

String Searching

Where is the ding in ramalamadingdong?

text $x = a_1 a_2 \dots a_n$

pattern $y = b_1 b_2 \dots b_p$

Naive

$i \leftarrow 0$

] offset into text

while $i < n-p$ **do**

$i \leftarrow i+1; j \leftarrow 1; k \leftarrow i$

while $j \leq p$ **and** $a_k = b_j$ **do**

$j \leftarrow j+1$

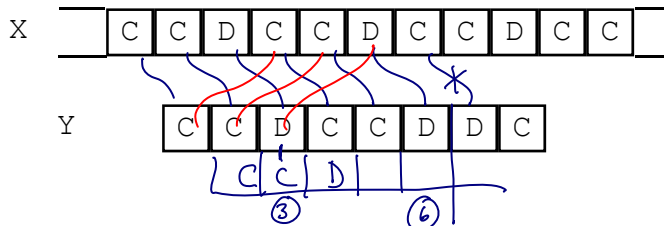
$k \leftarrow k+1$

if $j = p+1$ **then write** (found at i)

$O((n-p) \cdot p)$

index through pattern
index through text

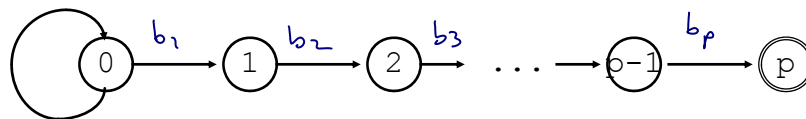
Matching Machine



Naive $O((n-p) \cdot p)$

Can do $O(n + p)$

at place i , have seen $b_1, b_2 \dots b_i$ in the text



Machine Operation

matched up to j

b_1	$b_2 \dots$	b_{j-1}	b_j	\dots	
a_1	a_2	$a_3 \dots$	a_{k-1}	a_k	\dots

for $m > j$ ~~b_1~~ $b_2 \dots b_{m-1} b_m \dots b_p$

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

If $a_{k+1} \neq b_{j+1}$ find largest i such
 that $b_1 \dots b_i$ is a suffix $b_1 \dots b_j$
 $b_1 \dots b_i$ $a_k a_{k+1} \dots$

Finding Suffixes

How to get from j back to i ?

Failure function $f(j)$

$f(j) =$ largest $s < j$
 such that $b_1 \dots b_s$ is a suffix of
 $b_1 \dots b_j$

b_1	$b_2 \dots$	b_{s-1}	b_s	\dots	
b_1	b_2	$b_3 \dots$	b_{j-1}	b_j	\dots

Defining Failure Function

	C	C	D	C	C	D	D	C
i	1	2	3	4	5	6	7	8
$f(i)$	0	1	0	1	2	3	0	1

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

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

b) $f^{(m)}(j) = f(f^{(m-1)}(j))$

$f^{(2)}(5) = f(f^{(1)}(5)) = f(f(5)) = f(2) = 1$

Using Failure Function

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

1. $f^{(m)}(j) = u \ \& \ a_{k+1} = b_{u+1}$

or

2. $f^{(m)}(j) = 0 \ \& \ a_{k+1} \neq b_1$

Algorithm Design & Analysis

Example

Input

$C \ C \ \bar{C} \ D \ C \ C \ D \ C \ C \ D \ D \ C$
0 1 2 2 3 4 5 6 4 5 6 7 8

How to compute f ?

$f(1) = 0$

Lecture Notes 7 David Maier 7

Algorithm Design & Analysis

Computing Failure Function

Have $f(1), f(2), \dots, f(j)$
 where $f(j) = i$

If $b_{j+1} = b_{i+1}$ then $f(j+1) = i+1$

If $b_{j+1} \neq b_{i+1}$, find smallest m where

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

or

2. $f^{(m)}(j) = 0$ and $b_{j+1} \neq b_1$ $f(j+1) = 0$

Lecture Notes 7 David Maier 8

Failure Function Algorithm

```

f(1) ← 0
for j = 2 to p do
  i ← f(j-1)
  while bj ≠ bi+1 and i > 0 do
    L i ← f(i)
  if bj ≠ bi+1 and i = 0 then
    L f(j) ← 0
  else f(j) ← i+1
    
```

	C	C	D	C	C	D	D	C
j	1	2	3	4	5	6	7	8
f(j)	0	1	0	1	2	3	0	1

Complexity of Constructing Failure Function

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

Proof: number of executions of for-loop = $p-1$ $O(p)$ - (exclusive of while-loop)

What about the while-loop?

Each time it executes we decrement i by 1, until we get to 0.
at least 1

What can increment i ?

$f(j) \leftarrow i+1$ then $i \leftarrow f(j-1)$ in the next loop iteration } happens at most $p-2$ times

(while loop is $O(p)$ over all executions $O(p) + O(p) = O(p)$)

Matching Algorithm: Knuth-Morris-Pratt

```

text x = a1 a2 ... an
pattern y = b1 b2 ... bp
f ← failure-function(y)
q ← 0 /* match counter
for i = 1 to n /* match L to R
  while q > 0 and bq+1 ≠ ai
    q ← f(q)
  if bq+1 = ai then q ← q + 1
  if q = mp then ("match at" i-p)
                 q ← f(q)

```

Complexity of Matching Process

How many times can q change in processing x ?

Claim: At most $2n$ times

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

So, charge each a_i \$2

\$1 for increment q
\$1 for bank

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

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

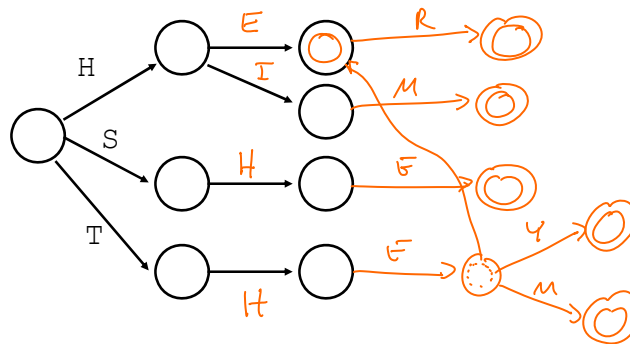
and we have enough \$ in bank to pay for that many decrements.

KMP complexity: $O(p + n)$

Variation: Multiple Patterns

Multiple search strings

HE, HIM, SHE, HER, THEM, THEY



Variation: Backwards Match

Go through pattern right-to-left:

Positions $p, p-1, p-2, \dots$

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

Slide pattern forward to that point

Might never look at some characters in the text

$O(p+n)$

Algorithm Design & Analysis

Pattern Shift

Figure out the shift for a mismatch at each position

Lecture Notes 7 David Maier **15**