

---

Searching for a pattern.

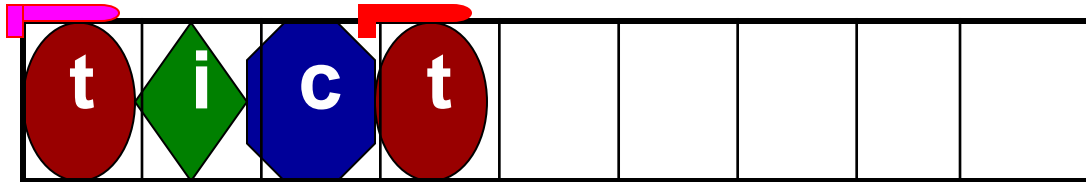
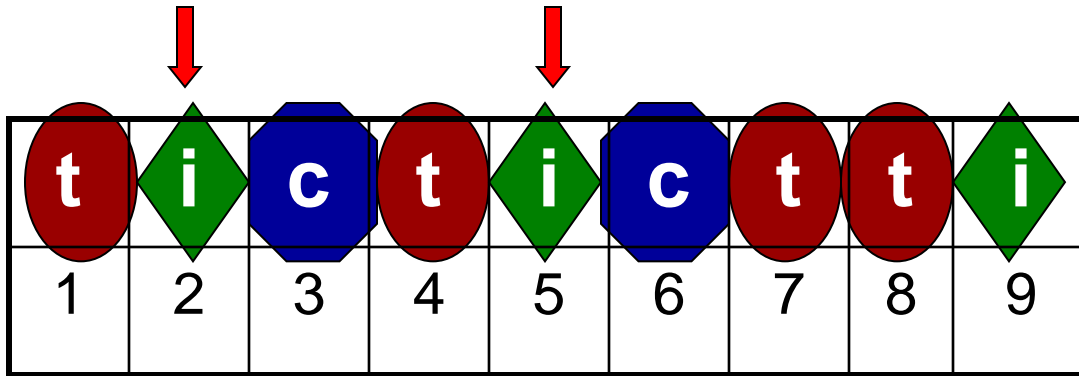
Knuth-Morris-Pratt.

Overlap function

---

Lecture 2.1

# How to compute the OF function



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

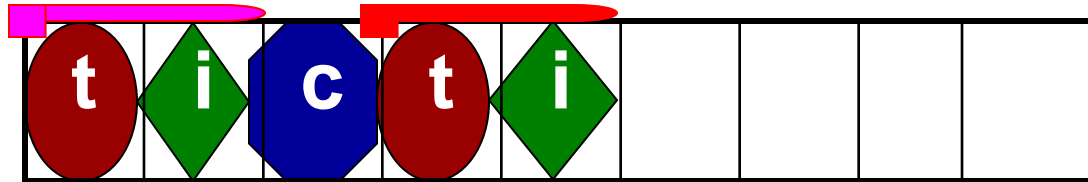
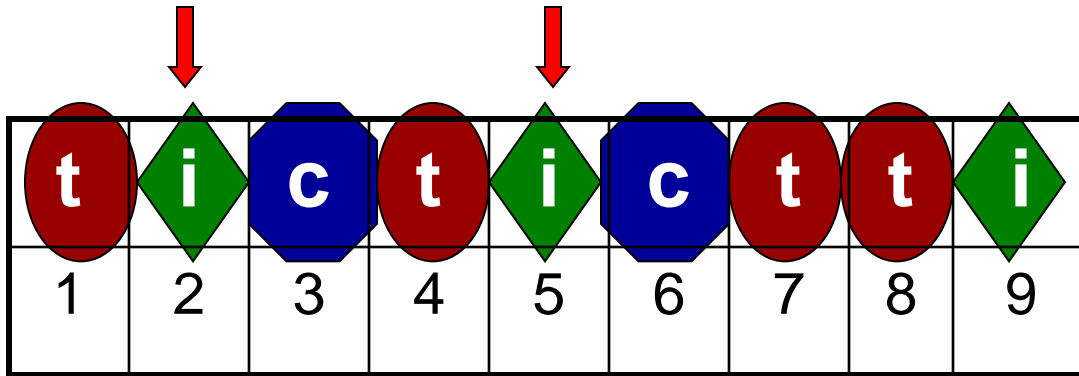
The easy case:

if we have  $OF(j-1)$ , and the characters

$P[j]$  and  $P[OF(j-1)+1]$  match

Then we just increase by 1  
 $OF(j) = OF(j-1) + 1$

# How to compute the OF function



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

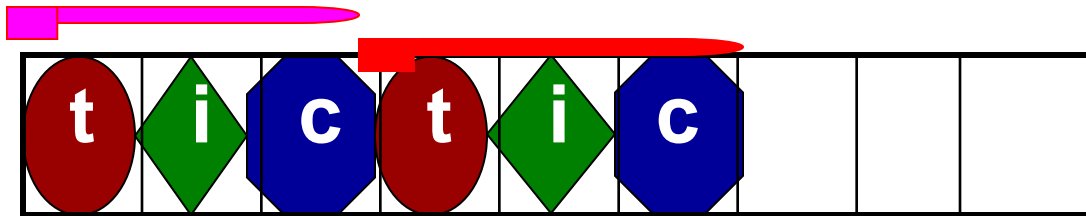
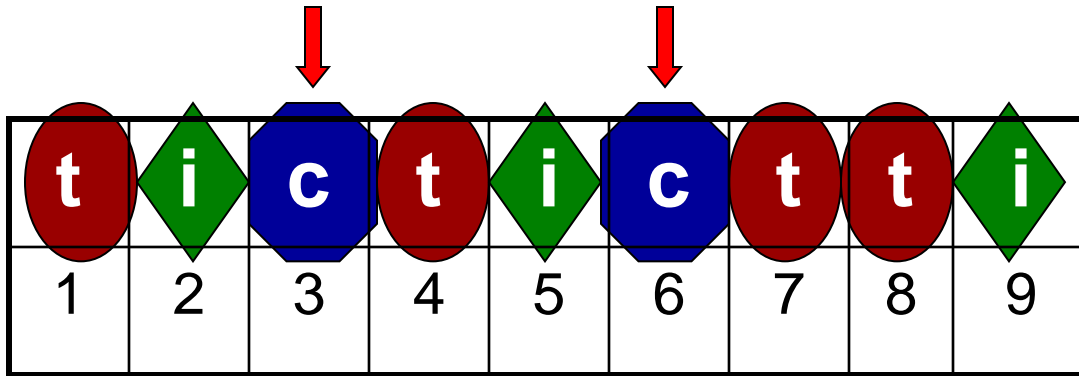
The easy case:

if we have  $OF(j-1)$ , and the characters

$P[j]$  and  $P[OF(j-1)+1]$  match

Then we just increase  
 $OF(j)=OF(j-1)+1$

# How to compute the OF function



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

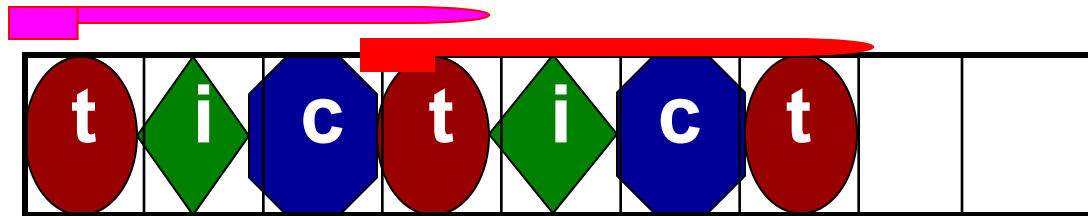
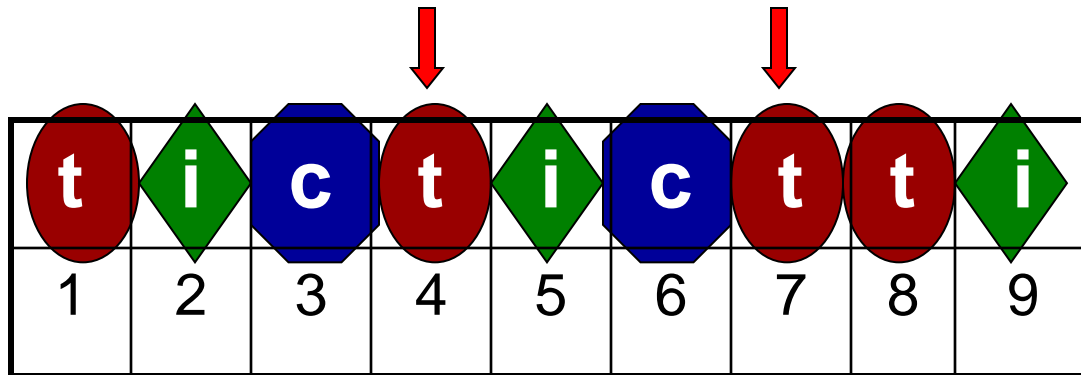
The easy case:

