컴퓨터공학 All in One

C/C++ 문법, 자료구조 및 심화 프로젝트 (나동빈)
제 43강 - KMP 문자열 매칭

학습 목표

KMP 문자열 매칭

1) 구현하기 쉬운 문자열 매칭 알고리즘의 원리와 구현 방법에 대해서 이해합니다.
2) 효율적인 문자열 매칭 알고리즘인 KMP 알고리즘의 원리에 대해서 이해합니다.
3) KMP 알고리즘을 C언어를 이용해 구현할 수 있습니다.

KMP 문자열 매칭

단순 비교 문자열 매칭

1) 단순 비교 문자열 매칭 알고리즘은 두 문자열을 처음부터 끝까지 계속 비교하는 알고리즘입니다.
2) 단순 비교 문자열 매칭 알고리즘은 $O(NM)$의 시간 복잡도를 가집니다.

Fast campus

KMP 문자열 매칭

단순 비교 문자열 매칭: 문자열 A에서 문자열 B를 찾는 과정

문자열 A: “ABCDEFG”
문자열 B: “EF”

문자열 A	A	B	C	D	E	F	G
문자열 B	E	F					

Fast campus

KMP 문자열 매칭

단순 비교 문자열 매칭: 문자열 A에서 문자열 B를 찾는 과정

문자열 A: “ABCDEFG”
문자열 B: “EF”

문자열 A	A	B	C	D	E	F	G
문자열 B		E	F				

KMP 문자열 매칭

단순 비교 문자열 매칭: 문자열 A에서 문자열 B를 찾는 과정

문자열 A: “ABCDEFG”
문자열 B: “EF”

문자열 A	A	B	C	D	E	F	G
문자열 B			E	F			

KMP 문자열 매칭

단순 비교 문자열 매칭: 문자열 A에서 문자열 B를 찾는 과정

문자열 A: “ABCDEFG”
문자열 B: “EF”

문자열 A	A	B	C	D	E	F	G
문자열 B				E	F		

Fast campus

KMP 문자열 매칭

단순 비교 문자열 매칭: 문자열 A에서 문자열 B를 찾는 과정

문자열 A: “ABCDEFG”
문자열 B: “EF”

문자열 A	A	B	C	D	E	F	G
문자열 B					E	F	

KMP 문자열 매칭

단순 비교 문자열 매칭 구현하기

```c
#include <stdio.h>
#include <string.h>

char *parent = "ABCDEFG";
char *pattern = "EF";

int main(void) {
  int parentSize = strlen(parent);
  int patternSize = strlen(pattern);
  for (int i = 0; i <= parentSize - patternSize; i++) {
    int found = 1;
    for (int j = 0; j < patternSize; j++) {
      if (parent[i + j] != pattern[j]) { found = 0; break; }
    }
    if (found) {
      printf("%d번째에서 찾았습니다.\n", i + 1);
    }
  }
  system("pause");
}
```

KMP 문자열 매칭

KMP 문자열 매칭

1) 단순 비교 문자열 매칭 알고리즘은 $O(NM)$의 시간 복잡도로 상당히 비효율적입니다.
2) KMP 문자열 매칭 알고리즘을 활용하면 $O(N + M)$의 시간 복잡도로 문제를 해결할 수 있습니다.

Fast campus

KMP 문자열 매칭

접두사와 접미사

KMP 알고리즘은 접두사와 접미사를 활용해 빠르게 문자열 매칭을 수행하는 알고리즘입니다.

