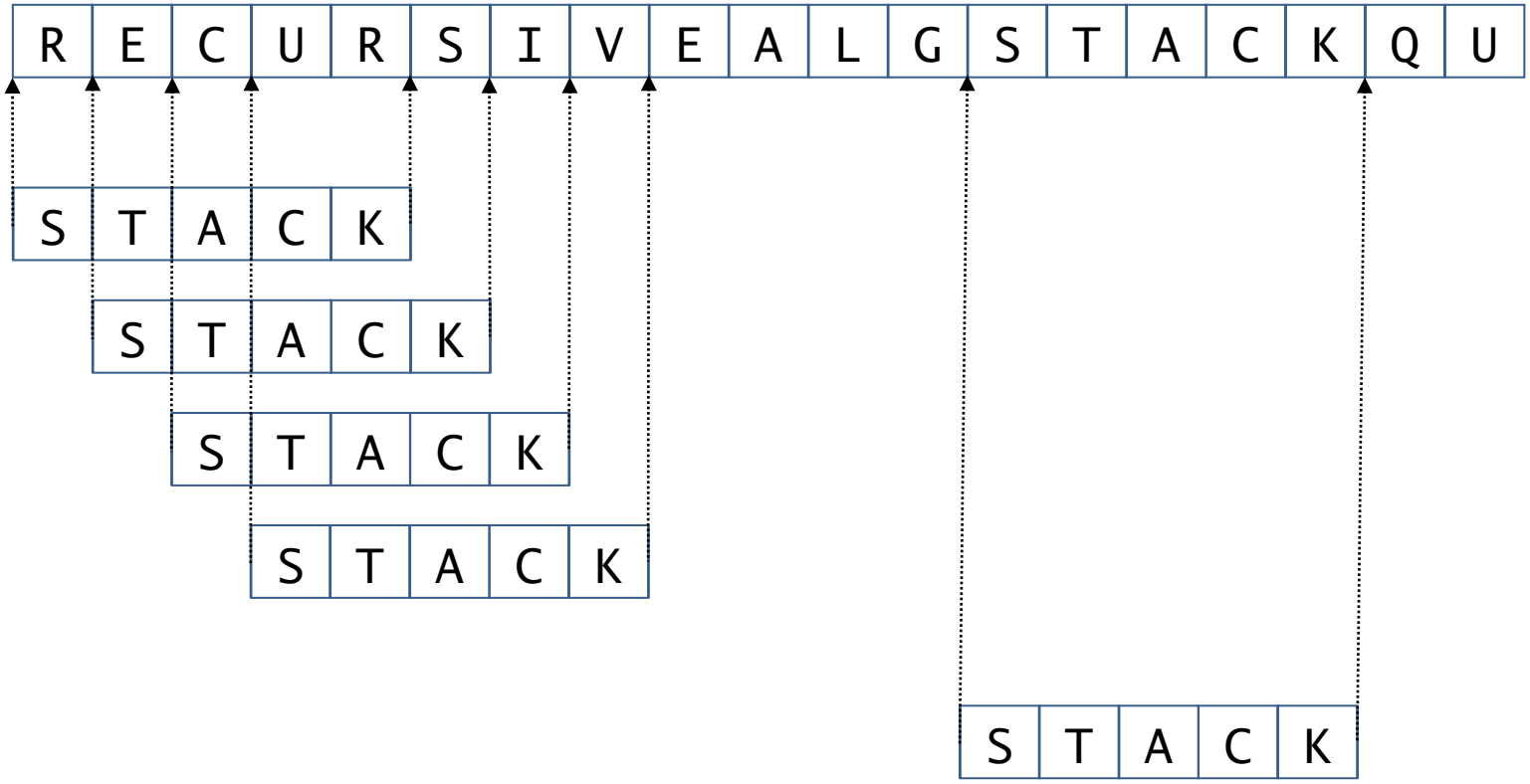
Ch. 32 String Matching : KMP algorithm

국민대학교 컴퓨터공학부 최준수
Modified by 임은진

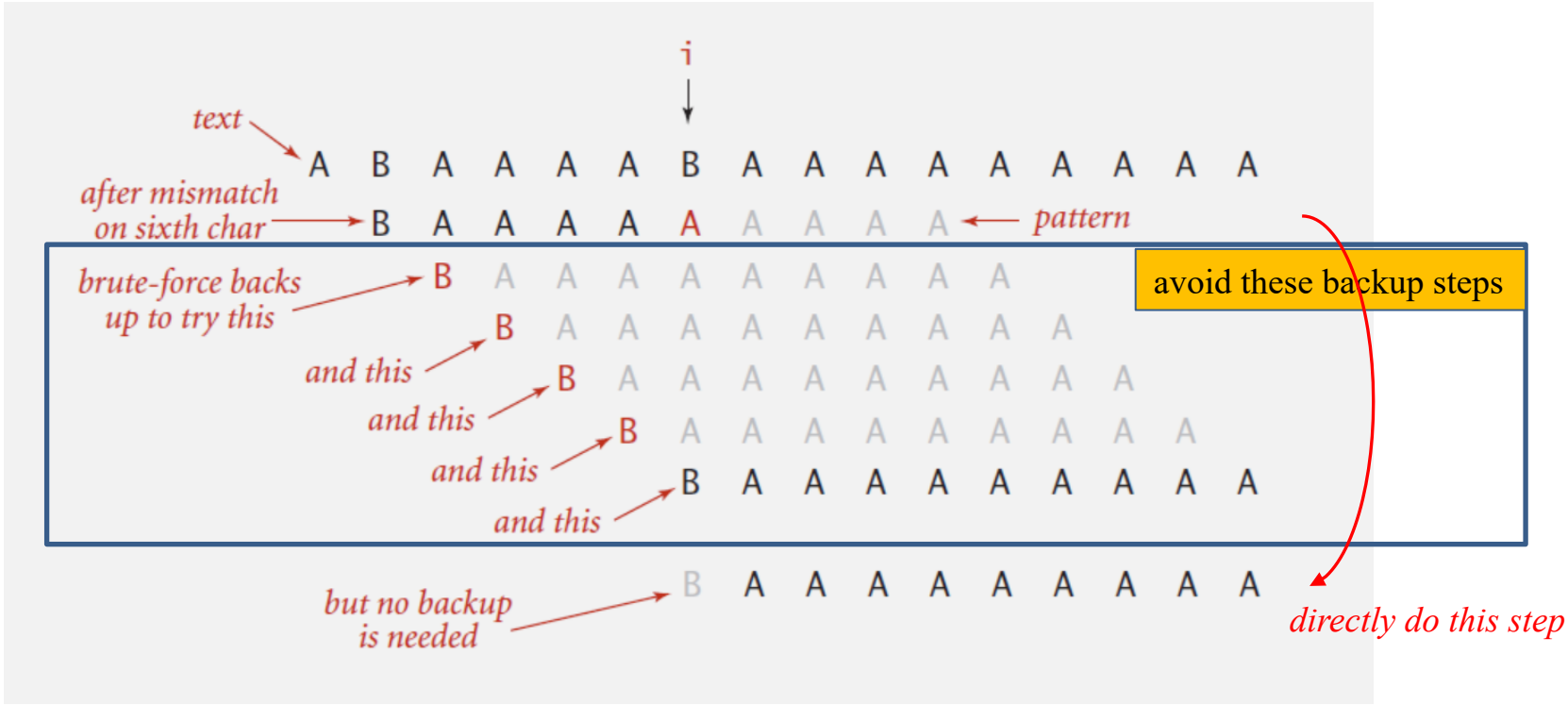
Brute-Force String Matching *O(MN)*

- Naïve Algorithm
  - Check for pattern starting at each text position



String Matching with DFA *O(N)*

– Clever method to avoid backup problem.

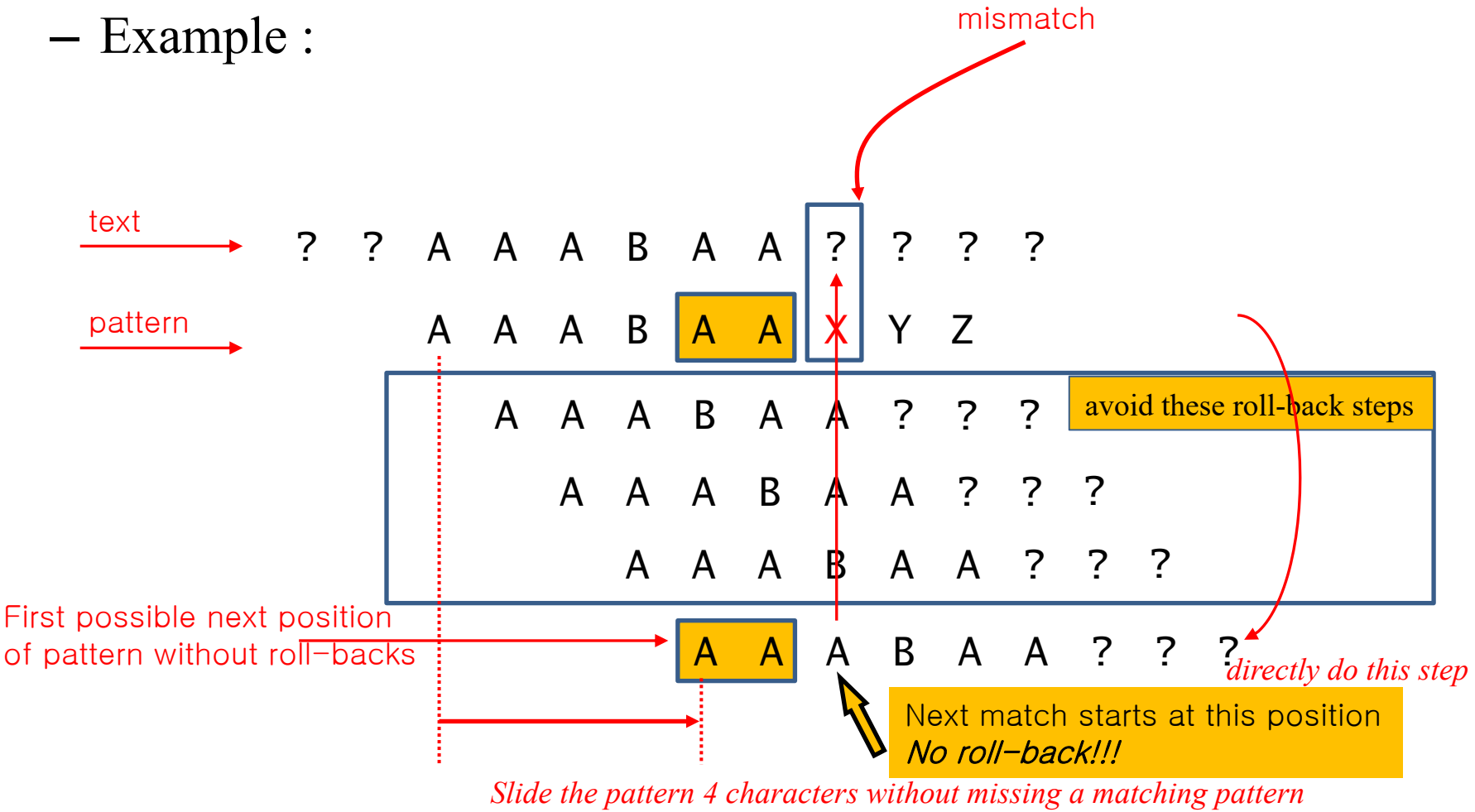


String Matching algorithms

Algorithm	Preprocessing time	Matching time
Naive	0	$O((n-m+1)m)$
Rabin-Karp	$\Theta(m)$	$O((n-m+1)m)$
Finite automaton	$O(m\lvert\Sigma\rvert)$	$\Theta(n)$
Knuth-Morris-Pratt	$\Theta(m)$	$\Theta(n)$

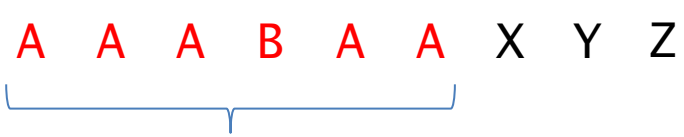
Knuth-Morris-Pratt(KMP) Algorithm

- Avoid roll-backs in naïve algorithm:
  - Example :