if we have  $OF(j-1)$ , and the characters

$P[j]$  and  $P[OF(j-1)+1]$  match

Then we just increase  
 $OF(j)=OF(j-1)+1$

# How to compute the OF function



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

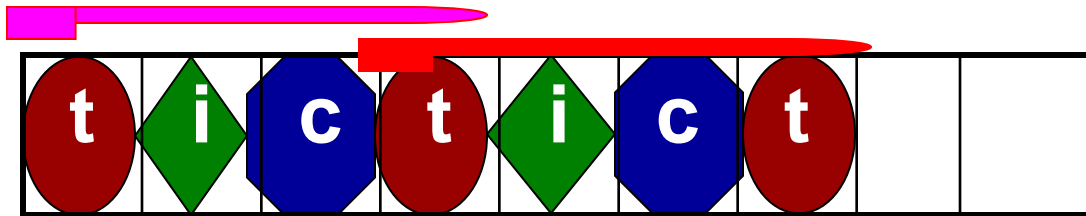
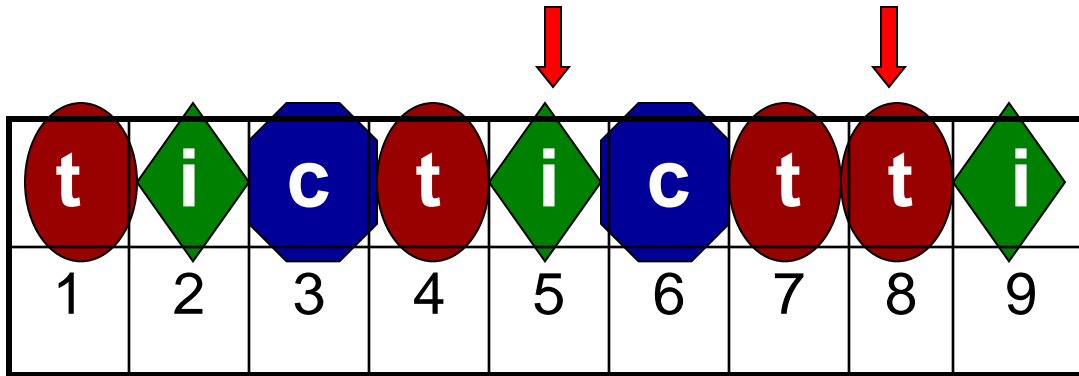
The easy case:

if we have  $OF(j-1)$ , and the characters

$P[j]$  and  $P[OF(j-1)+1]$  match

Then we just increase  
 $OF(j)=OF(j-1)+1$

# How to compute the OF function



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

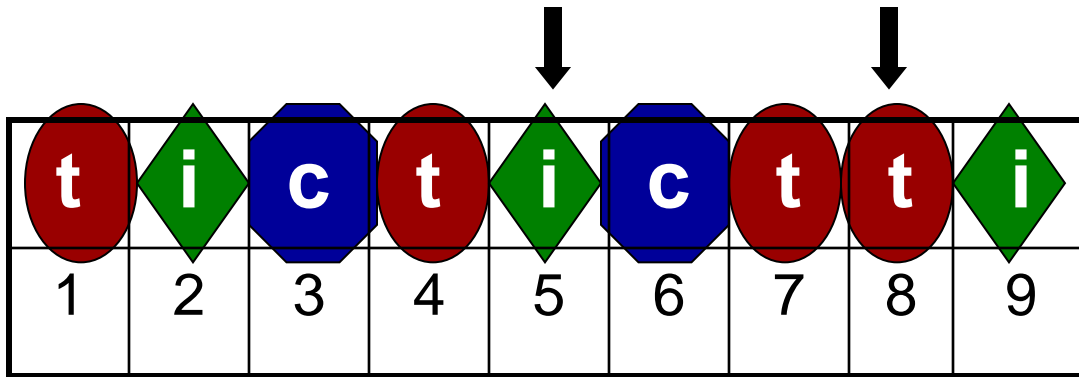
The *general* case:

If the characters

$P[j]$  and  $P[OF(j-1)+1]$  do not match

where do we find  $OF[j]$ ?

# How to compute the OF function



The general case:

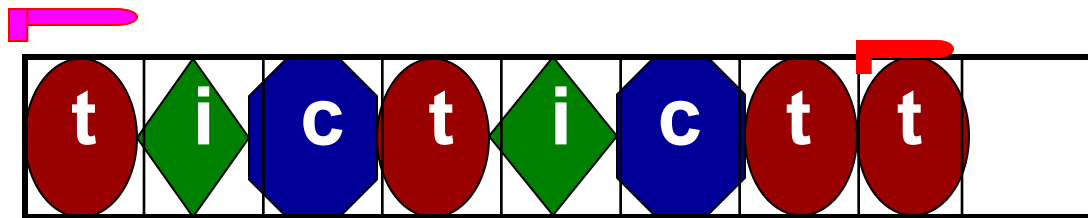
If the characters

$P[j]$  and  $P[OF(j-1)+1]$  do not match

then  $OF(j)$  is less than  $OF(j-1)$

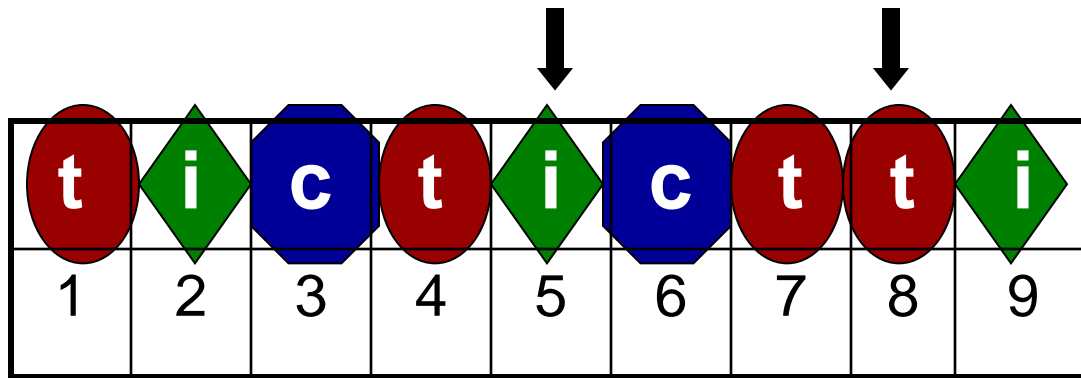
We look at  $v = OF(j-1)$  and check again the next character