KMP 문자열 매칭

접두사와 접미사

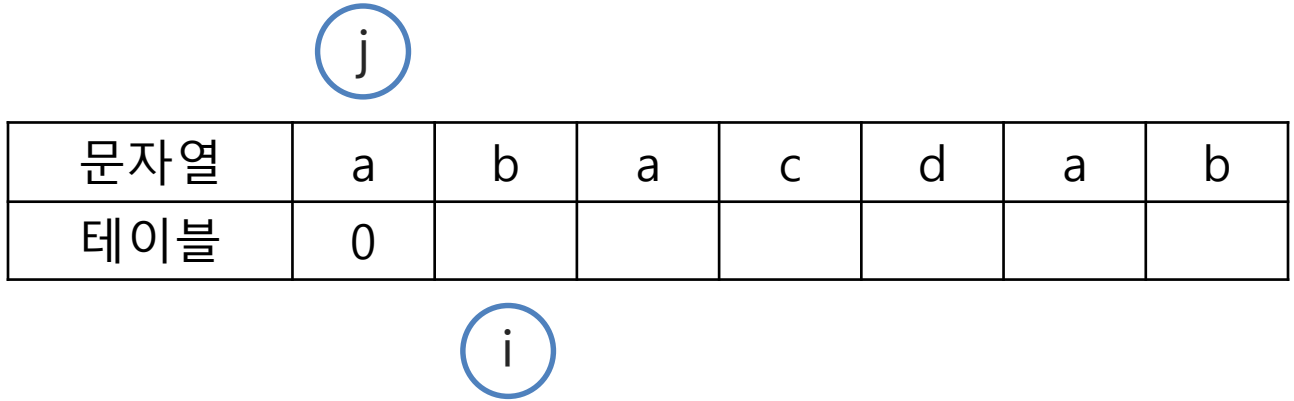
특정한 부모 문자열에서 찾고자 하는 패턴(Pattern) 문자열이 “abacdab”라고 가정해 봅시다. 이 때 각 길이에 따른 접두사와 접미사가 일치하는 최대 길이를 구합니다.

길이	문자열	최대 일치 길이
1	a	0
2	ab	0
3	aba	1
4	abac	0
5	abacd	0
6	abacda	1
7	abacdab	2

KMP 문자열 매칭

테이블 만들기

j = 0, i = 1로 설정하고 테이블 만들기를 진행합니다.



KMP 문자열 매칭

테이블 만들기

pattern[j]와 pattern[i]가 일치하는지 반복적으로 확인합니다. 일치하는 경우 j에 1을 더한 뒤에 j의 인덱스를 table[i]에 삽입합니다.

j

문자열	a	b	a	c	d	a	b
테이블	0						

i

KMP 문자열 매칭

테이블 만들기

pattern[j]와 pattern[i]가 일치하는지 반복적으로 확인합니다. 일치하는 경우 j에 1을 더한 뒤에 j의 인덱스를 table[i]에 삽입합니다.

j

문자열	a	b	a	c	d	a	b
테이블	0	0					

i

KMP 문자열 매칭

테이블 만들기

pattern[j]와 pattern[i]가 일치하는지 반복적으로 확인합니다. 일치하는 경우 j에 1을 더한 뒤에 j의 인덱스를 table[i]에 삽입합니다.

j

문자열	a	b	a	c	d	a	b
테이블	0	0					

i

KMP 문자열 매칭

테이블 만들기

pattern[j]와 pattern[i]가 일치하는지 반복적으로 확인합니다. 일치하는 경우 j에 1을 더한 뒤에 j의 인덱스를 table[i]에 삽입합니다.

j

문자열	a	b	a	c	d	a	b
테이블	0	0	1				

i

KMP 문자열 매칭

테이블 만들기

불일치하는 경우 j를 '마지막으로 일치했던 순간'까지의 인덱스로 이동해 다시 검사합니다. '마지막으로 일치했던 순간'은 table[j - 1]입니다.

문자열	a	b	a	c	d	a	b
테이블	0	0	1				

KMP 문자열 매칭

테이블 만들기

불일치하는 경우 j를 '마지막으로 일치했던 순간'까지의 인덱스로 이동해 다시 검사합니다. '마지막으로 일치했던 순간'은 table[j - 1]입니다.

j

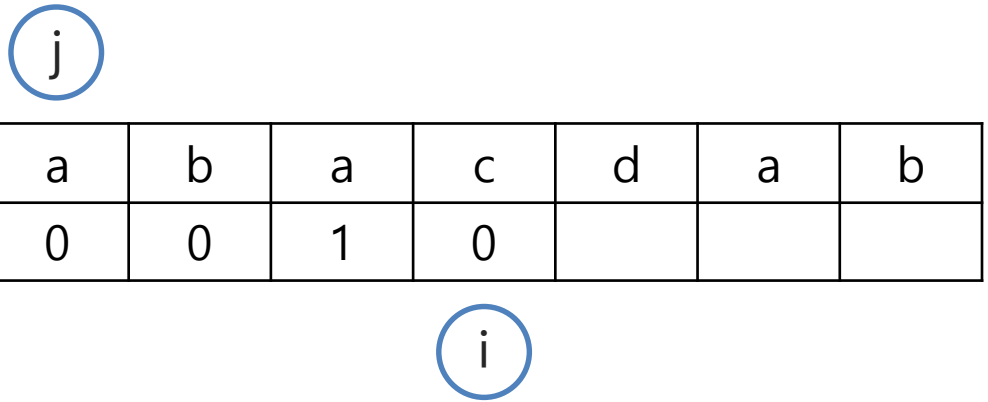
문자열	a	b	a	c	d	a	b
테이블	0	0	1				

i

KMP 문자열 매칭

테이블 만들기

이러한 과정을 마지막 문자까지 반복합니다.



