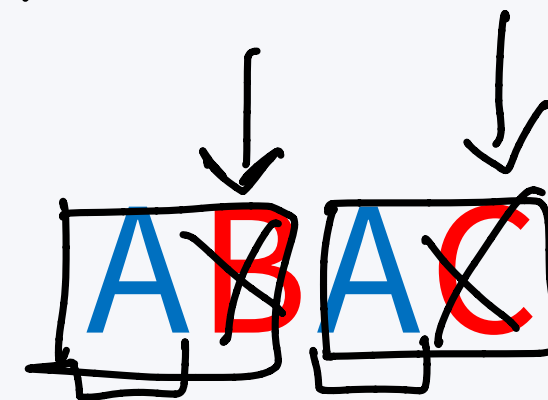


# KMP

## String Matching Algorithm

17

i	0	1	2	3	4	5	6	7	8	9
p[i]	A	B	A	C	A	B	A	B	A	C
pi[i]	0	0	1	0						



# KMP

## String Matching Algorithm

i	0	1	2	3	4	5	6	7	8	9
p[i]	A	B	A	C	A	B	A	B	A	C
pi[i]	0	0	1	0	1					

ABACAA

# KMP

## String Matching Algorithm

19

i	0	1	2	3	4	5	6	7	8	9
p[i]	A	B	A	C	A	B	A	B	A	C
pi[i]	0	0	1	0	1	2				

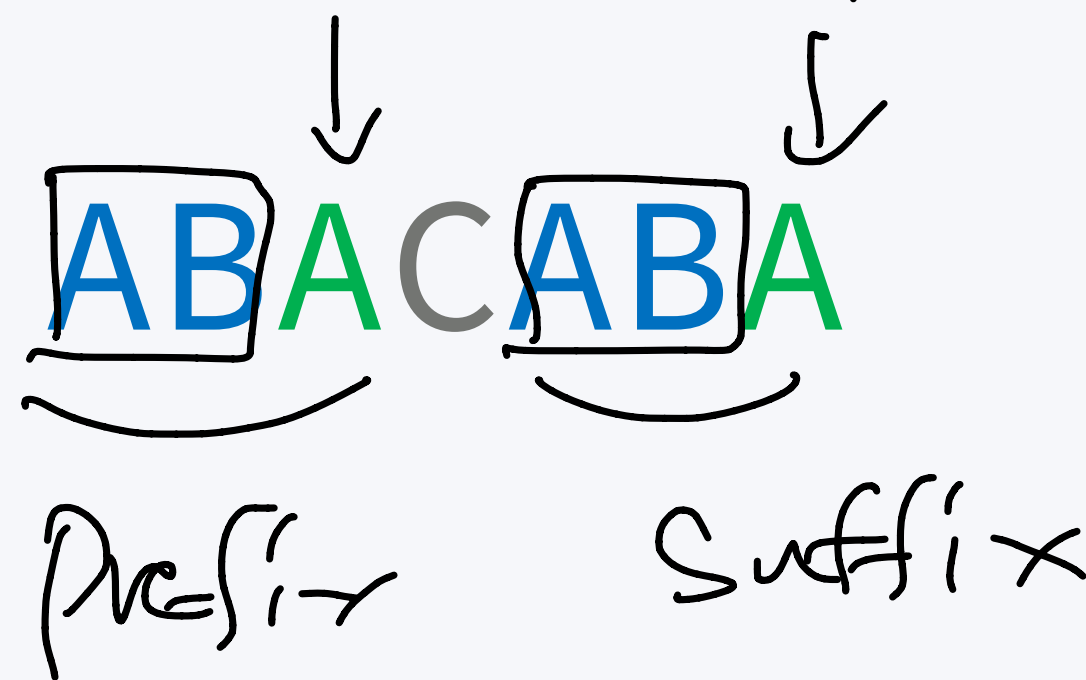
ABACAB

# KMP

## String Matching Algorithm

20

i	0	1	2	3	4	5	6	7	8	9
p[i]	A	B	A	C	A	B	A	B	A	C
pi[i]	0	0	1	0	1	2	3			



# KMP

## String Matching Algorithm

21

i	0	1	2	3	4	5	6	7	8	9
p[i]	A	B	A	C	A	B	A	B	A	C
pi[i]	0	0	1	0	1	2	3	<del>0</del>		

ABACABAB

# KMP

## String Matching Algorithm

22

i	0	1	2	3	4	5	6	7	8	9
p[i]	A	B	A	C	A	B	A	B	A	C
pi[i]	0	0	1							

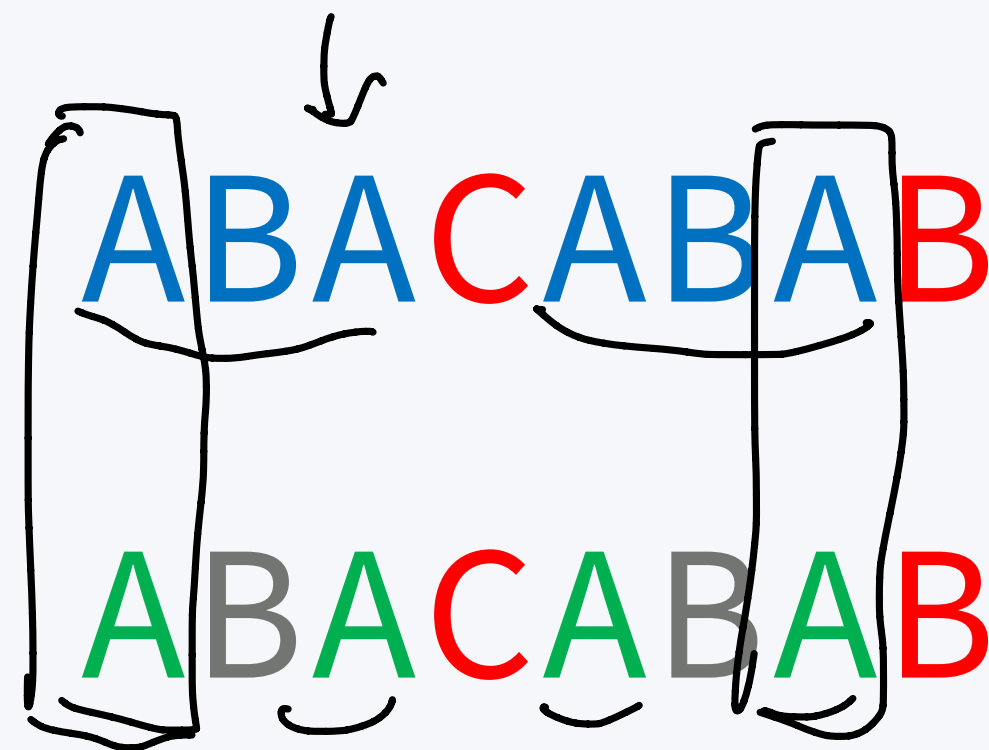
ABA

# KMP

## String Matching Algorithm

23

i	0	1	2	3	4	5	6	7	8	9
p[i]	A	B	A	C	A	B	A	B	A	C
pi[i]	0	0	1	0	1	2	3			



# KMP

## String Matching Algorithm

24

i	0	1	2	3	4	5	6	7	8	9
p[i]	A	B	A	C	A	B	A	B	A	C
pi[i]	0	0	1	0	1	2	3	2		

ABACABAB

ABACABAB

ABACABAB



# KMP

## String Matching Algorithm

25

i	0	1	2	3	4	5	6	7	8	9
p[i]	A	B	A	C	A	B	A	B	A	C
pi[i]	0	0	1	0	1	2	3	2	3	

ABACABABA



# KMP

## String Matching Algorithm

26

i	0	1	2	3	4	5	6	7	8	9
p[i]	A	B	A	C	A	B	A	B	A	C
pi[i]	0	0	1	0	1	2	3	2	3	4

ABACABABAC



OK! PD

# 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] = 2라는 것은
- p[0...1] == p[3...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
- $p[0 \cdots 1] == p[3 \dots 4]$  라는 것을 의미
- ABCABCABE에서 ABCABE를 찾을 때
- ABCAB**C**ABE
- ABCAB****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

- pi[4] = 2라는 것은
- $p[0 \cdots 1] == p[3 \cdots 4]$  라는 것을 의미
- ABCABCABE에서 ABCABE를 찾을 때
- ABCAB**C**ABE
- 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

- 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]

- 패턴 ABCABE의 pi

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

[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 ( $i = 5, j = 2$ ) 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 ABCABDABCABCABEF 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 ABCABDABCABCEEF 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				

A B C

# 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

- 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 ( $i = 14, j = 5$ ) 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  $O(N+M)$

$$|S| = N$$
$$|P| = M$$

# 찾기

52

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

- 문자열 T가 주어졌을 때 P가 몇 번 등장하는지 구하는 문제

# 찾기

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

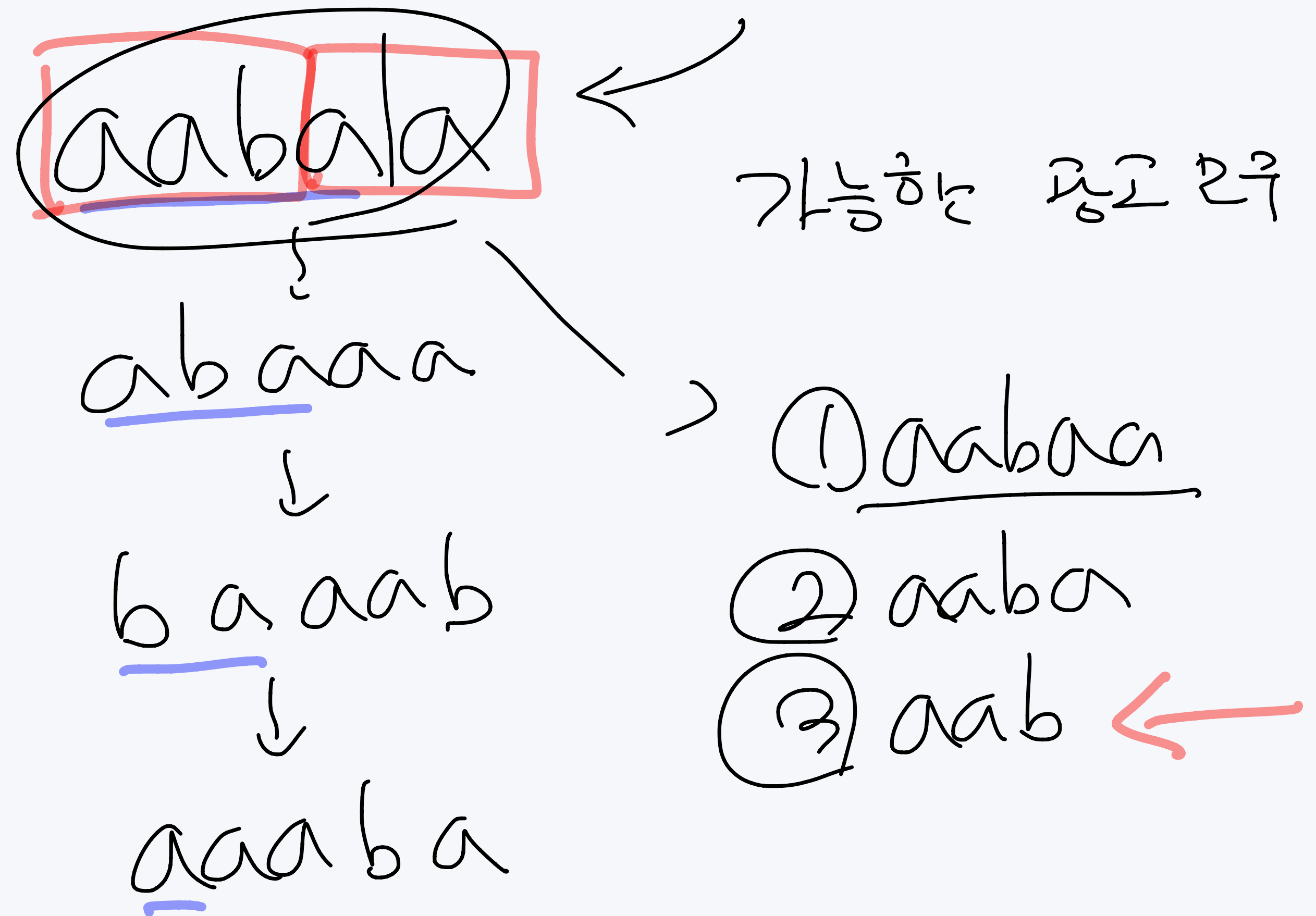
- <https://gist.github.com/Baekjoon/8d3e2a1b93e59879011f>

# 광고

<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 -> ...



# 광고

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

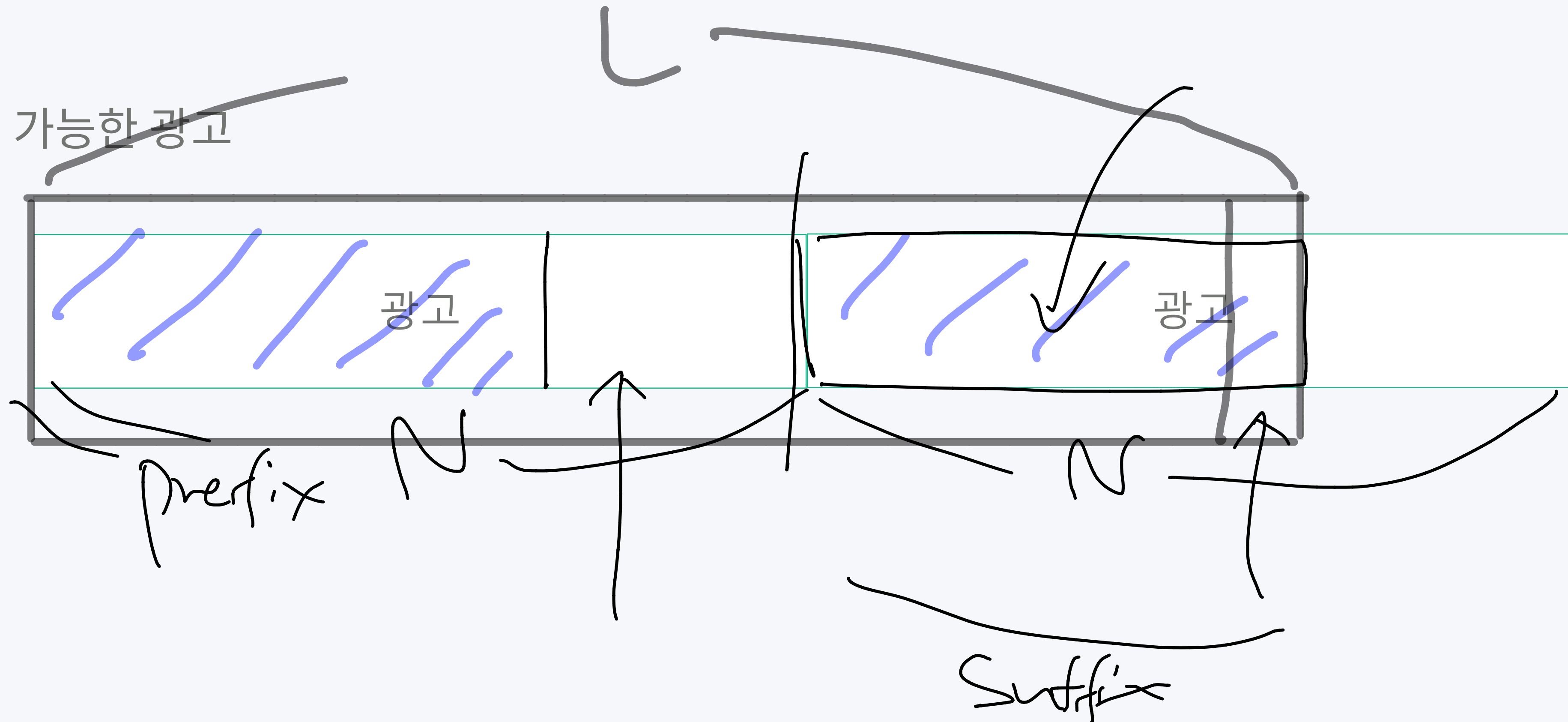
55

$P == S$

가장 긴?

