Knuth-Morris-Pratt

Include one state for each character in pattern (plus accept state).

```
            j   0   1   2   3   4   5
    pat.charAt(j)  A   B   A   B   A   C
                 | A
    dfa[][j]     | B
                 | C
```

⓪   ①   ②   ③   ④   ⑤   ⑥

Constructing the DFA for KMP substring search for A B A B A C

Knuth-Morris-Pratt construction

Match transition: advance to next state if `c == pat.charAt(j)`.

j	0	1	2	3	4	5
pat.charAt(j)	A	B	A	B	A	C
A	1		3		5	
dfa[][j] B		2		4		
C						6

Constructing the DFA for KMP substring search for A B A B A C

Mismatch transition: back up if `c != pat.charAt(j)`.

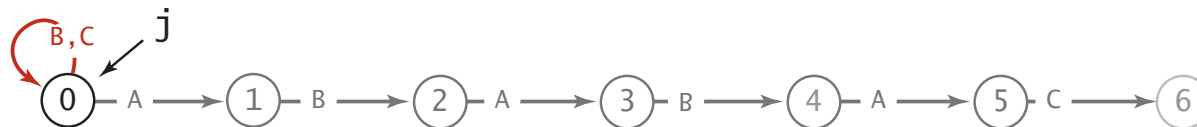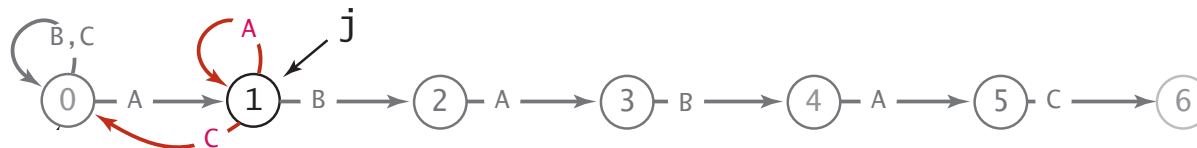| j	0	1	2	3	4	5
pat.charAt(j)	A	B	A	B	A	C
dfa[][j] A	1		3		5	
B	0	2		4		
C	0					6

Constructing the DFA for KMP substring search for A B A B A C

Knuth-Morris-Pratt construction

Mismatch transition: back up if `c != pat.charAt(j)`.

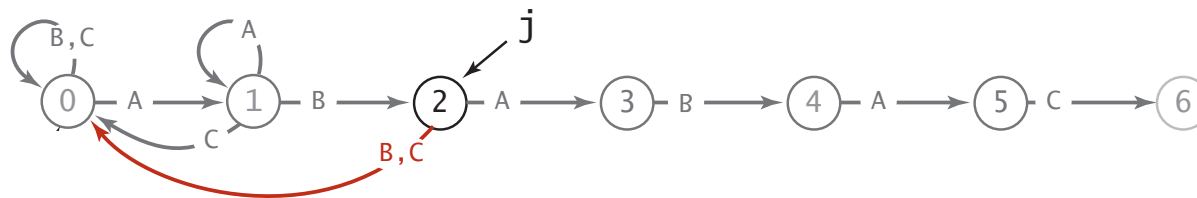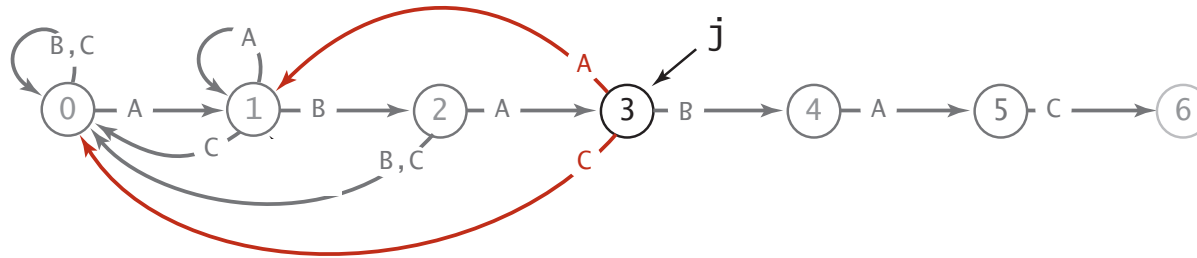| j	0	1	2	3	4	5
pat.charAt(j)	A	B	A	B	A	C
dfa[][j] A	1	1	3		5	
B	0	2		4		
C	0	0				6

Constructing the DFA for KMP substring search for A B A B A C

Knuth-Morris-Pratt construction

Mismatch transition: back up if `c != pat.charAt(j)`.

j	0	1	2	3	4	5
pat.charAt(j)	A	B	A	B	A	C
dfa[][j] A	1	1	3		5	
B	0	2	0	4		
C	0	0	0			6

Constructing the DFA for KMP substring search for A B A B A C

Mismatch transition: back up if `c != pat.charAt(j)`.