KMP 문자열 매칭

테이블 만들기

이러한 과정을 마지막 문자까지 반복합니다.

j

문자열	a	b	a	c	d	a	b
테이블	0	0	1	0			

i

KMP 문자열 매칭

테이블 만들기

이러한 과정을 마지막 문자까지 반복합니다.

j

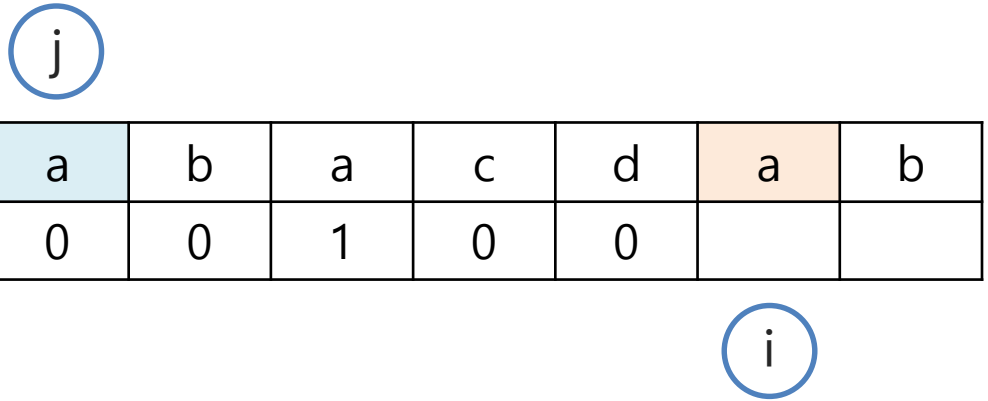
문자열	a	b	a	c	d	a	b
테이블	0	0	1	0	0		

i

KMP 문자열 매칭

테이블 만들기

이러한 과정을 마지막 문자까지 반복합니다.



KMP 문자열 매칭

테이블 만들기

이러한 과정을 마지막 문자까지 반복합니다.

j

문자열	a	b	a	c	d	a	b
테이블	0	0	1	0	0	1	

i

KMP 문자열 매칭

테이블 만들기

이러한 과정을 마지막 문자까지 반복합니다.

j

문자열	a	b	a	c	d	a	b
테이블	0	0	1	0	0	1	

i

KMP 문자열 매칭

테이블 만들기

완성된 테이블은 다음과 같습니다.

문자열	a	b	a	c	d	a	b
테이블	0	0	1	0	0	1	2

Fast campus

KMP 문자열 매칭

테이블 만들기: 문자열 정의

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char *parent = "acabacdabac";
char *pattern = "abacdab";
```

KMP 문자열 매칭

테이블 만들기: 테이블 생성 함수 구현하기