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

- 가능한 광고



# 광고

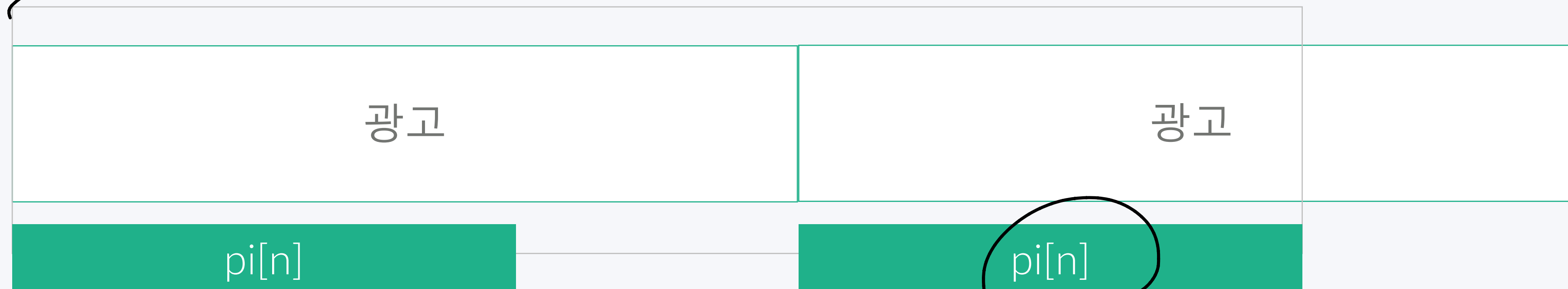
<https://www.acmicpc.net/problem/1305>

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

$N - \text{pi}[N]$

- 가능한 광고

$N$





# 광고

57

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

- 정답은  $N - \text{pi}[N]$  이 된다.

