O problema do *Pattern matching*

Antonio Fariña Martínez

Motivación

- A evolución tecnolóxica na área da computación e o amplio uso de Internet, están a propiciar un considerable aumento da información textual:
  - **Medios tecnolóxicos como:**
    - **Bibliotecas dixitais**
    - **Xornais Dixitais**
    - **Bases de Datos de Texto …**
- Existen estructuras de indexación (p.ex: índices invertidos, arrays de sufixos) que facilitan a recuperación …
- A busca de patróns permite acceder rápidamente a zonas de interés
  - **Ex1: Dado un documento ver exactamente en que posición aparece unha palabra (un índice invertido podería conter xa esta información)**
  - **Ex2: Cantas veces aparece a cadea "ssipi" nun texto ?**

Pattern matching

Definición do problema:

Dados

- Un patrón $\boldsymbol{P = p_1 \ldots p_m}$ (composto de m símbolos) e,

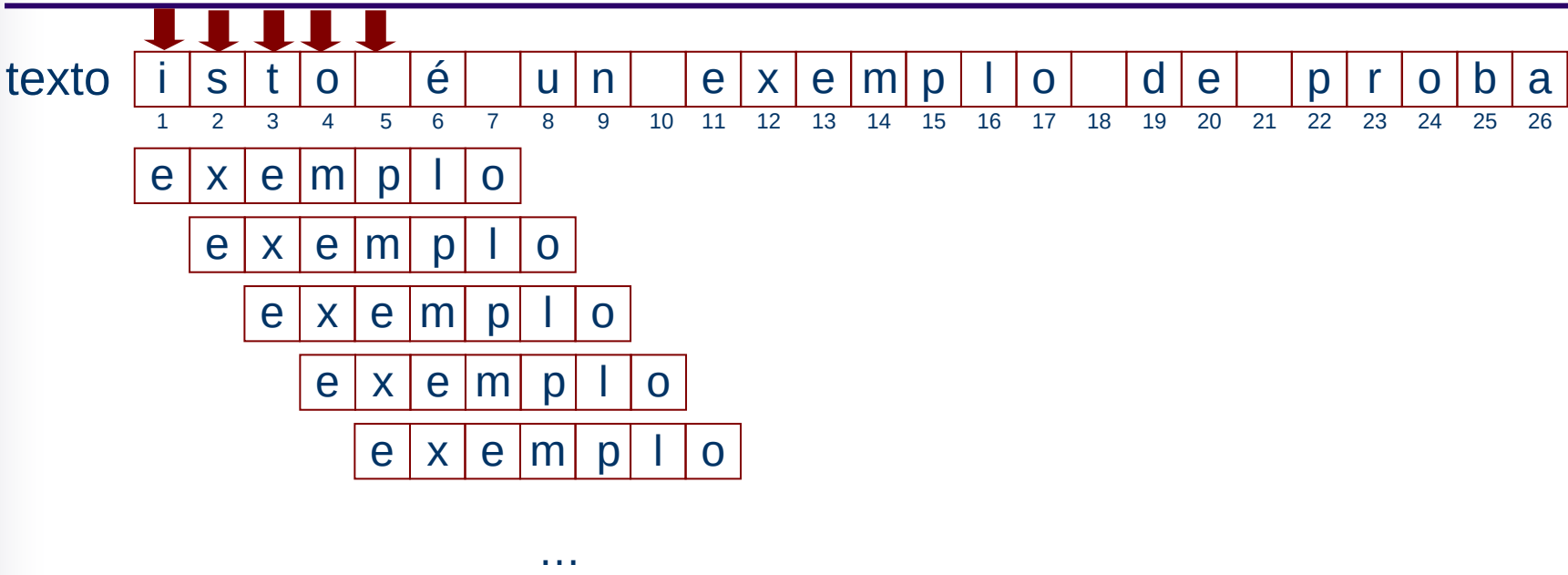
- Un texto $\boldsymbol{T = t_1 \ldots t_n}$, ambas secuencias de símbolos sobre un alfabeto finito **Σ**,

Preténdese

Encontrar todas as coincidencias de *P* en *T*; isto é:

Encontrar o conxunto {|x|, $T = \boldsymbol{x}Py$}

Forma trivial: Forza Bruta.