Prefix and Proper Suffix of the Prefix

pattern



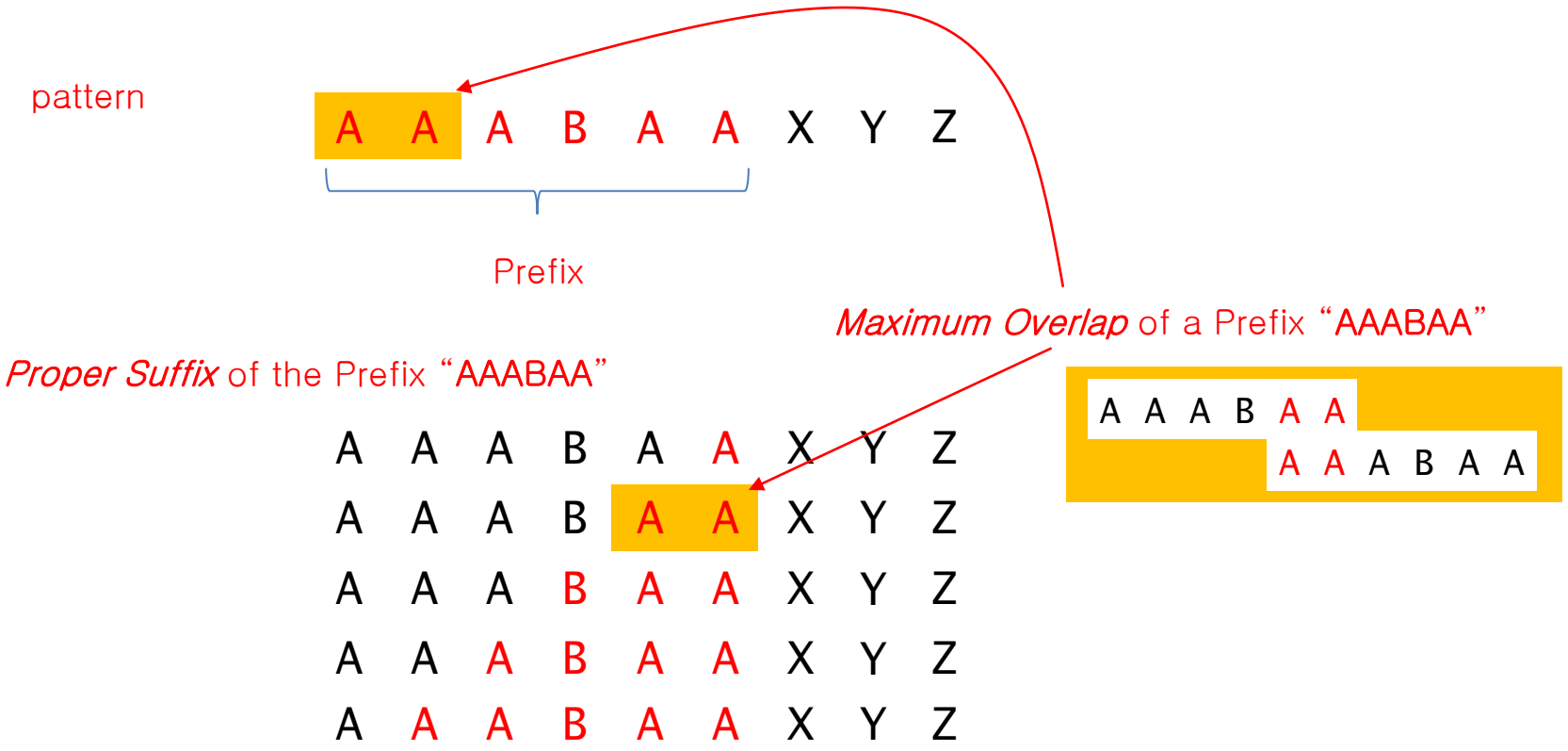
Proper Suffix of the Prefix "AAABAA"

A A A B A A X Y Z
A A A B A A X Y Z
A A A B A A X Y Z
A A A B A A X Y Z
A A A B A A X Y Z

A A A B A A X Y Z ← Not a *proper* suffix of the prefix (the same as the prefix)

Maximum Overlap of a Prefix

– the longest proper suffix that is equal to prefix of the prefix



Examples : Maximum Overlap of a Prefix

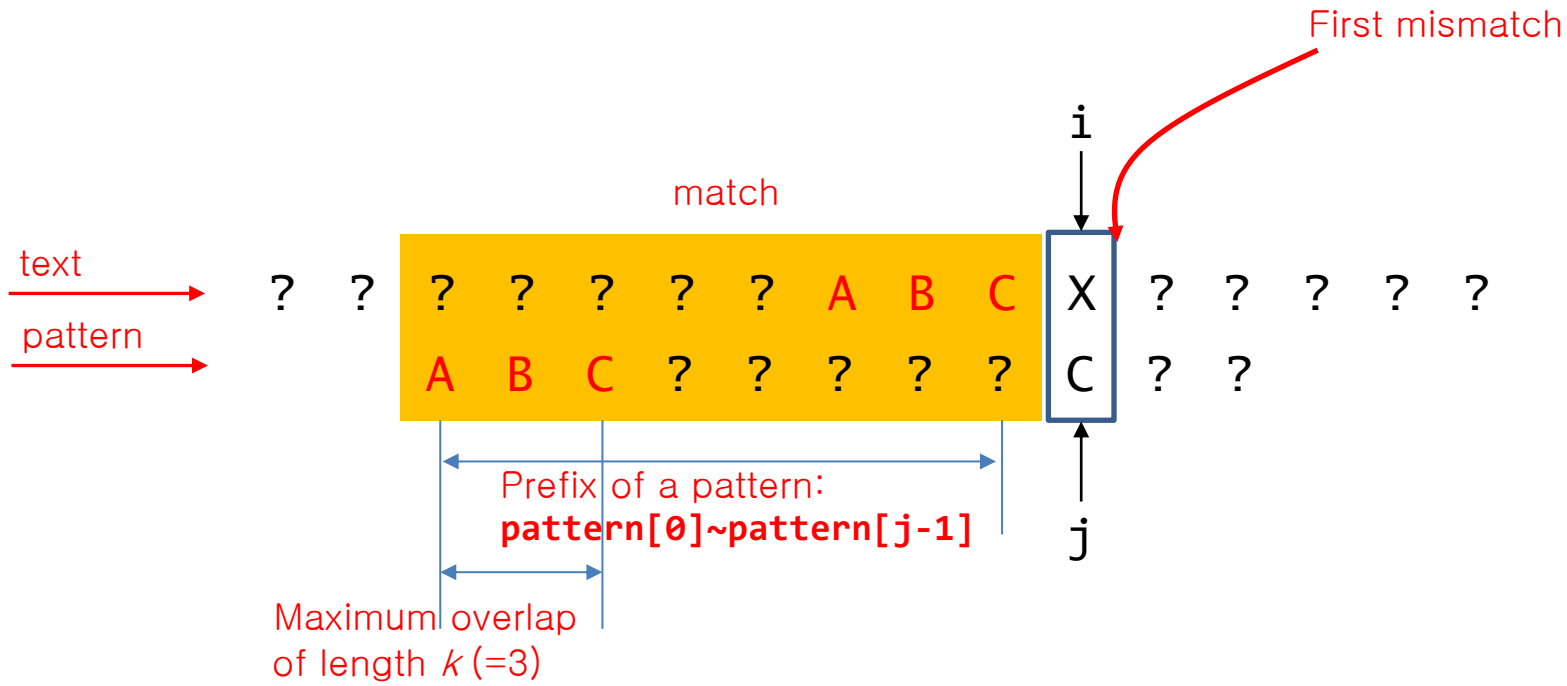
Prefix	Maximum Overlap
AAAAA	AAAA
AABA	A
AAAB	
ABABABAB	ABABAB

not **AAAAA**

NULL String

maximum overlap of a prefix in KMP Algorithm

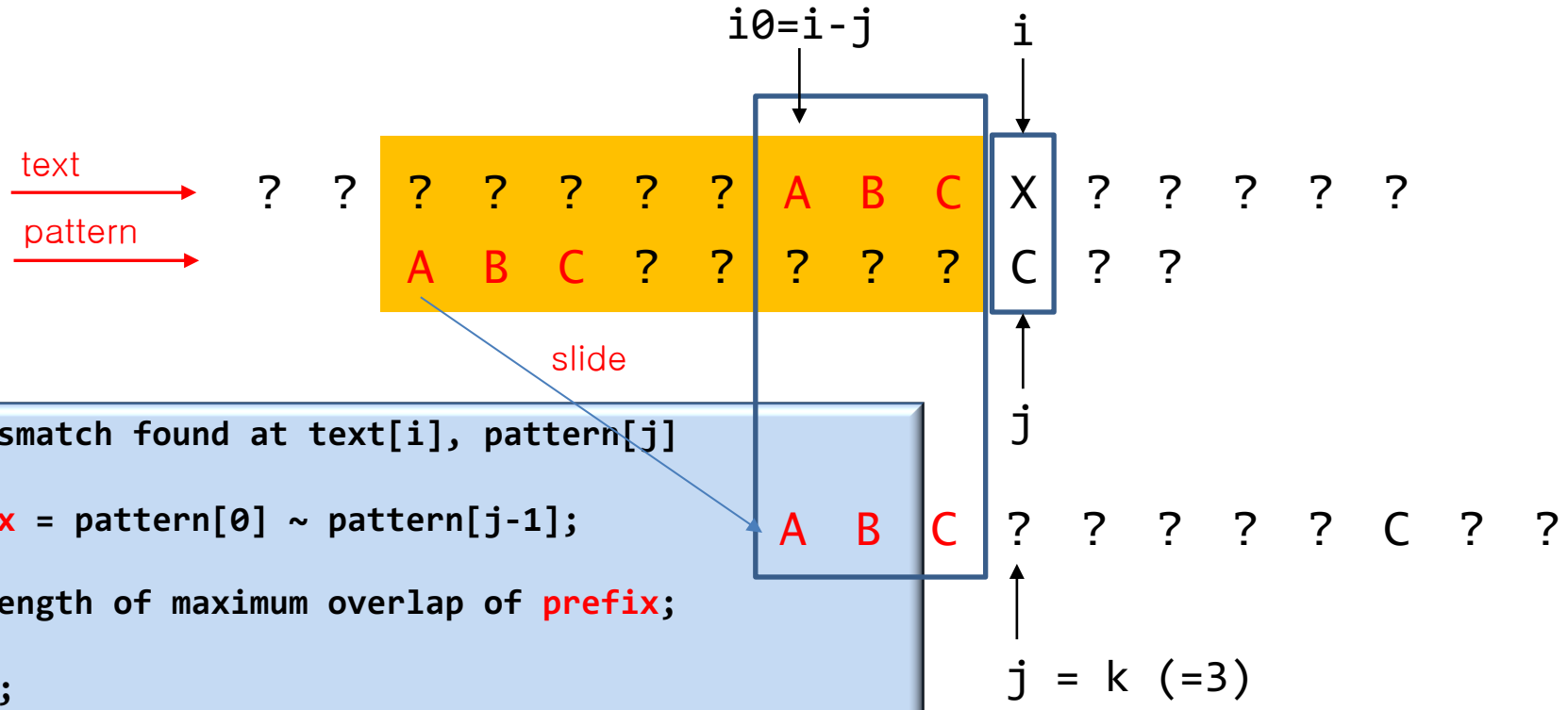
- Reuse of prefix information when there is a mismatch
  - Mismatch at `text[i]` and `pattern[j]`



Note that if the mismatched *location* is `pattern[j]`, then *prefix* is: `pattern[0]~pattern[j-1]`

KMP Algorithm

– Then we can slide the pattern so that the *suffix and prefix aligns without missing out on a match*:




```
// mismatch found at text[i], pattern[j]

prefix = pattern[0] ~ pattern[j-1];

k = Length of maximum overlap of prefix;

j = k;
// i is unchanged !

// Matched position i0 in text starts from (i - j);
i0 = i - j;
```


Algorithmic Engineering Laboratory

KMP Algorithm

- Fast sliding algorithm:

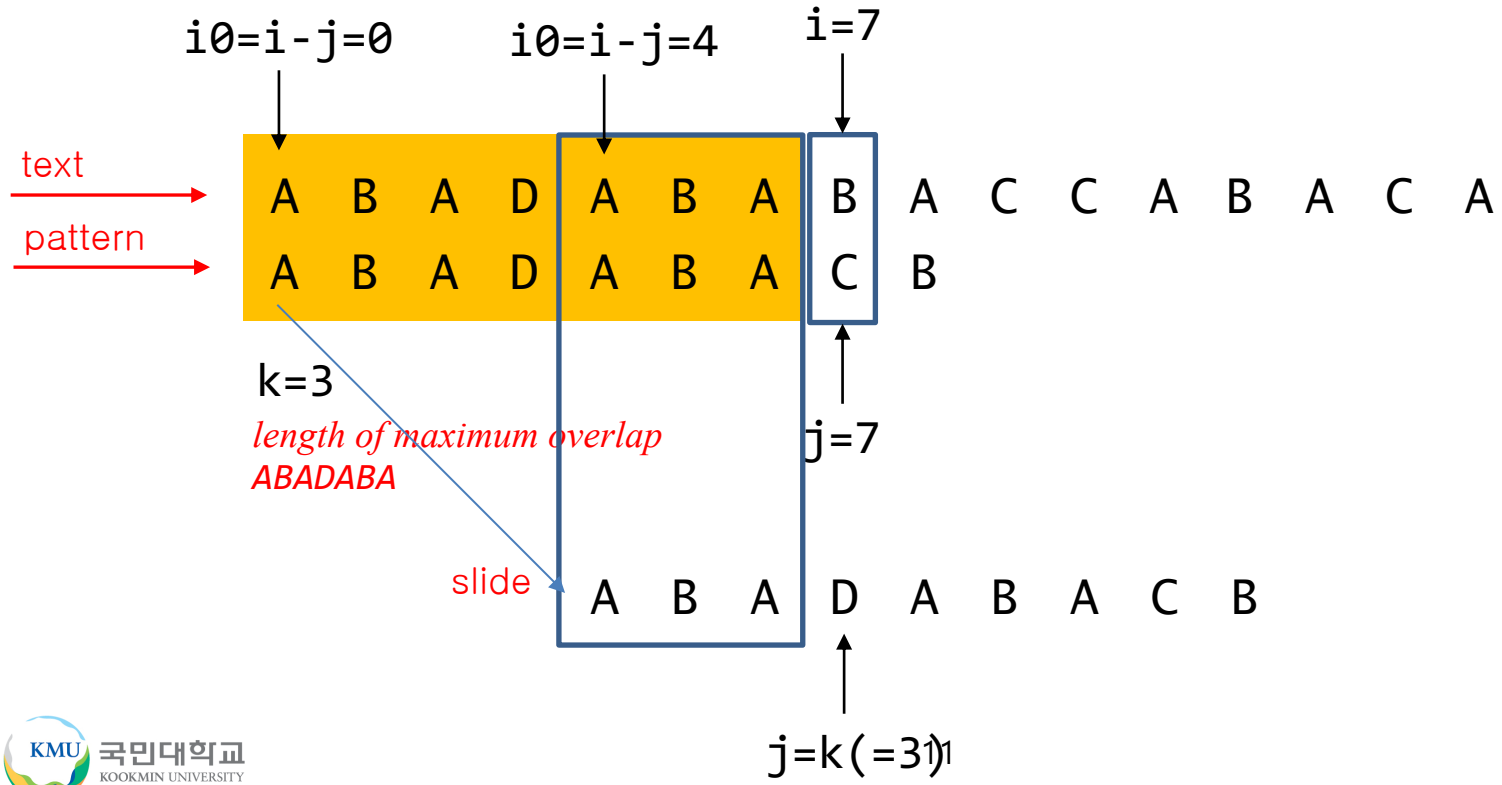
```
// mismatch found at text[i], pattern[j]

prefix = pattern[0] ~ pattern[j-1];

k = Length of maximum overlap of prefix;

j = k;
// i is unchanged !

// Matched position i0 in text starts from (i - j);
i0 = i - j;
```



failure function in KMP Algorithm

- Failure function:
  - M : the length of a pattern
  - For $0 < k < M$, the failure function $fail(k)$ is the length of maximum overlap of a prefix `pattern[0]` ~ `pattern[k]`
    - Note that $fail(0) = 0$

banabana

k	prefix	$fail(k)$
0	b	0
1	ba	0
2	ban	0
3	bana	0
4	banab	1
5	banaba	2
6	banaban	3
7	banabana	4