$P[OF(v)+1]$



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

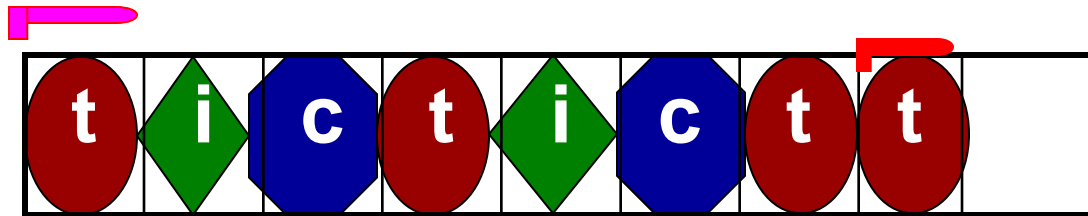
# How to compute the OF function



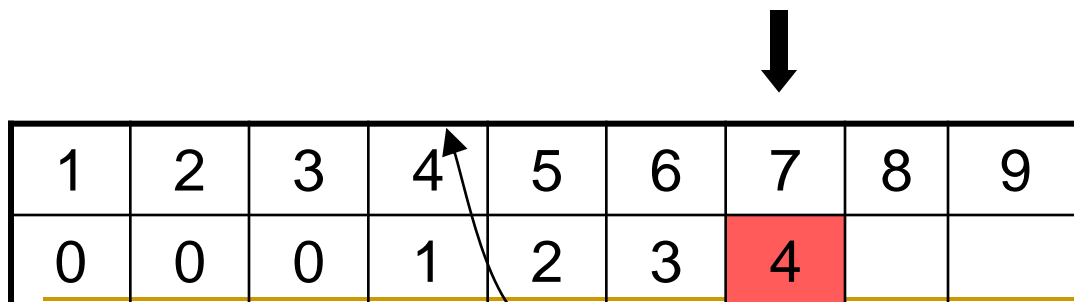
The general case:

If the characters

$P[j]$  and  $P[OF(j-1)+1]$  do not match



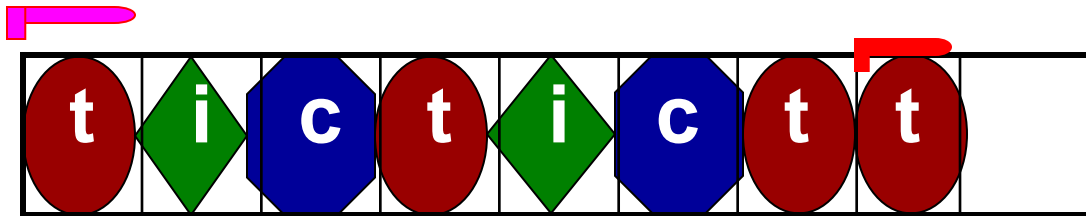
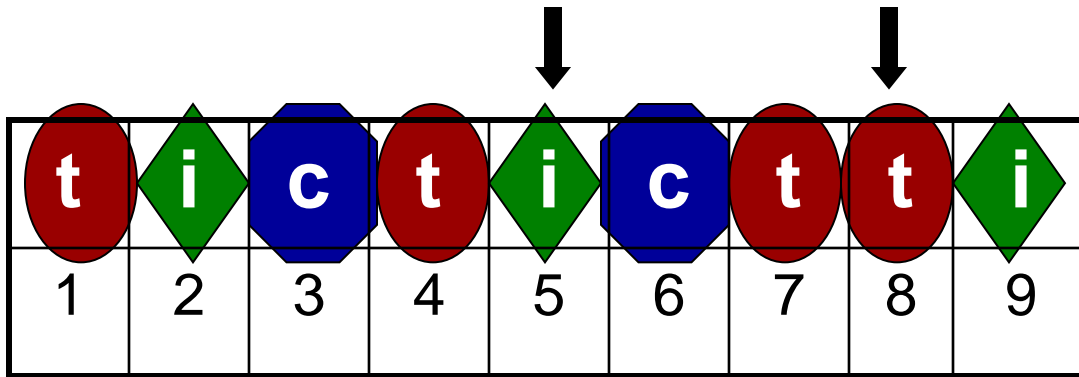
we look at  $v=OF(j-1)$  and check again the next character  $P[OF(v)+1]$



The pointer is bouncing through the entire OF table until it finds the symbol matching the current symbol after the next assignment of  $v=OF(v)$



# How to compute the OF function



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

The general case:

If the characters

$P[j]$  and  $P[OF(j-1)+1]$  do not match

then  $OF(j)$  is less than  $OF(j-1)$

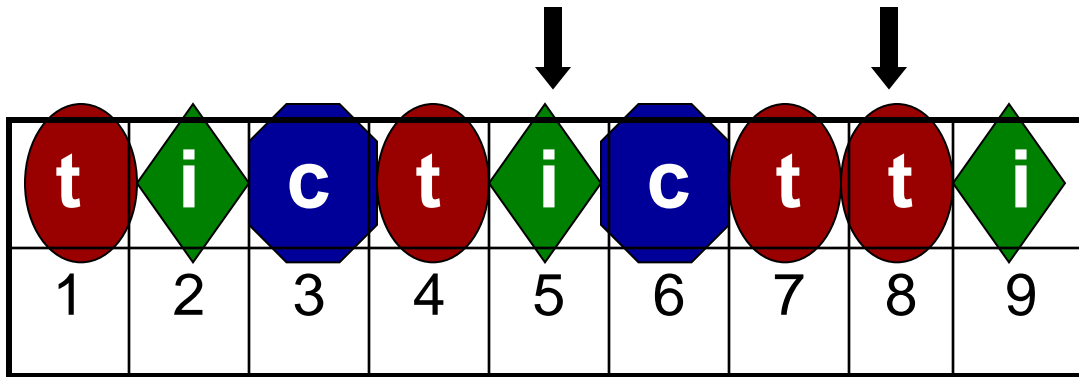
We look at  $v=OF(j-1)$  and check again the next character

The pointer is bouncing through the entire OF table until it finds the symbol matching the current symbol after the next assignment of  $v=OF(v)$

$P[2] \neq P[8]$

$v=OF(4)$

# How to compute the OF function

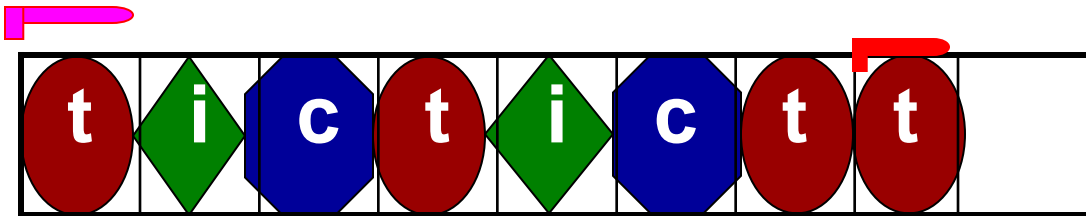


The general case:

$$v = \text{OF}(4)$$

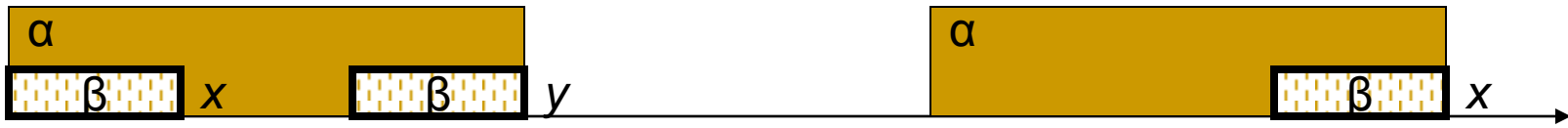
$P[1] = P[8]$ , thus

$$\text{OF}(8) = \text{OF}(1) + 1 = 1$$



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

# Why do we compute the OF value this way?



We know that since we could not extend suffix  $\alpha$ , so there is a smaller suffix,  $\beta$ , which starts somewhere inside  $\alpha$ .

What is the next smaller overlap for all suffixes starting inside  $\alpha$ ?