# 광고

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

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

# 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?

↓  
a b c d a b c a b b  
0 0 0 0 1 2 3 1 2 0  
~~~~~  
c d a b c a b b  
0 0 0 0 0 0 0 0  
↑

Prefix == Suffix <

$$O(N) \times N = O(N^2)$$

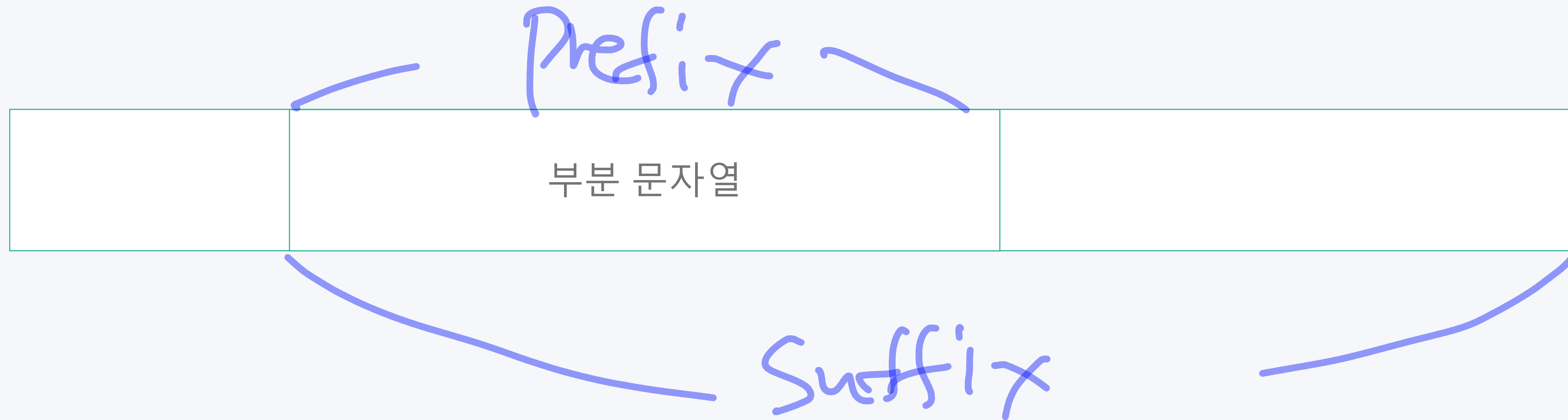
$$5000^2$$

# Cubeditor

60

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

- 모든 부분 문자열은
- 어딘가에서 시작해서 어딘가에서 끝난다



# 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

62

<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>

# Trie

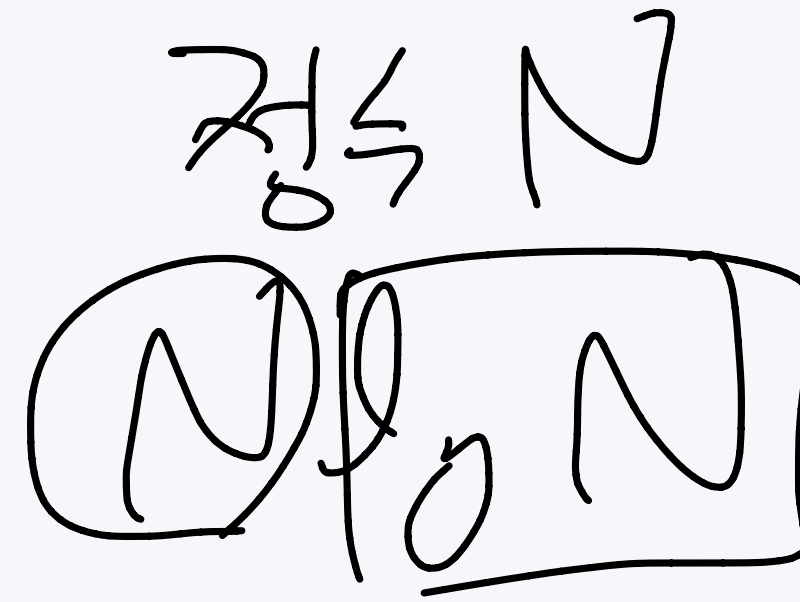
---



# Trie

## Trie

- 숫자 비교는  $O(1)$  이지만
- 문자열 비교는 최대  $O(\text{길이})$  가 걸린다.



길이 -  $\textcircled{M}$  문자열  $N$

$M N \lg N$

- 문자열  $N$ 개를 담고있는 BST에서 검색하는데 걸리는 시간은  $O(\lg N)$ 이 아니고  $O(\text{길이} * \lg N)$ 이다.

정수 비교  $O(1)$

문자열 비교

문자열 비교  $O(\text{길이})$

↓ ↓

# Trie

Trie

$O(2^b)$

66

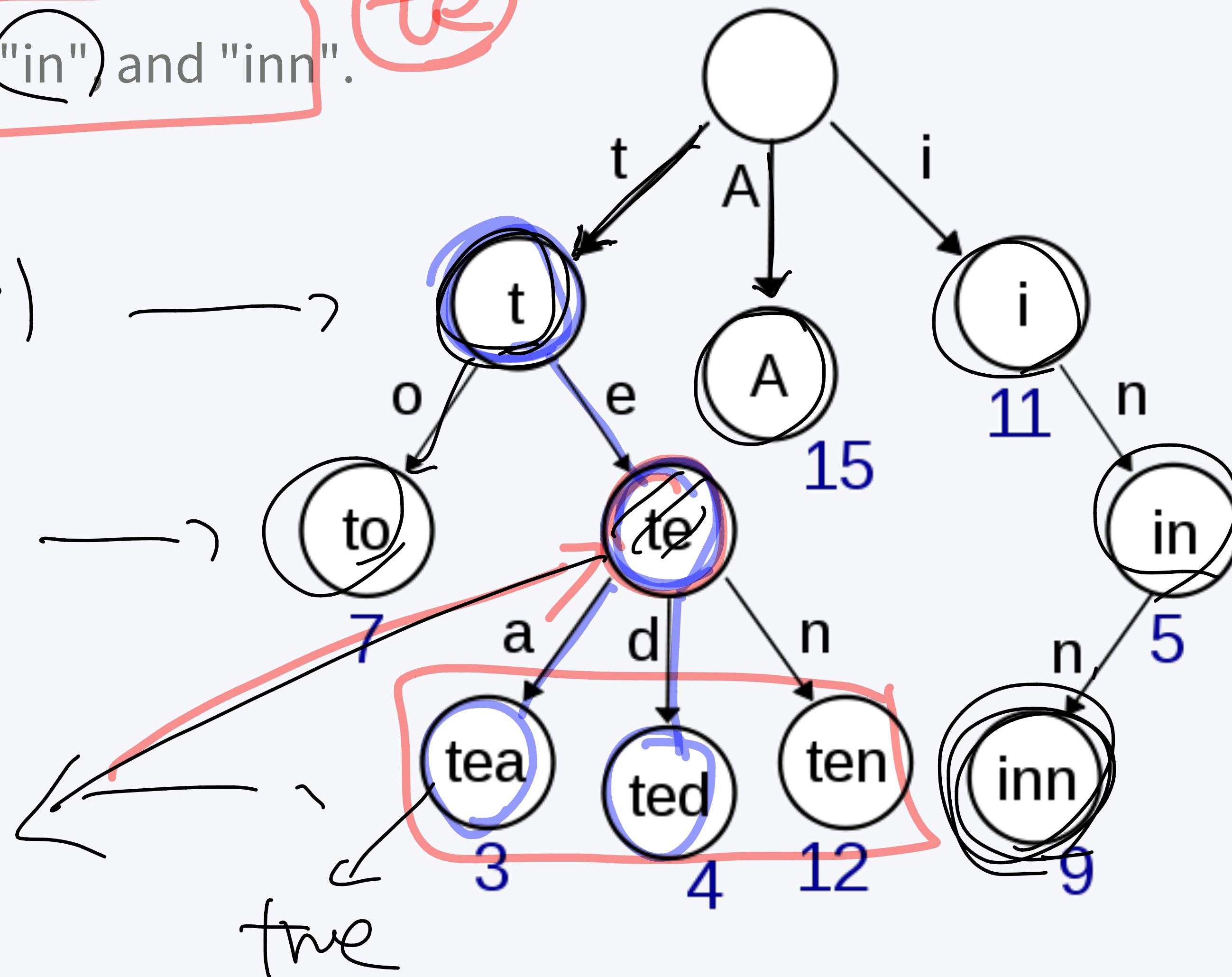
- "A", "to", "tea", "ted", "ten", "i", "in", and "inn".
- 로 만든 Trie

te

depth =  $2^b$

Prefix Tree

valid



# Boggle

<https://www.acmicpc.net/problem/9202>

- 가능한 모든 단어를 미리 다 만들어보고 단어 사전에 들어있는 단어를 모두 찾아보는 방법
- 가능한 단어의 개수: 673108개

# Boggle

<https://www.acmicpc.net/problem/9202>

- <https://gist.github.com/Baekjoon/6101e355b8e861c432c5>
- 8초 정도 걸린다

# Boggle

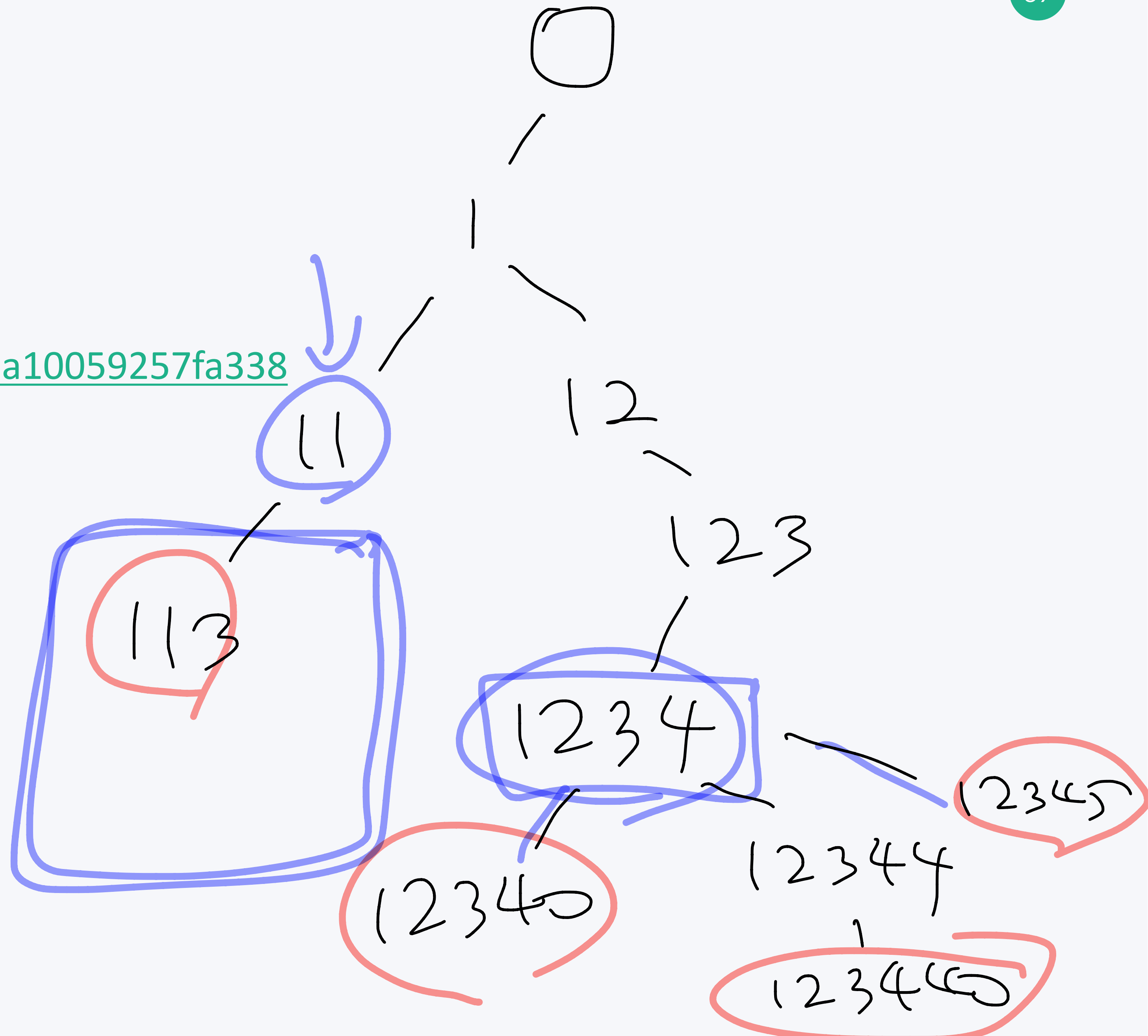
69

<https://www.acmicpc.net/problem/9202>

- Trie 구현!

- <https://gist.github.com/Baekjoon/9f1f0da10059257fa338>

Prefix Tree



# 전화번호 목록

70

<https://www.acmicpc.net/problem/5052>

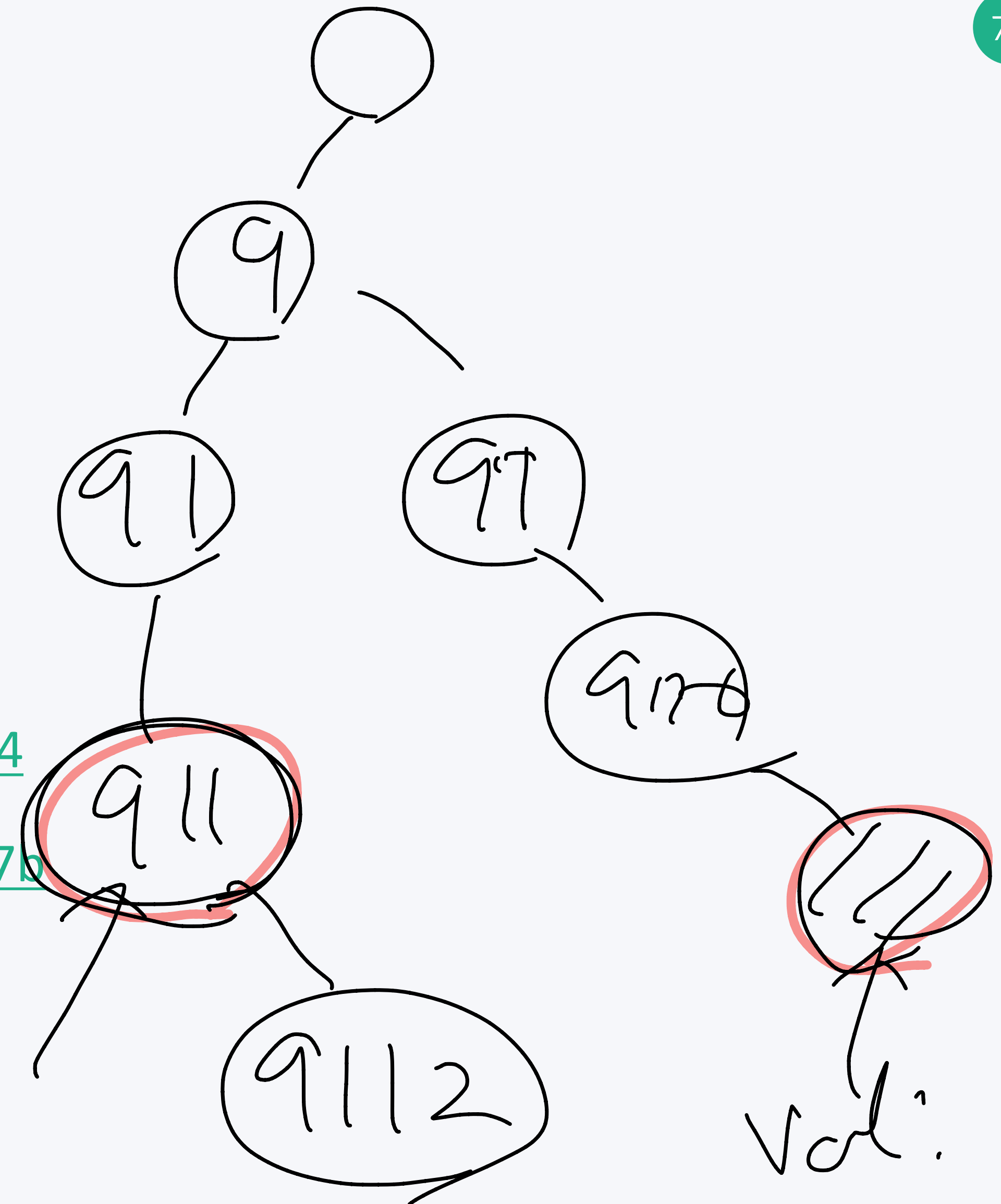
- 한 번호가 다른 번호의 접두어이면 안되다

Valid 한 노드  
자식

- Trie = Prefix Tree

- <https://gist.github.com/Baekjoon/cfd92ab423a2f3d3a894>

- <https://gist.github.com/Baekjoon/925e8e0f691b75506b7b>



# Aho-corasick

---

# Aho-corasick

72

Aho-corasick

- KMP에서의 pi를
- Trie에서도 구현하는 것
- $\text{pi}[\text{node}] = \text{node}$ 가 나타내는 문자열  $s$ 의 suffix이면서 trie에 포함된 가장 긴 문자열



# Aho-corasick

Aho-corasick

- $S = \text{"ABCDEABABABABCD"}$
- 에서
- "ABAB", "AD", "ABC", "BCD", "ABABC"가 몇 개 인지 찾는 문제
- KMP를 총 5번 돌린다?

$S$  124,  $P$  124, KMP  
 $S$  124,  $P$  124, KMP

$$(|S| + |P|)M.$$

Aho corasick

# Aho-corasick

Aho-corasick

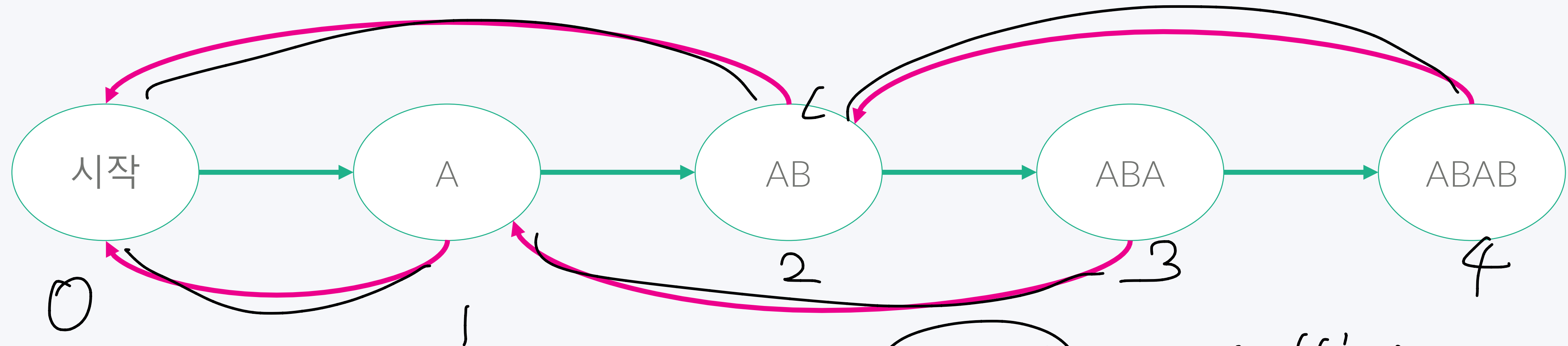
- $S = \text{"ABCDEABABABABCD"}$
- 에서
- “ABAB”, “AD”, “ABC”, “BCD”, “ABABC”가 몇 개 인지 찾는 문제
- KMP를 총 5번 돌린다?
- ABAB: 0 0 1 2
- AD: 0 0
- ABC: 0 0 0
- BCD: 0 0 0
- ABABC: 0 0 1 2 0

# Aho-corasick

Aho-corasick

75

ABAB  
0012



Prefix == Suffix

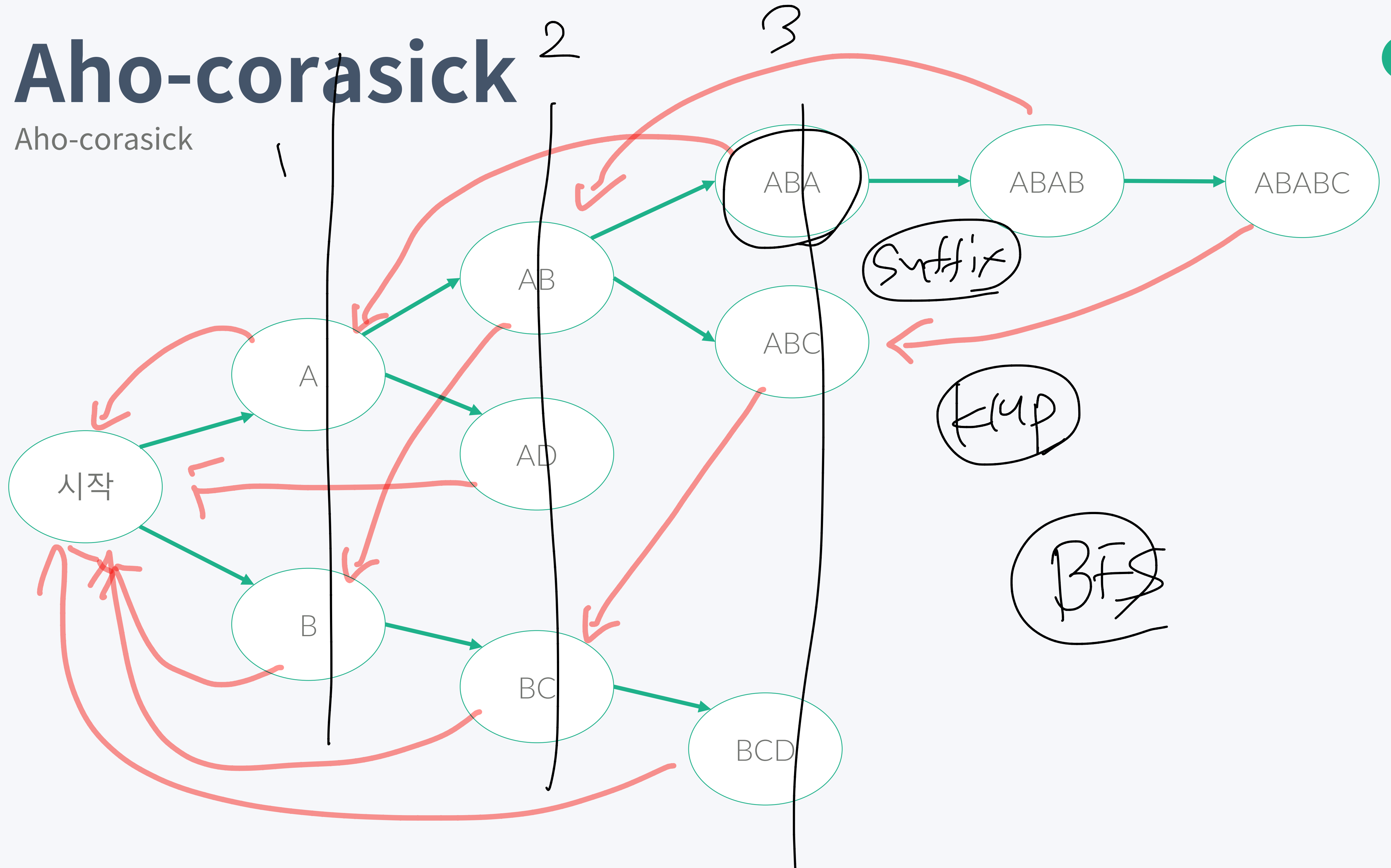
가장 긴 접미어

이러한 접미어 Node

# Aho-corasick

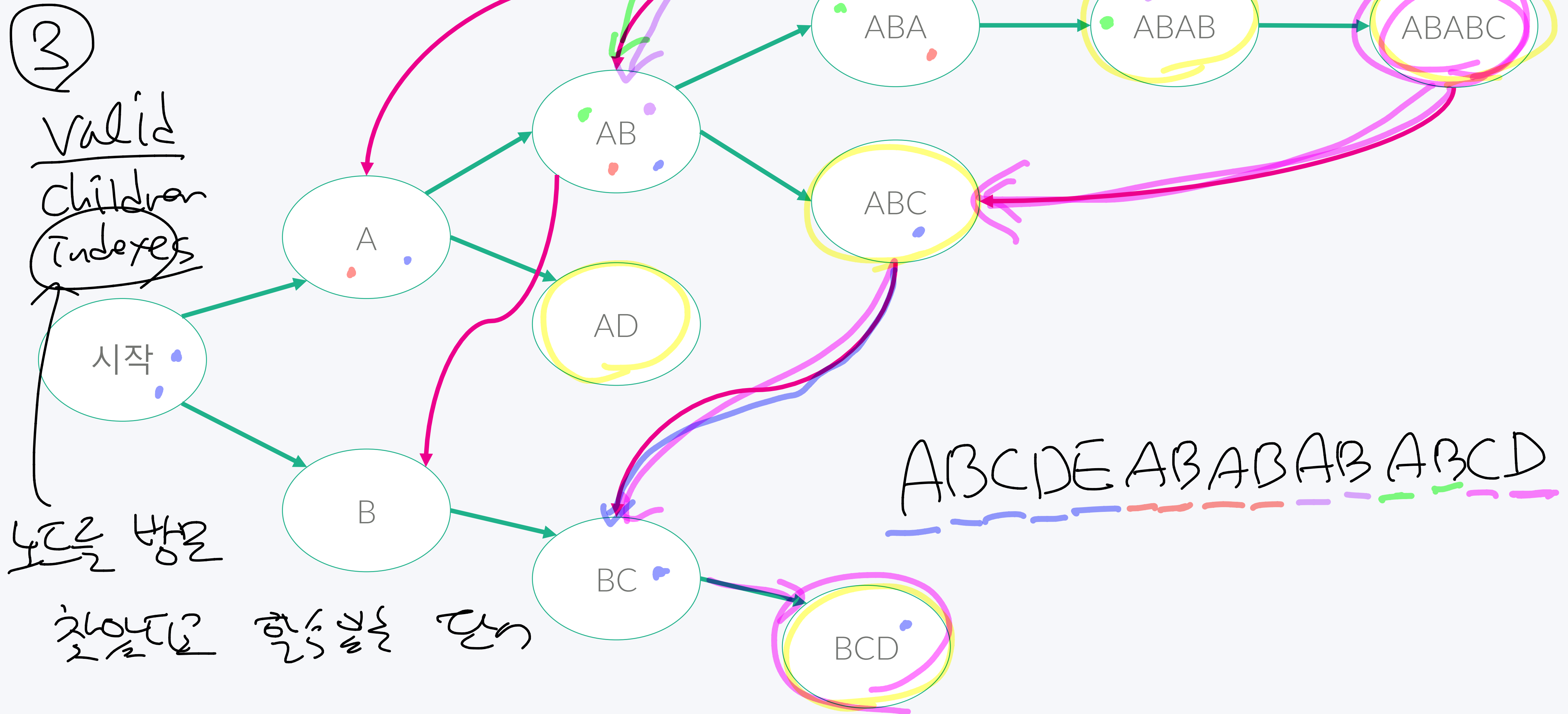
Aho-corasick

76



# Aho-corasick

Aho-corasick



# Aho-corasick

Aho-corasick

- <https://gist.github.com/Baekjoon/6561c788d5b8feee7f26>
- <https://gist.github.com/Baekjoon/d3ffd37864ec4616a45a>

# 문자열 집합 판별

<https://www.acmicpc.net/problem/9250>

- <https://gist.github.com/Baekjoon/e45c2335848ca781a8dd>

# 돌연변이

<https://www.acmicpc.net/problem/10256>

- <https://gist.github.com/Baekjoon/e2aab198a5990b55553a>



# Suffix Array

---

# Prefix

## Prefix

- baekjoon의 Prefix

1. (b)

2. (ba)

3. (bae)

4. (baek)

5. baekj

6. baekjo

7. baekjoo

8. baekjoon

# Suffix

Suffix

- baekjoon의 Suffix

- 1. baekjoon
- 2. aekjoon
- 3. ekjoon
- 4. kjoon
- 5. joon
- 6. oon
- 7. on
- 8. n

백준  
012345  
baekjoon  
54 : on

# 접미사 배열

Suffix Array

84

- 문자열 S의 모든 접미사를 구해서 사전순으로 정렬한 자료구조

Suffix

# 접미사 배열

Suffix Array

- aekjoon
- baekjoon
- ekjoon
- joon
- kjoon
- n
- on
- oon

# 접미사

Suffix

- 접미사는 정수로 나타낼 수 있다.
- $i$ 번 접미사:  $i$ 번째 글자에서 시작하는 접미사

# 접미사 배열

<https://www.acmicpc.net/problem/11656>

- 문자열 S의 모든 접미사를 구하는 문제

# 접미사 배열

<https://www.acmicpc.net/problem/11656>

- 정렬할 때, 매번 부분 문자열을 만들지 않기 위해 strcmp를 사용했다