KMP Algorithm

- Knuth-Morris-Pratt(KMP) Algorithm

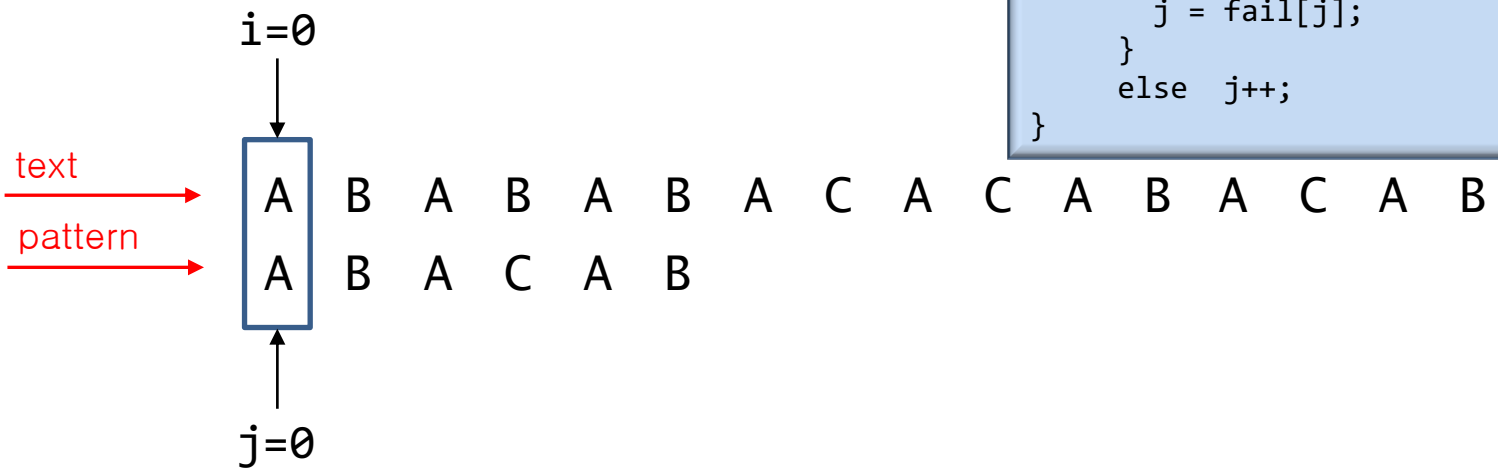
```
vector<int> kmp(string text, string pattern)
{
    vector<int> ans;
    fail = getFail(pattern);        // failure function
    int n = (int) text.size(), m = (int) pattern.size();
    int j =0;                       // j : index of pattern

    for(int i = 0 ; i < n ; i++)    // i : index of text
    {
        while(j>0 && text[i] != pattern[j])
            j = fail[j-1];
        if(text[i] == pattern[j])
            if(j==m-1)              // pattern matching is found
            {
                ans.push_back(i-j); // save the matched position
                j = fail[j];
            }
            else
                j++;
    }
    return ans;
}
```

KMP Algorithm

– Example:

```
for(int i = 0 ; i < n ; i++){
  while(j>0 && text[i] != pattern[j])
     j = fail[j-1];
  if(text[i] == pattern[j])
     if(j==m-1){
       ans.push_back(i-j);
       j = fail[j];
     }
     else  j++;
}
```



text[i] == pattern[j]

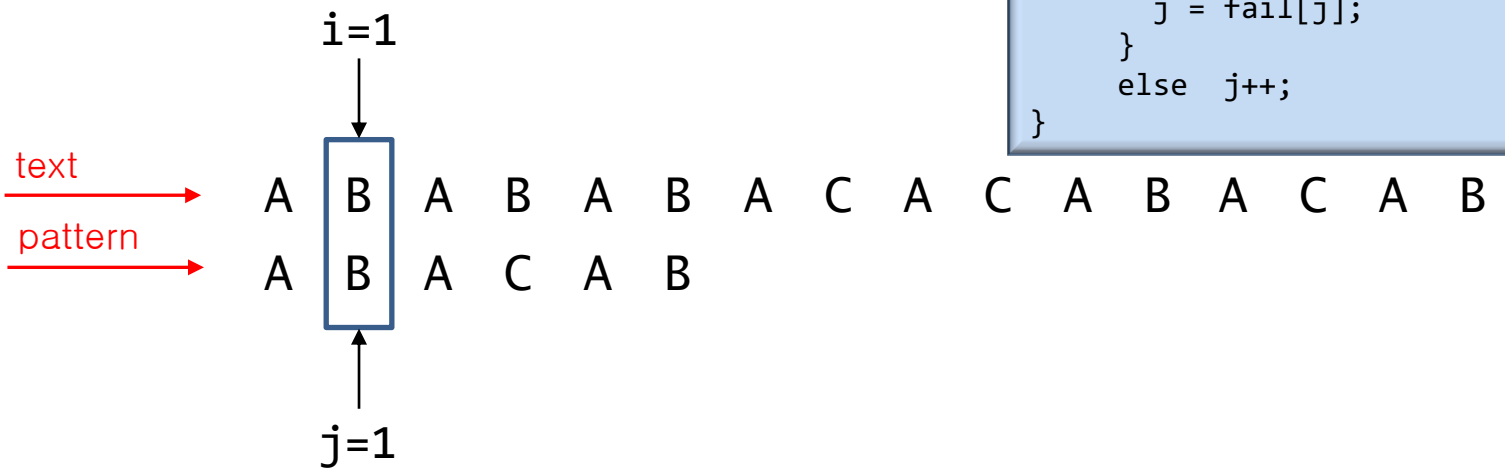
➔ i++, j++

i	0	1	2	3	4	5
fail(i)	0	0	1	0	1	2

KMP Algorithm

– Example:

```
for(int i = 0 ; i < n ; i++){
  while(j>0 && text[i] != pattern[j])
     j = fail[j-1];
  if(text[i] == pattern[j])
     if(j==m-1){
       ans.push_back(i-j);
       j = fail[j];
     }
     else  j++;
}
```



text[i] == pattern[j]

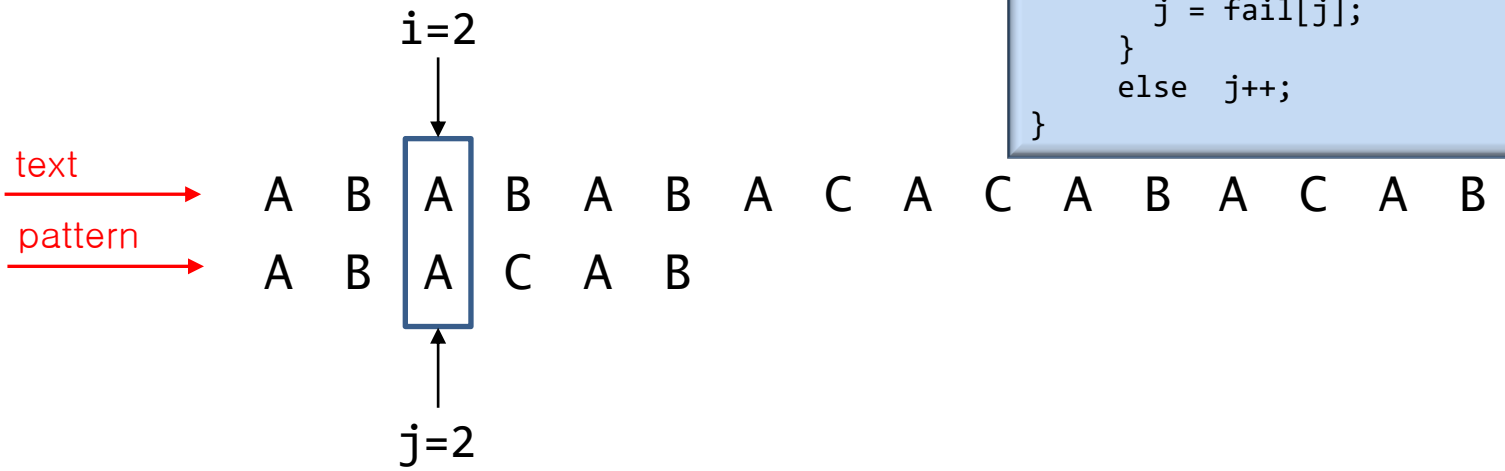
➔ i++, j++

i	0	1	2	3	4	5
fail(i)	0	0	1	0	1	2

KMP Algorithm

– Example:

```
for(int i = 0 ; i < n ; i++){
  while(j>0 && text[i] != pattern[j])
     j = fail[j-1];
  if(text[i] == pattern[j])
     if(j==m-1){
       ans.push_back(i-j);
       j = fail[j];
     }
     else  j++;
}
```



text[i] == pattern[j]

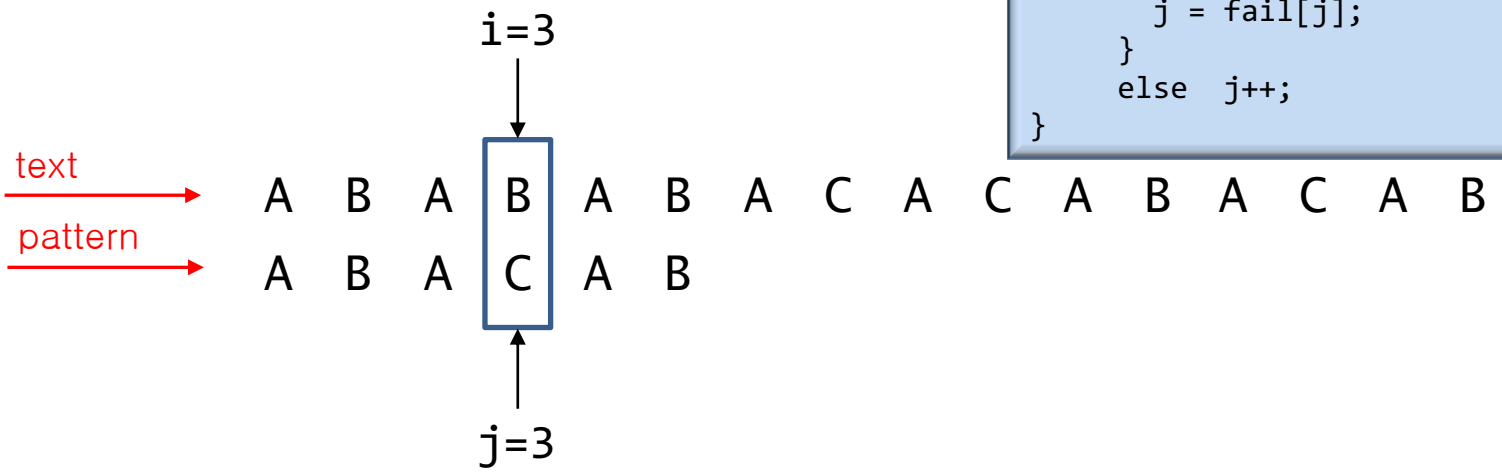
➔ i++, j++

i	0	1	2	3	4	5
fail(i)	0	0	1	0	1	2

KMP Algorithm

– Example:

```
for(int i = 0 ; i < n ; i++){
  while(j>0 && text[i] != pattern[j])
     j = fail[j-1];
  if(text[i] == pattern[j])
     if(j==m-1){
       ans.push_back(i-j);
       j = fail[j];
     }
     else  j++;
}
```



text[i] != pattern[j]

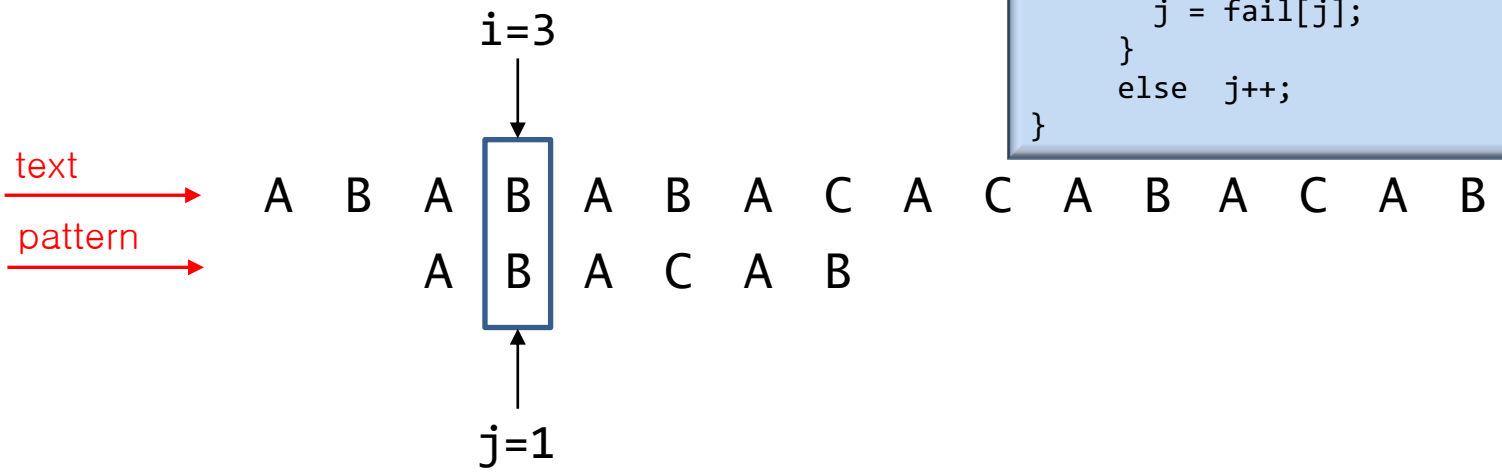
➔ j = fail[j-1]

i	0	1	2	3	4	5
fail(i)	0	0	1	0	1	2

KMP Algorithm

– Example:

```
for(int i = 0 ; i < n ; i++){
  while(j>0 && text[i] != pattern[j])
     j = fail[j-1];
  if(text[i] == pattern[j])
     if(j==m-1){
       ans.push_back(i-j);
       j = fail[j];
     }
     else  j++;
}
```



text[i] == pattern[j]

