

Pattern search. KMP. Computing Overlap Function in time $O(M)$

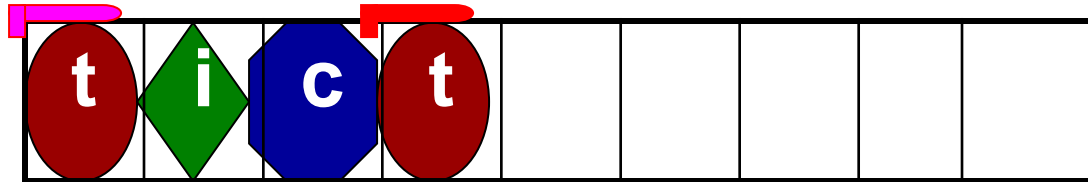
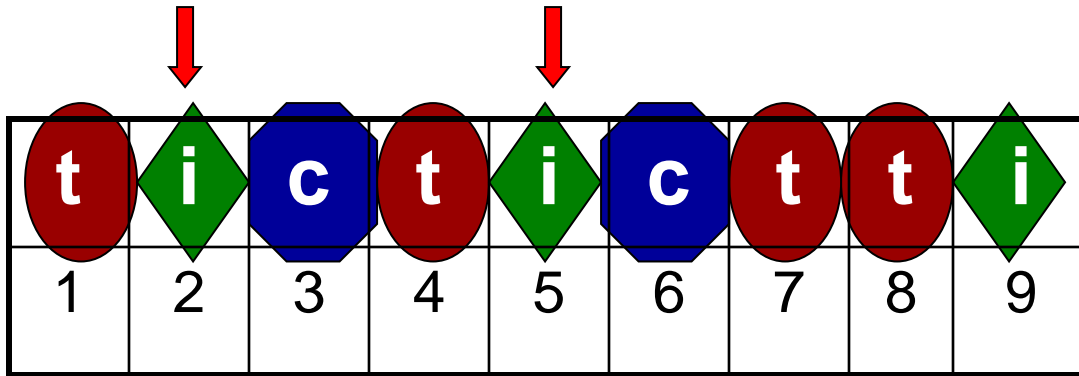
Lecture 2.2

by Marina Barsky

[Gusfield, Chapter 2.3.2](#)

Overlap function computation in
time $O(M)$

How to compute OF (overlap function)



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

Easy case:

if we have $OF(j-1)$,

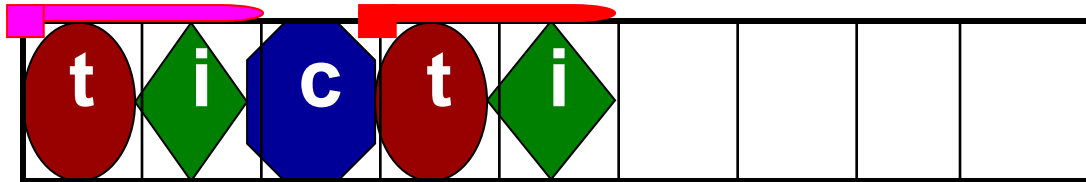
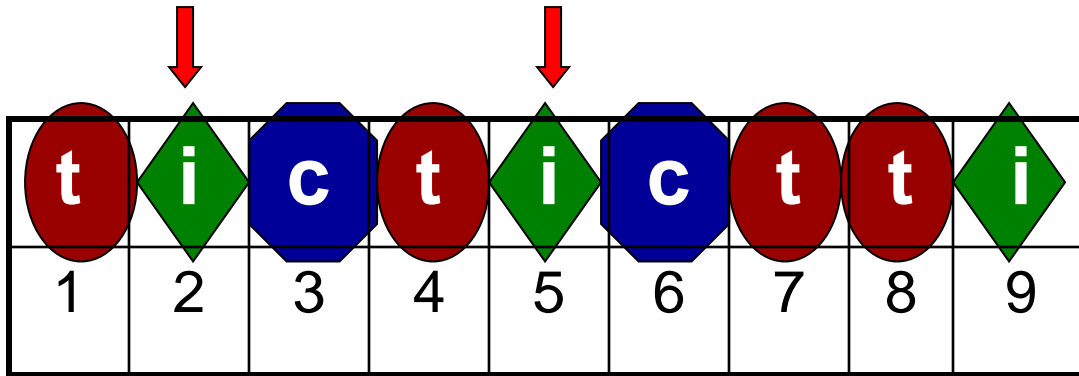
and the characters