```
vector<int> a(n);  
for (int i=0; i<n; i++) {  
    a[i] = i;  
}  
sort(a.begin(), a.end(), [&s](int u, int v) {  
    return strcmp(s.c_str()+u, s.c_str()+v) < 0;  
});
```

$$N^2 \lg N$$

$$N \lg^2 N$$

$$N \lg N$$

$$\text{정렬: } (N \lg N) \times \underline{N}$$



# 접미사 배열

<https://www.acmicpc.net/problem/11656>

- C++: <https://gist.github.com/Baekjoon/34ecd3cd0bd1a59d205c6a16f73ff634>
- C++11: <https://gist.github.com/Baekjoon/931ed853aebbdb8afed0c66dfe84b4ff>

# 접미사 배열

<https://www.acmicpc.net/problem/11656>

- 길이가  $N$ 인 문자열  $S$ 의 접미사는 총  $N$ 개가 있다.
- 따라서,  $N$ 개를 정렬하는데 걸리는 시간은  $O(N \lg N)$ 이다?

# 접미사 배열

<https://www.acmicpc.net/problem/11656>

- 길이가  $N$ 인 문자열  $S$ 의 접미사는 총  $N$ 개가 있다.
- 따라서,  $N$ 개를 정렬하는데 걸리는 시간은  $O(N \lg N)$ 이다?
- 아니다. 문자열을 비교하는데 걸리는 시간이  $O(N)$ 이기 때문에,  $O(N^2 \lg N)$ 이다.

# 접미사 배열 2

<https://www.acmicpc.net/problem/13013>

92

- 접미사 배열 SA가 주어졌을 때, 그러한 접미사 배열을 만드는 문자열 S를 만드는 문제
- 이 때, S에 포함된 서로 다른 문자의 개수가 최소가 되어야 한다

aaba



3: a  
0: abca  
1: bca  
2: Ca

3: a  
0: aaba  
1: aba  
2: ba

aaba

abca

=> [3, 0, 1, 2]

bcd a

3: a  
0: b c d a  
1: c d a  
2: d a

# 접미사 배열 2

<https://www.acmicpc.net/problem/13013>

- 접미사 배열 SA가 주어졌을 때, 그러한 접미사 배열을 만드는 문자열 S를 아무거나 만들어 보자

# 접미사 배열 2

<https://www.acmicpc.net/problem/13013>

- $SA[i]$ 번째에서 시작하는 접미사는  $SA[i+1]$ 번째에 시작하는 접미사보다 사전순으로 앞서야 한다
- $SA[i]$ 번째 접미사와  $SA[i+1]$ 번째 접미사를 비교한다고 하면
- 먼저,  $S[SA[i]]$ 와  $S[SA[i+1]]$ 을 비교해야 한다
- $S[SA[i]] > S[SA[i+1]]$  은 절대로 일어날 수 없다

# 접미사 배열 2

<https://www.acmicpc.net/problem/13013>

- 각각의  $i$ 에 대해서,  $S[SA[i]] < S[SA[i+1]]$ 가 되게  $S$ 를 만들 수 있다.

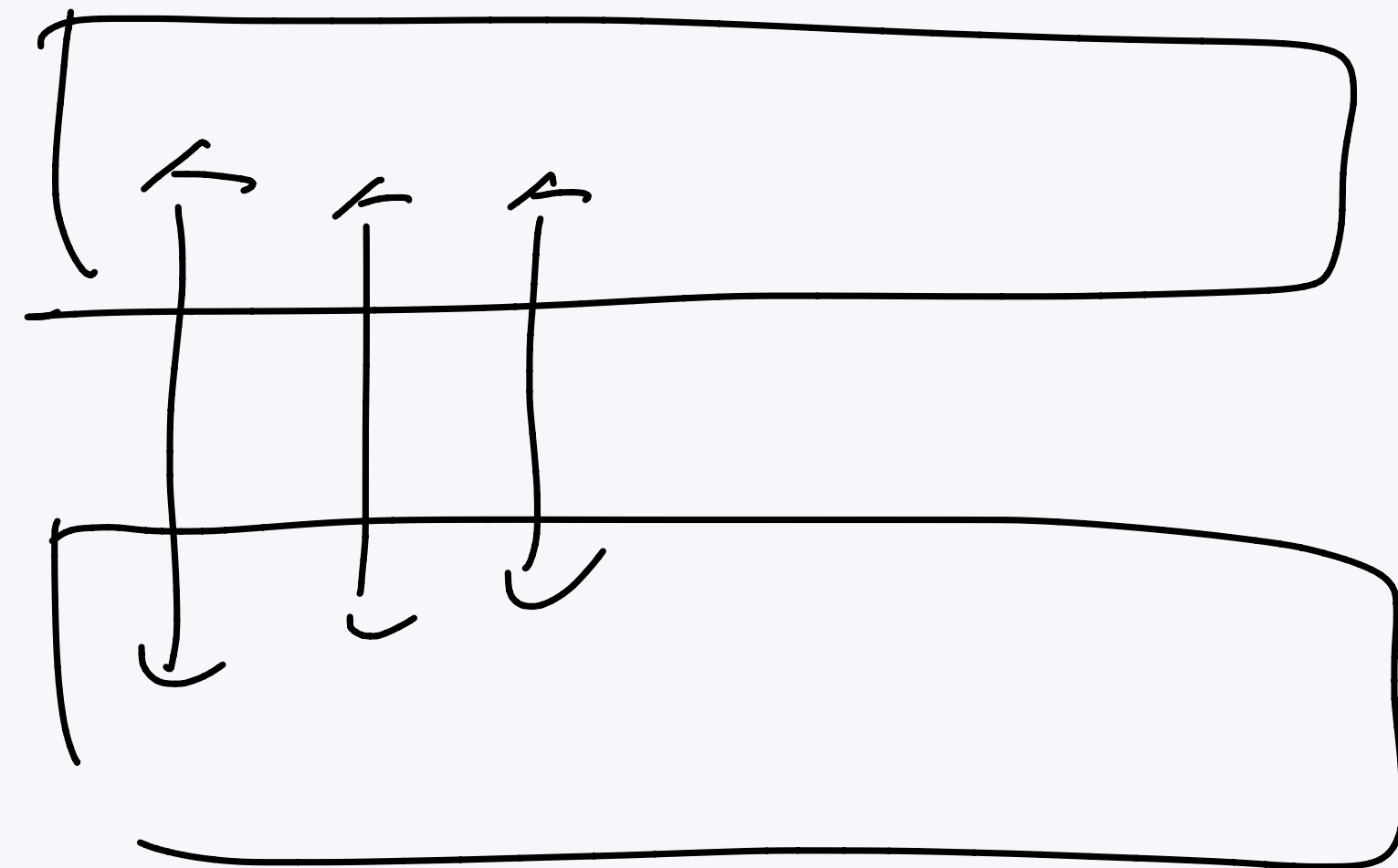
- $SA = (2, 3, 1, 0)$  인 경우에
- $SA[0] = 2$  이기 때문에,  $S[2] = a$ 를 넣고
- $SA[1] = 3$  이기 때문에,  $S[3] = b$ 를 넣고
- $SA[2] = 1$  이기 때문에,  $S[1] = c$ 를 넣고
- $SA[3] = 0$  이기 때문에,  $S[0] = d$ 를 넣는다.

$S$ 의 접미사의 접미사도  
 $S$ 의 접미사

# 접미사 배열 2

<https://www.acmicpc.net/problem/13013>

- $S[SA[i]] \leq S[SA[i+1]]$  인 경우에 문제를 풀어본다
- $SA[0]$ 은 아무 제약 조건이 없기 때문에, 그냥  $S[SA[0]] = a$ 를 넣는다
- $SA[1]$ 은 두 가지 경우가 가능하다
- $S[SA[0]] < S[SA[1]]$  또는  $S[SA[0]] == S[SA[1]]$





# 접미사 배열 2

<https://www.acmicpc.net/problem/13013>

- 두 문자열의 비교는 앞에서 부터 한 글자씩 비교를 한다

# 접미사 배열 2

<https://www.acmicpc.net/problem/13013>

- 두 접미사의 비교는 문자 하나와 접미사 비교로 바꿀 수 있다

i번 접미사:

|      |        |  |  |     |
|------|--------|--|--|-----|
| S[i] | S[i+1] |  |  | ... |
|------|--------|--|--|-----|

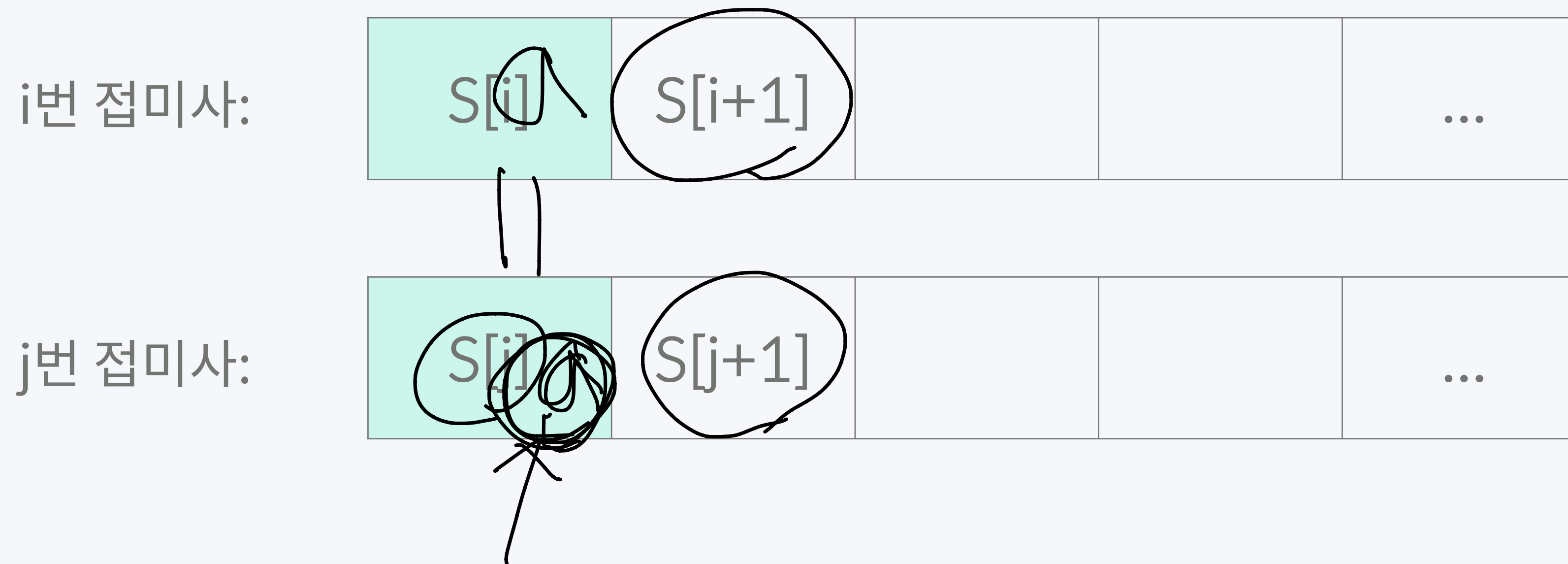
j번 접미사:

|      |        |  |  |     |
|------|--------|--|--|-----|
| S[j] | S[j+1] |  |  | ... |
|------|--------|--|--|-----|

# 접미사 배열 2

<https://www.acmicpc.net/problem/13013>

- 두 접미사의 비교는 문자 하나와 접미사 비교로 바꿀 수 있다
- $S[i]$ 와  $S[j]$ 를 비교한다

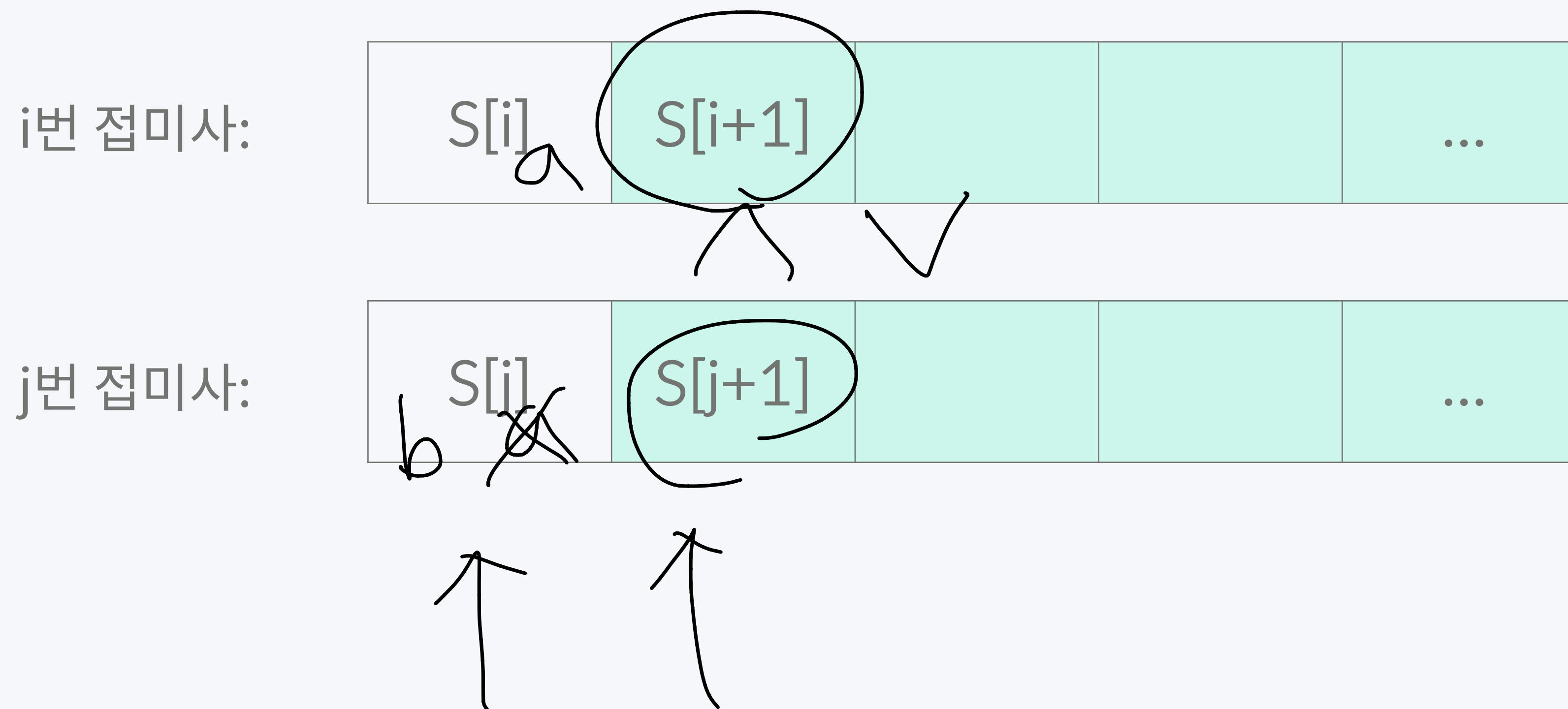


# 접미사 배열 2

100

<https://www.acmicpc.net/problem/13013>

- 두 접미사의 비교는 문자 하나와 접미사 비교로 바꿀 수 있다
- $i+1$ 번 접미사와  $j+1$ 번 접미사



# 접미사 배열 2

<https://www.acmicpc.net/problem/13013>

- 두 접미사의 비교는 문자 하나와 접미사 비교로 바꿀 수 있다
- $i+1$ 번 접미사와  $j+1$ 번 접미사

$i+1$ 번 접미사:

|          |  |  |     |
|----------|--|--|-----|
| $S[i+1]$ |  |  | ... |
|----------|--|--|-----|

$j+1$ 번 접미사:

|          |  |  |     |
|----------|--|--|-----|
| $S[j+1]$ |  |  | ... |
|----------|--|--|-----|

# 접미사 배열 2

102

<https://www.acmicpc.net/problem/13013>

- 각각의  $i$ 에 대해서
- $S[SA[i]] < S[SA[i+1]]$  이면 새로운 문자를 추가해야 하는 것 이기 때문에
- 되도록
- $S[SA[i]] = S[SA[i+1]]$  을 많이 사용하는 것이 좋다.

# 접미사 배열 2

<https://www.acmicpc.net/problem/13013>

- 각각의  $i$ 에 대해서
- $S[SA[i]] < S[SA[i+1]]$  을 최소로 하는 것은
- $S[SA[i+1]] = S[SA[i]] + 1$  을 최소로 하는 것과 같기 때문에
- 사전 순으로 앞서는 문자열을 만들 수 있게 된다

# 접미사 배열 2

<https://www.acmicpc.net/problem/13013>

- $S[SA[i]] == S[SA[i+1]]$  인 경우
- $i$ 번째 접미사와  $i+1$ 번째 접미사를 비교하는 것은
- $i$ 번째 문자와  $i+1$ 번째 글자를 비교하고 같은 경우에는
- $i+1$ 번째 접미사와  $i+2$ 번째 접미사를 비교하는 것과 같다



# 접미사 배열 2

105

<https://www.acmicpc.net/problem/13013>

- $S[SA[i]] == S[SA[i+1]]$  인 경우
- $i$ 번째 접미사와  $i+1$ 번째 접미사를 비교하는 것은
- $i$ 번째 문자와  $i+1$ 번째 글자를 비교하고 같은 경우에는
- $i+1$ 번째 접미사와  $i+2$ 번째 접미사를 비교하는 것과 같다
- 지금  $S[SA[i]] == S[SA[i+1]]$  인 경우라는 것은  $i$ 번째 문자와  $i+1$ 번째 글자가 같다는 것이기 때문에
- $i+1$ 번째 접미사와  $i+2$ 번째 접미사를 비교해야 한다

# 접미사 배열 2

<https://www.acmicpc.net/problem/13013>

- 그런데
- 우리는 이미 접미사 배열을 가지고 있다
- 즉, 접미사 배열에서  $i+1$ 이  $i+2$ 보다 앞에 있으면 된다

# 접미사 배열 2

107

<https://www.acmicpc.net/problem/13013>

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

접미사 배열:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 4 | 8 | 6 | 1 | 5 | 2 | 9 | 3 | 0 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|  |  |  |  |  |  |  |   |  |  |
|--|--|--|--|--|--|--|---|--|--|
|  |  |  |  |  |  |  | a |  |  |
|--|--|--|--|--|--|--|---|--|--|

- 7번 접미사가 제일 앞서야 한다.
- S의 7번째 글자는 a라고 할 수 있다

# 접미사 배열 2

108

<https://www.acmicpc.net/problem/13013>

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

접미사 배열:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 4 | 8 | 6 | 1 | 5 | 2 | 9 | 3 | 0 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|  |  |  |  |   |  |  |   |  |  |
|--|--|--|--|---|--|--|---|--|--|
|  |  |  |  | a |  |  | a |  |  |
|--|--|--|--|---|--|--|---|--|--|

- 7번 접미사 < 4번 접미사
- 4번째 글자를 a라고 하면 8번 접미사와 5번 접미사를 비교해야 한다
- 8번 접미사의 위치: 2, 5번 접미사의 위치: 5
- 4번째 글자가 a여도 7번 접미사 < 4번 접미사이다

7번: a [ ]  
4번: @ [ ]  
8번  
5번

# 접미사 배열 2

109

<https://www.acmicpc.net/problem/13013>

|         |   |   |   |   |   |   |   |   |   |   |
|---------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:    | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 접미사 배열: | 7 | 4 | 8 | 6 | 1 | 5 | 2 | 9 | 3 | 0 |
| 문자열 S:  |   |   |   |   | a |   |   | a | a |   |

- 4번 접미사 < 8번 접미사
- 8번째 글자를 a라고 하면 5번 접미사와 9번 접미사를 비교해야 한다
- 5번 접미사의 위치: 5, 9번 접미사의 위치: 7
- 8번째 글자가 a여도 4번 접미사 < 8번 접미사이다

4번: [a] [5]  
8번: [a] [9]

# 접미사 배열 2

110

<https://www.acmicpc.net/problem/13013>

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

접미사 배열:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 4 | 8 | 6 | 1 | 5 | 2 | 9 | 3 | 0 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|  |  |  |  |   |  |   |   |   |  |
|--|--|--|--|---|--|---|---|---|--|
|  |  |  |  | a |  | b | a | a |  |
|--|--|--|--|---|--|---|---|---|--|

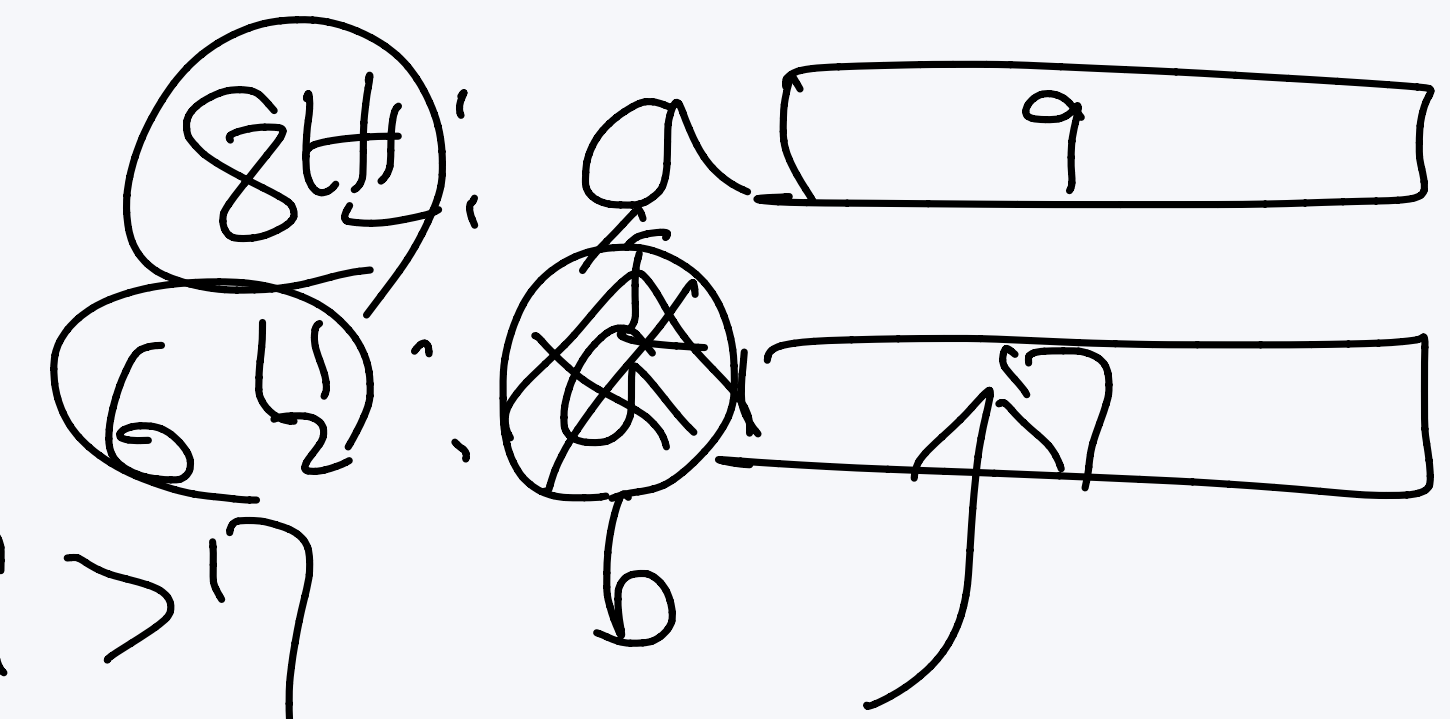
8번 접미사 < 6번 접미사

8번 > 6번

• 6번째 글자를 a라고 하면 9번 접미사와 7번 접미사를 비교해야 한다

• 9번 접미사의 위치: 7, 7번 접미사의 위치: 0

• 6번째 글자가 a이면 8번 접미사 < 6번 접미사가 될 수 없다. 따라서, 6번째 글자는 b이다



# 접미사 배열 2

111

<https://www.acmicpc.net/problem/13013>

|         |   |   |   |   |   |   |   |   |   |   |
|---------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:    | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 접미사 배열: | 7 | 4 | 8 | 6 | 1 | 5 | 2 | 9 | 3 | 0 |
| 문자열 S:  |   | b |   |   | a |   | b | a | a |   |

- 6번 접미사 < 1번 접미사
- 1번째 글자를 b라고 하면 7번 접미사와 2번 접미사를 비교해야 한다
- 7번 접미사의 위치: 0, 2번 접미사의 위치: 6
- 1번째 글자가 b여도 6번 접미사 < 1번 접미사이다

64 : b 7  
142 : b 2  
64 < 142

# 접미사 배열 2

112

<https://www.acmicpc.net/problem/13013>