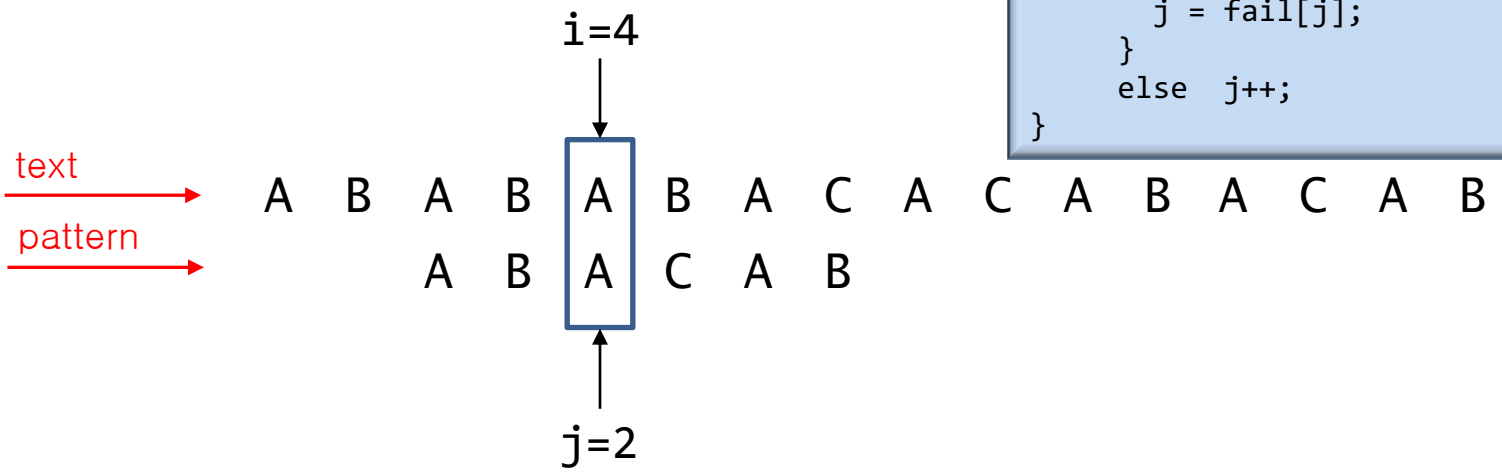
➔ i++, j++

i	0	1	2	3	4	5
fail(i)	0	0	1	0	1	2

KMP Algorithm

– Example:

```
for(int i = 0 ; i < n ; i++){
  while(j>0 && text[i] != pattern[j])
     j = fail[j-1];
  if(text[i] == pattern[j])
     if(j==m-1){
       ans.push_back(i-j);
       j = fail[j];
     }
     else  j++;
}
```



text[i] == pattern[j]

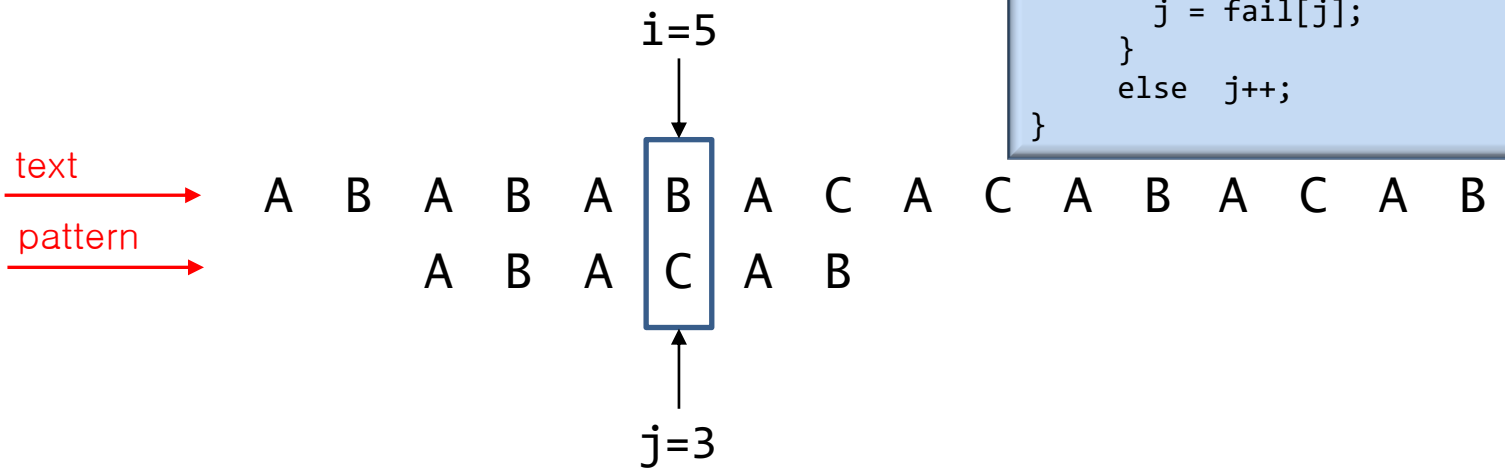
➔ i++, j++

i	0	1	2	3	4	5
fail(i)	0	0	1	0	1	2

KMP Algorithm

– Example:

```
for(int i = 0 ; i < n ; i++){
  while(j>0 && text[i] != pattern[j])
     j = fail[j-1];
  if(text[i] == pattern[j])
     if(j==m-1){
       ans.push_back(i-j);
       j = fail[j];
     }
     else  j++;
}
```



text[i] != pattern[j]

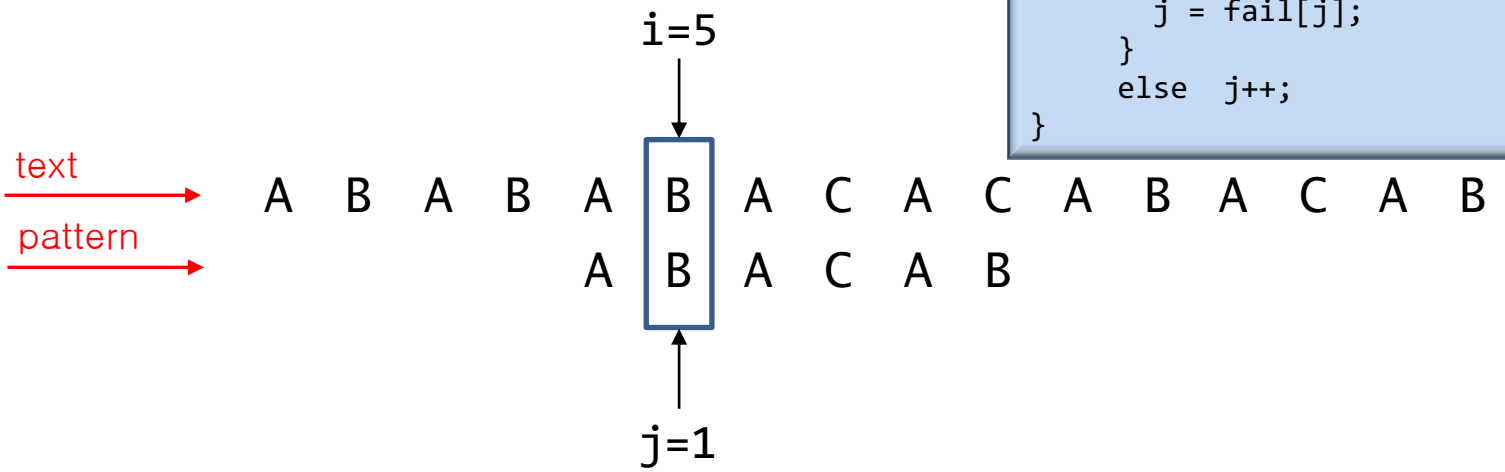
➔ j = fail[j-1]

i	0	1	2	3	4	5
fail(i)	0	0	1	0	1	2

KMP Algorithm

– Example:

```
for(int i = 0 ; i < n ; i++){
  while(j>0 && text[i] != pattern[j])
     j = fail[j-1];
  if(text[i] == pattern[j])
     if(j==m-1){
       ans.push_back(i-j);
       j = fail[j];
     }
     else  j++;
}
```



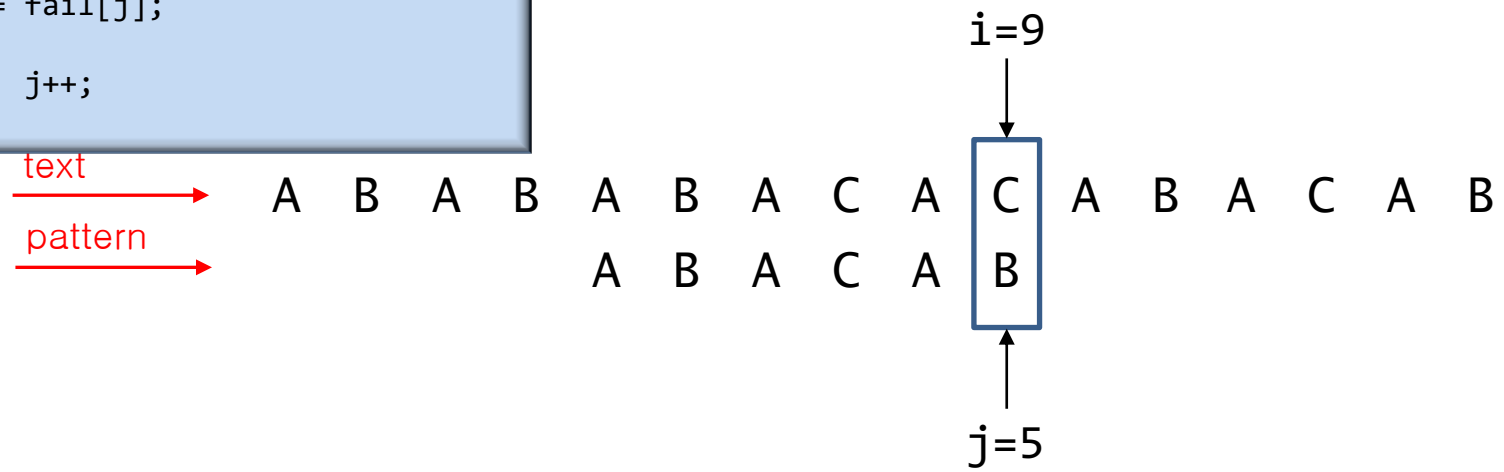
text[i] == pattern[j]

➔ i++, j++

i	0	1	2	3	4	5
fail(i)	0	0	1	0	1	2

KMP Algorithm

```
for(int i = 0 ; i < n ; i++){
  while(j>0 && text[i] != pattern[j])
     j = fail[j-1];
  if(text[i] == pattern[j])
     if(j==m-1){
       ans.push_back(i-j);
       j = fail[j];
     }
     else  j++;
}
```



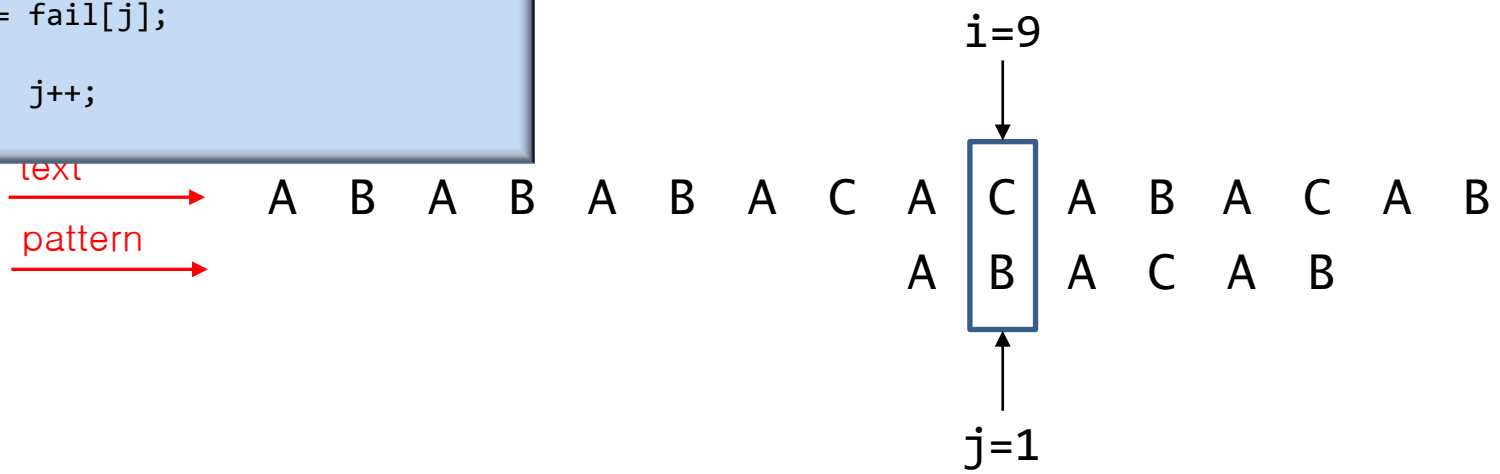
text[i] != pattern[j]

