String Matching

Cristian López Del Alamo [1],
Milton Condori Fernández [2]
Jose Luis Sotomayor Malqui [2]

Universidad Nacional de San Agustín
Escuela Profesional de Ciencia de la Computación

[1]Universidad La Salle
[2]Universidad Nacional de San Agustín

Índice

Motivación

- A menúdo los programas de edición de texto tienen que encontrar todas las ocurrencias de un patrón brindado por el usuario en dicho texto.

Motivación

- A menúdo los programas de edición de texto tienen que encontrar todas las ocurrencias de un patrón brindado por el usuario en dicho texto.
- Procesamiento de cadenas de ADN requiere el reconocimiento de patrones de secuencias específicas de ADNs.

Motivación

- A menúdo los programas de edición de texto tienen que encontrar todas las ocurrencias de un patrón brindado por el usuario en dicho texto.
- Procesamiento de cadenas de ADN requiere el reconocimiento de patrones de secuencias específicas de ADNs.
- Buscadores de internet necesitan encontrar palabras específicas para poder brindas las paginas web correspondientes ante una consulta.

String Matching

Definición

Algoritmos que solucionan el problema de reconocer y ubicar patrones de cadenas dentro de un texto son denominados algoritmos de coincidencia(String Matching Algorithms)

Notación

- Texto $\rightarrow$ T[1..n]

Notación

- Texto $\rightarrow$ T[1..n]
- Patrón $\rightarrow$ P[1..m]

Notación

- Texto $\rightarrow$ T[1..n]
- Patrón $\rightarrow$ P[1..m]
- Se dice que existe un matching entre P y T si: Existe un desplazamiento:

$$s \,,\ 0 \leq s \leq n\text{-}m,$$

Notación

- Texto $\rightarrow$ T[1..n]
- Patrón $\rightarrow$ P[1..m]
- Se dice que existe un matching entre P y T si: Existe un desplazamiento:

$$s\ ,\ 0 \leq s \leq n\text{-}m,$$

tal que:

$$T[s+1..m] = P[1..m]$$

Notación

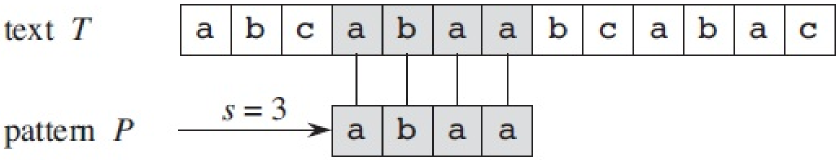
- Texto → T[1..n]
- Patrón → P[1..m]
- Se dice que existe un matching entre P y T si: Existe un desplazamiento:

$$s\ ,\ 0 \leq s \leq n\text{-}m,$$

tal que:

$$T[s+1..m] = P[1..m]$$

String Matching



Notación

- “w es prefijo de x”(w $\sqsubset$ x) si x = w.y (y$\varepsilon\Sigma$*).
  Si w $\sqsubset$ x luego, $\mid w \mid \leq \mid x \mid$
  Ejemplo:

$$ab \sqsubset abcac$$

Notación

- "w es prefijo de x" ($w \sqsubset x$) si $x = w.y$ ($y \varepsilon \Sigma^*$).
  Si $w \sqsubset x$ luego, $\mid w \mid \leq \mid x \mid$
  Ejemplo:
  $$ab \sqsubset abcac$$
- "w es sufijo de x" ($w \sqsupset x$) si $x = y.w$
  Ejemplo:
  $$cab \sqsupset abcab$$

Notación

- "w es prefijo de x" ($w \sqsubset x$) si $x = w.y$ ($y \varepsilon \Sigma^*$).
  Si $w \sqsubset x$ luego, $| w | \leq | x |$
  Ejemplo:
  $$ab \sqsubset abcac$$
- "w es sufijo de x" ($w \sqsupset x$) si $x = y.w$
  Ejemplo:
  $$cab \sqsupset abcab$$
- La notación P_k es equivalente a P[1..k], representa a los k caracteres prefijos de P