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

접미사 배열:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 4 | 8 | 6 | 1 | 5 | 2 | 9 | 3 | 0 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|  |   |  |  |   |   |   |   |   |  |
|--|---|--|--|---|---|---|---|---|--|
|  | b |  |  | a | c | b | a | a |  |
|--|---|--|--|---|---|---|---|---|--|

- 1번 접미사 < 5번 접미사
- 5번째 글자를 b라고 하면 2번 접미사와 6번 접미사를 비교해야 한다
- 2번 접미사의 위치: 6, 6번 접미사의 위치: 3
- 5번째 글자가 b이면 1번 접미사 < 5번 접미사가 될 수 없다. 따라서, 5번째 글자는 c이다





# 접미사 배열 2

113

<https://www.acmicpc.net/problem/13013>

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

접미사 배열:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 4 | 8 | 6 | 1 | 5 | 2 | 9 | 3 | 0 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|  |   |   |  |   |   |   |   |   |  |
|--|---|---|--|---|---|---|---|---|--|
|  | b | c |  | a | c | b | a | a |  |
|--|---|---|--|---|---|---|---|---|--|

- 5번 접미사 < 2번 접미사
- 2번째 글자를 c라고 하면 6번 접미사와 3번 접미사를 비교해야 한다
- 6번 접미사의 위치: 3, 3번 접미사의 위치: 8
- 2번째 글자가 c여도 5번 접미사 < 2번 접미사이다

# 접미사 배열 2

<https://www.acmicpc.net/problem/13013>

114

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

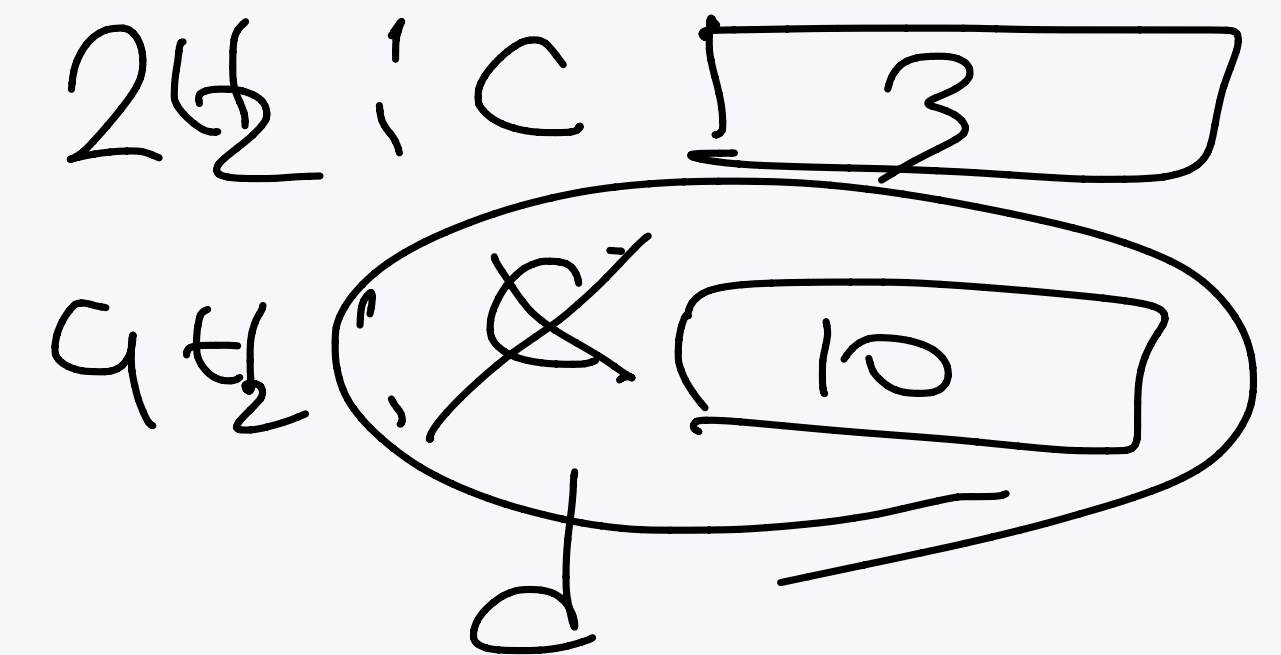
접미사 배열:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 4 | 8 | 6 | 1 | 5 | 2 | 9 | 3 | 0 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|  |   |   |  |   |   |   |   |   |   |
|--|---|---|--|---|---|---|---|---|---|
|  | b | c |  | a | c | b | a | a | d |
|--|---|---|--|---|---|---|---|---|---|

- 2번 접미사 < 9번 접미사
- 9번째 글자를 c라고 하면 3번 접미사와 10번 접미사를 비교해야 한다
- 3번 접미사의 위치: 8, 10번 접미사는 존재하지 않는다
- 9번째 글자가 c이면 2번 접미사 < 9번 접미사가 될 수 없다. 따라서, 9번째 글자는 d이다



# 접미사 배열 2

115

<https://www.acmicpc.net/problem/13013>

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

접미사 배열:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 7 | 4 | 8 | 6 | 1 | 5 | 2 | 9 | 3 | 0 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|  |   |   |   |   |   |   |   |   |   |
|--|---|---|---|---|---|---|---|---|---|
|  | b | c | d | a | c | b | a | a | d |
|--|---|---|---|---|---|---|---|---|---|

- 9번 접미사 < 3번 접미사
- 3번째 글자를 d라고 하면 10번 접미사와 4번 접미사를 비교해야 한다
- 10번 접미사는 존재하지 않고, 4번 접미사의 위치: 1
- 3번째 글자가 d여도 9번 접미사 < 3번 접미사이다

# 접미사 배열 2

116

<https://www.acmicpc.net/problem/13013>

|         |   |   |   |   |   |   |   |   |   |   |
|---------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:    | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 접미사 배열: | 7 | 4 | 8 | 6 | 1 | 5 | 2 | 9 | 3 | 0 |
| 문자열 S:  | d | b | c | d | a | c | b | a | a | d |

- 3번 접미사 < 0번 접미사
- 0번째 글자를 d라고 하면 4번 접미사와 1번 접미사를 비교해야 한다
- 4번 접미사의 위치: 1, 1번 접미사의 위치: 4
- 0번째 글자가 d여도 3번 접미사 < 0번 접미사이다

# 접미사 배열 2

117

<https://www.acmicpc.net/problem/13013>

- C/C++: <https://gist.github.com/Baekjoon/165da3bd0a0c42a4a7beb998c8fd017b>

# 접미사 배열 1

118

<https://www.acmicpc.net/problem/13012>

- C/C++: <https://gist.github.com/Baekjoon/308be99526e44da099db0fb1dacd616c>

$$O(N^2 \log N) \rightarrow N \log^2 N$$

# 접미사 배열

## Suffix Array

- $O(N(\lg N)^2)$  방법이 있다.
- 길이 1로 정렬
- 길이 2로 정렬
- 길이 4로 정렬
- ...

(N)

$\lg N$

$(N \lg N) \times \lg N$

~~$N^2 \lg N$~~

$N \lg^2 N$

# 접미사 배열

Suffix Array

- 0: abcdabcabb
- 1: bcdabcabb
- 2: cdabcabb
- 3: dabcabb
- 4: abcabb
- 5: bcabb
- 6: cabb
- 7: abb
- 8: bb
- 9: b

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: |   |   |   |   |   |   |   |   |   |   |



# 접미사 배열

Suffix Array

121

- 1글자를 기준으로 정렬하자

# 접미사 배열

Suffix Array

|   |               |
|---|---------------|
| 0 | 4: abcabbb    |
|   | 0: abcdabcabb |
|   | 7: abb        |
| 1 | 1: bcdabcabb  |
|   | 8: bb         |
|   | 5: bcabb      |
|   | 9: b          |
| 2 | 2: cdabcabb   |
|   | 6: cabb       |
| 3 | 3: dabcabb    |

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |

# 접미사 배열

Suffix Array

|   |      |
|---|------|
| 0 | 4: a |
|   | 0: a |
|   | 7: a |
| 1 | 1: b |
|   | 8: b |
|   | 5: b |
|   | 9: b |
| 2 | 2: c |
|   | 6: c |
| 3 | 3: d |

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |

- 1글자로 정렬이 모두 되어있는 상태이다
- 1글자로 정렬을 할 때는, 첫 번째 글자만 같으면 된다

# 접미사 배열

Suffix Array

|   |               |
|---|---------------|
| 0 | 4: abcabbb    |
|   | 0: abcdabcabb |
|   | 7: abb        |
| 1 | 1: bcdabcabb  |
|   | 8: bb         |
|   | 5: bcabb      |
|   | 9: b          |
| 2 | 2: cdabcabb   |
|   | 6: cabb       |
| 3 | 3: dabcabb    |

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |

- 2글자를 기준으로 정렬을 해보자

# 접미사 배열

Suffix Array

125

- 문자열  $S$ 의 접미사의 접미사도 접미사이다

# 접미사 배열

Suffix Array

|   |               |
|---|---------------|
| 0 | 4: abcabbb    |
|   | 0: abcdabcabb |
|   | 7: abb        |
| 1 | 1: bcdabcabb  |
|   | 8: bb         |
|   | 5: bcabb      |
|   | 9: b          |
| 2 | 2: cdabcabb   |
|   | 6: cabb       |
| 3 | 3: dabcabb    |

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |

- 2글자를 기준으로 정렬을 해보자
  - 4번 접미사와 0번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?
- 4: abcabbb
- 0: abcdabcabb

# 접미사 배열

Suffix Array

|   |                |
|---|----------------|
| 0 | 4: abccabb     |
|   | 0: abcdabccabb |
|   | 7: abb         |
| 1 | 1: bcdabccabb  |
|   | 8: bb          |
|   | 5: bcabb       |
|   | 9: b           |
| 2 | 2: cdabccabb   |
|   | 6: cabb        |
| 3 | 3: dabccabb    |

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| a | b | c | d | a | b | c | a | b | b |
|---|---|---|---|---|---|---|---|---|---|

그룹 번호:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|

127

- 2글자를 기준으로 정렬을 해보자
- 4번 접미사와 0번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?

4: ab

0: ab

- 같아야 한다
- 어떻게 알 수 있을까?

# 접미사 배열

Suffix Array

|   |                |
|---|----------------|
| 0 | 4: abccabb     |
|   | 0: abcdabccabb |
|   | 7: abb         |
| 1 | 1: bcdabccabb  |
|   | 8: bb          |
|   | 5: bcabb       |
|   | 9: b           |
| 2 | 2: cdabccabb   |
|   | 6: cabb        |
| 3 | 3: dabccabb    |

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| a | b | c | d | a | b | c | a | b | b |
|---|---|---|---|---|---|---|---|---|---|

그룹 번호:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|

128

- 2글자를 기준으로 정렬을 해보자
- 4번 접미사와 0번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?

4: abccabb

0: abcdabccabb

- 4번 접미사의 그룹 번호: 0
- 0번 접미사의 그룹 번호: 0





# 접미사 배열

Suffix Array

|   |                |
|---|----------------|
| 0 | 4: abccabb     |
|   | 0: abcdabccabb |
|   | 7: abb         |
| 1 | 1: bcdabccabb  |
|   | 8: bb          |
|   | 5: bcabb       |
|   | 9: b           |
| 2 | 2: cdabccabb   |
|   | 6: cabb        |
| 3 | 3: dabccabb    |

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| a | b | c | d | a | b | c | a | b | b |
|---|---|---|---|---|---|---|---|---|---|

그룹 번호:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|

129

- 2글자를 기준으로 정렬을 해보자
- 4번 접미사와 0번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?