➔ j = fail[j-1]

i	0	1	2	3	4	5
fail(i)	0	0	1	0	1	2

KMP Algorithm

```
for(int i = 0 ; i < n ; i++){
  while(j>0 && text[i] != pattern[j])
     j = fail[j-1];
  if(text[i] == pattern[j])
     if(j==m-1){
       ans.push_back(i-j);
       j = fail[j];
     }
     else  j++;
}
```



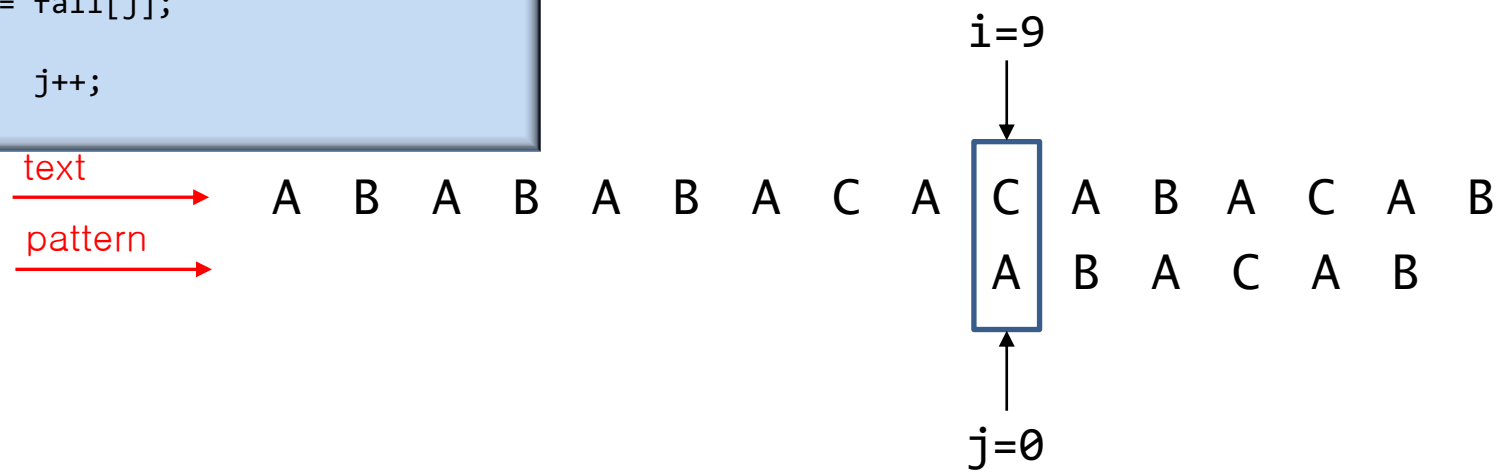
text[i] != pattern[j]

➔ j = fail[j-1]

i	0	1	2	3	4	5
fail(i)	0	0	1	0	1	2

KMP Algorithm

```
for(int i = 0 ; i < n ; i++){
  while(j>0 && text[i] != pattern[j])
     j = fail[j-1];
  if(text[i] == pattern[j])
     if(j==m-1){
       ans.push_back(i-j);
       j = fail[j];
     }
     else  j++;
}
```



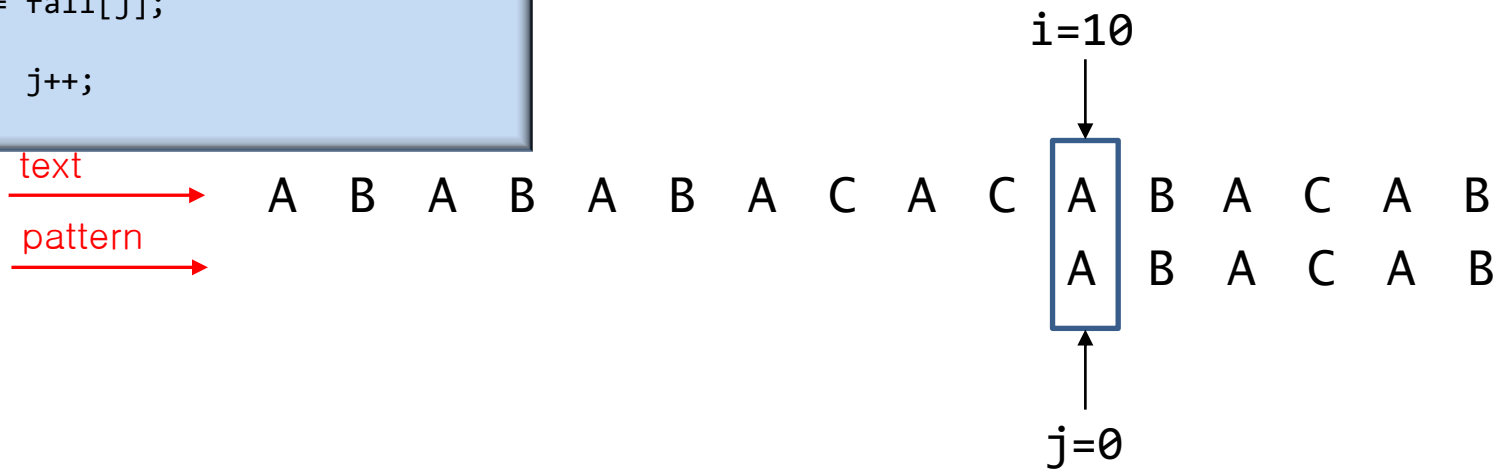
j==0 && text[i] != pattern[j]

➔ i++

i	0	1	2	3	4	5
fail(i)	0	0	1	0	1	2

KMP Algorithm

```
for(int i = 0 ; i < n ; i++){
  while(j>0 && text[i] != pattern[j])
     j = fail[j-1];
  if(text[i] == pattern[j])
     if(j==m-1){
       ans.push_back(i-j);
       j = fail[j];
     }
     else  j++;
}
```



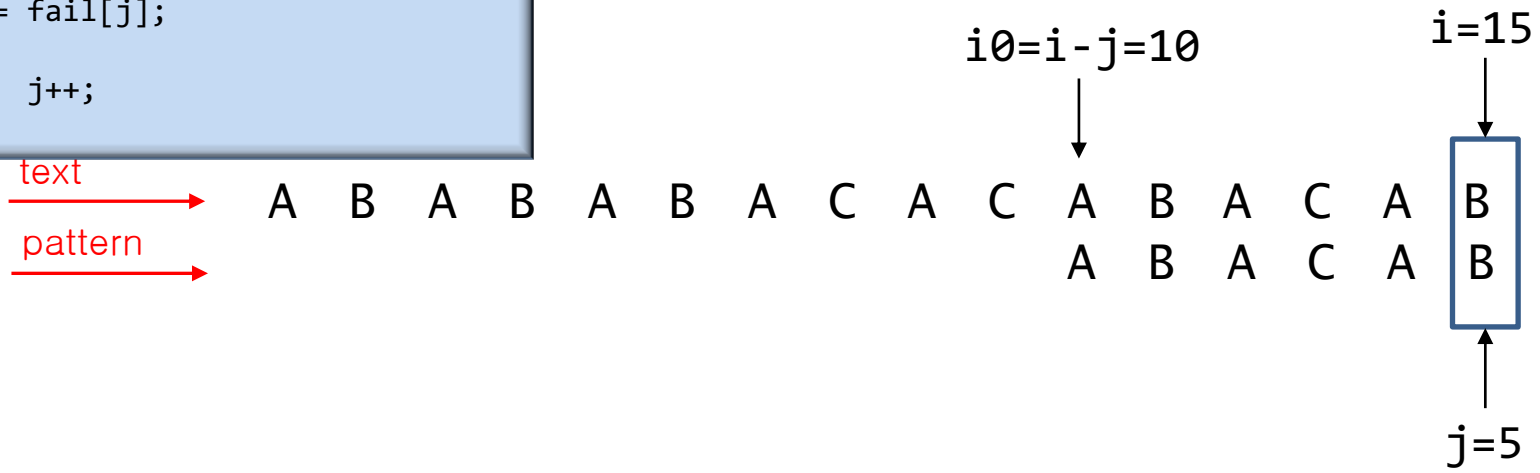
text[i] == pattern[j]

➔ i++, j++

i	0	1	2	3	4	5
fail(i)	0	0	1	0	1	2

KMP Algorithm

```
for(int i = 0 ; i < n ; i++){
  while(j>0 && text[i] != pattern[j])
     j = fail[j-1];
  if(text[i] == pattern[j])
     if(j==m-1){
       ans.push_back(i-j);
       j = fail[j];
     }
     else  j++;
}
```



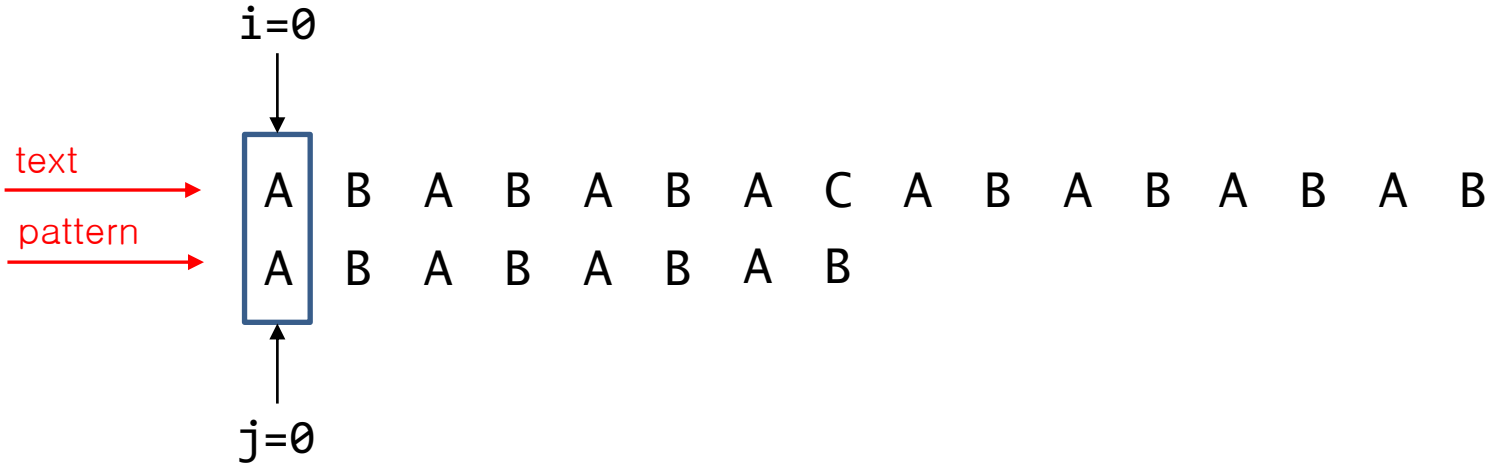
i	0	1	2	3	4	5
fail(i)	0	0	1	0	1	2

another example

KMP Algorithm

- Knuth-Morris-Pratt(KMP) Algorithm
  - Why?

```
while(j>0 && text[i] != pattern[j])
            j = fail[j-1];
```



text[i] == pattern[j]

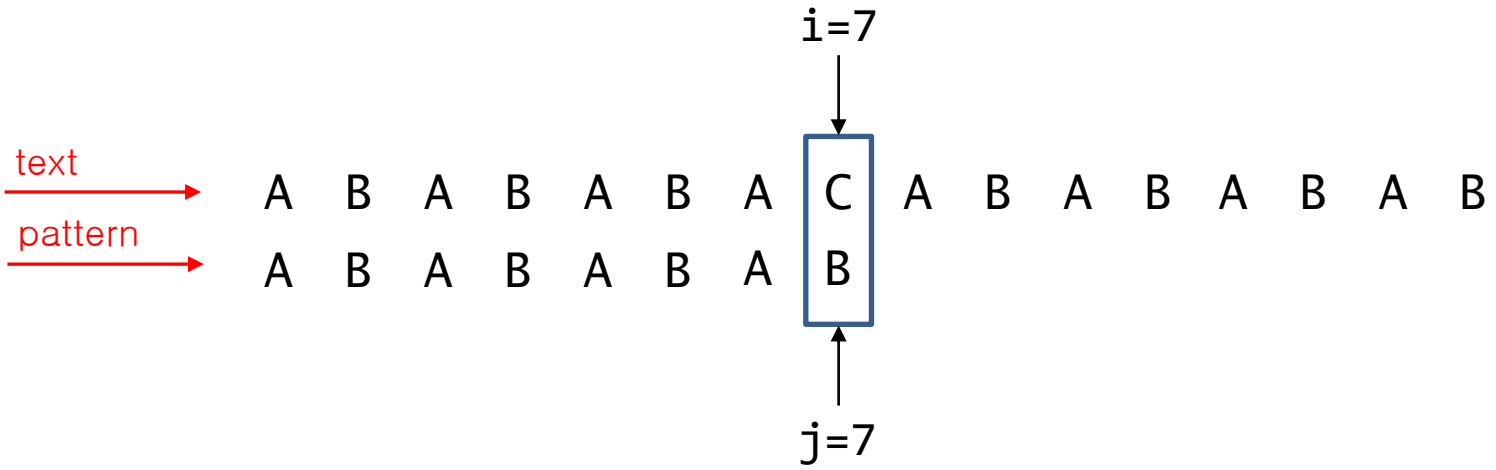
➔ i++, j++

i	0	1	2	3	4	5	6	7
fail(i)	0	0	1	2	3	4	5	6

KMP Algorithm

- Knuth-Morris-Pratt(KMP) Algorithm
  - Why?

```
while(j>0 && text[i] != pattern[j])
            j = fail[j-1];
```



text[i] != pattern[j]