Pattern matching

Xeralización: Pattern matching permitindo erros.

Dados

- Un patrón $\boldsymbol{P = p_1 \ldots p_m}$,un texto $\boldsymbol{T = t_1 \ldots t_n}$, secuencias de caracteres sobre un alfabeto finito **Σ**, e

- Unha marxe de erro ***K*** *(distancia entre un patrón e as súas coincidencias no texto: P. exemplo distancia de edición).*

Preténdese

Encontrar todas as coincidencias (<= marxe de erro) de *P* en *T*; isto é, atopar {|x|, *T*= **x**P'y *e ed(P,P') <=k}*

Tipos de patróns que se poden buscar

- **i)** Palabras exactas. P.ex: gato, can, etc.
- **ii)** Prefixos e sufixos. Palabras que comezan ou rematan con un **prefixo** ou sufixo dado. P.ex: [ante]-proxecto, [ante]-rior, Protec[ción], construc[ción]…
- **iii)** Subcadeas. Obter todas as palabras que conteñan unha cadea de caracteres. P.ex: patrón "**-sió-"** representa words como: compre[sió]n, pa[sió]n, tor[sió]n etc.
- **iv)** Rangos alfabéticos. Representan a todos os termos contidos un rango determinado (asumindo unha orde alfanumérica).

  P.ex1: [a, b) devolve as palabras que comezan por 'a'

  P.ex2: (zodiac, zona] devolve palabras como zombie, zona.

Tipos de patróns que se poden buscar

- **v)** Patróns permitindo erros. Permiten recuperar patróns que están a unha distancia de edición menor que o patrón de busca.

  P.ex1: buscar "**hot**" con 1 erro devolve as palabras:

  - **hot, hit, hat, pot, rot, hop, shot**

  Inserciónss, borrados ou substitucións de símbolos nunha determinada posición

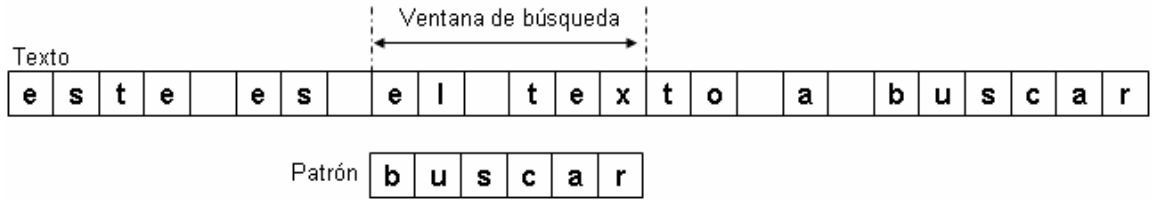
- **vi**) Expresións regulares. Inclúen todas as palabras que poden ser construidas a partires dunha palabra inicial e 3 operacións básicas:

  - **unión (e1|e2), concatenación (e1e2), e repetición (e*) 0+ veces do patrón e1, e2, e.**

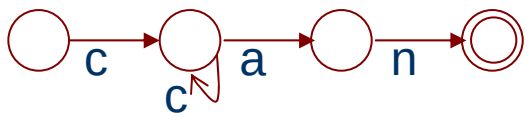
  P. ex1: (red)*(car|bike)(0|1)* atopará termos como: car, car0, car1, car01, bike, redbike, ...

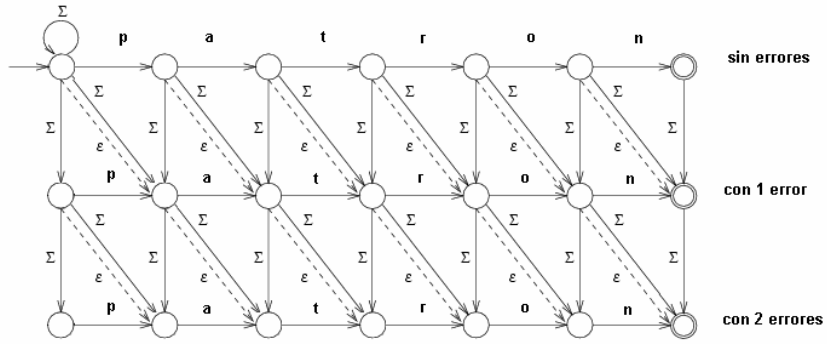
Técnicas de pattern matching.