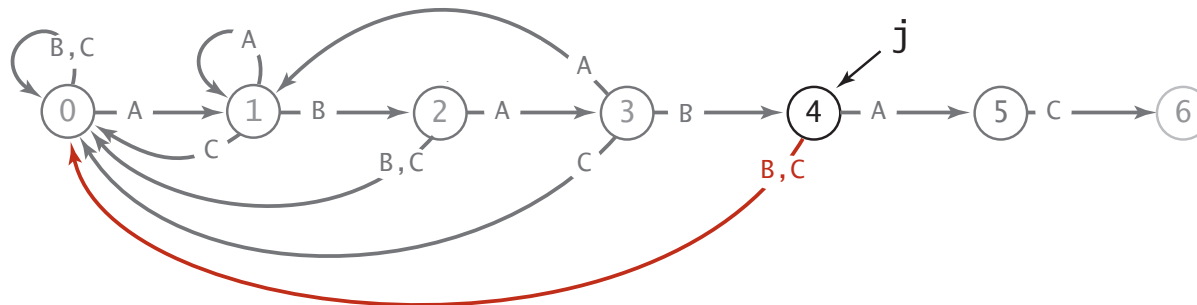| j	0	1	2	3	4	5
pat.charAt(j)	A	B	A	B	A	C
dfa[][j] A	1	1	3	1	5	
B	0	2	0	4		
C	0	0	0	0		6

Constructing the DFA for KMP substring search for A B A B A C

Knuth-Morris-Pratt construction
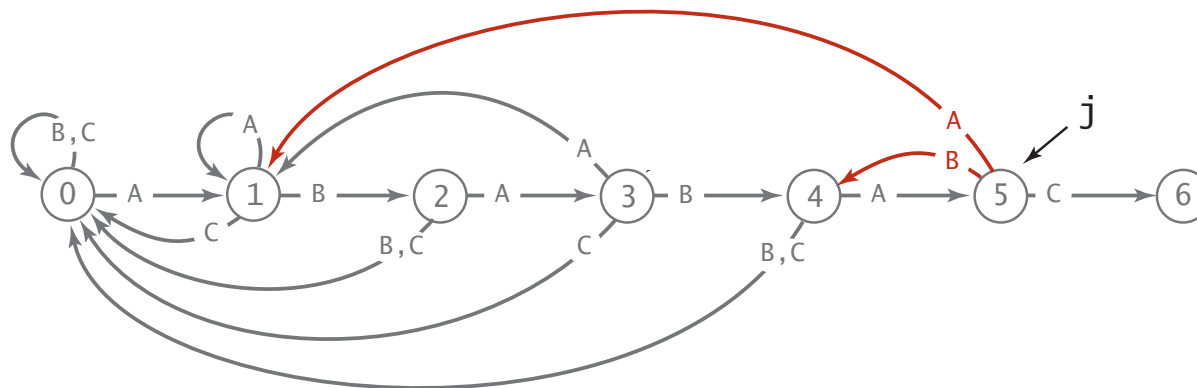
Mismatch transition: back up if `c != pat.charAt(j)`.

	j	0	1	2	3	4	5
	pat.charAt(j)	A	B	A	B	A	C
	A	1	1	3	1	5	
dfa[][j]	B	0	2	0	4	0	
	C	0	0	0	0	0	6

Mismatch transition: back up if `c != pat.charAt(j)`.

	j	0	1	2	3	4	5
	pat.charAt(j)	A	B	A	B	A	C
dfa[][j]	A	1	1	3	1	5	1
	B	0	2	0	4	0	4
	C	0	0	0	0	0	6

j	0	1	2	3	4	5
pat.charAt(j)	A	B	A	B	A	C
dfa[][j] A	1	1	3	1	5	1
B	0	2	0	4	0	4
C	0	0	0	0	0	6