5: bcabb

1: bcdabccabb

- 5번 접미사의 그룹 번호: 1
- 1번 접미사의 그룹 번호: 1

# 접미사 배열

Suffix Array

|   |               |
|---|---------------|
| 0 | 4: abcabbb    |
|   | 0: abcdabcabb |
|   | 7: abb        |
| 1 | 1: bcdabcabb  |
|   | 8: bb         |
|   | 5: bcabb      |
|   | 9: b          |
| 2 | 2: cdabcabb   |
|   | 6: cabb       |
| 3 | 3: dabcabb    |

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |

- 2글자를 기준으로 정렬을 해보자
- 4번 접미사와 0번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?
- 4: abcabbb
- 0: abcdabcabb
- 2글자를 기준으로 정렬했을 때
- 두 접미사의 그룹 번호는 같다

# 접미사 배열

Suffix Array

|   |                |
|---|----------------|
| 0 | 4: abccabb     |
|   | 0: abcdabccabb |
|   | 7: abb         |
| 1 | 1: bcdabccabb  |
|   | 8: bb          |
|   | 5: bcabb       |
|   | 9: b           |
| 2 | 2: cdabccabb   |
|   | 6: cabb        |
| 3 | 3: dabccabb    |

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| a | b | c | d | a | b | c | a | b | b |
|---|---|---|---|---|---|---|---|---|---|

그룹 번호:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|

131

- 2글자를 기준으로 정렬을 해보자
- 0번 접미사와 1번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?

0: abcdabccabb

1: bcdabccabb

- 0번 접미사의 그룹 번호: 0
- 1번 접미사의 그룹 번호: 1

# 접미사 배열

Suffix Array

|   |                |
|---|----------------|
| 0 | 4: abccabb     |
|   | 0: abcdabccabb |
|   | 7: abb         |
| 1 | 1: bcdabccabb  |
|   | 8: bb          |
|   | 5: bcabb       |
|   | 9: b           |
| 2 | 2: cdabccabb   |
|   | 6: cabb        |
| 3 | 3: dabccabb    |

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |

- 2글자를 기준으로 정렬을 해보자
  - 0번 접미사와 1번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?
- 0: abcdabccabb
- 1: bcdabccabb
- 2글자를 기준으로 정렬했을 때
  - 0번 접미사가 앞서야 한다

# 접미사 배열

Suffix Array

|   |                |
|---|----------------|
| 0 | 4: abccabb     |
|   | 0: abcdabccabb |
|   | 7: abb         |
| 1 | 1: bcdabccabb  |
|   | 8: bb          |
|   | 5: bcabb       |
|   | 9: b           |
| 2 | 2: cdabccabb   |
|   | 6: cabb        |
| 3 | 3: dabccabb    |

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |

- 2글자를 기준으로 정렬을 해보자
- 2번 접미사와 6번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?
- 2: cdabccabb
- 6: cabb
- 2번 접미사의 그룹 번호: 2
- 6번 접미사의 그룹 번호: 2

# 접미사 배열

Suffix Array

|   |                |
|---|----------------|
| 0 | 4: abccabb     |
|   | 0: abcdabccabb |
|   | 7: abb         |
| 1 | 1: bcdabccabb  |
|   | 8: bb          |
|   | 5: bcabb       |
|   | 9: b           |
| 2 | 2: cdabccabb   |
|   | 6: cabb        |
| 3 | 3: dabccabb    |

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |

- 2글자를 기준으로 정렬을 해보자
- 2번 접미사와 6번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?
- 3: dabccabb
- 7: abb
- 3번 접미사의 그룹 번호: 3
- 7번 접미사의 그룹 번호: 0

# 접미사 배열

Suffix Array

|   |                |
|---|----------------|
| 0 | 4: abccabb     |
|   | 0: abcdabccabb |
|   | 7: abb         |
| 1 | 1: bcdabccabb  |
|   | 8: bb          |
|   | 5: bcabb       |
|   | 9: b           |
| 2 | 2: cdabccabb   |
|   | 6: cabb        |
| 3 | 3: dabccabb    |

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| a | b | c | d | a | b | c | a | b | b |
|---|---|---|---|---|---|---|---|---|---|

그룹 번호:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|

135

- 2글자를 기준으로 정렬을 해보자
- 2번 접미사와 6번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?

2: cdabccabb

6: cabb

- 2글자를 기준으로 정렬했을 때
- 6번 접미사가 앞서야 한다

# 접미사 배열

Suffix Array

|   |                |
|---|----------------|
| 0 | 4: abccabb     |
|   | 0: abcdabccabb |
|   | 7: abb         |
| 1 | 1: bcdabccabb  |
|   | 8: bb          |
|   | 5: bcabb       |
|   | 9: b           |
| 2 | 2: cdabccabb   |
|   | 6: cabb        |
| 3 | 3: dabccabb    |

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |

- 2글자를 기준으로 정렬을 해보자
  - 8번 접미사와 9번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?
- 8: bb
- 9: b
- 8번 접미사의 그룹 번호: 1
  - 9번 접미사의 그룹 번호: 1



# 접미사 배열

Suffix Array

|   |                |
|---|----------------|
| 0 | 4: abccabb     |
|   | 0: abcdabccabb |
|   | 7: abb         |
| 1 | 1: bcdabccabb  |
|   | 8: bb          |
|   | 5: bcabb       |
|   | 9: b           |
| 2 | 2: cdabccabb   |
|   | 6: cabb        |
| 3 | 3: dabccabb    |

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| a | b | c | d | a | b | c | a | b | b |
|---|---|---|---|---|---|---|---|---|---|

그룹 번호:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|

137

- 2글자를 기준으로 정렬을 해보자
- 8번 접미사와 9번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?

9: b

10:

- 9번 접미사의 그룹 번호: 1
- 10번 접미사의 그룹 번호: ?

# 접미사 배열

Suffix Array

|   |                |
|---|----------------|
| 0 | 4: abccabb     |
|   | 0: abcdabccabb |
|   | 7: abb         |
| 1 | 1: bcdabccabb  |
|   | 8: bb          |
|   | 5: bcabb       |
|   | 9: b           |
| 2 | 2: cdabccabb   |
|   | 6: cabb        |
| 3 | 3: dabccabb    |

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| a | b | c | d | a | b | c | a | b | b |
|---|---|---|---|---|---|---|---|---|---|

그룹 번호:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |
|---|---|---|---|---|---|---|---|---|---|

138

- 10번 접미사처럼 문자열의 길이를 넘어가는 경우가 존재할 수 있다
- 이 경우 길이가  $n$ 일 때,  $n$ 번 접미사를 비교하는 일만 일어나기 때문에
- 즉,  $n+1$ 번 이상 접미사를 비교하는 일이 없기 때문에
- $\text{group}[n] = -1$ 로 넣어준다

# 접미사 배열

Suffix Array

|   |                |
|---|----------------|
| 0 | 4: abccabb     |
|   | 0: abcdabccabb |
|   | 7: abb         |
| 1 | 1: bcdabccabb  |
|   | 8: bb          |
|   | 5: bcabb       |
|   | 9: b           |
| 2 | 2: cdabccabb   |
|   | 6: cabb        |
| 3 | 3: dabccabb    |

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |

- 2글자를 기준으로 정렬을 해보자
- 8번 접미사와 9번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?
- 9:    b
- 10:
- 9번 접미사의 그룹 번호: 1
- 10번 접미사의 그룹 번호: -1

# 접미사 배열

Suffix Array

|   |                |
|---|----------------|
| 0 | 4: abccabb     |
|   | 0: abcdabccabb |
|   | 7: abb         |
| 1 | 1: bcdabccabb  |
|   | 8: bb          |
|   | 5: bcabb       |
|   | 9: b           |
| 2 | 2: cdabccabb   |
|   | 6: cabb        |
| 3 | 3: dabccabb    |

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 0 | 1 | 1 |

- 2글자를 기준으로 정렬을 해보자
- 8번 접미사와 9번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?
- 8: bb
- 9: b
- 2글자를 기준으로 정렬했을 때
- 9번 접미사가 앞서야 한다

# 접미사 배열

Suffix Array

|   |    |            |
|---|----|------------|
| 0 | 4: | abcabb     |
|   | 0: | abcdabcabb |
|   | 7: | abb        |
| 1 | 9: | b          |
| 2 | 8: | bb         |
| 3 | 1: | bcdabcabb  |
|   | 5: | bcabb      |
| 4 | 6: | cabb       |
| 5 | 2: | cdabcabb   |
| 6 | 3: | dabcabb    |

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: | 0 | 3 | 5 | 6 | 0 | 3 | 2 | 0 | 2 | 1 |

- 2글자를 기준으로 정렬을 해보자

# 접미사 배열

Suffix Array

|   |     |    |
|---|-----|----|
| 0 | 4 : | ab |
|   | 0 : | ab |
|   | 7 : | ab |
| 1 | 9 : | b  |
| 2 | 8 : | bb |
| 3 | 1 : | bc |