```
int* makeTable(char* pattern) {
  int patternSize = strlen(pattern);
  int* table = (int*)malloc(sizeof(int) * patternSize);
  for (int i = 0; i < patternSize; i++) {
    table[i] = 0;
  }
  int j = 0;
  for (int i = 1; i < patternSize; i++) {
    while (j > 0 && pattern[i] != pattern[j]) {
      j = table[j - 1];
    }
    if (pattern[i] == pattern[j]) {
      table[i] = ++j;
    }
  }
  return table;
}
```

KMP 문자열 매칭

문자열 매칭 진행하기

실제로 문자열 매칭을 진행할 때에도 불일치하는 경우 j를 '마지막으로 일치했던 순간'까지의 인덱스로 이동해 다시 검사합니다. '마지막으로 일치했던 순간'은 table[j - 1]입니다.

테이블	0	0	1	0	0	1	2

부모 문자열	a	c	a	b	a	c	d	a	b	a	c
패턴 문자열	a	b	a	c	d	a	b				

KMP 문자열 매칭

문자열 매칭 진행하기

실제로 문자열 매칭을 진행할 때에도 불일치하는 경우 j를 '마지막으로 일치했던 순간'까지의 인덱스로 이동해 다시 검사합니다. '마지막으로 일치했던 순간'은 table[j - 1]입니다.

테이블	0	0	1	0	0	1	2

부모 문자열	a	c	a	b	a	c	d	a	b	a	c
패턴 문자열	a	b	a	c	d	a	b				

KMP 문자열 매칭

문자열 매칭 진행하기