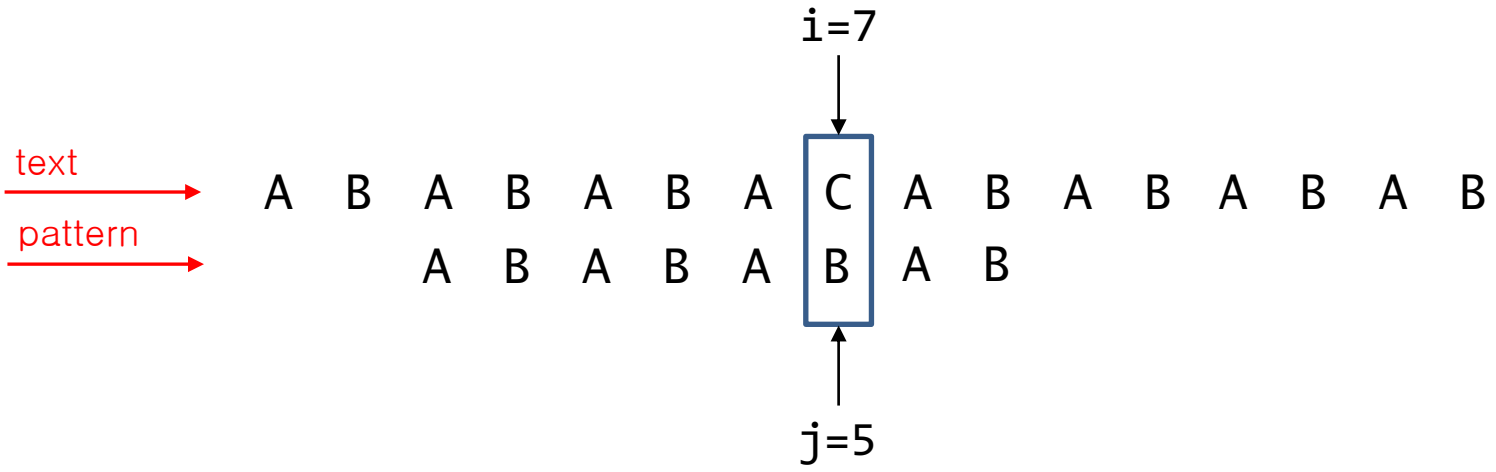
➔ j = fail[j-1]

i	0	1	2	3	4	5	6	7
fail(i)	0	0	1	2	3	4	5	6

KMP Algorithm

- Knuth-Morris-Pratt(KMP) Algorithm
  - Why?

```
while(j>0 && text[i] != pattern[j])
            j = fail[j-1];
```



text[i] != pattern[j]

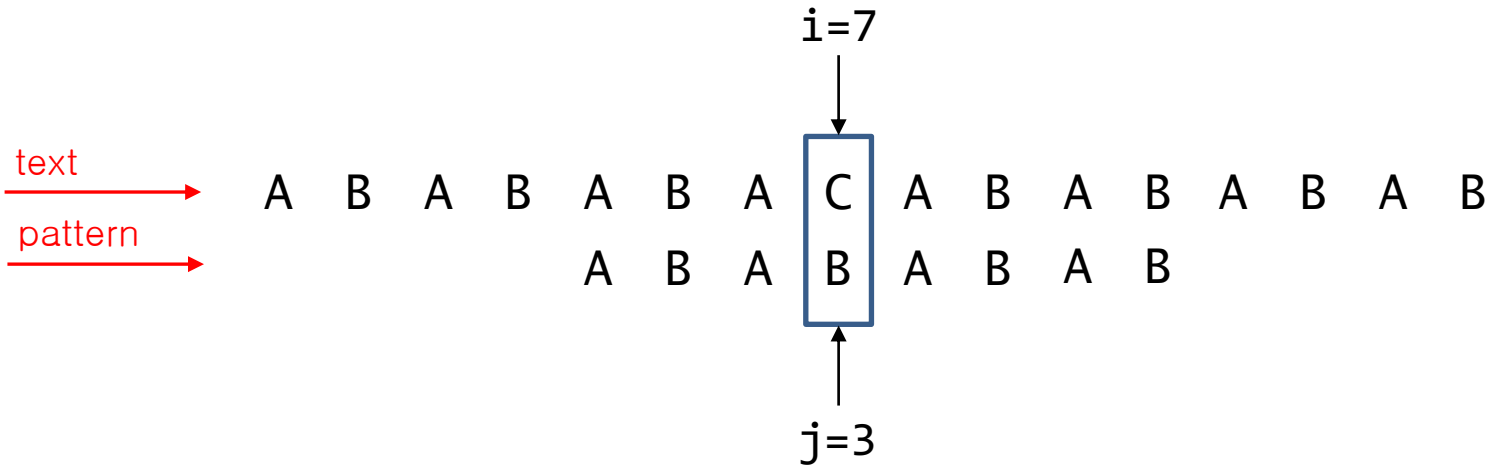
➔ j = fail[j-1]

i	0	1	2	3	4	5	6	7
fail(i)	0	0	1	2	3	4	5	6

KMP Algorithm

- Knuth-Morris-Pratt(KMP) Algorithm
  - Why?

```
while(j>0 && text[i] != pattern[j])
            j = fail[j-1];
```



text[i] != pattern[j]

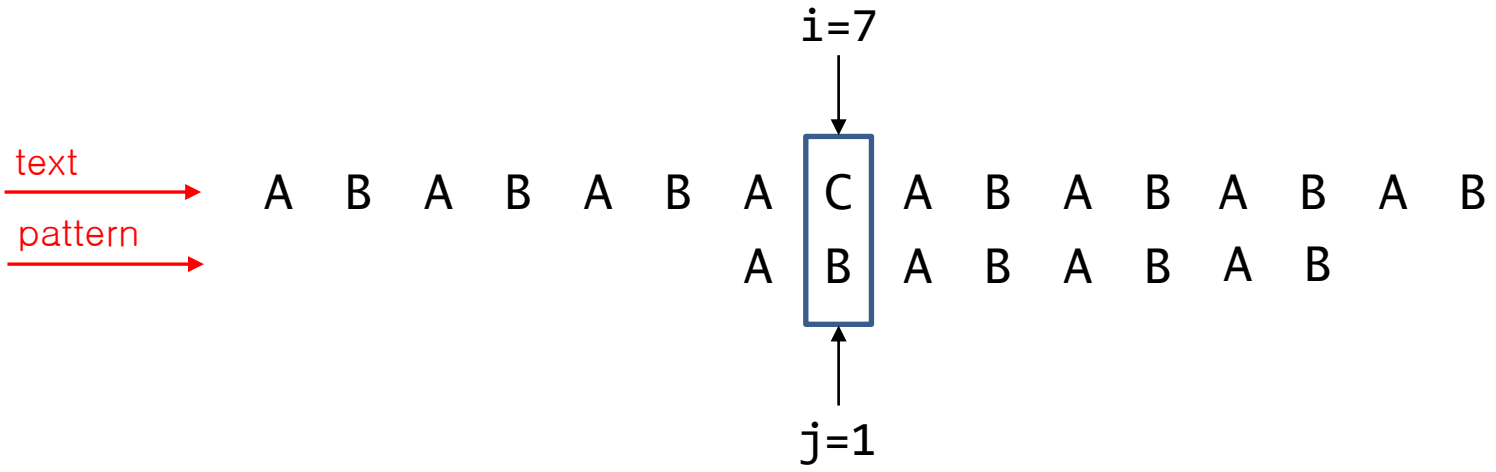
➔ j = fail[j-1]

i	0	1	2	3	4	5	6	7
fail(i)	0	0	1	2	3	4	5	6

KMP Algorithm

- Knuth-Morris-Pratt(KMP) Algorithm
  - Why?

```
while(j>0 && text[i] != pattern[j])
            j = fail[j-1];
```



text[i] != pattern[j]

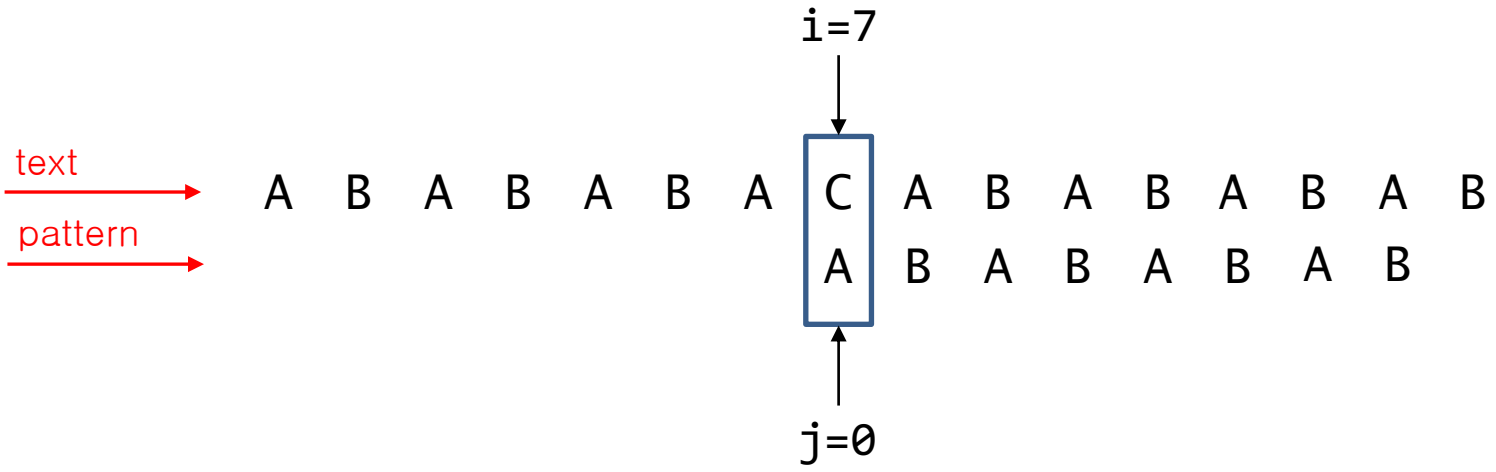
➔ j = fail[j-1]

i	0	1	2	3	4	5	6	7
fail(i)	0	0	1	2	3	4	5	6

KMP Algorithm

- Knuth-Morris-Pratt(KMP) Algorithm
  - Why?

```
while(j>0 && text[i] != pattern[j])
            j = fail[j-1];
```



text[i] != pattern[j]

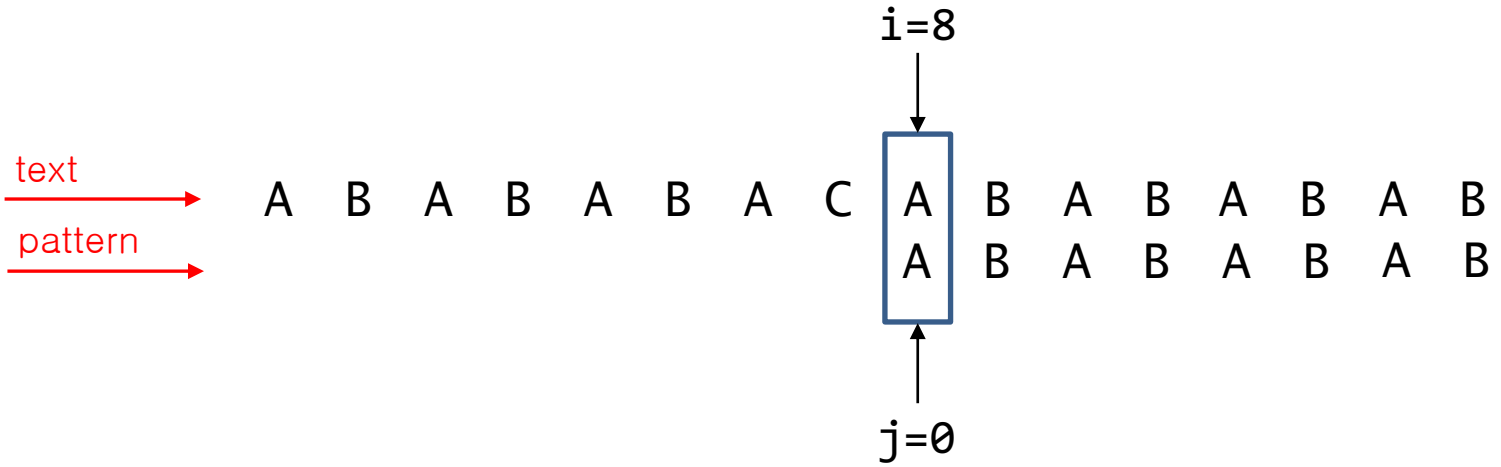
➔ j = fail[j-1]

i	0	1	2	3	4	5	6	7
fail(i)	0	0	1	2	3	4	5	6

KMP Algorithm

- Knuth-Morris-Pratt(KMP) Algorithm
  - Why?

```
while(j>0 && text[i] != pattern[j])
            j = fail[j-1];
```




```
j==0 && text[i] != pattern[j]
```

➔ i++

i	0	1	2	3	4	5	6	7
fail(i)	0	0	1	2	3	4	5	6

computation of fail[] (= π[])

- `getFail()` function
  - Simple `fail[]` computation needs $O(M^3)$ time.
  - $O(M)$ time algorithm: very similar to KMP algorithm itself

