0

String Matching

Given:

a short string Pattern P of size m

a long string Text T of size n

Find: if P is a substring of T

return index i if

$P[0] == T[i]$ & $P[1] == T[i+1]$ ....

Applications:

Simple nested loops algorithm $O(\quad)$?

?

Knuth - Morris Pratt

(1) KMP Algorithm $O(n)$

avoid back tracking in T

$P =$ "a b b b a b b b"

failure

$$
\begin{array}{lcccccccc}
T = \ldots\ldots\ldots & a & b & b & b & a & b & b & x \\
 & \shortparallel & \shortparallel & \shortparallel & \shortparallel & \shortparallel & \shortparallel & \shortparallel & \neq \\
P = & a & b & b & b & a & b & b & b
\end{array}
$$

$O(n\,m)$ algorithm would start ?

draw?

Can we do better? what do we already know?

all match !

$$
\begin{array}{lcccccccc}
T = & a & b & b & b & a & b & b & x \\
P = & a & b & b & b & a & b & b &
\end{array}
$$

(2) Failure function

start again $\Leftarrow F(j)$ fail at j

largest prefix of $P[0 \ldots j]$

such that

$P[0 .. F(j)]$ matches sufix of $P[0 .. j]$

$$\begin{array}{l} \qquad\qquad\qquad\qquad\qquad\quad j \\ \qquad\qquad\qquad\qquad\qquad\quad \downarrow \\ P = a\,b\,b\,b\,a\,b\,b\,x \\ \qquad\qquad\quad a\,b\,b\,\Box \\ \qquad\qquad\qquad\qquad\qquad\quad \uparrow \\ \qquad\qquad\qquad\qquad\qquad\quad f(j) \end{array}$$

start matching at $f(j)$ at failure i in Text T

Complexity ?

Best situation / problem

3 Boyer Moore

- pre process <u>characters</u> in P
- match P "backwards" in T

P = "r i t h m"

fail at t

T = a ␣ p a t t e r n ␣ m a t c h i n g a l g o r i t

P = r i t h m

start here → r i t h m ← line up here

last

④ Last - Occurrance Function

Σ alphabet = { a . . . . , }

P = "r i t h m"

Σ alphabet = a b c d e f g h i j k l m n o p q r s t u v w x

LO(j) -1 -1 -1 -1 -1 -1 -1 1 3 -1 -1 -1 0

?

Complexity ? worst/best case

Best problem distribution ?

5

Rabin Karp Algorithm

Revisit the simple nested loop alg.

for each substring length m in T
if equal(substring, P)
then found

Efficient equal function?

hashing
function

Recompute rolling hash