실제로 문자열 매칭을 진행할 때에도 불일치하는 경우 j를 '마지막으로 일치했던 순간'까지의 인덱스로 이동해 다시 검사합니다. '마지막으로 일치했던 순간'은 table[j - 1]입니다.

테이블	0	0	1	0	0	1	2

부모 문자열	a	c	a	b	a	c	d	a	b	a	c
패턴 문자열	a	b	a	c	d	a	b				

KMP 문자열 매칭

문자열 매칭 진행하기

실제로 문자열 매칭을 진행할 때에도 불일치하는 경우 j를 '마지막으로 일치했던 순간'까지의 인덱스로 이동해 다시 검사합니다. '마지막으로 일치했던 순간'은 table[j - 1]입니다.

테이블	0	0	1	0	0	1	2

부모 문자열	a	c	a	b	a	c	d	a	b	a	c
패턴 문자열	a	b	a	c	d	a	b				

KMP 문자열 매칭

문자열 매칭 진행하기

실제로 문자열 매칭을 진행할 때에도 불일치하는 경우 j를 '마지막으로 일치했던 순간'까지의 인덱스로 이동해 다시 검사합니다. '마지막으로 일치했던 순간'은 table[j - 1]입니다.

테이블	0	0	1	0	0	1	2

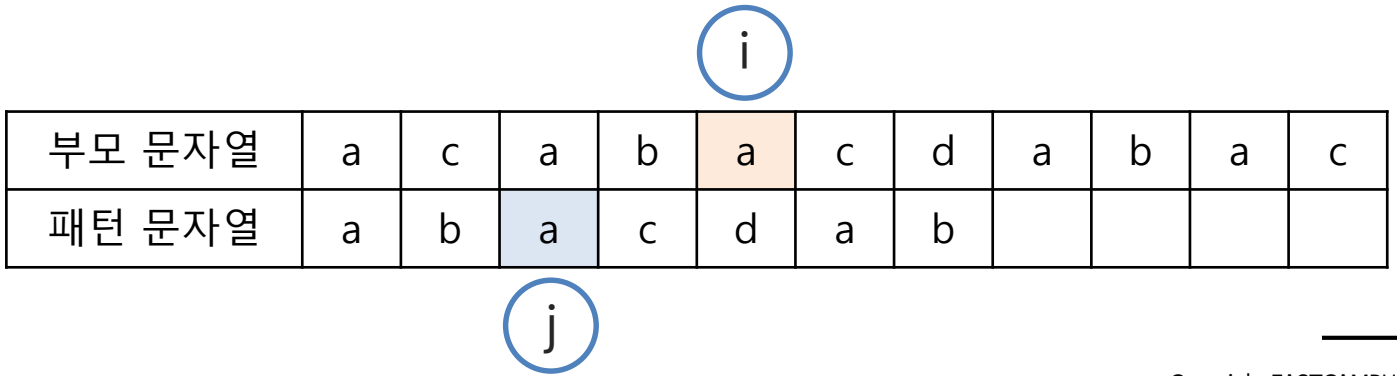
부모 문자열	a	c	a	b	a	c	d	a	b	a	c
패턴 문자열	a	b	a	c	d	a	b				

KMP 문자열 매칭

문자열 매칭 진행하기

실제로 문자열 매칭을 진행할 때에도 불일치하는 경우 j를 '마지막으로 일치했던 순간'까지의 인덱스로 이동해 다시 검사합니다. '마지막으로 일치했던 순간'은 table[j - 1]입니다.

테이블	0	0	1	0	0	1	2

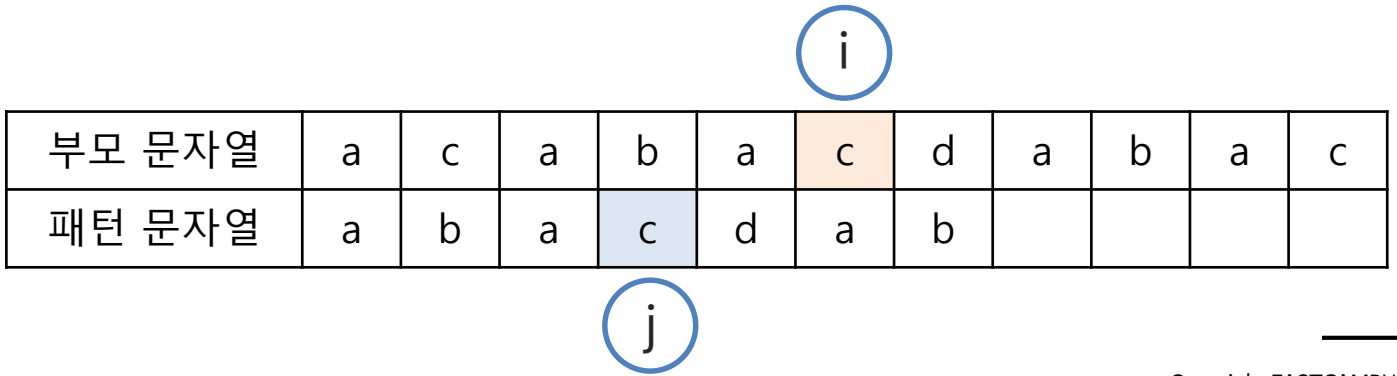


KMP 문자열 매칭

문자열 매칭 진행하기