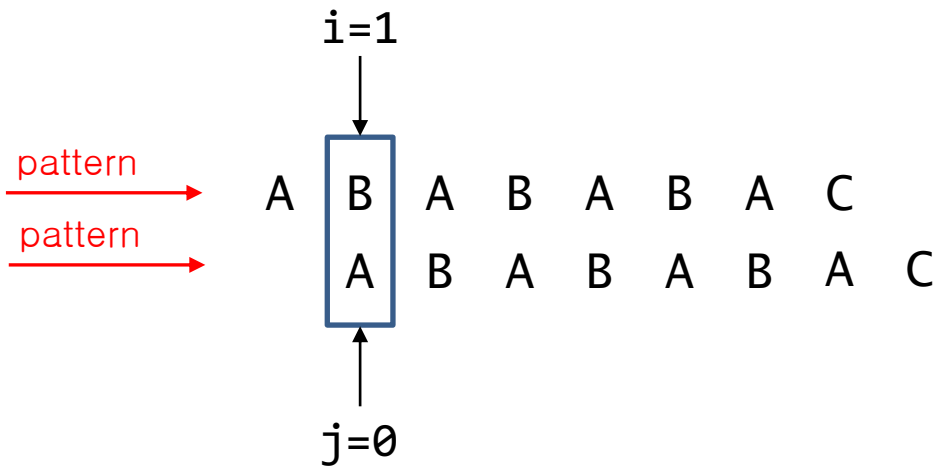
```
vector<int> getFail(string pattern)
{
    int m = (int) pattern.size();
    int j=0;
    vector<int> fail(m, 0);

    for(int i = 1; i< m ; i++)
    {
        while(j > 0 && pattern[i] != pattern[j])
            j = fail[j-1];
        if(pattern[i] == pattern[j])
              fail[i] = ++j;
        else  fail[i] = j;
    }
    return fail;
}
```

computation of fail[] (= π[])

- `getFail()` function
  - Example:

i	0	1	2	3	4	5	6	7
fail[i]	0	0						



pattern[i] != pattern[j]

➔ fail[i] = j
i++

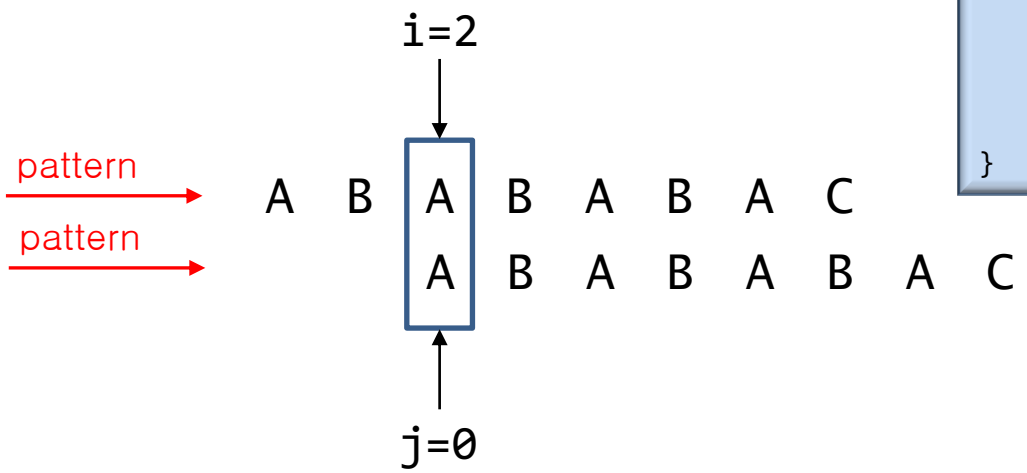
```
vector<int> getFail(string pattern)
{
    int m = (int) pattern.size();
    int j=0;
    vector<int> fail(m, 0);

    for(int i = 1; i< m ; i++)
    {
       while(j > 0 &&
          pattern[i] != pattern[j])
            j = fail[j-1];
       if(pattern[i] ==pattern[j])
             fail[i] = ++j;
       else  fail[i] = j;
    }
    return fail;
}
```

computation of fail[] (= π[])

- `getFail()` function
  - Example:

i	0	1	2	3	4	5	6	7
fail[i]	0	0	1					




```
vector<int> getFail(string pattern)
{
    int m = (int) pattern.size();
    int j=0;
    vector<int> fail(m, 0);

    for(int i = 1; i< m ; i++)
    {
       while(j > 0 &&
          pattern[i] != pattern[j])
            j = fail[j-1];
       if(pattern[i] ==pattern[j])
             fail[i] = ++j;
       else  fail[i] = j;
    }
    return fail;
}
```

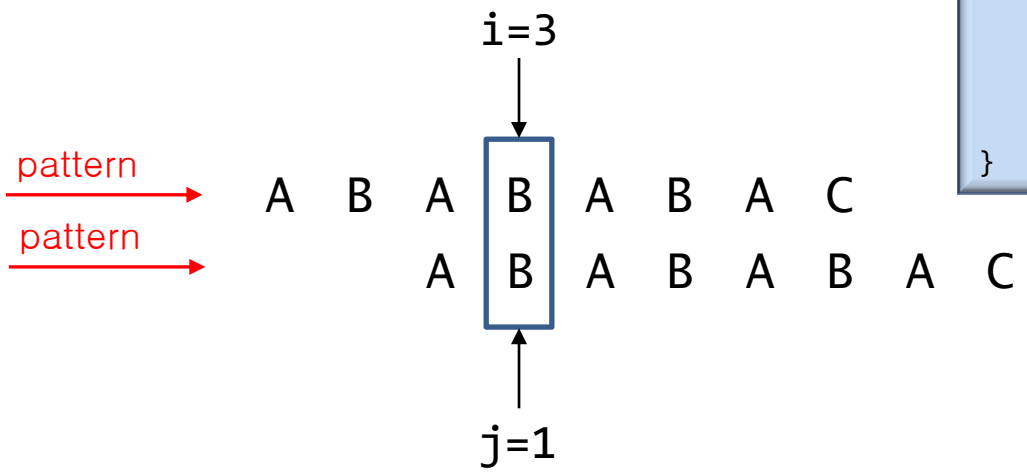
`pattern[i] == pattern[j]`

➔ `fail[i] = ++j`
`i++`

computation of fail[] (= π[])

- `getFail()` function
  - Example:

i	0	1	2	3	4	5	6	7
fail[i]	0	0	1	2				




```
vector<int> getFail(string pattern)
{
    int m = (int) pattern.size();
    int j=0;
    vector<int> fail(m, 0);

    for(int i = 1; i< m ; i++)
    {
       while(j > 0 &&
          pattern[i] != pattern[j])
            j = fail[j-1];
       if(pattern[i] ==pattern[j])
             fail[i] = ++j;
       else  fail[i] = j;
    }
    return fail;
}
```

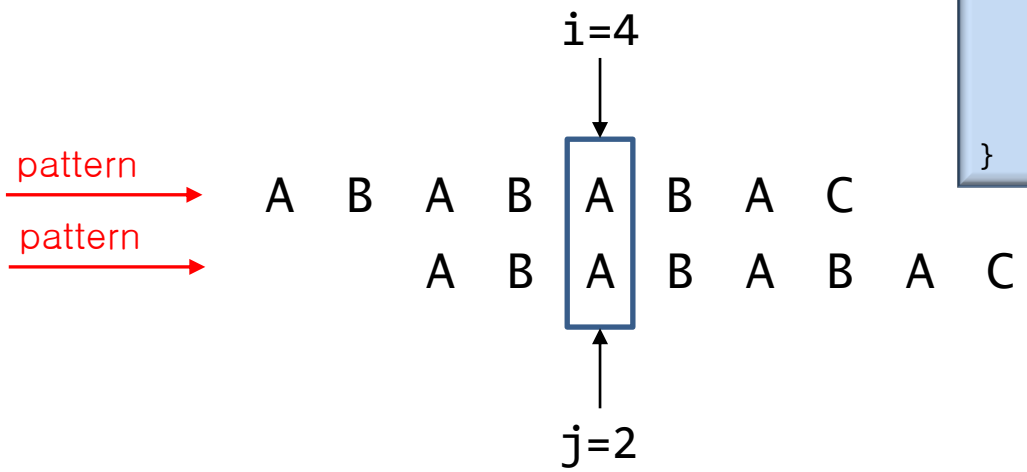
`pattern[i] == pattern[j]`

➔ `fail[i] = ++j`
`i++`

computation of fail[] (= π[])

- `getFail()` function
  - Example:

i	0	1	2	3	4	5	6	7
fail[i]	0	0	1	2	3			




```
vector<int> getFail(string pattern)
{
    int m = (int) pattern.size();
    int j=0;
    vector<int> fail(m, 0);

    for(int i = 1; i< m ; i++)
    {
       while(j > 0 &&
          pattern[i] != pattern[j])
            j = fail[j-1];
       if(pattern[i] ==pattern[j])
             fail[i] = ++j;
       else  fail[i] = j;
    }
    return fail;
}
```

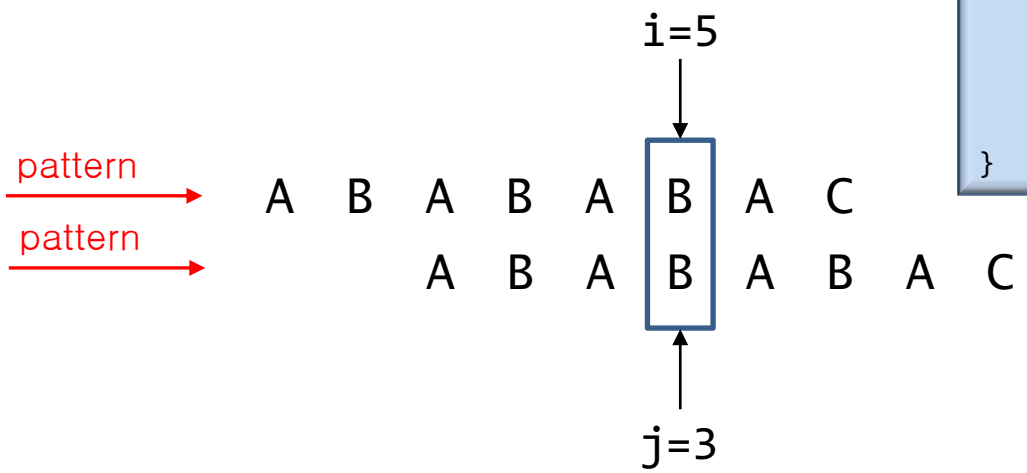
`pattern[i] == pattern[j]`

➔ `fail[i] = ++j`
`i++`

computation of fail[] (= π[])

- `getFail()` function
  - Example:

i	0	1	2	3	4	5	6	7
fail[i]	0	0	1	2	3	4		




```
vector<int> getFail(string pattern)
{
    int m = (int) pattern.size();
    int j=0;
    vector<int> fail(m, 0);

    for(int i = 1; i< m ; i++)
    {
       while(j > 0 &&
          pattern[i] != pattern[j])
            j = fail[j-1];
       if(pattern[i] ==pattern[j])
             fail[i] = ++j;
       else  fail[i] = j;
    }
    return fail;
}
```

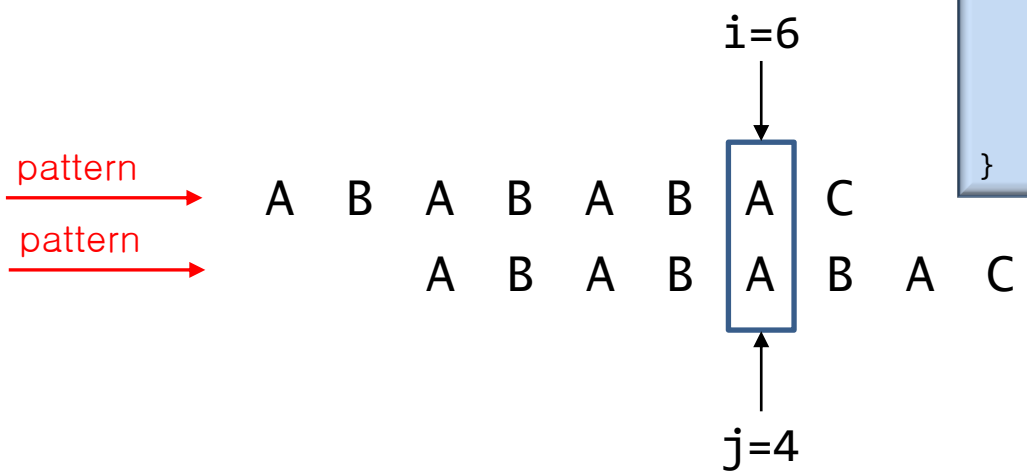
`pattern[i] == pattern[j]`

➔ `fail[i] = ++j`
`i++`

computation of fail[] (= π[])

- `getFail()` function
  - Example:

i	0	1	2	3	4	5	6	7
fail[i]	0	0	1	2	3	4	5	




```
vector<int> getFail(string pattern)
{
    int m = (int) pattern.size();
    int j=0;
    vector<int> fail(m, 0);

    for(int i = 1; i< m ; i++)
    {
       while(j > 0 &&
          pattern[i] != pattern[j])
            j = fail[j-1];
       if(pattern[i] ==pattern[j])
            fail[i] = ++j;
       else  fail[i] = j;
    }
    return fail;
}
```