|   | 5 : | bc |
| 4 | 6 : | ca |
| 5 | 2 : | cd |
| 6 | 3 : | da |

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: | 0 | 3 | 5 | 6 | 0 | 3 | 2 | 0 | 2 | 1 |

- 2글자를 기준으로 정렬이 되어있는 상태이다

# 접미사 배열

Suffix Array

|   |     |            |
|---|-----|------------|
| 0 | 4 : | abcabb     |
|   | 0 : | abcdabcabb |
|   | 7 : | abb        |
| 1 | 9 : | b          |
| 2 | 8 : | bb         |
| 3 | 1 : | bcdabcabb  |
|   | 5 : | bcabb      |
| 4 | 6 : | cabb       |
| 5 | 2 : | cdabcabb   |
| 6 | 3 : | dabcabb    |

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: | 0 | 3 | 5 | 6 | 0 | 3 | 2 | 0 | 2 | 1 |

- 4글자를 기준으로 정렬을 해보자

# 접미사 배열

Suffix Array

|   |    |            |
|---|----|------------|
| 0 | 4: | abcabb     |
|   | 0: | abcdabcabb |
|   | 7: | abb        |
| 1 | 9: | b          |
| 2 | 8: | bb         |
| 3 | 1: | bcdabcabb  |
|   | 5: | bcabb      |
| 4 | 6: | cabb       |
| 5 | 2: | cdabcabb   |
| 6 | 3: | dabcabb    |

|        |   |   |   |   |   |   |   |   |   |   |
|--------|---|---|---|---|---|---|---|---|---|---|
| 인덱스:   | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 문자열 S: | a | b | c | d | a | b | c | a | b | b |
| 그룹 번호: | 0 | 3 | 5 | 6 | 0 | 3 | 2 | 0 | 2 | 1 |

- 4글자를 기준으로 정렬을 해보자
  - 4번 접미사와 0번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?
- 4: abcabb
- 0: abcdabcabb
- 4번 접미사의 그룹 번호: 0
  - 0번 접미사의 그룹 번호: 0



# 접미사 배열

Suffix Array

|   |                |
|---|----------------|
| 0 | 4: abccabb     |
|   | 0: abcdabccabb |
|   | 7: abb         |
| 1 | 9: b           |
| 2 | 8: bb          |
| 3 | 1: bcdabccabb  |
|   | 5: bcabb       |
| 4 | 6: cabb        |
| 5 | 2: cdabccabb   |
| 6 | 3: dabccabb    |

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| a | b | c | d | a | b | c | a | b | b |
|---|---|---|---|---|---|---|---|---|---|

그룹 번호:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 3 | 5 | 6 | 0 | 3 | 2 | 0 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|

- 4글자를 기준으로 정렬을 해보자
- 4번 접미사와 0번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?

6:    cabb

2:    cdabccabb

- 6번 접미사의 그룹 번호: 2
- 2번 접미사의 그룹 번호: 5

# 접미사 배열

Suffix Array

|   |     |            |
|---|-----|------------|
| 0 | 4 : | abcabb     |
|   | 0 : | abcdabcabb |
|   | 7 : | abb        |
| 1 | 9 : | b          |
| 2 | 8 : | bb         |
| 3 | 1 : | bcdabcabb  |
|   | 5 : | bcabb      |
| 4 | 6 : | cabb       |
| 5 | 2 : | cdabcabb   |
| 6 | 3 : | dabcabb    |

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| a | b | c | d | a | b | c | a | b | b |
|---|---|---|---|---|---|---|---|---|---|

그룹 번호:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 3 | 5 | 6 | 0 | 3 | 2 | 0 | 2 | 1 |
|---|---|---|---|---|---|---|---|---|---|

146

- 4글자를 기준으로 정렬을 해보자
- 4번 접미사와 0번 접미사는 두 글자를 기준으로 정렬하면 누가 앞에 와야 할까?

4 : abcabb

0 : abcdabcabb

- 2글자를 기준으로 정렬했을 때
- 4번 접미사가 앞서야 한다

# 접미사 배열

Suffix Array

0 | 7: abb  
1 | 4: abcabb  
2 | 0: abcdabcabb  
3 | 9: b  
4 | 8: bb  
5 | 5: bcabb  
6 | 1: bcdabcabb  
7 | 6: cabb  
8 | 2: cdabcabb  
9 | 3: dabcabb

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| a | b | c | d | a | b | c | a | b | b |
|---|---|---|---|---|---|---|---|---|---|

그룹 번호:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 6 | 8 | 9 | 1 | 5 | 7 | 0 | 4 | 3 |
|---|---|---|---|---|---|---|---|---|---|

- 4글자를 기준으로 정렬을 해보자

# 접미사 배열

Suffix Array

0 | 7: abb  
1 | 4: abca  
2 | 0: abcd  
3 | 9: b  
4 | 8: bb  
5 | 5: bcab  
6 | 1: bcda  
7 | 6: cabb  
8 | 2: cdab  
9 | 3: dabc

인덱스:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|

문자열 S:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| a | b | c | d | a | b | c | a | b | b |
|---|---|---|---|---|---|---|---|---|---|

그룹 번호:

|   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|
| 2 | 6 | 8 | 9 | 1 | 5 | 7 | 0 | 4 | 3 |
|---|---|---|---|---|---|---|---|---|---|

- 4글자를 기준으로 정렬이 되어있는 상태

문자열 정렬 2기

$O(1)$   $O(N \log N)$   $O(N)$   
 $N \log^2 N$

# 접미사 배열

## Suffix Array

149

- t글자를 기준으로 정렬되어 있다면, t\*2글자를 기준으로도 정렬할 수 있다
- N개의 문자열을 정렬하는데 걸리는 시간:  $O(N \lg N)$
- 그런데, 문자열 비교가 아니고 정수 비교 2번으로 정렬할 수 있다.
- 따라서,  $O(N^2 \lg N)$ 이 아니고  $O(N \lg N)$
- 정렬은 총  $\lg N$ 번 반복된다.
- 따라서,  $O(N(\lg N)^2)$  이다.

# 접미사 배열 2

150

<https://www.acmicpc.net/problem/13264>

- <https://gist.github.com/Baekjoon/3bf266d04f7cf4244a48192cae079f89>
- <https://gist.github.com/Baekjoon/8f496d0648c97eb42d94e135e635f930>

# 서로 다른 부분 문자열의 개수

<https://www.acmicpc.net/problem/11478>

- 모든 부분 문자열은 suffix의 prefix 이다.
- abcde의 서로 다른 부분 문자열은 몇 개일까?

# 서로 다른 부분 문자열의 개수

<https://www.acmicpc.net/problem/11478>

- 모든 부분 문자열은 suffix의 prefix 이다.
- abcd의 서로 다른 부분 문자열은 몇 개일까?
- 10개
- abcd
- bcd
- cd
- d



# 서로 다른 부분 문자열의 개수

<https://www.acmicpc.net/problem/11478>

- 모든 부분 문자열은 suffix의 prefix 이다.
- abcd의 서로 다른 부분 문자열은 몇 개일까?
- 10개

- abcd (a, ab, abc, abcd)
- bcd (b, bc, bcd)
- cd (c, cd)
- d (d)

$\sum_{i=1}^N N_i$  prefix N

# 서로 다른 부분 문자열의 개수

<https://www.acmicpc.net/problem/11478>

- 모든 부분 문자열은 suffix의 prefix 이다.
- abab의 서로 다른 부분 문자열은 몇 개일까?
- 7개

$\begin{array}{l} \rightarrow ab(a, ab) \\ \rightarrow abab(\text{ab}, aba, abab) \end{array}$

LCP

$\begin{array}{l} \bullet b(b) \\ \bullet \underline{bab}(b, ba, bab) \end{array}$

-2

-0

-1

a b ~~a~~ ~~b~~  
 ab ba ~~ab~~  
 abab bab

# 서로 다른 부분 문자열의 개수

<https://www.acmicpc.net/problem/11478>

- prefix가 같은 것의 개수를 빼줘야 한다.
- suffix array는 사전순 정렬이기 때문에, 바로 인접한 것과 비교를 해야 한다.
- $i$ 와  $i-1$ 이 prefix가 몇개 까지 겹치는 지를 세어줘야 한다.

# 서로 다른 부분 문자열의 개수

156

<https://www.acmicpc.net/problem/11478>

- <https://gist.github.com/Baekjoon/6db55d4083f0fb4c9226>

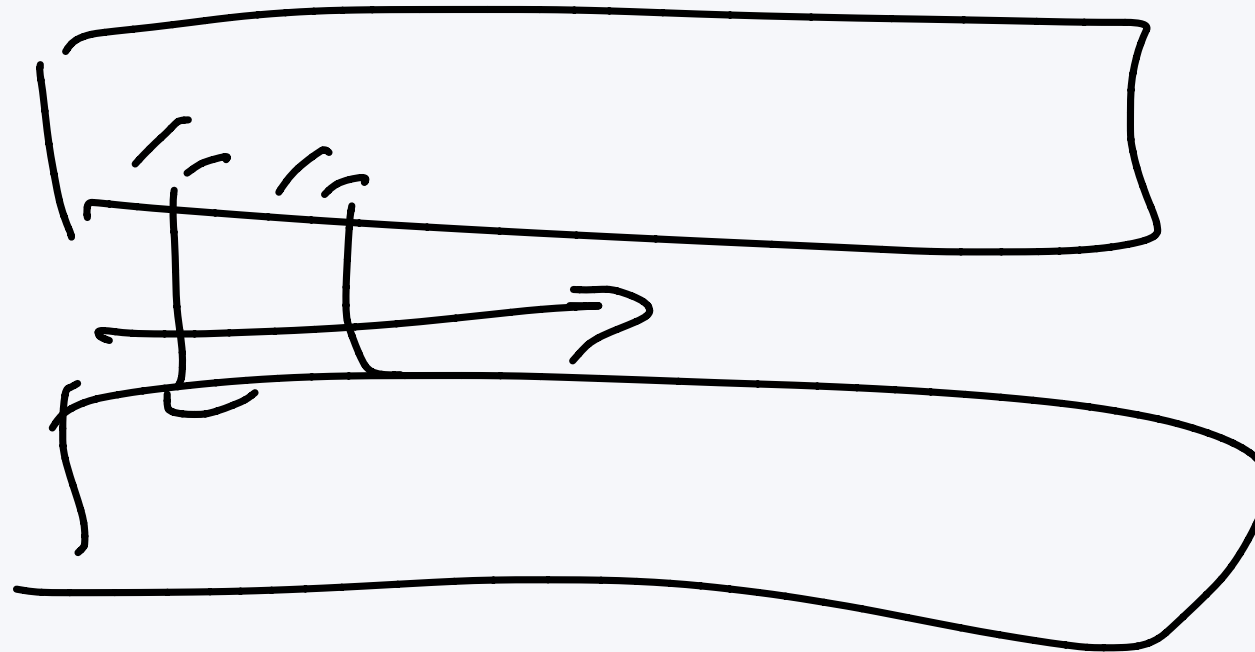
# LCP

## Longest Common Prefix

- 가장 긴 prefix의 길이
- 앞에서 구한 P배열을 이용하면 LCP를  $\lg N$ 만에 구할 수 있다.
- 두 suffix  $i$ 와  $i+1$ 가 있을 때, 큰  $k$ 부터 1씩 감소해나가면서 비교할 수 있다.

LCP

$O(\text{길이})$

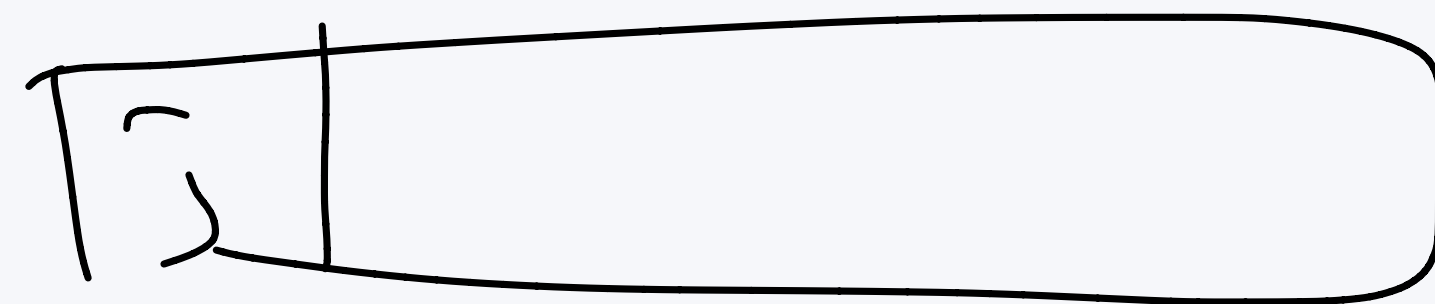
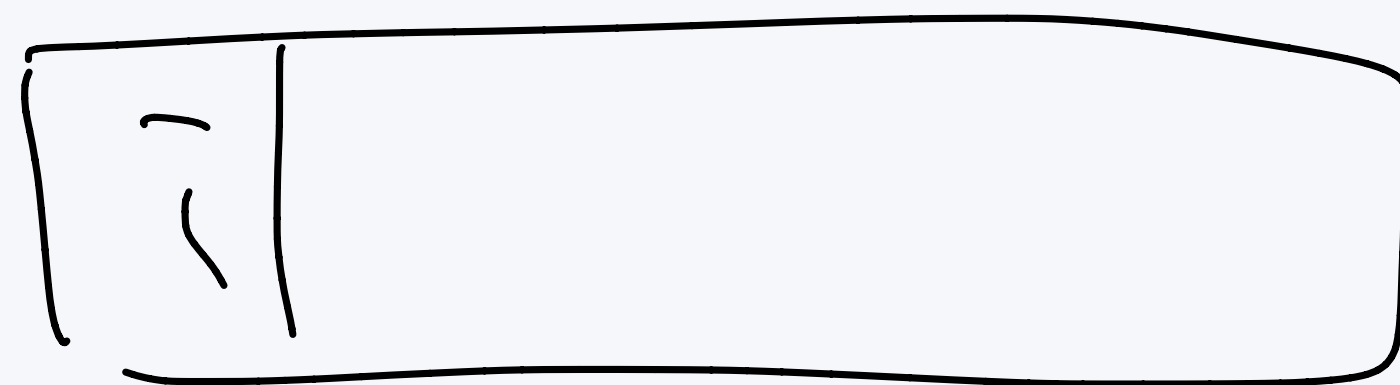


# 서로 다른 부분 문자열의 개수 2

<https://www.acmicpc.net/problem/11479>

- LCP를  $\lg N$ 만에 구하면 된다.

(LCP)  $\lg N$



group[k]

문자열  $2^k$  3 3개  
길이  $2^k$  23

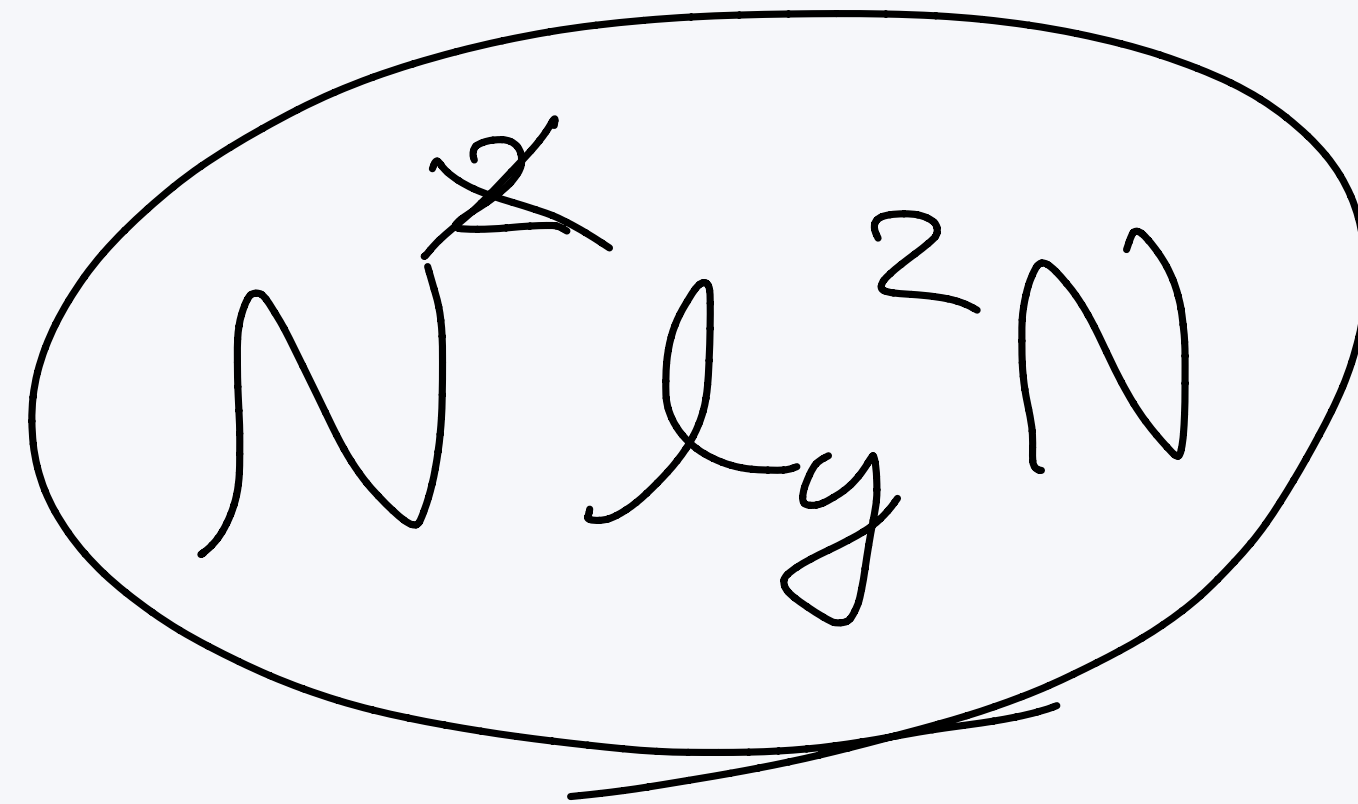
$$\text{group}[2][i] = 2 \quad \text{group}[2][j]$$

# 서로 다른 부분 문자열의 개수 2

159

<https://www.acmicpc.net/problem/11479>

- <https://gist.github.com/Baekjoon/45770e6c00d4fd7009b7dd7f24a6f80e>



A handwritten mathematical expression  $N^2 \lg^2 N$  is enclosed within a hand-drawn oval. The expression is written in a cursive, handwritten style.

# 기수 정렬

Radix Sort

160

- 낮은 자리수 부터 비교하면서 정렬해나가는 방법



# 기수 정렬

Radix Sort

161

- 260, 35, 25, 160, 8, 90, 33, 180, 23, 45, 10, 98을 정렬해보자

# 기수 정렬

Radix Sort

162

- 260, 35, 25, 160, 8, 90, 33, 180, 23, 45, 10, 98을 정렬해보자
- 0: 260, 160, 90, 180, 10
- 1:
- 2:
- 3: 33, 23
- 4:
- 5: 35, 25, 45
- 6:
- 7:
- 8: 8, 98
- 9:

# 기수 정렬

Radix Sort

163

- 0: 260, 160, 90, 180, 10
- 1:
- 2:
- 3: 33, 23
- 4:
- 5: 35, 25, 45
- 6:
- 7:
- 8: 8, 98
- 9:
