Sequential String Matching

Maxime Crochemore

King's College London

Maxime.Crochemore@kcl.ac.uk
http://www.dcs.kcl.ac.uk/staff/mac/

Examples

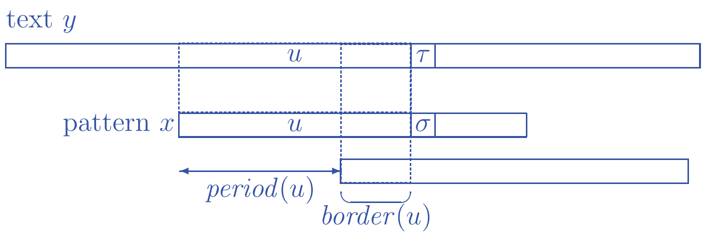
⋆ Naive search (1)

```
 a b a b a a a b a a b a a b a b . .
 a b a a a b a b
   a b a a a b a b
     a b a a a b a b
       a b a a a b a b
         a b a a a b a b
            . . .
```

⋆ Naive search (2)

```
 a a a b a a a a b a b a a b a b . .
 a a a b a a
   a a a b a a
     a a a b a a
       a a a b a a
         a a a b a a
            . . .
```

Left-to-right scan — shift



- ⋆ Mismatch situation: $\sigma \neq \tau$
- ⋆ $period(u) = |u| - |border(u)|$
- ⋆ Optimal shift length $= period(u\tau)$
- ⋆ Valid if $u = x$

Periods and borders

- ⋆ Non-empty string u, integer p, $0 < p \leq |u|$
- ⋆ p is a period of u if any of these equivalent conditions is satisfied:
  - [1] $u[i] = u[i+p]$, for $0 \leq i < |u| - p$
  - [2] u is a prefix of some y^k, $k > 0$, $|y| = p$
  - [3] $u = yw = wz$, for some strings y, z, w with $|y| = |z| = p$
    String w is called **a border** of u
- ⋆ **The** period of u, $period(u)$, is its smallest period (can be $|u|$)
- ⋆ **The** border of u, $border(u)$, is its longest border (can be empty)
- ⋆ Periods and borders of `abacabacaba`

4	abacaba
8	aba
10	a
11	empty string

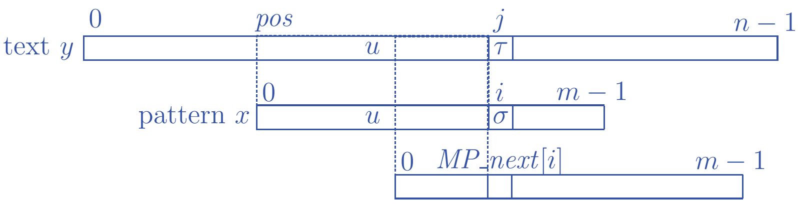
Sequential search

- ⋆ Simple online search
- ⋆ Length of shift = period
- ⋆ Memorization of borders

---

while window on text **do**
    $u \longleftarrow$ longest common prefix of window and pattern
    if $u =$ pattern **then** report a match
    shift window $period(u)$ places to right
    memorize $border(u)$

MP algorithm



MP(string x, y; integer m, n)
    $i \longleftarrow 0$; $j \longleftarrow 0$
    while $j < n$ **do**
        while $(i = m)$ **or** $(i \geq 0$ **and** $x[i] \neq y[j])$ **do**
            $i \longleftarrow MP\_next[i]$
        $i \longleftarrow i + 1$; $j \longleftarrow j + 1$
        if $i = m$ **then** output('x occurs in y at position', $j - i$)

Example of MP run

- MP_next table

i	0	1	2	3	4	5	6	7	8	9	10
$x[i]$	a	b	a	c	a	b	a	c	a	b	
$MP\_next[i]$	-1	0	0	1	0	1	2	3	4	5	6

- Run of MP algorithm

```
y a b a b a c a b a d a b . .
x a b a c a b a c a b
      a b a c a b a c a b
              a b a c a b a c a b
                  a b a c a b a c a b
                    a b a c a b a c a b
                      a b a c a b a c a b
```

- If end of y, MP algorithm gives the longest overlap between y and x.