실제로 문자열 매칭을 진행할 때에도 불일치하는 경우 j를 '마지막으로 일치했던 순간'까지의 인덱스로 이동해 다시 검사합니다. '마지막으로 일치했던 순간'은 table[j - 1]입니다.

테이블	0	0	1	0	0	1	2

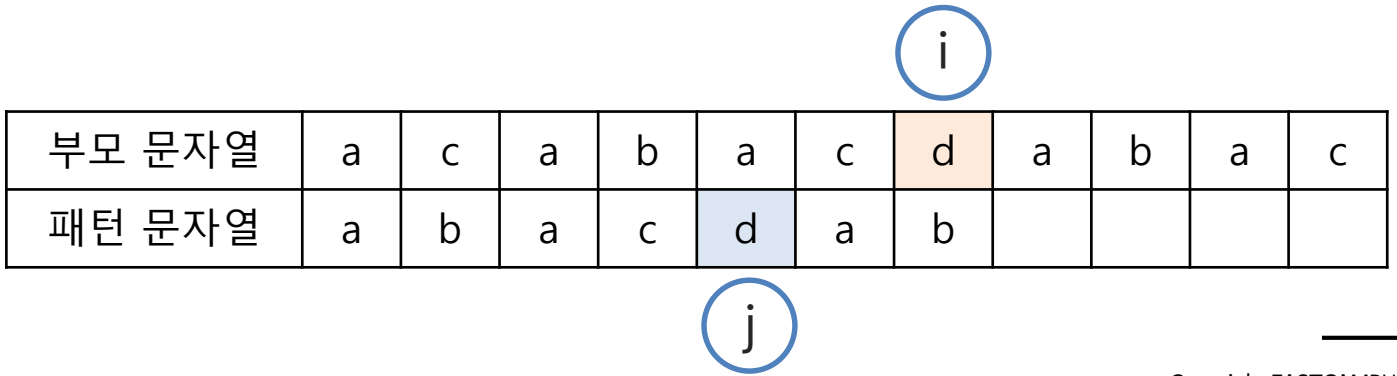


KMP 문자열 매칭

문자열 매칭 진행하기

실제로 문자열 매칭을 진행할 때에도 불일치하는 경우 j를 '마지막으로 일치했던 순간'까지의 인덱스로 이동해 다시 검사합니다. '마지막으로 일치했던 순간'은 table[j - 1]입니다.

테이블	0	0	1	0	0	1	2

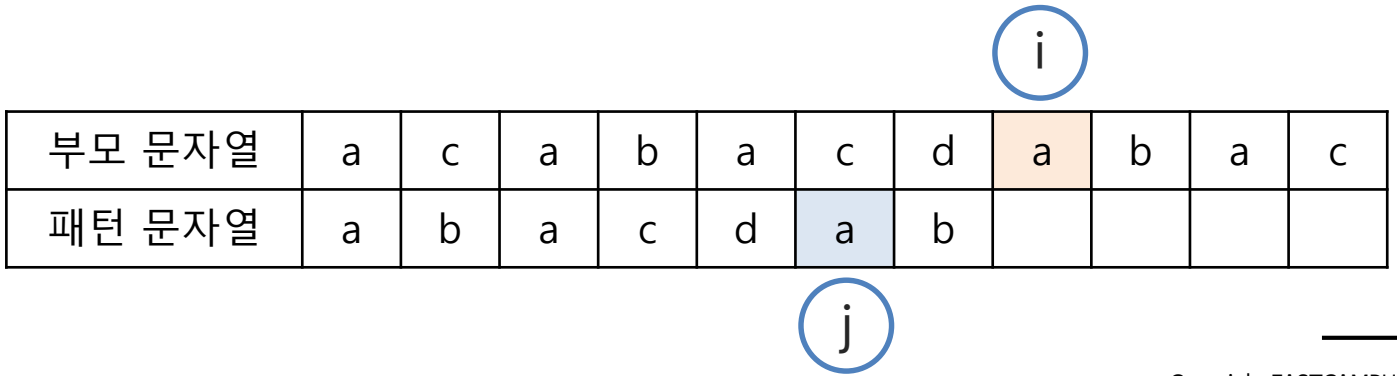


KMP 문자열 매칭

문자열 매칭 진행하기

실제로 문자열 매칭을 진행할 때에도 불일치하는 경우 j를 '마지막으로 일치했던 순간'까지의 인덱스로 이동해 다시 검사합니다. '마지막으로 일치했던 순간'은 table[j - 1]입니다.

테이블	0	0	1	0	0	1	2

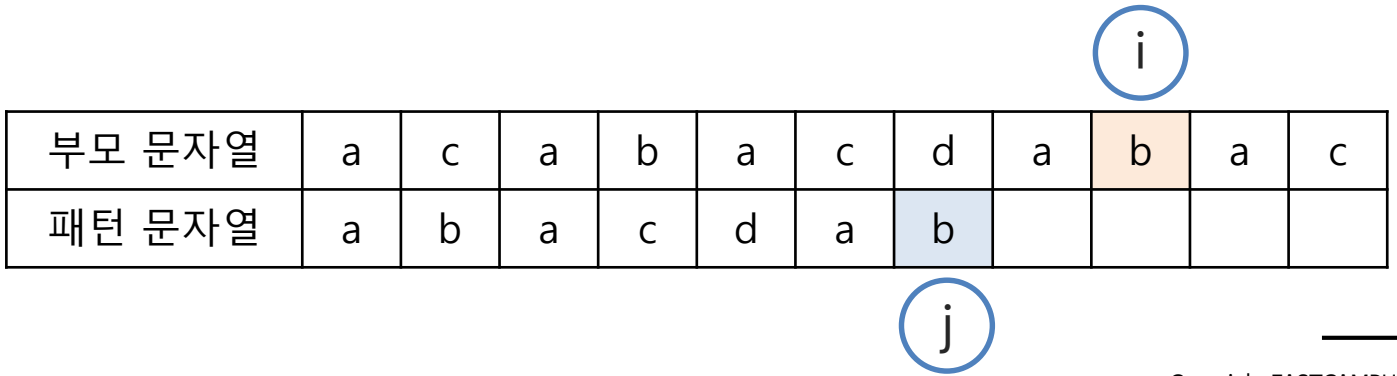


KMP 문자열 매칭

문자열 매칭 진행하기

매칭에 성공했습니다.

테이블	0	0	1	0	0	1	2

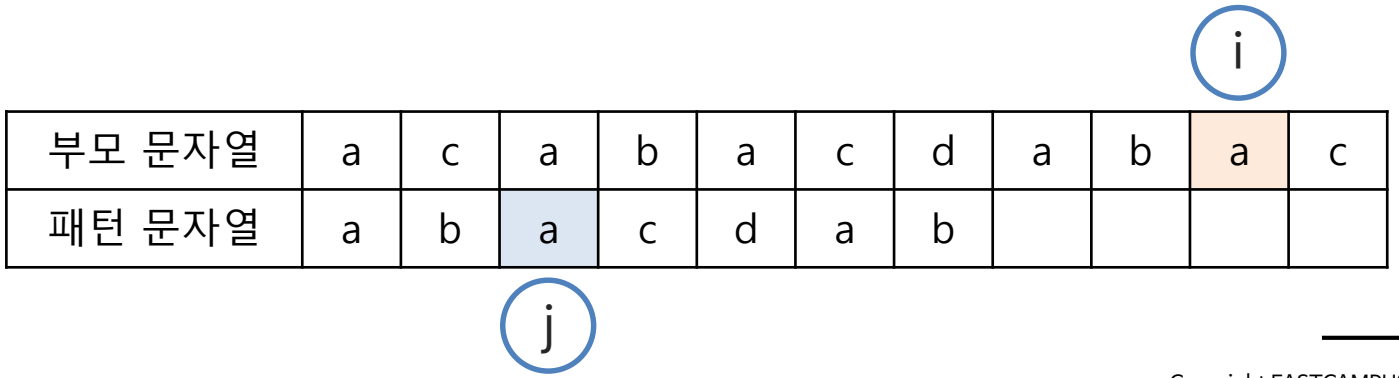


KMP 문자열 매칭

문자열 매칭 진행하기

매칭에 성공한 경우에는 table[j]의 위치로 이동하여 다시 검사하면 됩니다.

테이블	0	0	1	0	0	1	2

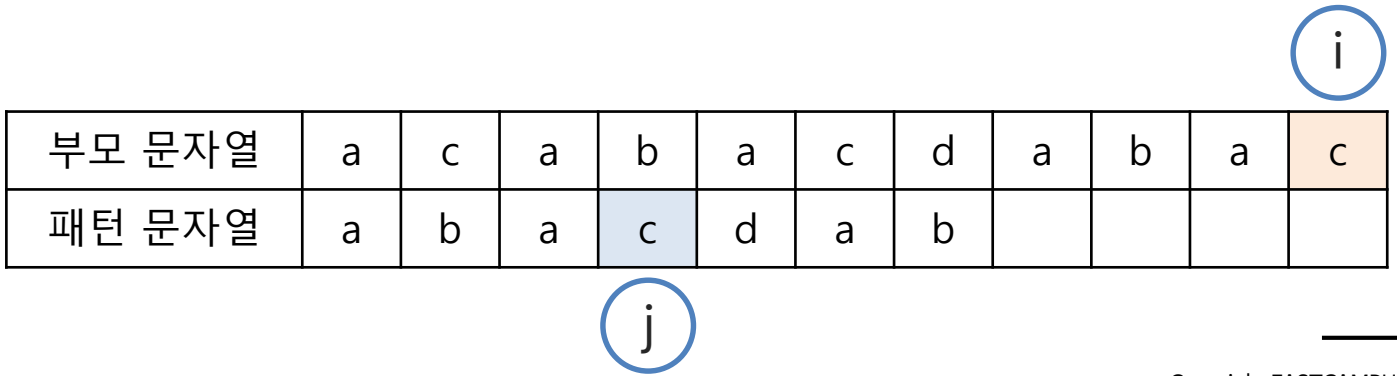


KMP 문자열 매칭

문자열 매칭 진행하기

매칭에 성공한 경우에는 table[j]의 위치로 이동하여 다시 검사하면 됩니다.