- 결과: 260, 160, 90, 180, 10, 33, 23, 35, 25, 45, 8, 98

# 기수 정렬

Radix Sort

164

- 260, 160, 90, 180, 10, 33, 23, 35, 25, 45, 8, 98을 정렬해보자
- 0: 8
- 1: 10
- 2: 23, 25
- 3: 33
- 4: 45
- 5:
- 6: 260, 160
- 7:
- 8: 180
- 9: 90, 98

# 기수 정렬

Radix Sort

165

- 0: 8
- 1: 10
- 2: 23, 25
- 3: 33
- 4: 45
- 5:
- 6: 260, 160
- 7:
- 8: 180
- 9: 90, 98
- 결과: 8, 10, 23, 25, 33, 45, 260, 160, 180, 90, 98

# 기수 정렬

Radix Sort

166

- 8, 10, 23, 25, 33, 45, 260, 160, 180, 90, 98을 정렬해보자
- 0: 8, 10, 23, 25, 33, 45, 90, 98
- 1: 160, 180
- 2: 260
- 3:
- 4:
- 5:
- 6:
- 7:
- 8:
- 9:

# 기수 정렬

Radix Sort

167

- 0: 8, 10, 23, 25, 33, 45, 90, 98
- 1: 160, 180
- 2: 260
- 3:
- 4:
- 5:
- 6:
- 7:
- 8:
- 9:
- 결과: 8, 10, 23, 25, 33, 45, 90, 98, 160, 180, 260

# 기수 정렬

Radix Sort

168

- 기수 정렬의 시간 복잡도는  $O(dN)$  이다
- 여기서  $N$ 은 수의 개수,  $d$ 는 수의 자리



# 접미사 배열

## Suffix Array

- t글자를 기준으로 정렬되어 있다면, t\*2글자를 기준으로도 정렬할 수 있다
- N개의 문자열을 기수 정렬하는데 걸리는 시간:  $O(N)$
- 두 번 정렬해야 하기 때문에  $O(N)$
- 정렬은 총  $\lg N$ 번 반복된다.
- 따라서,  $O(N \lg N)$  이다.

정렬

반복

$$O(N \lg N) \times \lg N$$

$$O(N) \times \lg N$$

$$O(N \lg N)$$

# 접미사 배열

Suffix Array

170

- <https://gist.github.com/Baekjoon/822fd97fb81ebce548854d76f19df933>

# 접미사 배열

Suffix Array

171

- $O(N)$ 만에 접미사 배열을 구현할 수 있다
- [https://github.com/kcm1700/algorithms/blob/master/string/suffix\\_array.cpp](https://github.com/kcm1700/algorithms/blob/master/string/suffix_array.cpp)

# Suffix Array

<https://www.acmicpc.net/problem/9248>

172

- 접미사 배열과 LCP 배열을 만드는 문제

# Suffix Array

173

<https://www.acmicpc.net/problem/9248>

- $LCP[i] = i$ 번째 Suffix와  $i-1$ 번째 Suffix의 LCP 길이

# Suffix Array

174

<https://www.acmicpc.net/problem/9248>

- <https://gist.github.com/Baekjoon/f83f42df84084dffc7c610f54bc37c71>

# Cubeditor

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

175

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

4

Longest

KMP

부분 문자열

Suffix의

prefix

CP

Suffix Array

# Cubeditor

176

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

- 모든 부분 문자열은
- Suffix의 Prefix이다

|  |        |  |
|--|--------|--|
|  | 부분 문자열 |  |
|--|--------|--|



# Cubeditor

177

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

- 즉, 모든 접미사를 구하고 LCP 길이의 최대값이 정답이 된다

# Cubeditor

178

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

- <https://gist.github.com/Baekjoon/a393f18ad365de2e1274933cd2283d9d>

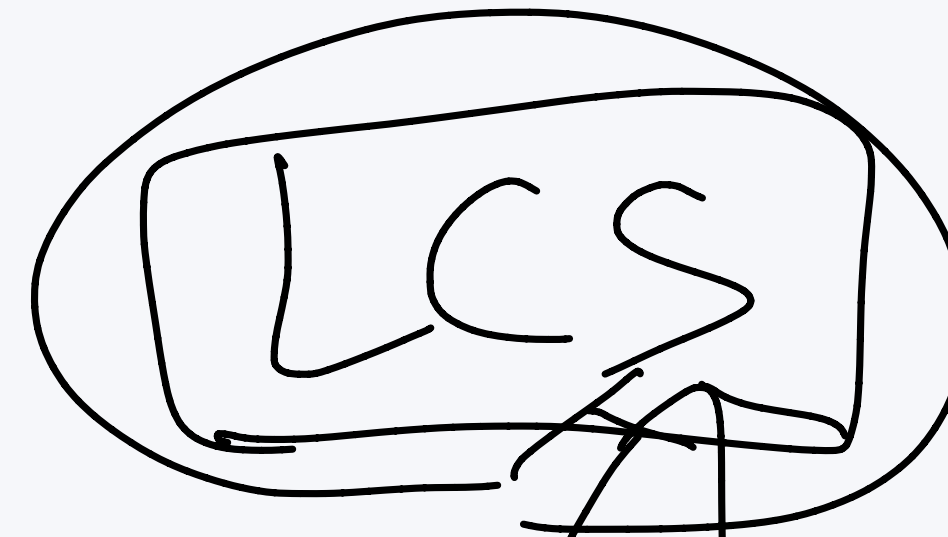
# 공통 부분 문자열

<https://www.acmicpc.net/problem/5582>

- 두 문자열에 등장하는 부분 문자열 (연속) 중에서 가장 긴 것을 찾는 문제

- ABRACADABRA

- ECADADABRBCRDARA



SubSequence

SubString

# 공통 부분 문자열

180

<https://www.acmicpc.net/problem/5582>

- 두 문자열에 등장하는 부분 문자열 (연속) 중에서 가장 긴 것을 찾는 문제

A ABRACADABRA

B ECADADABRBCRDARA

$$A[i] == B[j]$$

$$DP[i][j] = \frac{DP[i-1][j-1] + 1}{0}$$

# 공통 부분 문자열

181

<https://www.acmicpc.net/problem/5582>

- 두 문자열에 등장하는 부분 문자열 (연속) 중에서 가장 긴 것을 찾는 문제
- ABRACADABRA
- ECADADABRBCRDARA
- $D[i][j] = D[i-1][j-1] + 1 \text{ (} A[i] == A[j] \text{) else } 0$

# 공통 부분 문자열

182

<https://www.acmicpc.net/problem/5582>

- A
- ABRA
- ABRACADABRA
- ACADABRA
- ADABRA
- BRA
- BRACADABRA
- CADABRA
- DABRA
- RA
- RACADABRA
- ABRBC
- ADABRBC
- ADADABRBC
- BC
- BRBC
- C
- CADADABRBC
- DABRBC
- DADABRBC
- ECADADABRBC
- RBC

# 공통 부분 문자열

<https://www.acmicpc.net/problem/5582>

- A의 suffix x에 대해서, s보다 크면서 가장 작은 B의 suffix y를 찾는다
- x와 y의 LCP가 공통 부분 문자열이 된다.
- 이 값 중에서 가장 큰 값을 찾으면 된다.

# 공통 부분 문자열

184

<https://www.acmicpc.net/problem/5582>

- 이 방법은 LCP를 구하는 것이  $O(N)$  이다



# 공통 부분 문자열

185

<https://www.acmicpc.net/problem/5582>

- 앞의 방법은 아래와 같이 두 문자열의 접미사 배열을 구하고 그것을 합친 것으로 볼 수 있다
- abc의 접미사 배열
  - abc
  - bc
  - c
- bcd의 접미사 배열
  - bcd
  - cd
  - d

# 공통 부분 문자열

<https://www.acmicpc.net/problem/5582>

- 앞의 방법은 아래와 같이 두 문자열의 접미사 배열을 구하고 그것을 합친 것으로 볼 수 있다

- abc의 접미사 배열

- abc

- bc

- c

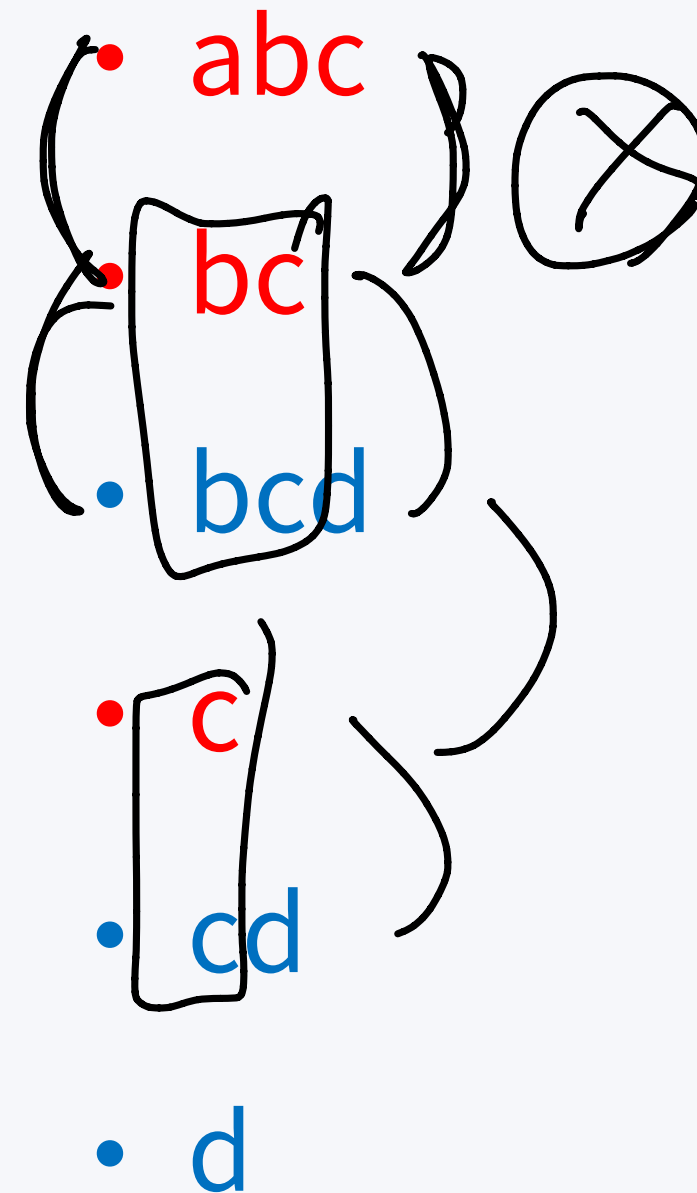
- bcd의 접미사 배열

- bcd

- cd

- d

- abc와 bcd의 접미사 배열을 합친 결과



# 공통 부분 문자열

187

<https://www.acmicpc.net/problem/5582>

- 이것은 두 문자열을 합친 다음 접미사 배열을 구한 결과와 같다
- **abcbcd**의 접미사 배열
  - **abcbcd**
  - **bcbcd**
  - **bcd**
  - **cbcd**
  - **cd**
  - **d**
- abc와 bcd의 접미사 배열을 합친 결과
  - **abc**
  - **bc**
  - **bcd**
  - **c**
  - **cd**
  - **d**

# 공통 부분 문자열

188

<https://www.acmicpc.net/problem/5582>

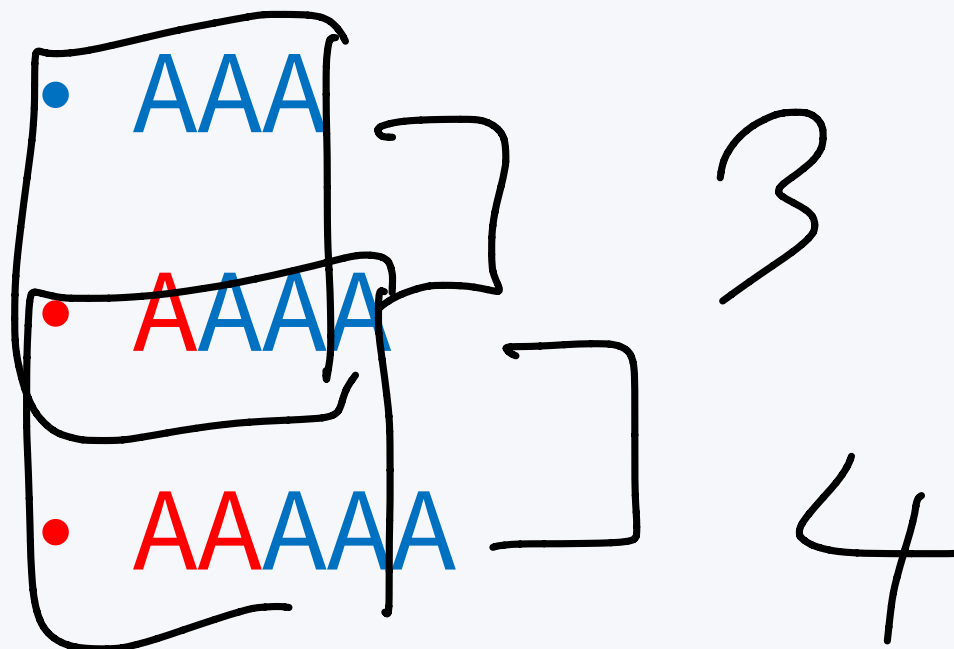
- 정답은 두 문자열을 합친 다음, 접미사 배열의 LCP의 최댓값이 된다

# 공통 부분 문자열

189

<https://www.acmicpc.net/problem/5582>

- 반례가 존재한다
- $A = AA$
- $B = AAA$
- $AA$ 의 접미사
- $A$
- $AA$



# 공통 부분 문자열

190

<https://www.acmicpc.net/problem/5582>

- 사이에 A와 B에 등장할 수 없는 문자를 추가하면 된다
- $S = A + \text{'\#' } + B$

# 공통 부분 문자열

191

<https://www.acmicpc.net/problem/5582>

- <https://gist.github.com/Baekjoon/3dc030c811c306c2636113534935eca9>

# Hidden Password

<https://www.acmicpc.net/problem/3789>

- 길이가 N인 문자열 S가 주어진다.
- 문자열 S의 가장 왼쪽 글자를 오른쪽에 붙여가면서 만들 수 있는 단어는 총 N개가 있다
- 예: S = alabala
- alabala
- labalaa
- abalaal
- balaala
- aalabal
- laalaba
- aalabal



# Hidden Password

193

<https://www.acmicpc.net/problem/3789>

- 길이가  $N$ 인 문자열  $S$ 가 주어진다.
- 문자열  $S$ 의 가장 왼쪽 글자를 오른쪽에 붙여가면서 만들 수 있는 단어는 총  $N$ 개가 있다
- 이 때, 사전 순으로 가장 앞서는 단어를 찾는 문제

# Hidden Password

194

<https://www.acmicpc.net/problem/3789>

- 글자의 길이를 2배로 한 다음, 접미사 배열을 만든다
- 접미사의 번호가 원래 문자열의 길이보다 작은 것 중 첫 번째가 정답이 된다.

# Hidden Password

<https://www.acmicpc.net/problem/3789>

•  $S = \text{dbac}$ ,  $S*2 = \text{dbacdbac}$

• ac

• acdbac

• bac

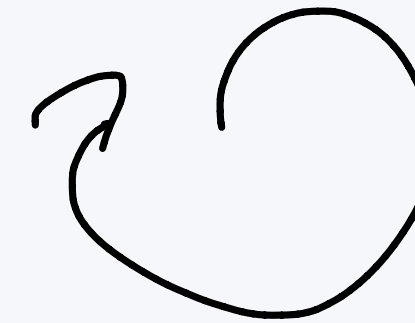
• bacdbac

• c

• cdbac

• dbac

• dbacdbac



# Hidden Password

196

<https://www.acmicpc.net/problem/3789>

- $S = \text{baba}$ ,  $S*2 = \text{babababa}$
- a
- aba
- ababa
- abababa
- ba
- baba
- bababa
- babababa

# Hidden Password

197

<https://www.acmicpc.net/problem/3789>

- 같을 때는, 인덱스가 뒤에 있는것을 먼저 오게 정렬하면 된다

# Hidden Password

198

<https://www.acmicpc.net/problem/3789>

- <https://gist.github.com/Baekjoon/ac8a8ad7295db9f42889113c2d4b74ca>

# 가장 긴 팰린드롬 부분 문자열

199

<https://www.acmicpc.net/problem/13275>

- 문자열 S의 가장 긴 팰린드롬 부분 문자열을 찾는 문제

# 가장 긴 팰린드롬 부분 문자열

200

<https://www.acmicpc.net/problem/13275>

- 문자열  $S$ 와  $S'$  ( $S$ 를 뒤집은 것)을 만들고
- 공통 부분 문자열 문제와 비슷하게 풀면 된다



# 가장 긴 팰린드롬 부분 문자열

201

<https://www.acmicpc.net/problem/13275>

- 문자열  $S$ 와  $S'$  ( $S$ 를 뒤집은 것)을 만들고
- 공통 부분 문자열 문제와 비슷하게 풀면 된다

# 가장 긴 팰린드롬 부분 문자열

202

<https://www.acmicpc.net/problem/13275>

- $S = \text{banana}$
- $S' = \text{ananab}$
- $S + \text{'\#'} + S' = \text{banana\#ananab}$

# 가장 긴 팰린드롬 부분 문자열

<https://www.acmicpc.net/problem/13275>

- 6 #ananab
- 5 a#ananab (LCP = 0)
- 11 ab (LCP = 1)
- 3 ana#ananab (LCP = 1)
- 9 anab (LCP = 3)
- 1 anana#ananab (LCP = 3)
- 7 ananab (LCP = 5)
- 12 b (LCP = 0)
- 0 banana#ananab (LCP = 1)
- 4 na#ananab (LCP = 0)
- 10 nab (LCP = 2)
- 2 nana#ananab (LCP = 2)
- 8 nanab (LCP = 4)

# 가장 긴 팰린드롬 부분 문자열

204

<https://www.acmicpc.net/problem/13275>

- <https://gist.github.com/Baekjoon/320400c05b01f4c890e8cdf5be5ee639>