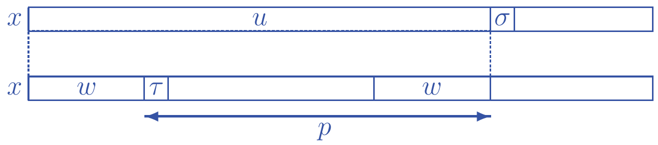
Computing borders of prefixes

- A border of a border of u is a border of u
  A border of u is either $border(u)$ or a border of it
- $\text{Border}[i] = |border(x[0\,.\,.\,i-1])|$
- j runs through decreasing lengths of borders

```
COMPUTE_BORDERS(string x; integer m)
    Border[0] ⟵ −1
    for i ⟵ 0 to m − 1 do
        j ⟵ Border[i]
        while j ≥ 0 and x[i] ≠ x[j] do
            j ⟵ Border[j]
        Border[i + 1] ⟵ j + 1
    return Border
```

- $MP\_next[i] = \text{Border}[i]$ for $i = 0, \ldots, m$

Improvement



- ⋆ Interrupted periods — strict borders
- ⋆ Changes only the preprocessing of MP algorithm

---

while window on text **do**
  $u \longleftarrow$ longest common prefix of window and pattern
  if $u =$ pattern **then** report a match
  shift window $interrupt\_period(u)$ places to the right
  memorize $strict\_border(u)$

Interrupted periods and strict borders

- ⋆ Fixed string x, non-empty prefix u of x
- ⋆ w is a strict border of u if both:
  - w is a border of u
  - $w\tau$ is a prefix of x, but $u\tau$ is not
- ⋆ p is an interrupted period of u if $p = |u| - |w|$ for some strict border $|w|$ of u
- ⋆ Prefix `abacabacaba` of `abacabacabacc`

  Interrupted periods and strict borders of `abacabacaba`

10	a
11	empty string

KMP preprocessing

- $k = MP\_next[i]$
- $KMP\_next[i] = \begin{cases} k, & \text{if } x[i] \neq x[k] \text{ or if } i = m, \\ KMP\_next[k], & \text{if } x[i] = x[k]. \end{cases}$

COMPUTE_KMP_NEXT(string x; integer m);
    $KMP\_next[0] \longleftarrow -1$; $k \longleftarrow 0$
    for $i \longleftarrow 1$ **to** $m - 1$ **do** {here: $k = MP\_next[i]$}
        if $x[i] = x[k]$ **then**
            $KMP\_next[i] \longleftarrow KMP\_next[k]$
        else $KMP\_next[i] \longleftarrow k$
            do $k \longleftarrow KMP\_next[k]$
            while $k \geq 0$ **and** $x[i] \neq x[k]$
        $k \longleftarrow k + 1$
    $KMP\_next[m] \longleftarrow k$
    return KMP_next

Example of KMP run

- KMP_next table

i	0	1	2	3	4	5	6	7	8	9	10
$x[i]$	a	b	a	c	a	b	a	c	a	b	
$MP\_next[i]$	-1	0	0	1	0	1	2	3	4	5	6
$KMP\_next[i]$	-1	0	-1	1	-1	0	-1	1	-1	0	6

- Run of KMP algorithm

```
y a b a b a c a b a d a b . .
x a b a c a b a c a b
      a b a c a b a c a b
                  a b a c a b a c a b
                    a b a c a b a c a b
                      a b a c a b a c a b
```

- If end of y, KMP algorithm gives the longest overlap between y and x.

Running times of MP and KMP

Theorem 1 *On a text of length n, MP and KMP string-searching algorithm run in time $O(n)$.*
They make less than $2n$ symbol comparisons.

Proof Positive comparisons increase the value of j
Negative comparisons increase the value of $j - i$ (shift)

⋆ Delay = maximum number of comparisons on a text symbol

Theorem 2 *Pattern of length m. The delay for MP algorithm is no more than m. The delay for KMP algorithm is no more than $\log_\Phi(m+1)$, where Φ is the golden ratio, $(1+\sqrt{5})/2$.*

Proof For KMP, use the periodicity theorem

⋆ A worst-case pattern of length 19: `abaababaabaababaaba`

Periodicities

Theorem 3 *If p and q are periods of a word x and satisfy $p + q - GCD(p,q) \leq |x|$ then $GCD(p,q)$ is a period of x.*

[Fine, Wilf, 1965]

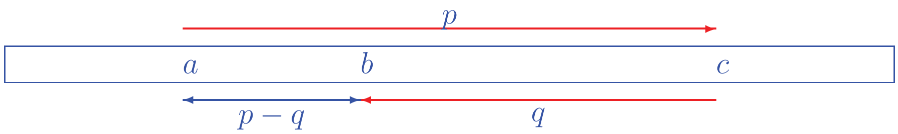
Used in the analysis of KMP algorithm and in the analysis of many other pattern matching algorithms.