- **Funcionamento básico:**
  - Utilización dunha “xanela de busca” que se vai deslizando sobre o texto, e dentro da cal se busca o patrón.



  - Utilización de autómatas, que cambian de estado con cada carácter lido.





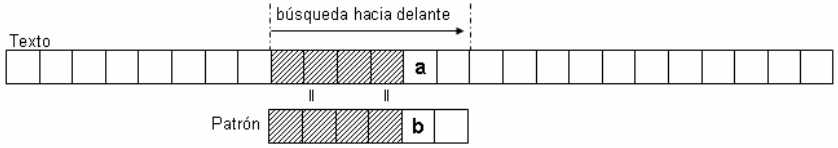
  - Outros: programación dinámica…

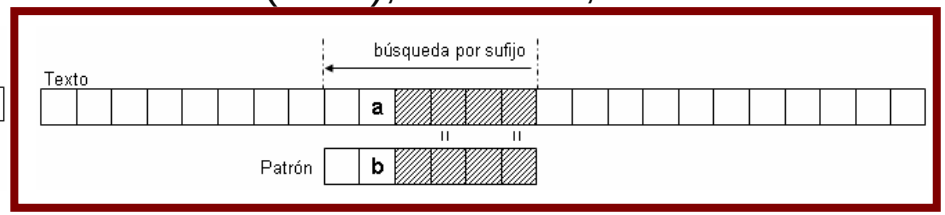
Clasificación das técnicas de pattern matching

- **Dependendo do número / tipo de patróns que permiten buscar:**
  - Monopatrón:
    - Só se busca un único patrón P a través do texto T.
  - Multipatrón:
    - Permiten buscar varios patróns P', P'', … P^r ao longo do texto T.
  - Aproximadas ou permitindo erros (cun marxe de erro <=k).
    - Atopan dos os patróns "similares" a un patrón dado no texto T.
    - As solución más rápidas utilizan programación dinámica, ou baséanse na construcción de autómatas (de forma que os estados terminais indican o recoñecemento dun patrón, e cada vez que se le un carácter do texto o autómata fai unha transición de estado).

Clasificación das técnicas de pattern matching

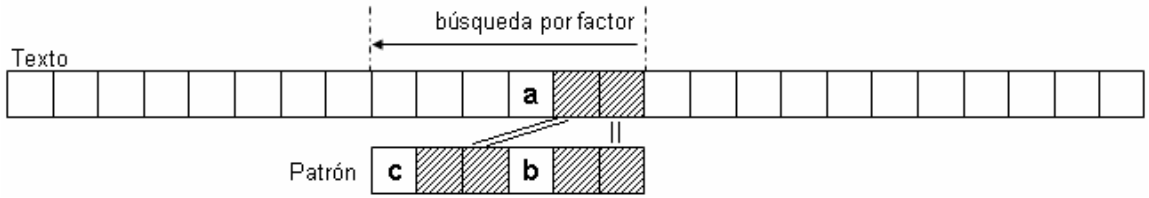
- **Clasificación pola forma que teñen de buscar no texto:**
  - Busca cara adiante na ventá: Buscan prefixos do patrón de busca o máis largos posible dentro da ventá
    - Exemplos moi coñecidos: Knuth-Morris-Pratt (KMP), Shift-And, Shift-Or…





  - Busca cara atrás: Buscan sufixos do patrón de busca dentro da ventá. Isto permite facer "saltos" no texto, evitando procesar todos os "caracteres" do texto.
    - Exemplos típicos: Boyer-Moore, Horspool, etc.