Notación

- "w es prefijo de x" ($w \sqsubset x$) si $x = w.y$ ($y \varepsilon \Sigma^*$).
  Si $w \sqsubset x$ luego, $| w | \leq | x |$
  Ejemplo:

$$ab \sqsubset abcac$$

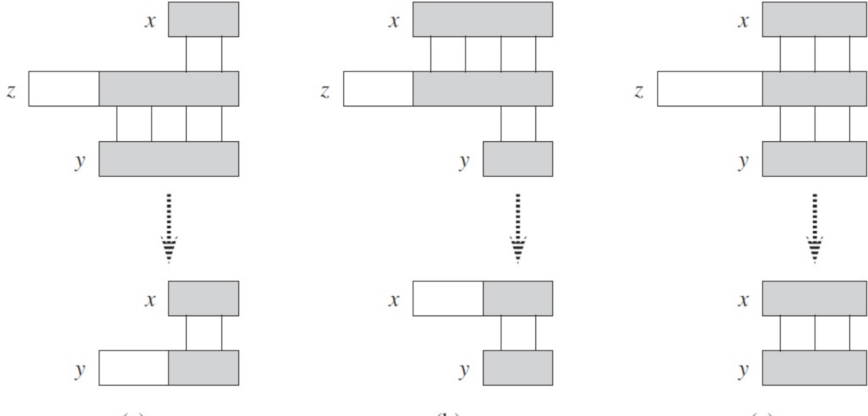
- "w es sufijo de x" ($w \sqsupset x$) si $x = y.w$
  Ejemplo:

$$cab \sqsupset abcab$$

- La notación P_k es equivalente a P[1..k], representa a los k caracteres prefijos de P
- T_k representa a los k caracteres prefijo de T

Notación

- x,y,z: cadenas tal que :
  $x \sqsupset z$ y $y \sqsupset z$. Si $|x|$ $|y|$, luego $x \sqsupset y$. Si $|x| \geq |y|$, luego $y \sqsupset x$. Si $|x|=|y|$, luego $x=y$



(a) (b) (c)

Algunas Soluciones

Algorithm	Preprocessing time	Matching time
Naive	0	$O((n-m+1)m)$
Rabin-Karp	$\Theta(m)$	$O((n-m+1)m)$
Finite automaton	$O(m\,\lvert\Sigma\rvert)$	$\Theta(n)$
Knuth-Morris-Pratt	$\Theta(m)$	$\Theta(n)$

Algoritmo obvio

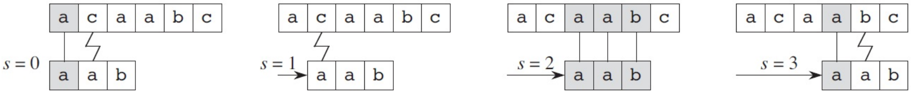
Este algoritmo es el que uno piensa naturalmente, posee un costo de computacional de $O((n-m+1).m)$ en el peor de los casos.

Algoritmo obvio

Este algoritmo es el que uno piensa naturalmente, posee un costo de computacional de O((n − m + 1).m) en el peor de los casos.

```
NAIVE-STRING-MATCHER(T, P)
n = T.length
m = P.length
for s = 0 to n − m
    if P[1..m] == T[s + 1..s + m]
        print "Pattern occurs with shift" s
```

Ejemplo



(a) (b) (c) (d)

Algoritmo de Rabin Karp

Este algoritmo, utiliza la matemática teórica para comparar 2 números en módulo de un tercero. Para esto asume que cada carácter del texto es un número entre 0 y 9.

Notación:

- Texto $\rightarrow$ T[1..n] $\rightarrow$ Valor decimal del texto del mismo tamaño que el patrón t_s.

Algoritmo de Rabin Karp

Este algoritmo, utiliza la matemática teórica para comparar 2 números en módulo de un tercero. Para esto asume que cada carácter del texto es un número entre 0 y 9.

Notación:

- Texto $\rightarrow$ T[1..n] $\rightarrow$ Valor decimal del texto del mismo tamaño que el patrón t_s.
- Patrón $\rightarrow$ P[1..m] $\rightarrow$ Valor decimal del patrón es p.

Algoritmo de Rabin Karp

Este algoritmo, utiliza la matemática teórica para comparar 2 números en módulo de un tercero. Para esto asume que cada carácter del texto es un número entre 0 y 9.

Notación:

- Texto $\rightarrow$ T[1..n] $\rightarrow$ Valor decimal del texto del mismo tamaño que el patrón t_s.
- Patrón $\rightarrow$ P[1..m] $\rightarrow$ Valor decimal del patrón es p.
- d $\rightarrow$ cantidad de números en el alfabeto.
  En general [0, 1, ... , d-1]

Algoritmo de Rabin Karp

Este algoritmo, utiliza la matemática teórica para comparar 2 números en módulo de un tercero. Para esto asume que cada carácter del texto es un número entre 0 y 9.

