CS 410/584, Algorithm Design & Analysis, Lecture Notes 7

String Searching

Where is the `ding` in `ramalamadingdong`?

text $x = a_1\ a_2\ \ldots\ a_n$
pattern $y = b_1\ b_2\ \ldots\ b_p$

```
i ← 0
while i < n-p do
     i ← i+1; j ← 1; k ← i
     while j ≤ p and a_k = b_j do



     if j = p+1 then write(          )
```


Matching Machine

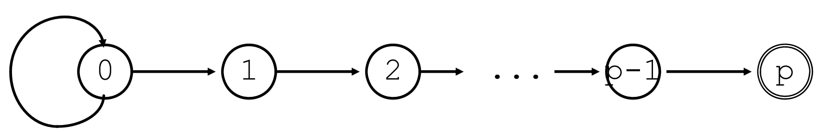
X | C | C | D | C | C | D | C | C | D | C | C |

Y | C | C | D | C | C | D | D | C |

Naive O(   )

Can do O(   )

at place i, have seen



Machine Operation

$b_1\ b_2 ..\ b_{j-1}\ b_j\ ...$

$a_1\ a_2\ a_3\ ....\ a_{k-1}\ a_k\ ...$

$b_1\ b_2\ .....\ b_{m-1}\ b_m\ ...\ b_p$

If $a_{k+1} = b_{j+1}$, go to
advance input to

If $a_{k+1} \neq b_{j+1}$ find

$a_1\ a_2\ a_3\ ....\ a_k\ a_{k+1}\ ...$

Finding Suffixes

How to get from j back to i?

<u>Failure function</u> f(j)

f(j) =

$b_1\ b_2\ ...\ b_{s-1}\ b_s\ ...$

$b_1\ b_2\ b_3\ ....\ b_{j-1}\ b_j\ ...$

Defining Failure Function

	C	C	D	C	C	D	D	C
i	1	2	3	4	5	6	7	8
f(i)								

Define $f^{(m)}(j)$

a) $f^{(1)}(j) =$

b) $f^{(m)}(j) =$

$f^{(2)}(5)$

Using Failure Function

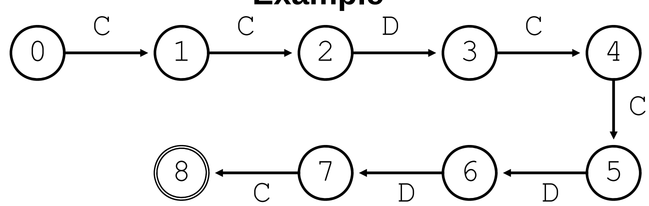
Have read $a_1 \; a_2 \ldots a_k$ of text, at state j and a_{k+1} b_{j+1}. Apply f repeatedly to j to find smallest m where

1. $f^{(m)}(j) =$

or

2. $f^{(m)}(j) =$

Example



Input
C C C D C C D C C D D C

How to compute f?
f(1) = 0

Computing Failure Function

Have f(1), f(2), ... f(j)
where f(j) = i

If $b_{j+1} = b_{i+1}$ then

If $b_{j+1} \neq b_{i+1}$, find smallest m where
1. $f^{(m)}(j) =$

or
2. $f^{(m)}(j) =$

Failure Function Algorithm

```
f(1) ← 0
for j = 2 to p do
   i ← f(j-1)
   while b_j ≠ b_{i+1} and i > 0 do

   if b_j ≠ b_{i+1} and i = 0 then

   else
```

j	1	2	3	4	5	6	7	8
	C	C	D	C	C	D	D	C
f(j)								

Complexity of Constructing Failure Function

Theorem: Failure function can be constructed in O(p) time.

Proof: number of executions of **for**-loop =

What about the **while**-loop?

Each time it executes

What can increment i?

Matching Algorithm: Knuth-Morris-Pratt

```
text x = a1 a2 ... an
pattern y = b1 b2 ... bp
f ← failure-function(y)
q ← 0
for i = 1 to n
 while q > 0 and bq+1 ≠ ai
  q ← f(q)
 if bq+1 = ai then
 if q = m then
```


Complexity of Matching Process

How many times can q change in processing x?

Claim: At most times

Note that q only increases if we match a new position of x.

So, charge each a_i \$2
- \$1 for
- \$1 for

When $q = j$, have at least in the bank.

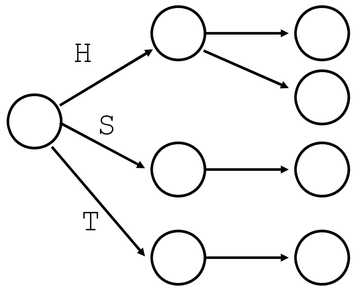
What is the maximum number of $q \leftarrow f(q)$ we can do at this point?

KMP complexity:

Variation: Multiple Patterns

Multiple seach strings

HE, HIM, SHE, HER, THEM, THEY



Variation: Backwards Match

Go through pattern right-to-left:

Positions p, p-1, p-2, …

If you get a mismatch at p-i, see if there is an earlier place in the pattern that has b_{p-i} … b_{p-1} b_p

Slide pattern forward to that point

Might never look at some characters in the text

Pattern Shift

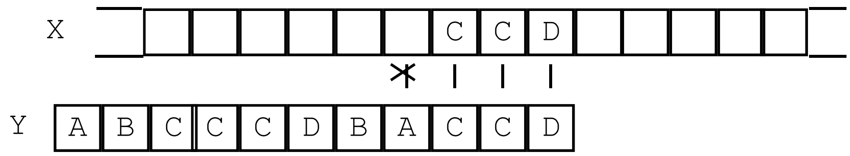


Figure out the shift for a mismatch at each position