The same as for all suffixes inside the prefix of length  $|\alpha|$

Thus, if we check the OF value for the position  $|\alpha|$ , we see the next smaller maximal overlap

We check if this is a desired maximal overlap by checking the next character after the prefix of size  $|\beta|$

If this character is  $x$ , we are done

If not, we continue by the same logic

# Example

|          |          |          |          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| <b>t</b> | <b>i</b> | <b>c</b> | <b>t</b> | <b>i</b> | <b>c</b> | <b>t</b> | <b>t</b> | <b>i</b> |
| 1        | 2        | 3        | 4        | 5        | 6        | 7        | 8        | 9        |

|          |          |          |          |          |          |          |          |   |
|----------|----------|----------|----------|----------|----------|----------|----------|---|
| <b>t</b> | <b>i</b> | <b>c</b> | <b>t</b> | <b>i</b> | <b>c</b> | <b>t</b> | <b>t</b> |   |
| 1        | 2        | 3        | 4        | 5        | 6        | 7        | 8        | 9 |

|   |   |   |          |   |   |   |   |   |
|---|---|---|----------|---|---|---|---|---|
| 1 | 2 | 3 | 4        | 5 | 6 | 7 | 8 | 9 |
| 0 | 0 | 0 | <b>1</b> | 2 | 3 | 4 |   |   |

We know that the substring *tictict* ending at position 7 had suffix *tict* which is overlapping with the prefix *tict* of the pattern

We also know that we cannot extend this overlap since  $P[8]$  and  $P[5]$  do not match

Now we want to check what overlap had the prefix *tict* with the prefix of the entire pattern, since the suffix start for a new overlap is somewhere inside *tict*

We look at position 4 in OF table and find that the next overlap for substring of length 4 is of length 1

# Example

|          |          |          |          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|----------|----------|----------|
| <b>t</b> | <b>i</b> | <b>c</b> | <b>t</b> | <b>i</b> | <b>c</b> | <b>t</b> | <b>t</b> | <b>i</b> |
| 1        | 2        | 3        | 4        | 5        | 6        | 7        | 8        | 9        |

|          |          |          |          |          |          |          |          |   |
|----------|----------|----------|----------|----------|----------|----------|----------|---|
| <b>t</b> | <b>i</b> | <b>c</b> | <b>t</b> | <b>i</b> | <b>c</b> | <b>t</b> | <b>t</b> |   |
| 1        | 2        | 3        | 4        | 5        | 6        | 7        | 8        | 9 |

|          |   |   |   |   |   |   |   |   |
|----------|---|---|---|---|---|---|---|---|
| 1        | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| <b>0</b> | 0 | 0 | 1 | 2 | 3 | 4 |   |   |

We check if  $P[1+1]$  matches  $P[8]$

They do not

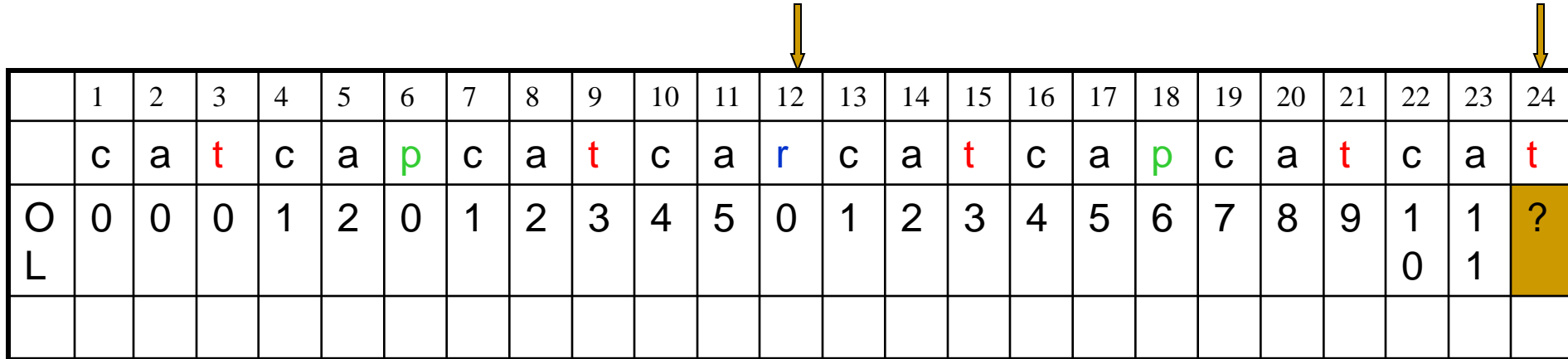
We repeat and by the same logic we are going to the entry 1 of the OF table, and find that there is no overlap for this value:  $OF[1]=0$

So we check if

$P[0+1]$  matches  $P[8]$

They do, so the  $OF[8]=OF[1]+1=1$

# A more complex example of the OL computation



|    | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|----|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|    | c | a | t | c | a | p | c | a | t | c  | a  | r  | c  | a  | t  | c  | a  | p  | c  | a  | t  | c  | a  | t  |
| OL | 0 | 0 | 0 | 1 | 2 | 0 | 1 | 2 | 3 | 4  | 5  | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 1  | 1  | ?  |
|    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

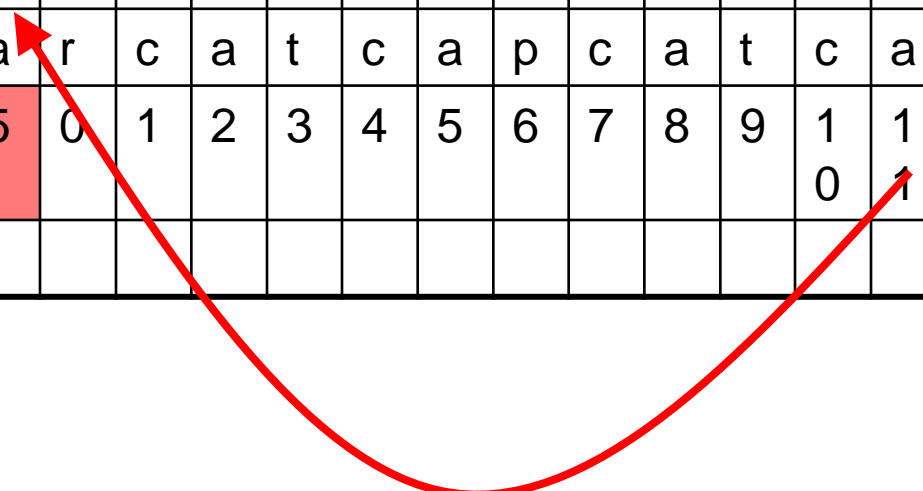
We know that  $OL(23)=11$

This means that the sequence of the first 11 characters of P is the same as that of the last 11 characters of P[1....23]

However, the character  $P[11+1]=r$  does not match the character  $P[23+1]=t$

# A more complex example of the OL computation

|    | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|----|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|    | c | a | t | c | a | p | c | a | t | c  | a  | r  | c  | a  | t  | c  | a  | p  | c  | a  | t  | c  | a  | t  |
| OL | 0 | 0 | 0 | 1 | 2 | 0 | 1 | 2 | 3 | 4  | 5  | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 1  | 1  | ?  |
|    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |



The maximum possible overlap is less than 11

The next maximum possible overlap can be found if we look at position 11 of the OF table and see what overlap this substring had

The substring  $P[1\dots 11]$  has a maximum overlap of length 5

# A more complex example of the OL computation

|    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |
|----|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|    | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|    | c | a | t | c | a | p | c | a | t | c  | a  | r  | c  | a  | t  | c  | a  | p  | c  | a  | t  | c  | a  | t  |
| OL | 0 | 0 | 0 | 1 | 2 | 0 | 1 | 2 | 3 | 4  | 5  | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 1  | 1  | ?  |
|    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

Let us check if this value is also the maximum overlap for the substring  $P[1\dots 24]$

For this we check the character next to  $P[5]$ , which is p, and it does not match our t

Therefore, the overlap we are looking for is less than 5

---



# A more complex example of the OL computation



|    | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|----|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|    | c | a | t | c | a | p | c | a | t | c  | a  | r  | c  | a  | t  | c  | a  | p  | c  | a  | t  | c  | a  | t  |
| OL | 0 | 0 | 0 | 1 | 2 | 0 | 1 | 2 | 3 | 4  | 5  | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | 1  | 1  | 3  |
|    |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |

We check the next possible value by considering the overlap value for the substring  $P[1\dots 5]$

This value is 2. Is this value of an overlap good for  $P[1\dots 24]$ ?