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

Then we just increment by 1

$$OF(j) = OF(j-1) + 1$$

How to compute OF



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

Easy case:

if we have $OF(j-1)$,

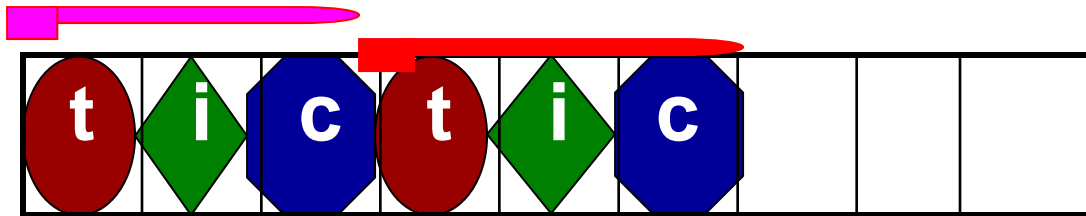
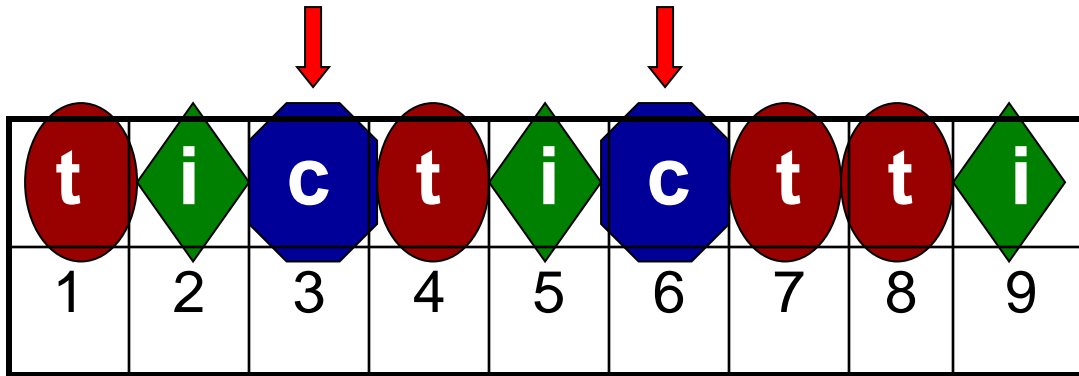
and the characters

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

Then we just increment by 1

$$OF(j) = OF(j-1) + 1$$

How to compute OF



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

Easy case:

if we have $OF(j-1)$,

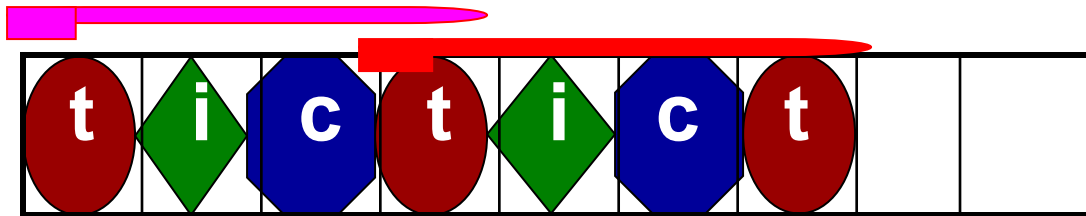
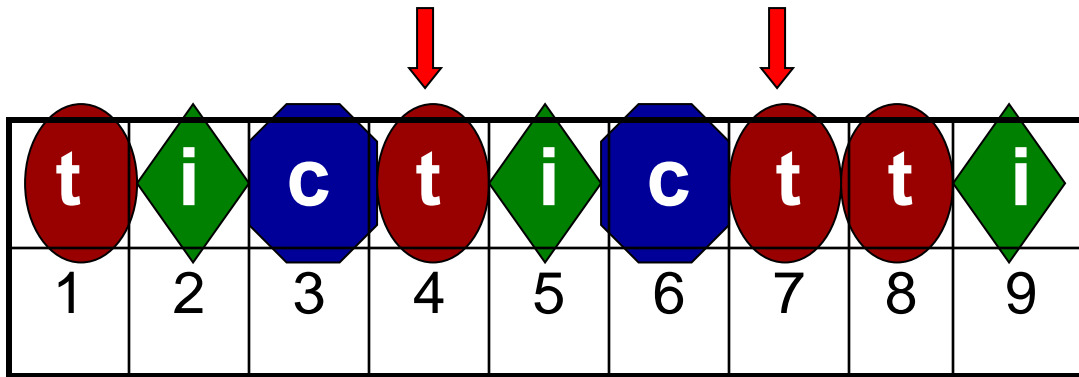
and the characters