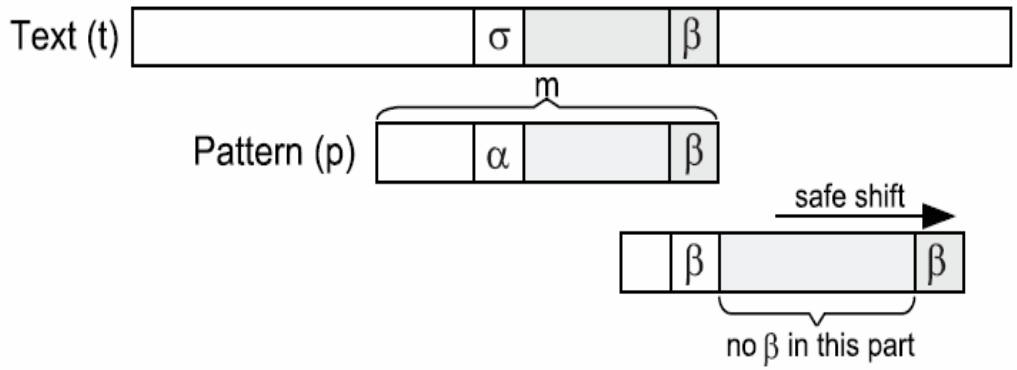
  - Busca por factor: Buscan cara atrás dentro da venta, o sufixo máis largo da ventá que tamén é un factor do patrón de busca
    - Exemplos: Backward Oracle Matching algorithm (BOM)



Familia Boyer-Moore: Horspool

Busca cara atrás na ventá.

- Se emparellan os m símbolos do patrón → ocurrencia de P.
- Se o emparellamento non se produce (σ != α) desplázase a ventá cara á dereita (**shift** ou salto) un número de símbolos que dependerá do último símbolo da ventá (β).
- Desplázase a ventá *shift(β)* símbolos á dereita.



De que depende o shift ?

Horspool: Preprocesado ou determinación do salto

- **Depende só do patrón buscado.**
- **Para cada símbolo do alfabeto que pode aparecer no texto → indica o salto a realizar.**

```
preprocessHorspool (d, p, m, t, Σ)
(1)  for k ∈ Σ do d[k] ← m;
(2)  for i ∈ 1...m − 1 do d[p[i]] ← m − i;
```

- **Saltar m, en caso de que un símbolo non aparezca no patrón *p*.**
- **Se un símbolo do alfabeto aparece no patrón o salto é menor.**

Horspool: fase de busca.

```
preprocessHorspool (d, p, m, t, Σ)
(1)  for k ∈ Σ do d[k] ← m;
(2)  for i ∈ 1...m − 1 do d[p[i]] ← m − i;
```

```
Searching phase(p, m, t, n, Σ, d)
(1)  pos ← 1;
(2)  while pos < n − m
(3)       j ← m;
(4)       while j > 0 and t_{pos+j} = p_j
(5)            j ← j − 1;
(6)       if j = 0 then
(7)            occurrence appeared at position pos;
(8)       pos ← pos + d[t_{pos+m}];
```

- **Recorrer o texto.**
- **Comparar ventá e patrón (cara atrás)**
- **Sinalar ocurrencias**

Horspool. Exemplo. Fase Preprocesado

preprocessHorspool (d, p, m, t, Σ)

(1) **for** $k \in \Sigma$ **do** $d[k] \leftarrow m$;

(2) **for** $i \in 1 \ldots m-1$ **do** $d[p[i]] \leftarrow m - i$;

Σ	␣	a	b	d	e	i	l	m	n	o	p	r	s	t	é

texto	i	s	t	o	␣	é	␣	u	n	␣	e	x	e	m	p	l	o	␣	d	e	␣	p	r	o	b	a
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