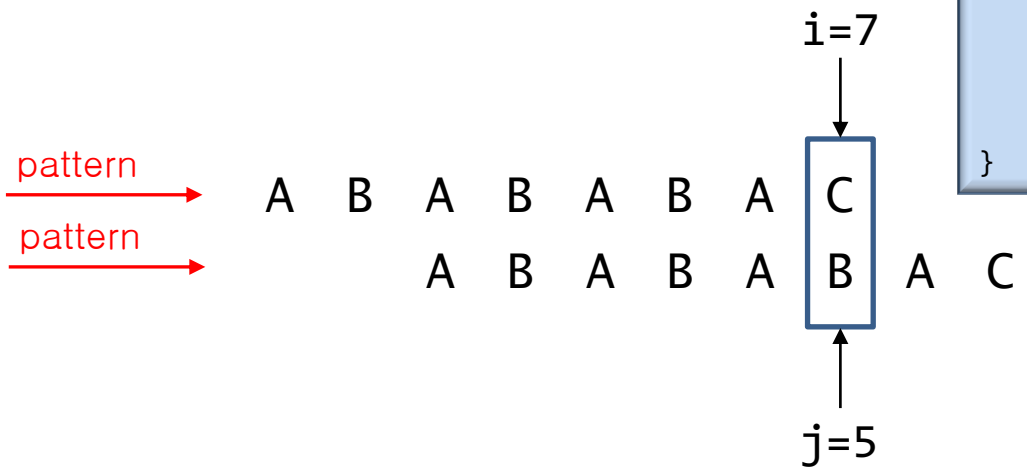
`pattern[i] == pattern[j]`

➔ `fail[i] = ++j`
`i++`

computation of fail[] (= π[])

- `getFail()` function
  - Example:

i	0	1	2	3	4	5	6	7
fail[i]	0	0	1	2	3	4	5	




```
vector<int> getFail(string pattern)
{
    int m = (int) pattern.size();
    int j=0;
    vector<int> fail(m, 0);

    for(int i = 1; i< m ; i++)
    {
       while(j > 0 &&
          pattern[i] != pattern[j])
            j = fail[j-1];
       if(pattern[i] ==pattern[j])
            fail[i] = ++j;
       else  fail[i] = j;
    }
    return fail;
}
```

`pattern[i] != pattern[j]`

➔ `j = fail[j-1] (=3)`

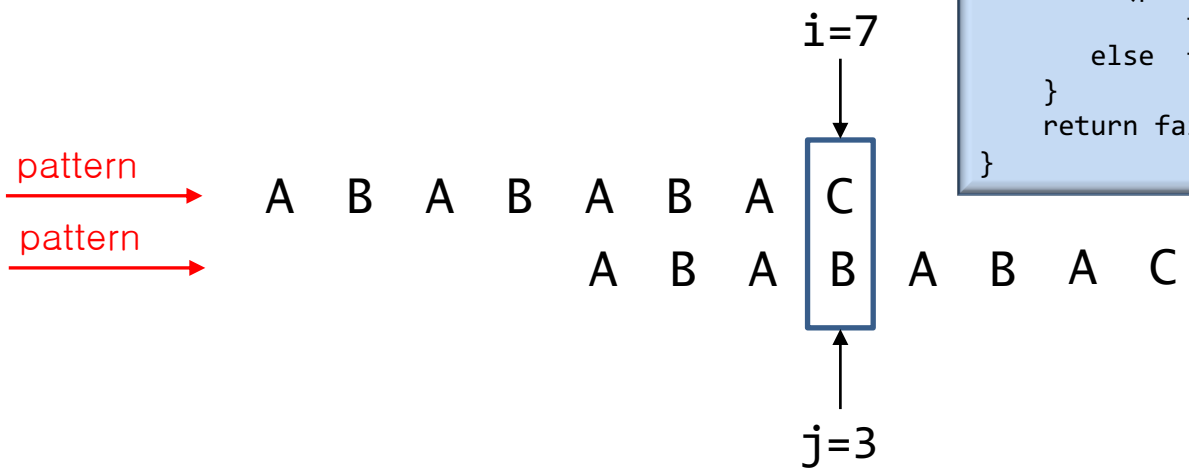
computation of fail[] (= π[])

- `getFail()` function
  - Example:

i	0	1	2	3	4	5	6	7
fail[i]	0	0	1	2	3	4	5	

```
vector<int> getFail(string pattern)
{
    int m = (int) pattern.size();
    int j=0;
    vector<int> fail(m, 0);

    for(int i = 1; i< m ; i++)
    {
       while(j > 0 &&
          pattern[i] != pattern[j])
            j = fail[j-1];
       if(pattern[i] ==pattern[j])
            fail[i] = ++j;
       else  fail[i] = j;
    }
    return fail;
}
```



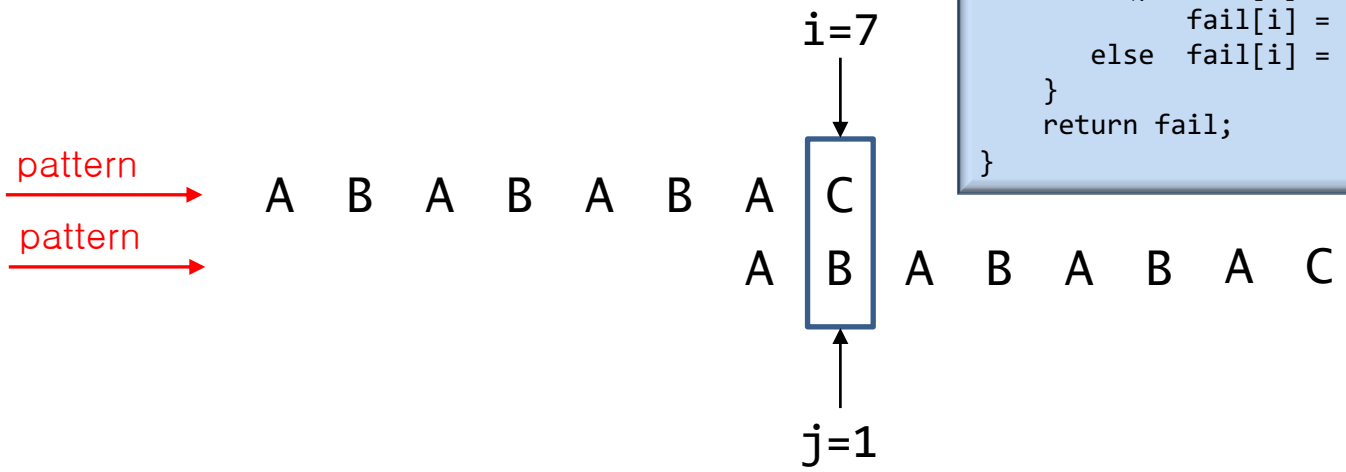
pattern[i] != pattern[j]

➔ j = fail[j-1] (=1)

computation of fail[] (= π[])

- `getFail()` function
  - Example:

i	0	1	2	3	4	5	6	7
fail[i]	0	0	1	2	3	4	5	




```
vector<int> getFail(string pattern)
{
    int m = (int) pattern.size();
    int j=0;
    vector<int> fail(m, 0);

    for(int i = 1; i< m ; i++)
    {
       while(j > 0 &&
          pattern[i] != pattern[j])
            j = fail[j-1];
       if(pattern[i] ==pattern[j])
             fail[i] = ++j;
       else  fail[i] = j;
    }
    return fail;
}
```

`pattern[i] != pattern[j]`

➔ `j = fail[j-1] (=1)`

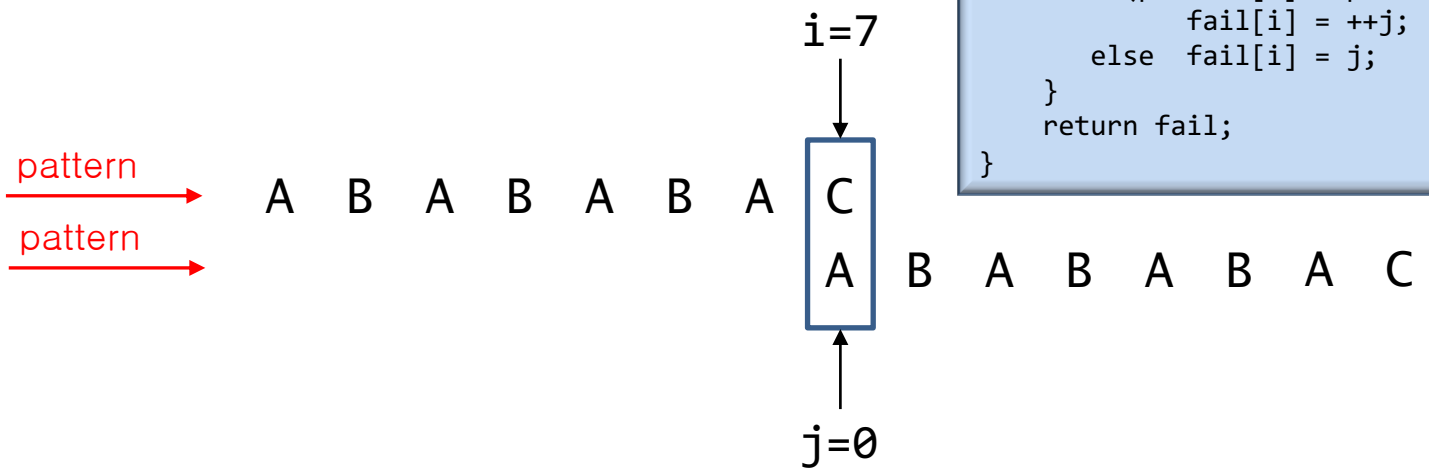
computation of fail[] (= π[])

- `getFail()` function
  - Example:

i	0	1	2	3	4	5	6	7
fail[i]	0	0	1	2	3	4	5	0

```
vector<int> getFail(string pattern)
{
    int m = (int) pattern.size();
    int j=0;
    vector<int> fail(m, 0);

    for(int i = 1; i< m ; i++)
    {
       while(j > 0 &&
          pattern[i] != pattern[j])
            j = fail[j-1];
       if(pattern[i] ==pattern[j])
            fail[i] = ++j;
       else  fail[i] = j;
    }
    return fail;
}
```



pattern[i] != pattern[j]

➔ j = fail[j-1] (=0)
fail[i] = j

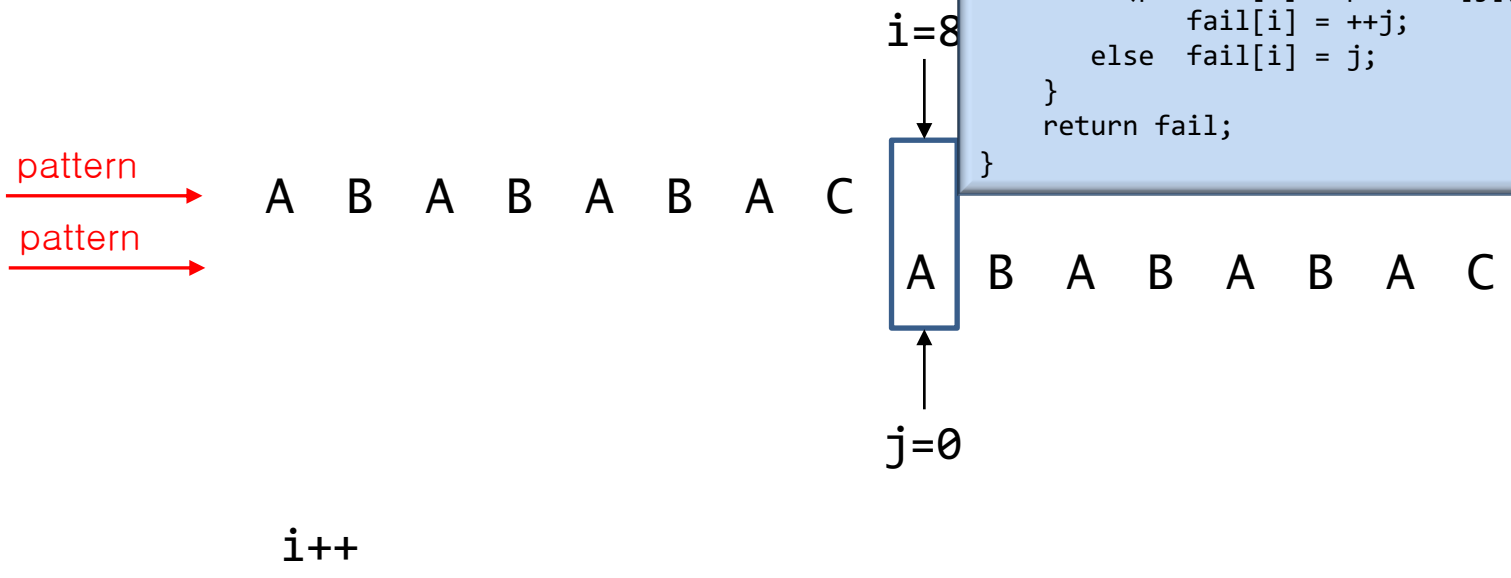
computation of fail[] (= π[])

- `getFail()` function
  - Example:

i	0	1	2	3	4	5	6	7
fail[i]	0	0	1	2	3	4	5	0

```
vector<int> getFail(string pattern)
{
    int m = (int) pattern.size();
    int j=0;
    vector<int> fail(m, 0);

    for(int i = 1; i< m ; i++)
    {
       while(j > 0 &&
          pattern[i] != pattern[j])
            j = fail[j-1];
       if(pattern[i] ==pattern[j])
             fail[i] = ++j;
       else  fail[i] = j;
    }
    return fail;
}
```



- `getFail()` function
  - Example: `aabaabac`

i	0	1	2	3	4	5	6	7
fail[i]	0	1	0	1	2	3	4	0

```
vector<int> getFail(string pattern)
{
    int m = (int) pattern.size();
    int j=0;
    vector<int> fail(m, 0);

    for(int i = 1; i< m ; i++)
    {
       while(j > 0 &&
          pattern[i] != pattern[j])
            j = fail[j-1];
       if(pattern[i] ==pattern[j])
             fail[i] = ++j;
       else  fail[i] = j;
    }
    return fail;
}
```

Analysis of KMP Algorithm

- `getFail()` function
  - Why $O(M)$?
    - Index `i` increases from 1 to M-1
    - Index `j` *increases* maximally as many as `i` increases
    - Also index `j` *decreases* maximally as many as `j` increases

- KMP algorithm
  - Why $O(N)$ algorithm?
    - Similar logic to get the time complexity of `getFail()` function