Knuth-Morris-Pratt

Include one state for each character in pattern (plus accept state).

```
       j   0   1   2   3   4   5
pat.charAt(j)  A   B   A   B   A   C
            | A
dfa[][j]    | B
            | C
```
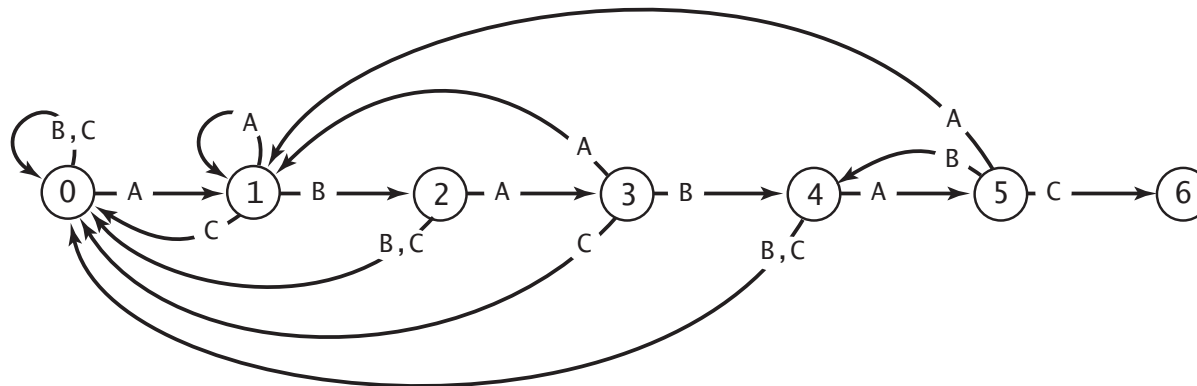
⓪   ①   ②   ③   ④   ⑤   ⑥

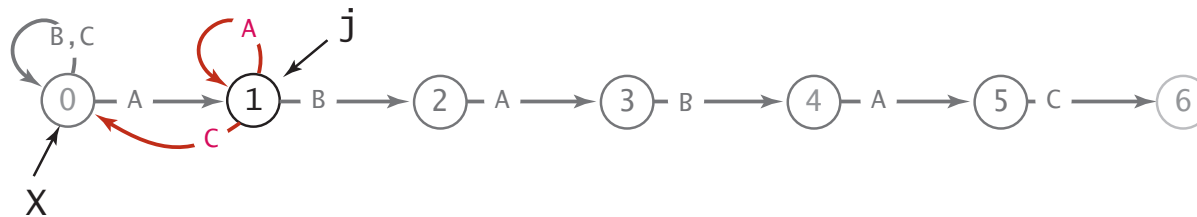**Constructing the DFA for KMP substring search for A B A B A C**

Knuth-Morris-Pratt construction

Match transition. For each state `j`, `dfa[pat.charAt(j)][j] = j+1`.

↑
first j characters of pattern
have already been matched

↑
now first j+1 characters of
pattern have been matched

	j	0	1	2	3	4	5
	pat.charAt(j)	A	B	A	B	A	C
	A	1		3		5	
dfa[][j]	B		2		4		
	C						6

```
0 ─ A ─→ 1 ─ B ─→ 2 ─ A ─→ 3 ─ B ─→ 4 ─ A ─→ 5 ─ C ─→ 6
```

Constructing the DFA for KMP substring search for A B A B A C

Mismatch transition.

	j	0	1	2	3	4	5
	pat.charAt(j)	A	B	A	B	A	C
	A	1		3		5	
dfa[][j]	B	0	2		4		
	C	0					6