We check  $P[2+1]=t$ , and  $P[24]=t$

Thus, the overlap for the substring  $P[1\dots 24]$  is  $2+1=3$

# Practice jumps on the following pattern

- aaahamaaahamamaaahamaaaa

|        | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 |
|--------|---|---|---|---|---|---|---|---|---|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
|        | a | a | a | h | a | m | a | a | a | h  | a  | m  | a  | m  | a  | a  | a  | h  | a  | m  | a  | a  | a  | a  |
| O<br>L | 0 | 1 | 2 | 0 | 1 | 0 | 1 | 2 | 3 | 4  | 5  | 6  | 7  | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  | 9  | ?  |
|        |   |   |   |   |   |   |   |   |   |    |    |    |    |    |    |    |    |    |    |    |    |    |    |    |









# Overlap function - pseudocode

**algorithm computeOverlapFunction** (pattern  $P$  of length  $M$ )

$OF[1]=0$

**for**  $k:=1$  **to**  $M-1$

$c:=P[k+1]$  // current character of  $P$

$v:=OF[k]$

**while:**  $P[v+1] \neq c$  **and**  $v \neq 0$

$v:=OF[v]$

**if**  $P[v+1]=c$

$OF[k+1]:=v+1$

**else**

$OF[k+1]:=0$

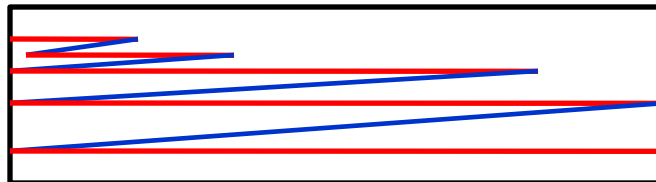
**return**  $OF$  table

# Overlap function: time complexity

The computation of  $OF$  is performed in time  $O(M)$  since:

- the total complexity is proportional to the total number of times the value of  $v$  is changed
- this value is increasing by one (or remains zero) in the *for* loop, and in total, during the entire algorithm, it is increasing not more than by  $M$  units
- in addition, the value of  $v$  is decreasing inside the *while* loop, but since  $v$  is never less than zero, the total number of units by which it is decreasing can not be more than the number it has been increasing, therefore it is bounded by  $M$  too.

The time is therefore less than  $2M$ :  $O(M)$



If we sum up the length of all the red lines (increasing value of  $v$ ), the result will be  $\leq M$ . Therefore, the total length of blue lines (decreasing value of  $v$ ) cannot be more than  $M$  in total

---

# Overlap function table

- Is called in the modern literature the *border array*



# Overlap Function - again

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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

The *OF* values tell where to position the start of the next comparison

They also tell which character to compare in *P* and whether to advance or not the pointer in *T*

For example, if mismatch occurred at pattern position  $j=5$ , from  $OF(5-1)=1$  the start  $k$  is 1 position backwards from a current position  $i$  in *T*, and we compare the same character in *T* with the character  $OF(5-1)+1=2$  in *P*, since we know that the first 1 character matches *T* starting from  $k$

| pos | OF |              |              |
|-----|----|--------------|--------------|
| 1   | 0  | Advance in T | Compare P[1] |
| 2   | 1  | Stay in T    | Compare P[1] |
| 3   | 0  | Stay in T    | Compare P[2] |
| 4   | 1  | Stay in T    | Compare P[1] |
| 5   | 2  | Stay in T    | Compare P[2] |
| 6   | 2  | Stay in T    | Compare P[3] |
|     |    | Advance in T | Compare P[3] |

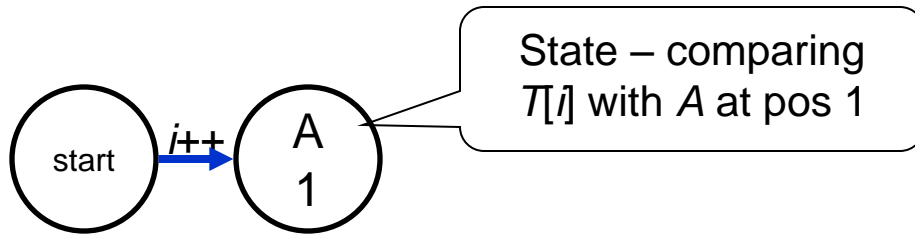
---

# Finite state automaton

- *FSA* is a model of behavior composed of a finite number of *states*, *transitions* between those states, and *actions*.
  - It is similar to a "flow graph" where we can inspect the way in which the logic runs when certain conditions are met.
-

# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

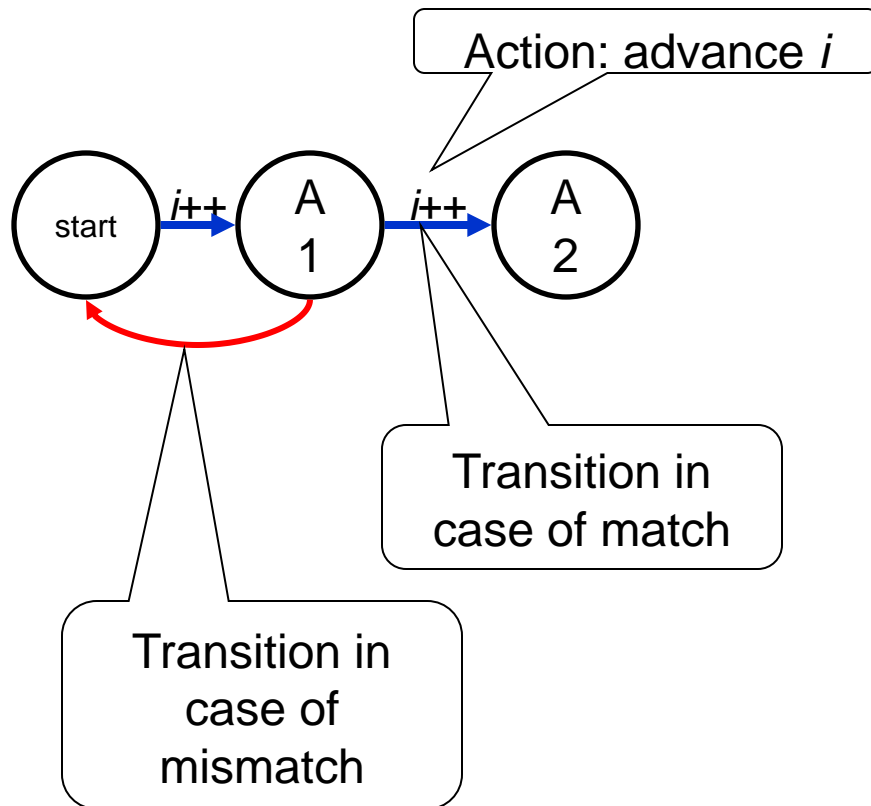


# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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



Where the transition in case of failure is directed, is determined by the value of an overlap function

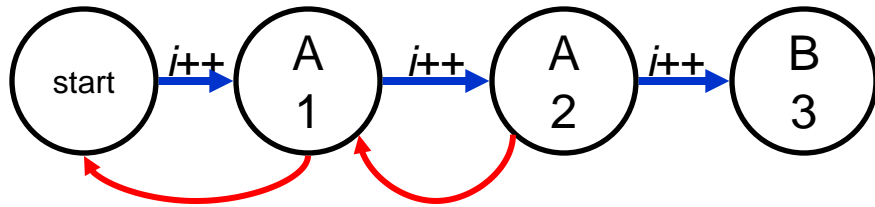
That is why OL function is called also *a failure function*

# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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