Notación:

- Texto → T[1..n] → Valor decimal del texto del mismo tamaño que el patrón t_s.
- Patrón → P[1..m] → Valor decimal del patrón es p.
- d → cantidad de números en el alfabeto.
  En general [0, 1, ... , d-1]
- q → Número primo con el que se calcula el módulo para hacer las comparaciones.

Preprocesamiento

Cálculo de p en O(m) usando la regla de Horner

p = P[m]+10(P[m-1]+10(P[m-2]+ ...10(P[2]+ 10 (P[2] + 10P[1])...))

valor de t_0 puede ser cálculado desde T[1..m] en O(m). Para calcular t_1, t_2,...,t_{n-m} en tiempo O(n-m) es suficiente observar que ts+1 puede ser calculado desde t_s en un tiempo constante:

Preprocesamiento

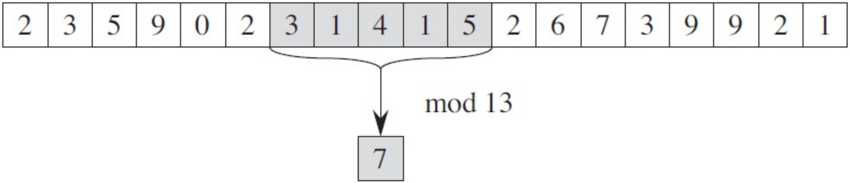
Cálculo de p en O(m) usando la regla de Horner

p = P[m]+10(P[m-1]+10(P[m-2]+ ...10(P[2]+ 10 (P[2] + 10P[1])...))

valor de t_0 puede ser cálculado desde T[1..m] en O(m). Para calcular t_1, t_2,...,t_{n-m} en tiempo O(n-m) es suficiente observar que ts+1 puede ser calculado desde t_s en un tiempo constante:

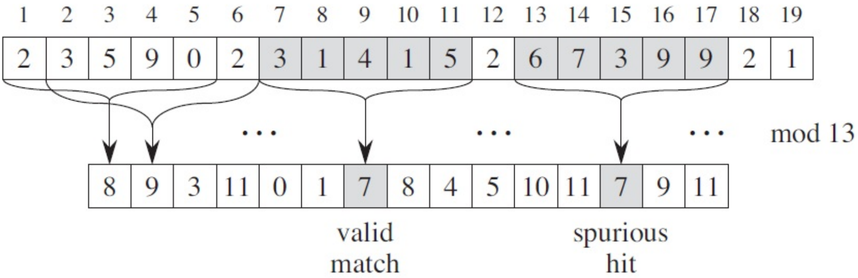
$$t_{s+1}=10(t_s - 10^{m-1}T[s+1]) + T[s+m+1]$$

Ejemplo - Parte 1/3



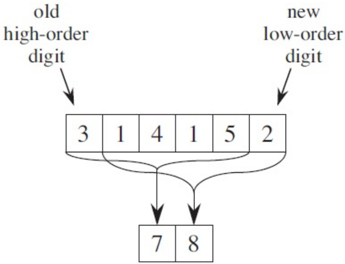
(a)

Ejemplo - Parte 2/3



(b)

Ejemplo - Parte 3/4



(c)

old high-order digit — shift — new low-order digit

$$
\begin{aligned}
14152 &\equiv (31415 - 3\cdot 10000)\cdot 10 + 2 \pmod{13} \\
&\equiv (7 - 3\cdot 3)\cdot 10 + 2 \pmod{13} \\
&\equiv 8 \pmod{13}
\end{aligned}
$$

Algoritmo del Automata Finito

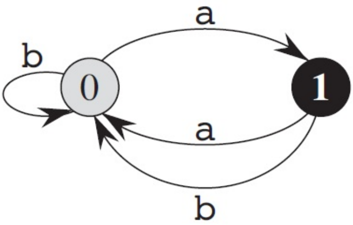
Este algoritmo se basa en la utilización de un autómata para encontrar un patrón. A medida que vamos leyendo caracteres, vamos avanzando de estado, y una vez que llegamos a un estado final significa que hubo matching.