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

Then we just increment by 1

$$OF(j) = OF(j-1) + 1$$

How to compute OF



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

Easy case:

if we have $OF(j-1)$,

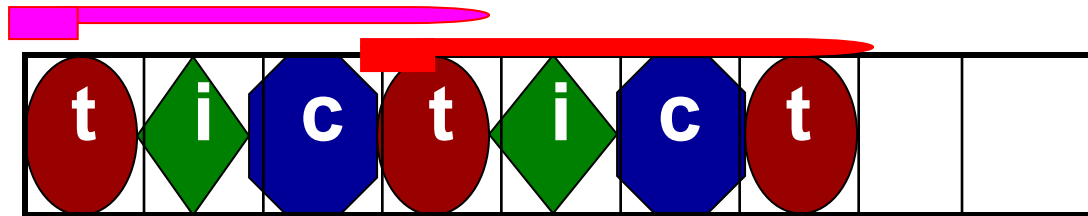
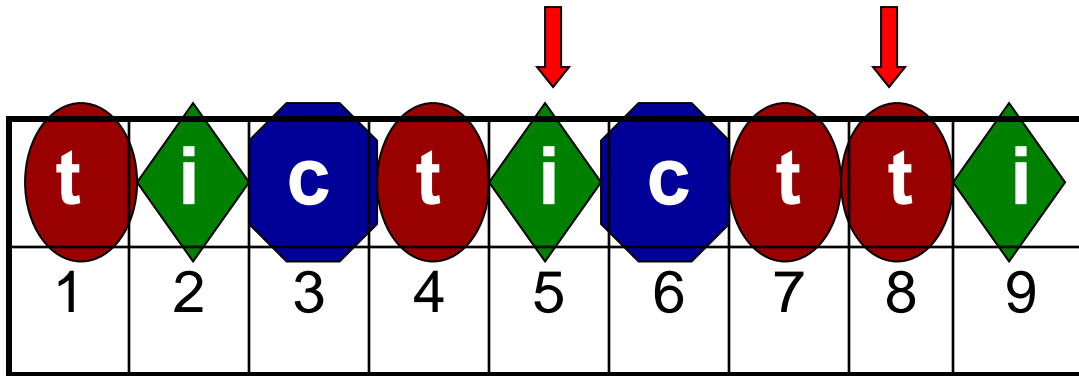
and the characters

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

Then we just increment by 1

$$OF(j) = OF(j-1) + 1$$

How to compute OF



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

General case:

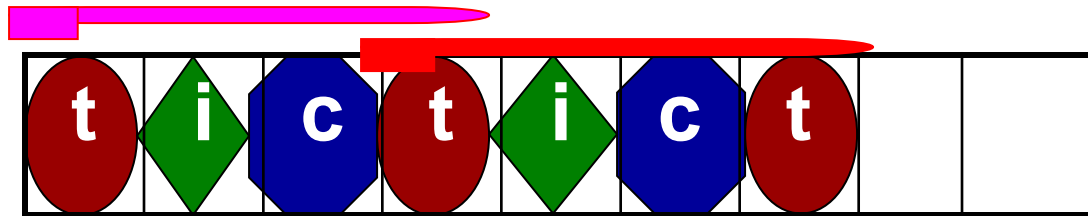
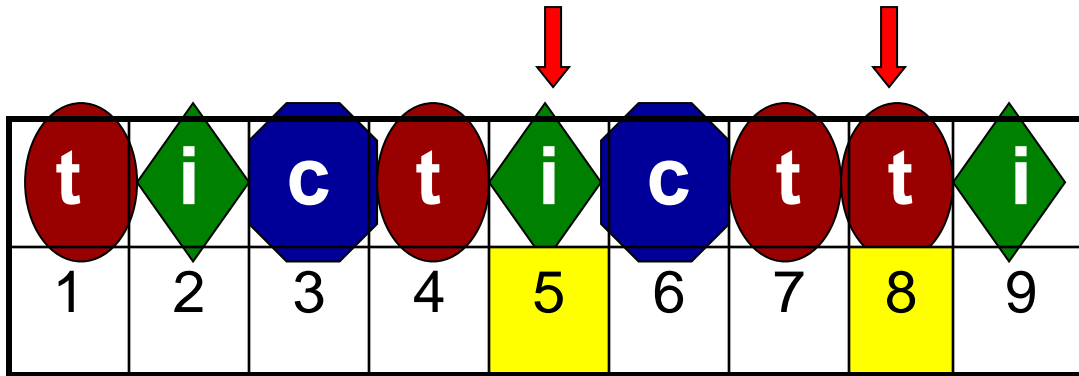
If the characters

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

do not match

- where do we find $OF[j]$?

How to compute OF



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

General case:

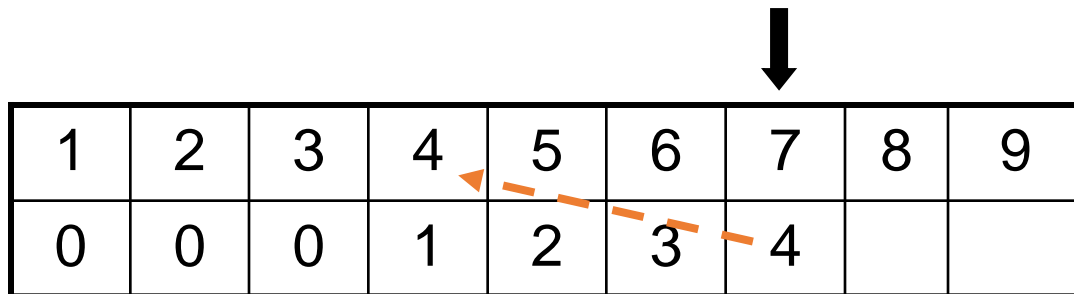
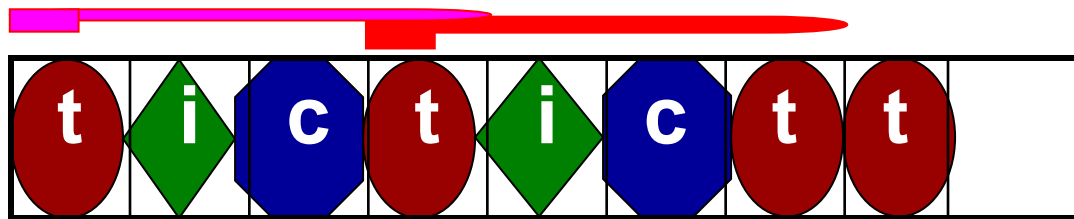
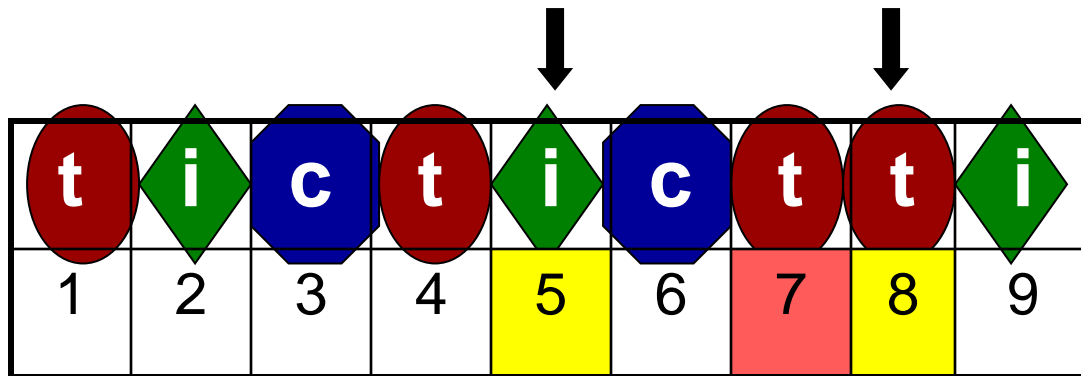
If the characters

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

do not match

- where do we find $OF[j]$?