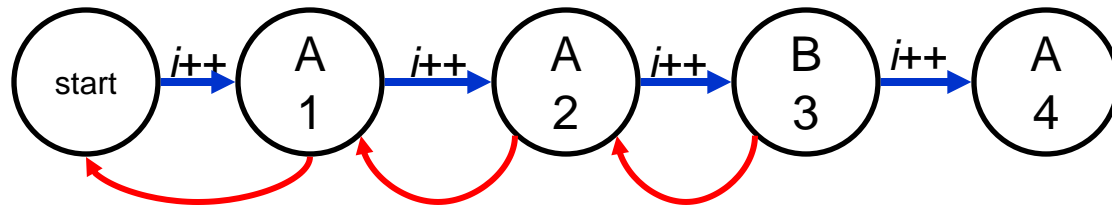


# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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

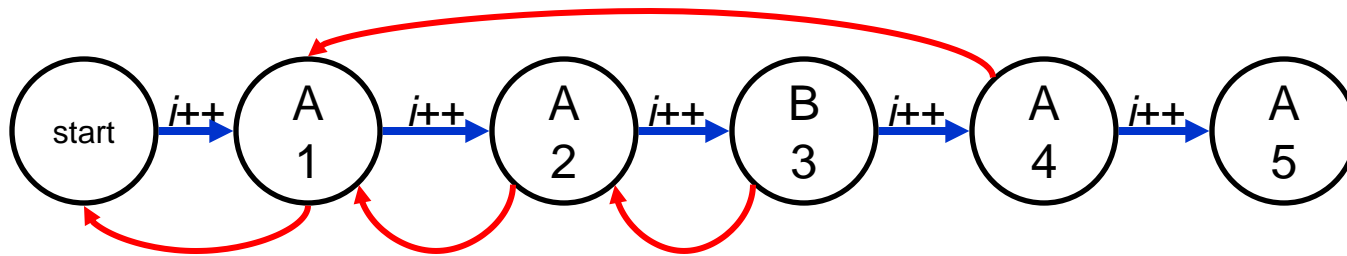


# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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

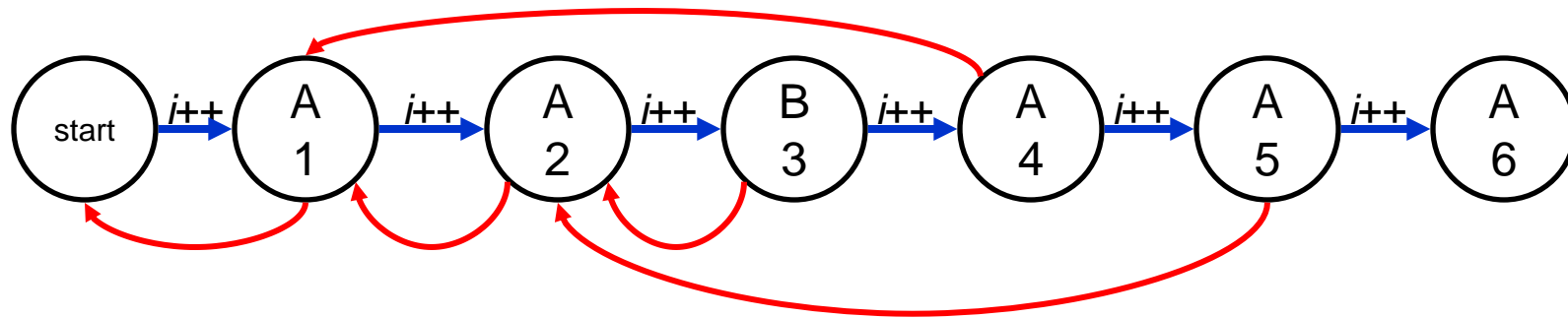


# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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



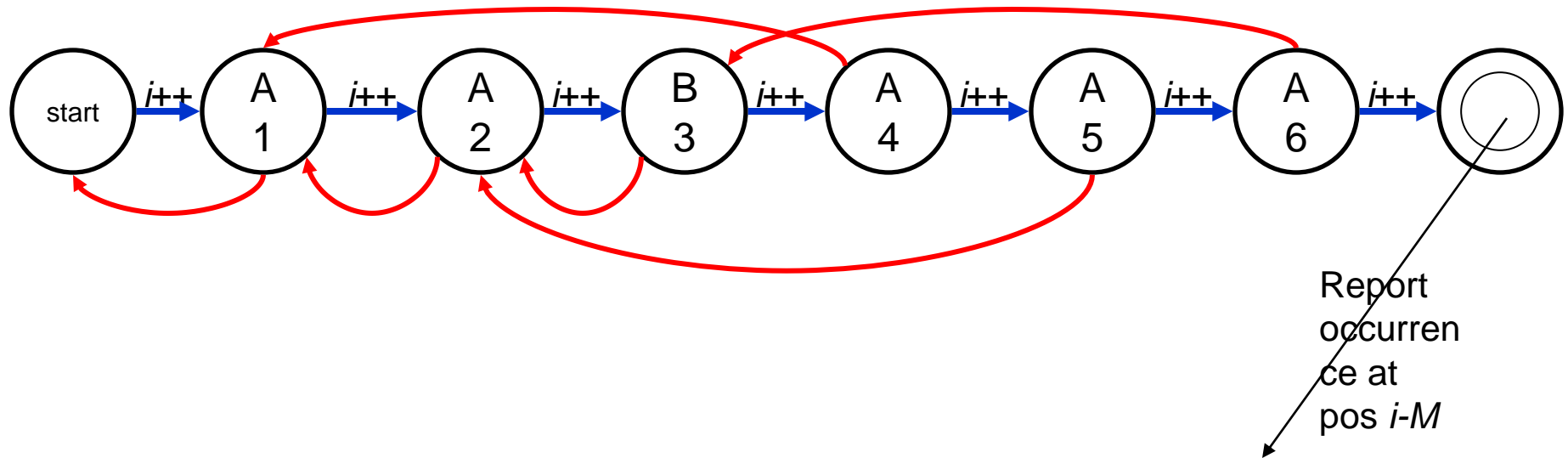


# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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

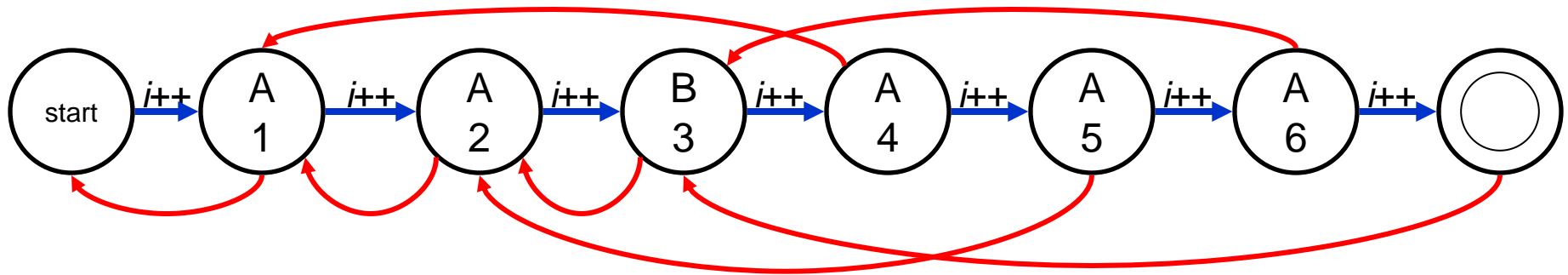


# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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

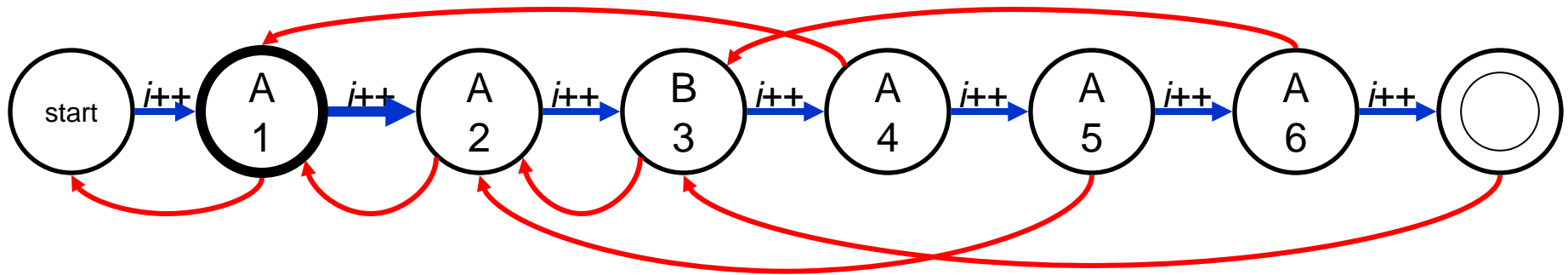


# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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