Algoritmo del Automata Finito

Este algoritmo se basa en la utilización de un autómata para encontrar un patrón. A medida que vamos leyendo caracteres, vamos avanzando de estado, y una vez que llegamos a un estado final significa que hubo matching.

	input	
state	a	b
0	1	0
1	0	0

(a)



(b)

Notación de un Autómata Finito

Un autómata finito M se define mediante la tupla :

Notación de un Autómata Finito

Un autómata finito M se define mediante la tupla :

$$< \text{Q}, q_0 , \text{A} , \sum , \delta >$$

donde :

Notación de un Autómata Finito

Un autómata finito M se define mediante la tupla :

$$< \mathsf{Q},\, q_0 \,,\, \mathsf{A} \,,\, \sum \,,\, \delta >$$

donde :

- Q es un conjunto finito de estados,

Notación de un Autómata Finito

Un autómata finito M se define mediante la tupla :

$$< \mathsf{Q},\ q_0\ ,\ \mathsf{A}\ ,\ \sum\ ,\ \delta >$$

donde :

- Q es un conjunto finito de estados,
- $q_0 \in$ Q, es el estado inicial

Notación de un Autómata Finito

Un autómata finito M se define mediante la tupla :

$$< Q, q_0 , A , \sum , \delta >$$

donde :

- Q es un conjunto finito de estados,
- $q_0 \in Q$, es el estado inicial
- $A \subseteq Q$, conjunto de estados finales

Notación de un Autómata Finito

Un autómata finito M se define mediante la tupla :

$$< Q, q_0 , A , \sum , \delta >$$

donde :

- Q es un conjunto finito de estados,
- $q_0 \in Q$, es el estado inicial
- $A \subseteq Q$, conjunto de estados finales
- $\sum$, alfabeto de entrada

Notación de un Autómata Finito

Un autómata finito M se define mediante la tupla :

$$< Q, q_0 \ , A \ , \sum \ , \delta >$$

donde :

- Q es un conjunto finito de estados,
- $q_0 \in$ Q, es el estado inicial
- A $\subseteq$ Q, conjunto de estados finales
- $\sum$, alfabeto de entrada
- δ : Q x $\sum \rightarrow$ Q es la función de transición

Notación de un Autómata Finito

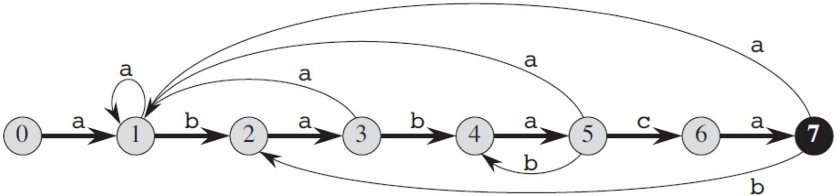
Un autómata finito M se define mediante la tupla :

$$< \text{Q}, q_0 \text{ , A , } \sum \text{ , } \delta >$$

donde :

- Q es un conjunto finito de estados,
- $q_0 \in$ Q, es el estado inicial
- A $\subseteq$ Q, conjunto de estados finales
- $\sum$, alfabeto de entrada
- δ : Q x $\sum \rightarrow$ Q es la función de transición
- $\phi(\varepsilon) = q_0$ ϕ(wa)= δ(ϕ(w),a) (transiciones sobre cadenas)

Ejemplo - Parte 1/2



(a)

Ejemplo - Parte 2/2

state	input a	input b	input c	P
0	1	0	0	a
1	1	2	0	b
2	3	0	0	a
3	1	4	0	b
4	5	0	0	a
5	1	4	6	c
6	7	0	0	a
7	1	2	0	

(b)

i	—	1	2	3	4	5	6	7	8	9	10	11
$T[i]$	—	a	b	a	b	a	b	a	c	a	b	a
state $\phi(T_i)$	0	1	2	3	4	5	4	5	6	**7**	2	3

(c)

Algoritmo

FINITE-AUTOMATON-MATCHER(T, δ, m)

```
n = T.length
q = 0
for i = 1 to n
    q = δ(q, T[i])
    if q == m
        print "Pattern occurs with shift" i − m
```

Preprocesamiento

Función sufijo correspondiente a un patrón P[1..m] :

σ : $\sum$* $\rightarrow$ { 0,1,...,m }

donde :

Preprocesamiento

Función sufijo correspondiente a un patrón P[1..m] :

$\sigma : \sum^* \rightarrow \{ 0,1,...,m \}$

donde :

- σ(x) es la longitud del prefijo más largo de P que es sufijo de x:

Preprocesamiento

Función sufijo correspondiente a un patrón P[1..m] :

σ : $\sum$* $\rightarrow$ { 0,1,...,m }

donde :

- σ(x) es la longitud del prefijo más largo de P que es sufijo de x:
- σ(x) = { k / $P_k \sqsupset$ x }

Ejemplo

Si P = ab

- $\sigma(\varepsilon) =$

Ejemplo

Si P = ab

- $\sigma(\varepsilon) = 0$
- $\sigma(\text{ccaca}) =$

Ejemplo

Si P = ab

- $\sigma(\varepsilon) = 0$
- $\sigma(\text{ccaca}) = 1$
- $\sigma(\text{ccaab}) =$

Ejemplo

Si P = ab

- $\sigma(\varepsilon) = 0$
- $\sigma(\text{ccaca}) = 1$
- $\sigma(\text{ccaab}) = 2$

$P_0 = \varepsilon$ es sufijo de cualquier x

Dado un patrón P de longitud m

$$\sigma(\text{x}) = \text{m}$$

si y sólo sí $\text{P} \sqsupset \text{x}$.