patrón	e	x	e	m	p	l	o
	1	2	3	4	5	6	7

m=7

Σ	␣	a	b	d	e	i	l	m	n	o	p	r	s	t	x	é	
d	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	**(inicialmente)**
d	7	7	7	7	6	7	7	7	7	7	7	7	7	7	7	7	**i=1 → p[i]=e**
d	7	7	7	7	6	7	7	7	7	7	7	7	7	7	5	7	**i=2 → p[i]=x**
d	7	7	7	7	4	7	7	7	7	7	7	7	7	7	5	7	**i=3 → p[i]=e**
d	7	7	7	7	4	7	7	3	7	7	7	7	7	7	7	7	**i=4 → p[i]=m**
d	7	7	7	7	4	7	7	3	7	7	2	7	7	7	5	7	**i=5 → p[i]=p**
d	7	7	7	7	4	7	1	3	7	7	2	7	7	7	5	7	**i=6 → p[i]=l**
d	7	7	7	7	**4**	7	**1**	**3**	7	**7**	**2**	7	7	7	**5**	7	**(final)**

Horspool. Exemplo. Fase Preprocesado

```
Searching phase(p, m, t, n, Σ, d)
(1)  pos ← 1;
(2)  while pos < n − m
(3)      j ← m;
(4)      while j > 0 and t_{pos+j} = p_j
(5)          j ← j − 1;
(6)      if j = 0 then
(7)          occurrence appeared at position pos;
(8)      pos ← pos + d[t_{pos+m}];
```

patrón

e	x	e	m	p	l	o
1	2	3	4	5	6	7

m=7

Σ		a	b	d	e	i	l	m	n	o	p	r	s	t	x	é
d	7	7	7	7	**4**	7	**1**	**3**	7	**7**	**2**	7	7	7	**5**	7

texto

i	s	t	o		é		u	n		e	x	e	m	p	l	o		d	e		p	r	o	b	a
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26

e	x	e	m	p	l	o

salto = d[‘ ’] = 7

e	x	e	m	p	l	o

salto = d[‘m’] = 3

e	x	e	m	p	l	o

salto = d[‘o’] = 7

e	x	e	m	p	l	o

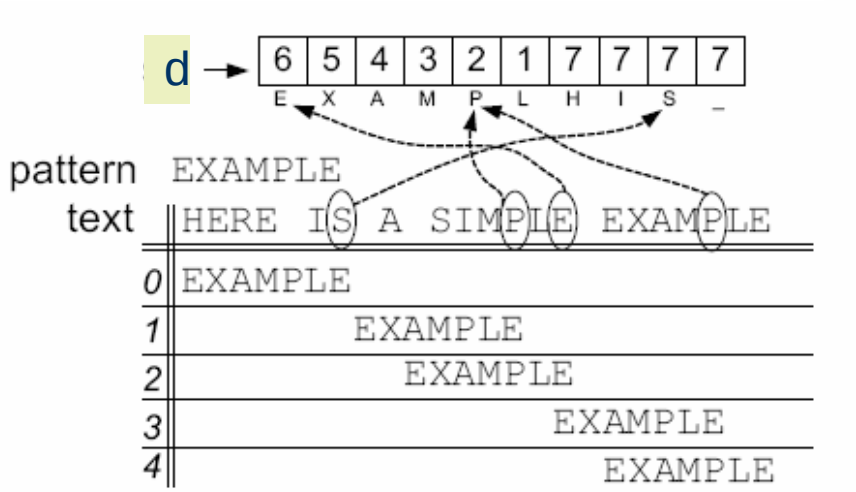
salto= d[‘o’] = 7

Fin

Outro exemplo.

HERE IS A SIMPLE EXAMPLE

$\Sigma = \{\texttt{E}, \texttt{X}, \texttt{A}, \texttt{M}, \texttt{P}, \texttt{L}, \texttt{H}, \texttt{I}, \texttt{S}, \_\}$



Once the preprocessing phase ends, and vector sv contains the shifts that correspond to each symbol in $\Sigma = \{\texttt{E}, \texttt{X}, \texttt{A}, \texttt{M}, \texttt{P}, \texttt{L}, \texttt{H}, \texttt{I}, \texttt{S}, \_\}$, the searching phase can start (see Figure 3.6).

In step 0, the search window contains '`HERE IS`'. Since '`E`'$\neq$'`S`' then $\sigma =$'`S`' and $\beta =$'`S`', then a match cannot occur, so the search window is shifted by $d[S] = 7$.