Constructing the DFA for KMP substring search for A B A B A C

14

Knuth-Morris-Pratt construction

Mismatch transition. For each state `j` and char `c != pat.charAt(j)`,
`dfa[c][j] = dfa[c][X]`; then update `x = dfa[pat.charAt(j)][X]`.

```
                              X
                              ↓
                 j    0    1    2    3    4    5
        pat.charAt(j) A    B    A    B    A    C
                     │A    1    1    3         5
           dfa[][j]  │B   (0)   2         4
                     │C    0    0              6
```
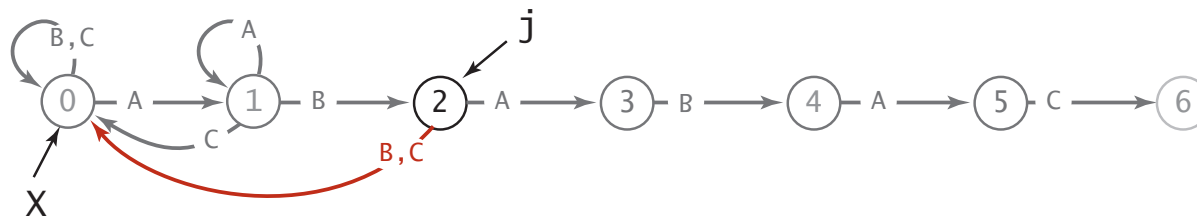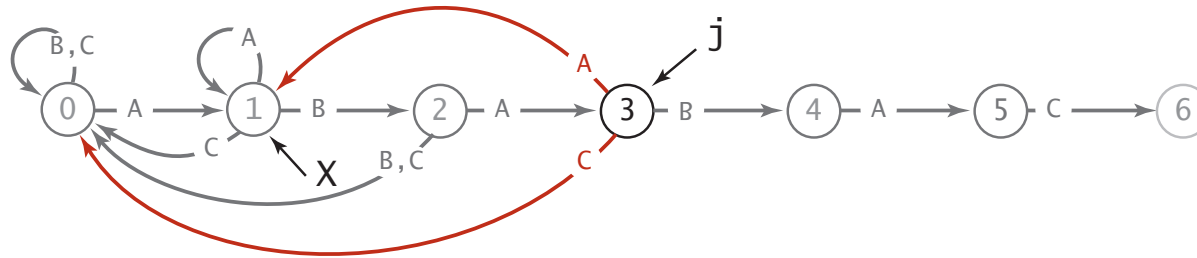

Constructing the DFA for KMP substring search for A B A B A C

Knuth-Morris-Pratt construction

Mismatch transition. For each state `j` and char `c != pat.charAt(j)`,

`dfa[c][j] = dfa[c][X];` then update `X = dfa[pat.charAt(j)][X].`
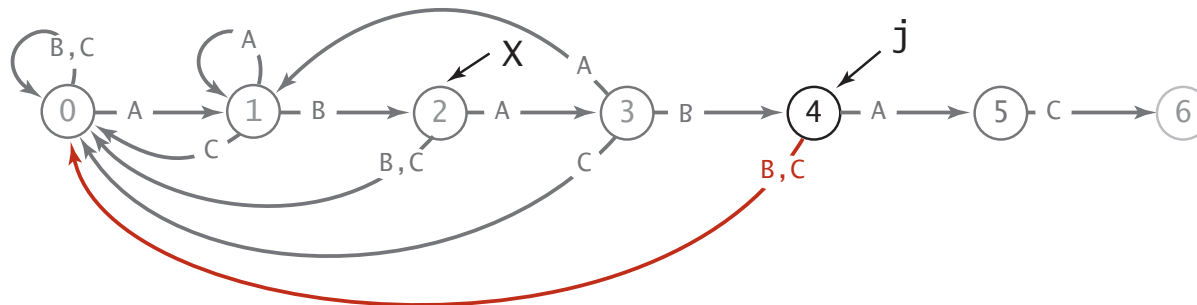

Constructing the DFA for KMP substring search for A B A B A C

Knuth-Morris-Pratt construction

Mismatch transition. For each state `j` and char `c != pat.charAt(j)`,
`dfa[c][j] = dfa[c][X];` then update `x = dfa[pat.charAt(j)][X]`.

X
↓

j	0	1	2	3	4	5
pat.charAt(j)	A	B	A	B	A	C
A	1	1	3	1	5	
dfa[][j] B	0	②	0	4		
C	0	0	0	0		6

Constructing the DFA for KMP substring search for A B A B A C

Knuth-Morris-Pratt construction

Mismatch transition. For each state `j` and char `c != pat.charAt(j)`,

`dfa[c][j] = dfa[c][X]`; then update `x = dfa[pat.charAt(j)][X]`.

Knuth-Morris-Pratt construction

Mismatch transition. For each state `j` and char `c != pat.charAt(j)`,
`dfa[c][j] = dfa[c][X];` then update `x = dfa[pat.charAt(j)][X]`.

j	0	1	2	3	4	5
pat.charAt(j)	A	B	A	B	A	C
dfa[][j] A	1	1	3	1	5	1
B	0	2	0	4	0	4
C	0	0	0	0	0	6