Definicion de AF

Definición del AF para P[1..m]

Definicion de AF

Definición del AF para P[1..m]

- Q es

Definicion de AF

Definición del AF para P[1..m]

- Q es { 0,1,...,m }

Definicion de AF

Definición del AF para P[1..m]

- Q es { 0,1,...,m }
- q_0 es

Definicion de AF

Definición del AF para P[1..m]

- Q es { 0,1,...,m }
- q_0 es 0

Definicion de AF

Definición del AF para P[1..m]

- Q es { 0,1,...,m }
- q_0 es 0
- A $=$

Definicion de AF

Definición del AF para P[1..m]

- Q es { 0,1,...,m }
- q_0 es 0
- A = { m }

Definicion de AF

Definición del AF para P[1..m]

- Q es { 0,1,...,m }
- q_0 es 0
- A = { m }
- δ(q,a) =

Definicion de AF

Definición del AF para P[1..m]

- Q es { 0,1,...,m }
- q_0 es 0
- A = { m }
- δ(q,a) = $\sigma(P_q$a)

Definicion de AF

Definición del AF para P[1..m]

- Q es { 0,1,...,m }
- q_0 es 0
- A = { m }
- δ(q,a) = $\sigma(P_q$a)
- El AF mantiene invariante las transiciones:

$$\phi(T_i) = \sigma(T_i)$$

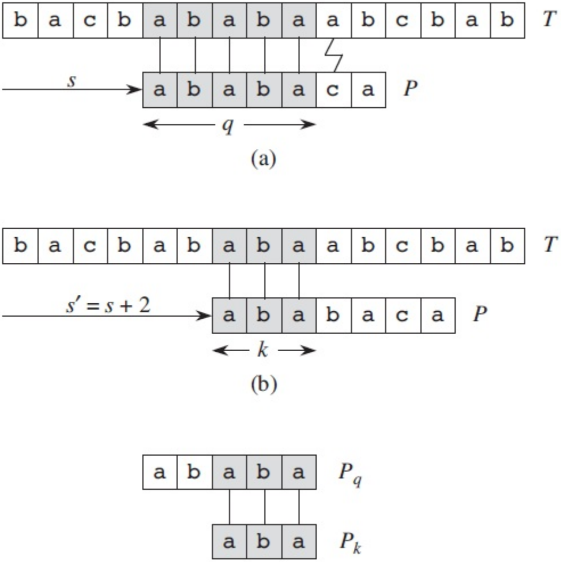
Algoritmo

```
COMPUTE-TRANSITION-FUNCTION(P, Σ)
m = P.length
for q = 0 to m
    for each character a ∈ Σ
        k = min(m + 1, q + 2)
        repeat
            k = k − 1
        until P_k ⊐ P_q a
        δ(q, a) = k
return δ
```

Algoritmo de Knuth Morris Pratt

Este algoritmo, utiliza un arreglo auxiliar kmpNext[0 ... m-1] el cual es generado con un costo de O(m). Este arreglo brinda información adicional sobre el patrón a la hora de hacer las comparaciones, permite saber el corrimiento que hay que hacer en el patrón hasta la próxima comparación evitando preguntas innecesarias. Generando así un algoritmo eficiente de costo O(n + m).

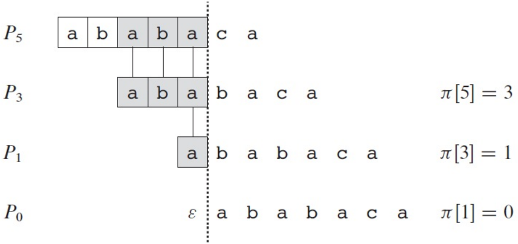
Ejemplo


b a c b a b a b a a b c b a b
T
s
a b a b a c a
P
q
(a)
b a c b a b a b a a b c b a b
T
s′ = s + 2
a b a b a c a
P
k
(b)
a b a b a
P_q
a b a
P_k

Ejemplo

i	1	2	3	4	5	6	7
$P[i]$	a	b	a	b	a	c	a
$\pi[i]$	0	0	1	2	3	0	1

(a)



(b)