In step 1, the search window contains '` A SIMP`'. Since '`E`'$\neq$'`P`' then $\sigma =$'`P`' and $\beta =$'`P`'. We shift the search window $d[P] = 2$ positions to the right.

In step 2, the search window contains '` SIMPLE`'. Then the backwards verification fails with $\sigma =$'`I`'. Since $\beta =$'`E`', a shift of $d[E] = 6$ positions is done.

In step 3, the search window contains '`E EXAMP`'. Since '`E`'$\neq$'`P`' then $\sigma =$'`P`' and $\beta =$'`P`'. We shift the search window by $d[P] = 2$ positions to the right.

The pattern and the search window match in step 4. Then the pattern is shifted $d[E] = 6$ positions to the right and the searching phase finishes. □

Máis exemplos

Σ		a	b	r	c	d	p	t
d	3	3	2	1	3	3	3	3

texto	a	b	r	a		c	a	d	a	b	r	a		p	a	t	a		t	r	a	b	a
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23

Patrón

b	r	a																					
	b	r	a																				
			b	r	a																		
					b	r	a																
							b	r	a														
									b	r	a												
											b	r	a										
													b	r	a								
															b	r	a						
																	b	r	a				

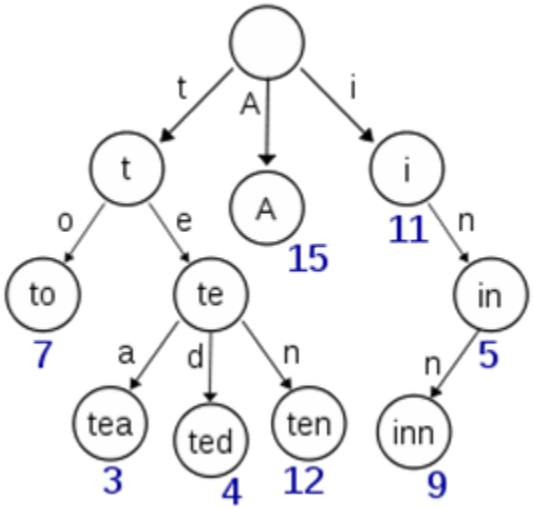
match en 2

match en 10

Horspool Multipatrón

- **Similar a Horspool simple**
- **Preprocesado**
  - Elexir o *shift* tendo en conta os diferentes patróns buscados
- **Fase de busca.**
  - Emparellamento “cara atrás” tendo en conta os diferentes patróns buscados.
  - Usar un trie construido sobre os patróns reversos (axiliza comparación coa ventá)
    - Mentres haxa transición|se chegue a un estado final
- **Trie:**

Horspool multipatrón

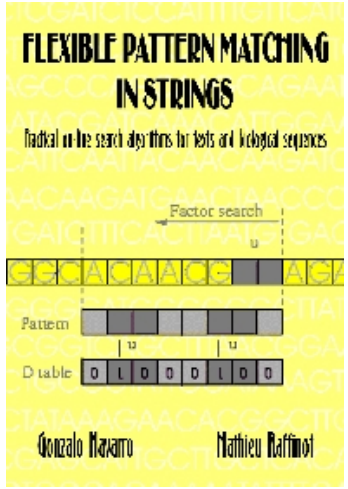


- **trie para as claves "A", "to", "tea", "ted", "ten", "i", "in", "inn".**
- **Pódese ver como un Autómata Finito determinista**

- **Exemplo**
- **Sobre o texto: abra cadabra pata traba**
- **Buscar ós patróns: abra, abro, pata**

Referencias

- **Un libro**



Flexible Pattern Matching in Strings. Practical on-line search algorithms for texts and biological sequences

Gonzalo Navarro and Mathieu Raffinot

"Cambridge University Press", 2002

ISBN 0-521-81307-7

- **Unha web con exemplos, pseudocódigos e animacións Java…**

  http://www-igm.univ-mlv.fr/~lecroq/string/index.html

- **Unha Revista Internacional**

  Gonzalo Navarro. P**attern Matching**.
  Journal of Applied Statistics 31(8):925-949, 2004.

- **…**