How to compute OF



General case:

If the characters

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

do not match

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

The current overlap cannot be extended.

We are looking for a shorter prefix that matches last characters of a new suffix:

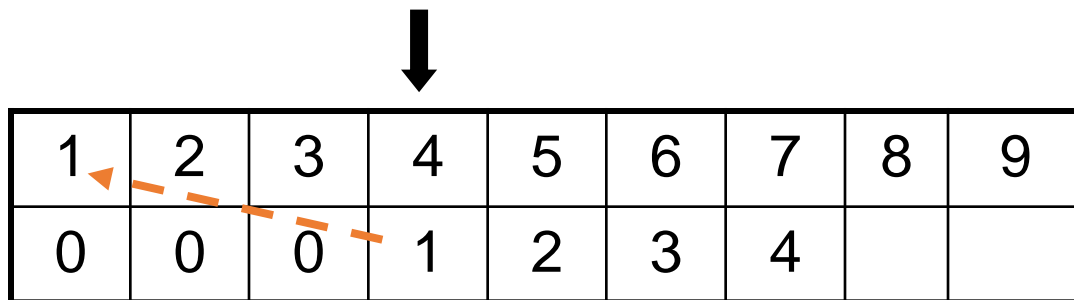
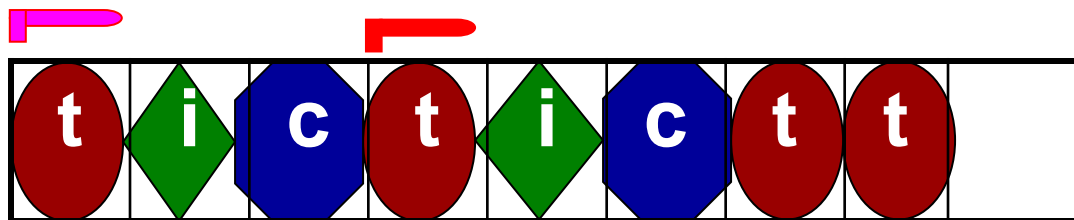
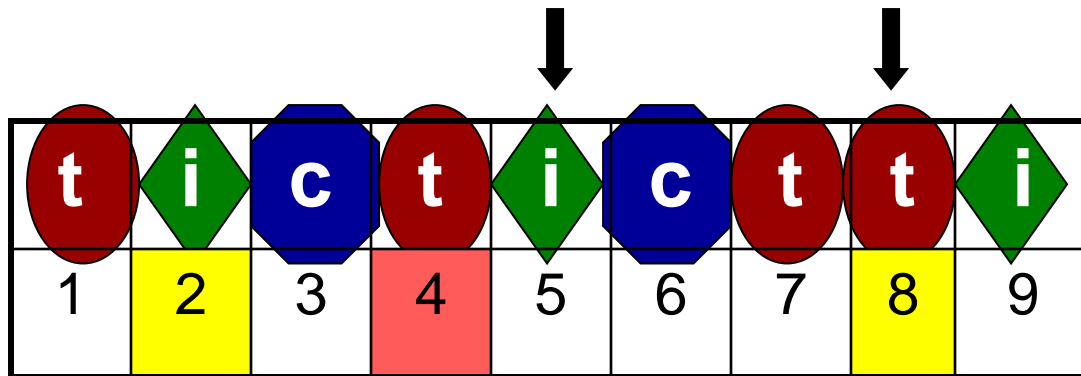
We look at $v = OF(j-1)$ and check again the next character