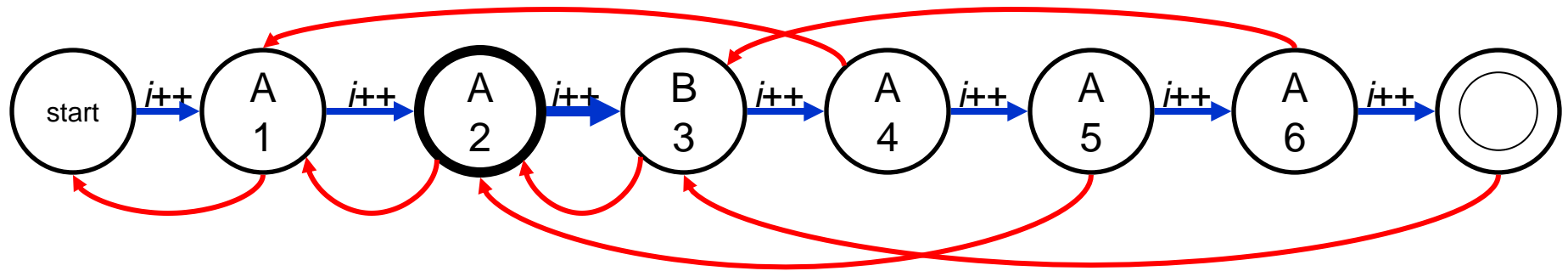
Example: streaming text  $T = \underline{a}aabaaaaabaaaa$  through the automaton

# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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



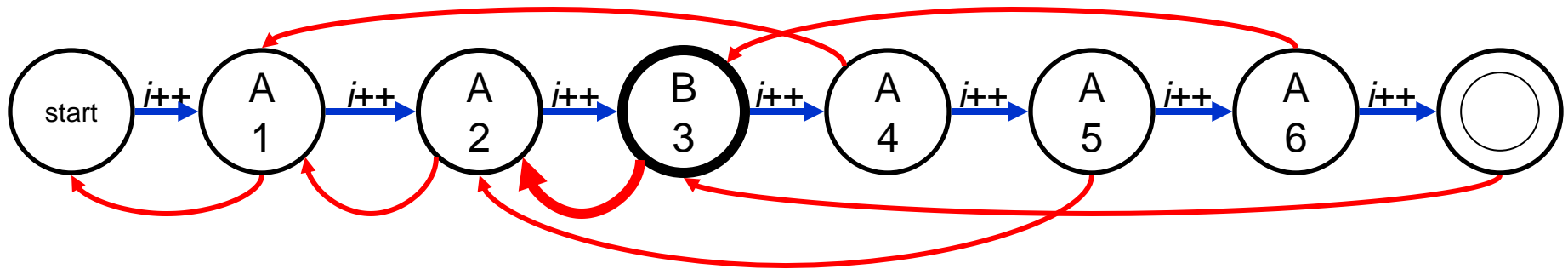
Example: streaming text  $T = aabaaaaabaaaa$  through the automaton

# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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



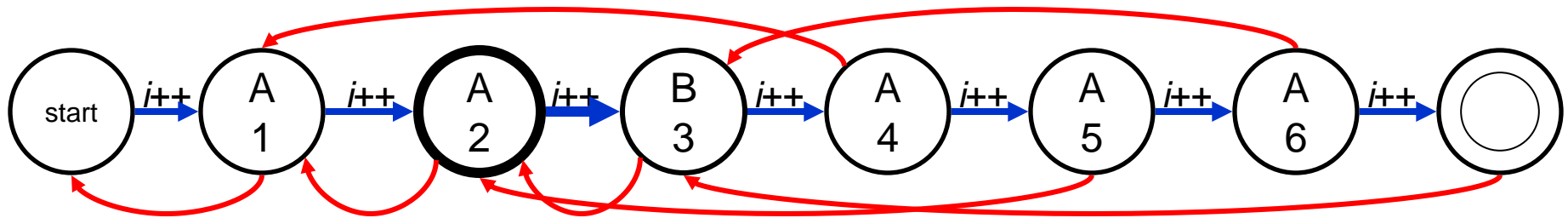
Example: streaming text  $T=aa**a**baaaaabaaaa$  through the automaton

# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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



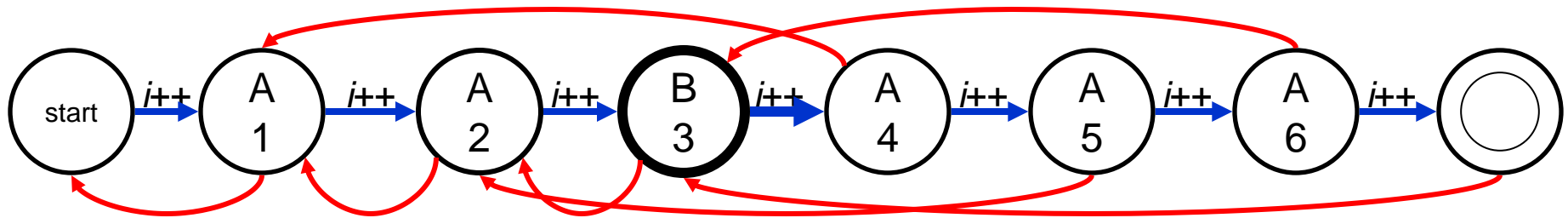
Example: streaming text  $T=aa**a**baaaaabaaaa$  through the automaton

# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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



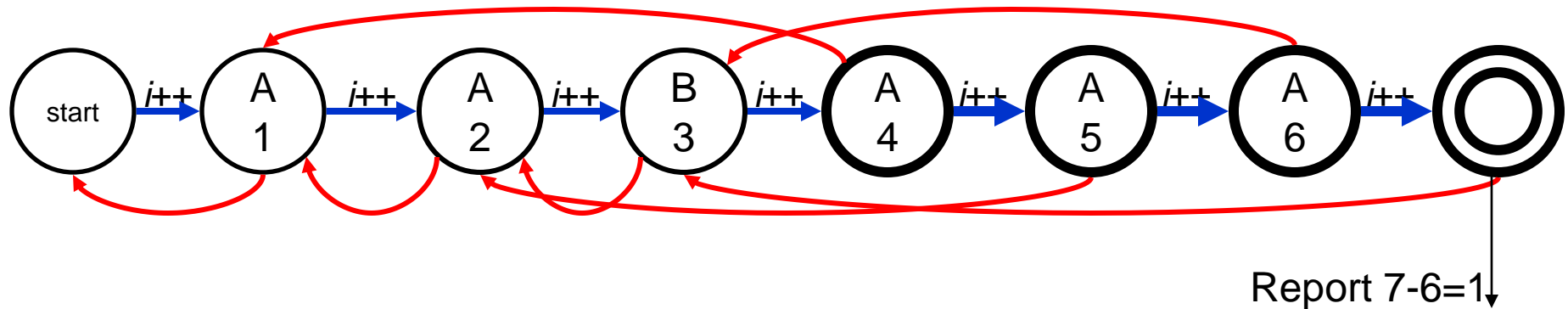
Example: streaming text  $T=aaabaaaaabaaaa$  through the automaton

# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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