테이블	0	0	1	0	0	1	2



KMP 문자열 매칭

문자열 매칭 진행하기

결과적으로 한 번의 매칭이 발생한 사실을 알 수 있습니다.

테이블	0	0	1	0	0	1	2

부모 문자열	a	c	a	b	a	c	d	a	b	a	c
패턴 문자열	a	b	a	c	d	a	b				

KMP 문자열 매칭

문자열 매칭 진행하기: KMP 함수 구현하기

```
void KMP(char* parent, char* pattern) {
  int* table = makeTable(pattern);
  int parentSize = strlen(parent);
  int patternSize = strlen(pattern);
  int j = 0;
  for (int i = 0; i < parentSize; i++) {
    while (j > 0 && parent[i] != pattern[j]) {
      j = table[j - 1];
    }
    if (parent[i] == pattern[j]) {
      if (j == patternSize - 1) {
        printf("[인덱스 %d]에서 매칭 성공\n", i - patternSize + 2);
        j = table[j];
      }
      else {
        j++;
      }
    }
  }
}
```

KMP 문자열 매칭

KMP 문자열 매칭 알고리즘 사용해보기

```
int main(void) {
  KMP(parent, pattern);
  system("pause");
}
```

배운 내용 정리하기

KMP 문자열 매칭

1) KMP 문자열 매칭 알고리즘은 $O(N + M)$의 시간 복잡도를 가집니다.