$P[OF(v)+1]$

$v = OF(7) = 4$

Maybe we can extend the overlap of substring $S[1:4]$

How to compute OF



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

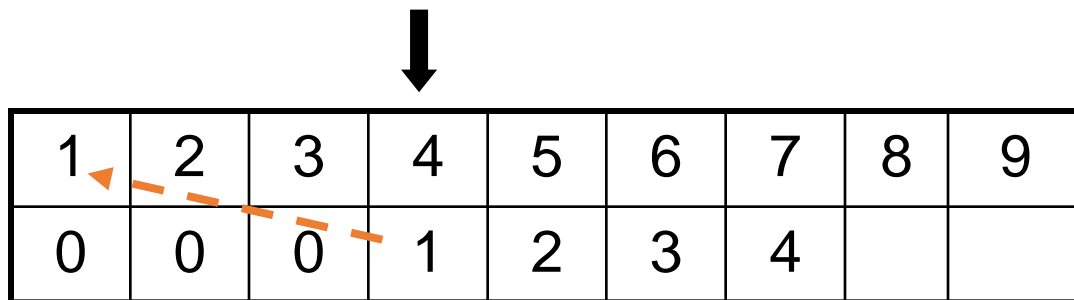
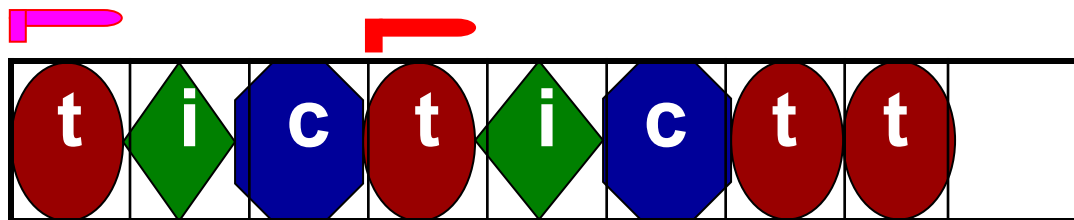
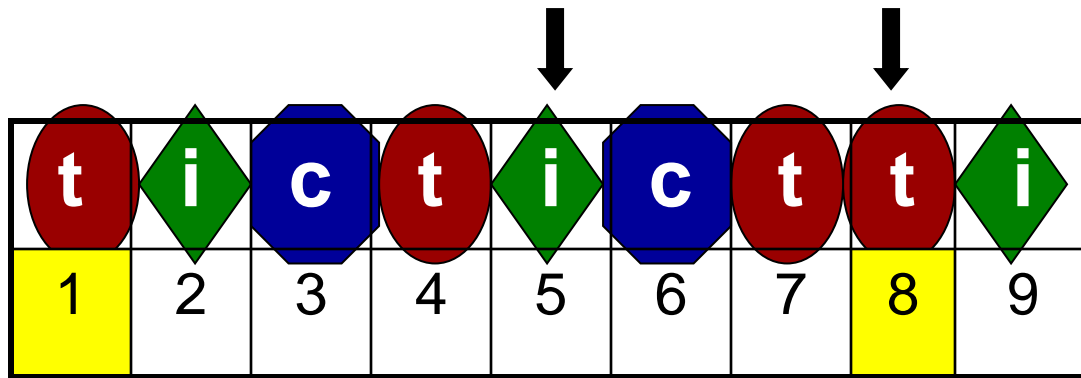
$P[OF(v)+1]$

$v = OF(7) = 4$

Maybe we can extend the overlap of substring $S[1:4]$

This substring 'tict' has overlap 1. We can extend it if the characters $S[2]$ and $S[8]$ match:
 $S[2] \neq S[8]$