Example: streaming text  $T=aaabaaaabaaaa$  through the automaton

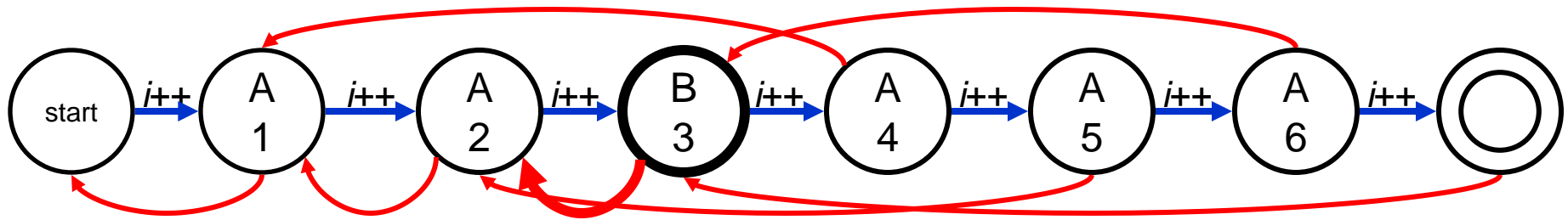


# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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



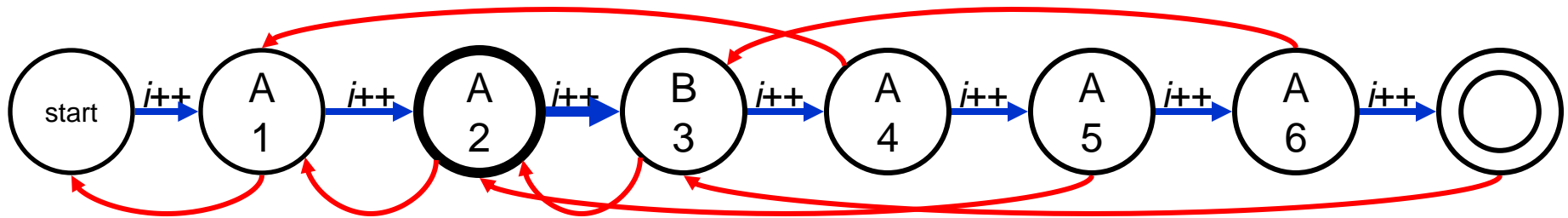
Example: streaming text  $T=aaabaaa$ **a**abaaaa through the automaton

# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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



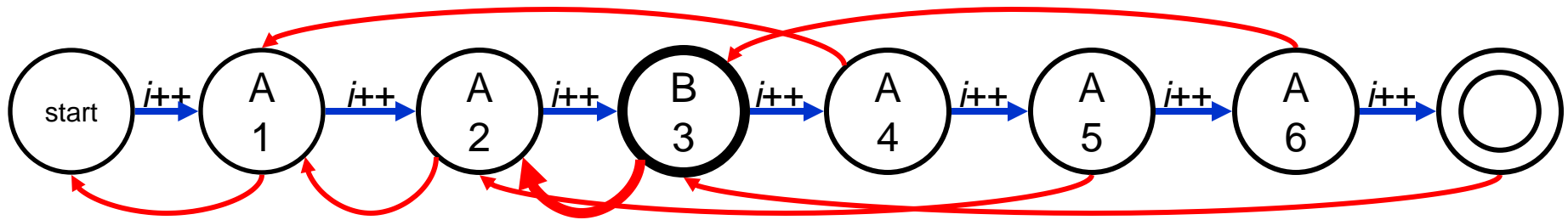
Example: streaming text  $T=aaabaaa$ **a** $abaaaa$  through the automaton

# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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



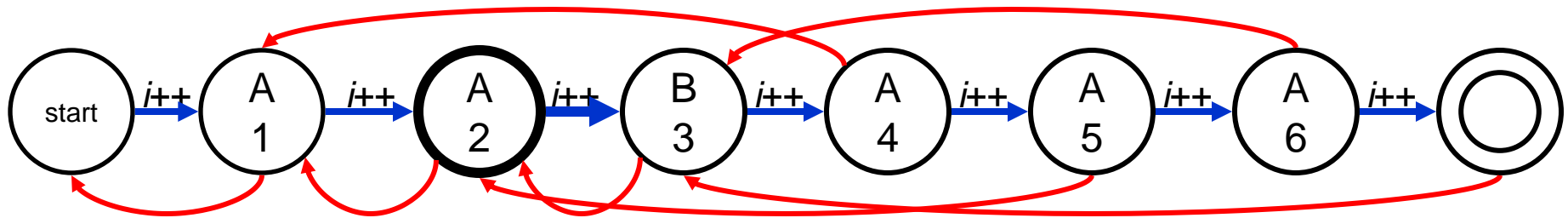
Example: streaming text  $T=aaabaaaa**a**baaaa$  through the automaton

# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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



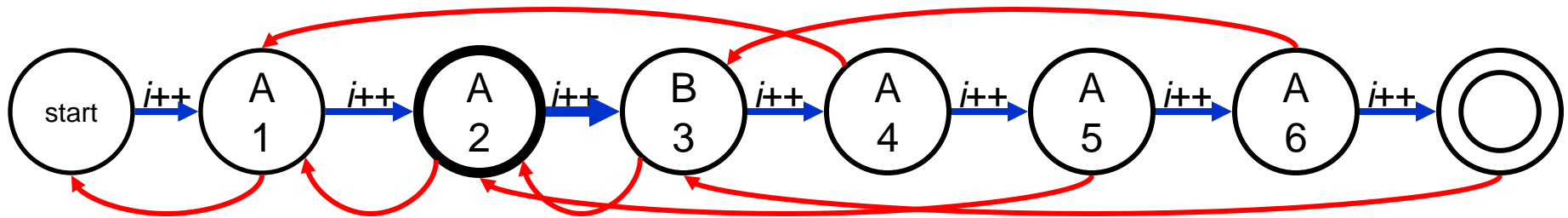
Example: streaming text  $T=aaabaaaa**a**baaaa$  through the automaton

# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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



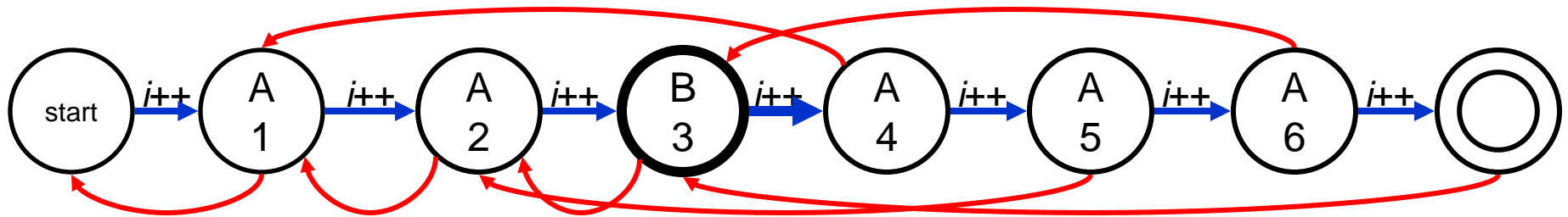
Example: streaming text  $T=aaabaaaa**a**baaaa$  through the automaton

# KMP pattern matching automaton

|          |          |          |          |          |          |
|----------|----------|----------|----------|----------|----------|
| 1        | 2        | 3        | 4        | 5        | 6        |
| <b>a</b> | <b>a</b> | <b>b</b> | <b>a</b> | <b>a</b> | <b>a</b> |

OF

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



Example: streaming text  $T=aaabaaaa**b**aaaa$  through the automaton

Etc...

---

# Automaton for a set of patterns

- The KMP automaton can be build for a set of patterns
  - In this case we are simultaneously finding the positions of several patterns in  $T$  by streaming  $T$  through the automaton
  - The automaton for a set of patterns is left as an exercise for you and may be chosen as a project (the Aho-Corasick algorithm)
-

---

# References

- [http://en.wikipedia.org/wiki/Knuth-Morris-Pratt\\_algorithm](http://en.wikipedia.org/wiki/Knuth-Morris-Pratt_algorithm)
  - <http://www.ics.uci.edu/~eppstein/161/960227.html>
  - Dan Gusfield. Algorithms on strings, trees, and sequences. Computer science and computational biology. Cambridge University press, 1999.
-