How to compute OF



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

$P[OF(v)+1]$

$v = OF(1) = 0$

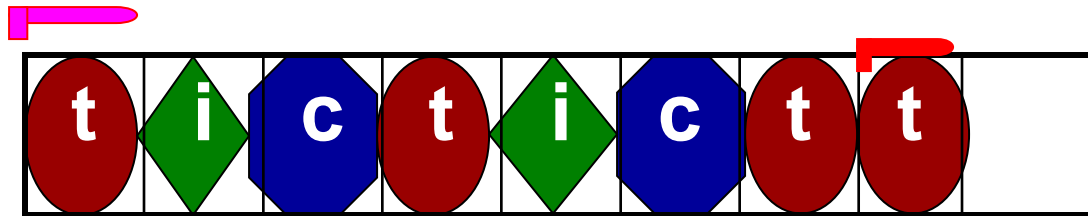
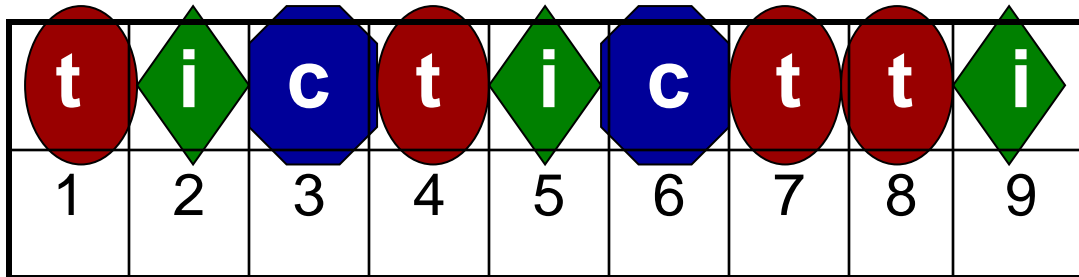
Maybe we can extend the overlap of substring $S[0:0]$

We can extend it if the characters $S[1]$ and $S[8]$ match:

$S[1] = S[8]$

New overlap is $0+1 = 1$

How to compute OF



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

General case:

If the characters

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

do not match

then 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 $v=OF(v)$

In other words

t	i	c	t	i	c	t	t	i
1	2	3	4	5	6	7	8	9

t	i	c	t	i	c	t	t	
1	2	3	4	5	6	7	8	9

1	2	3	4	5	6	7	8	9
0	0	0	1	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 substring *tict* with the prefix of the entire pattern, since the new overlap we are looking for is less than these 4 letters

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

In other words

t	i	c	t	i	c	t	t	i
1	2	3	4	5	6	7	8	9

t	i	c	t	i	c	t	t	
1	2	3	4	5	6	7	8	9

1	2	3	4	5	6	7	8	9
0	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 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
 $OF[8]=OF[1]+1=1$

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 M times
- in addition, the value of v is decreasing inside the *while* loop, but since v is never less than zero, the total number it is decreasing can not be more than the number it is increasing, therefore is bounded by M too.

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

Another example of the OF 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 $P[23+1]=t$

Another example of the OF 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
O L	0	0	0	1	2	0	1	2	3	4	5	0	1	2	3	4	5	6	7	8	9	1	1	?

The new 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

Another example of the OF 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
O L	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

Another example of the OF 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
O L	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 $2+1=3$

Check your understanding:

Practice jumps on the following pattern

- $P = \text{aaahamaaahamamaaahamaaaa}$

	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 L	0	1	2	0	1	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	8	9	?



Solution step 1

	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 L	0	1	2	0	1	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	8	9	?

Can overlap for $P[1:23]$ be extended for $P[1:24]$?

No, because $P[10] \neq P[24]$

We will check if we can extend overlap for $P[1:9]$

Solution step 2

	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 L	0	1	2	0	1	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	8	9	?

P[1:9] has overlap 3, can this be extended for P[1:24]?

No, because $P[4] \neq P[24]$

We will check if we can extend overlap for P[1:3]

Solution step 3

	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 L	0	1	2	0	1	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	8	9	?

P[1:3] has overlap 2, can this be extended for P[1:24]?

Yes, because P[3] = P[24]

Solution



	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 L	0	1	2	0	1	0	1	2	3	4	5	6	7	0	1	2	3	4	5	6	7	8	9	3



Overlap function computation (0-based)

```
of: = table of size M with all zeroes

of[0]: = 0 # first overlap is always 0

for pos from 1 to M -1:
    prev_overlap: = of[pos - 1]

    if P[pos] = P[prev_overlap]: # if next character is the same
        of[pos]: = prev_overlap + 1 # overlap becomes bigger

    else: # the suffix does not extend previous suffix
        while P[pos] != P[prev_overlap] and prev_overlap ≥ 1:
            # try extend a smaller prefix - based on P [of[pos-1]]
            prev_overlap: = of[prev_overlap - 1]

            if P[pos] = P[prev_overlap]:
                of[pos] = prev_overlap + 1
            # if we did not find any overlap to extend
            # then of[pos] remains 0

return of
```