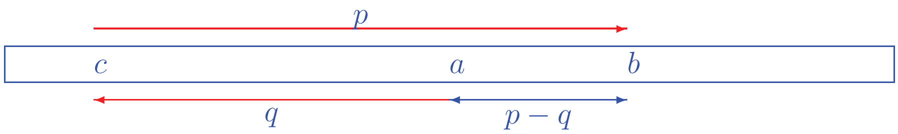
Theorem 4 *(Weak version) If p and q are periods of a word x and satisfy $p + q \leq |x|$ then $GCD(p,q)$ is a period of x.*

Proof If p and q are periods of x, $p > q$, then $p - q$ is also a period of x. Rest of the proof analogous to correctness of Euclid's gcd algorithm.

Proof of the weak statement

- p and q periods of x with $p + q \leq |x|$ and $p > q$
- $p - q$ period of x because:

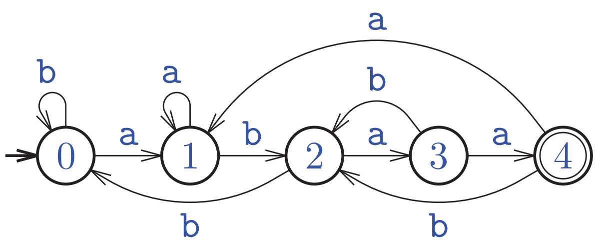




- rest of the proof like Euclid's induction

Searching with an automaton

- Uses the string-matching automaton $SMA(x)$:
  smallest deterministic automaton accepting $\Sigma^* x$
- Example $x =$ `abaa`



- Search for `abaa` in:

		b	a	b	b	a	a	b	a	a	b	a	a	b	b	a	···
state	0	0	1	2	0	1	1	2	3	4	2	3	4	2	0	1	···

Searching algorithm

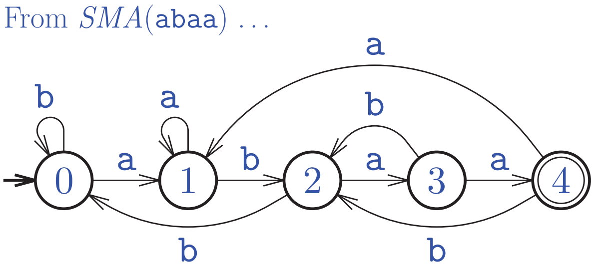
- ⋆ Simple online parsing of the text with the string-matching automaton $SMA(x)$

SEARCH(string x, y; integer m, n)
    $(Q, \Sigma, initial, \{terminal\}, \delta)$ is the automaton $SMA(x)$
    $q \longleftarrow$ *initial state*
    if $q =$ *terminal* **then** report an occurrence of x in y
    while not end of y **do**
        $\sigma \longleftarrow$ next symbol of y
        $q \longleftarrow \delta(q, \sigma)$
        if $q =$ *terminal* **then** report an occurrence of x in y

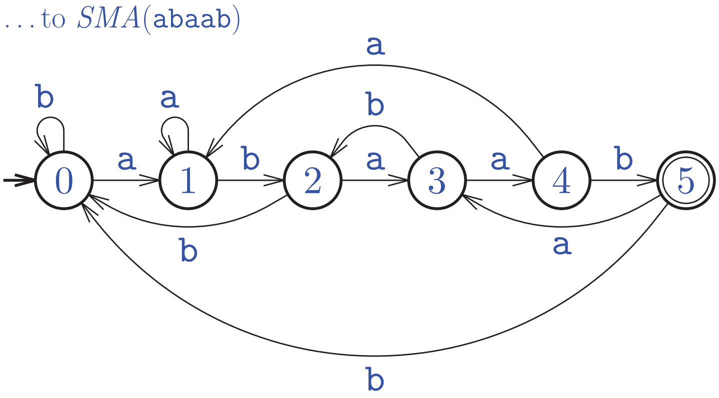
---

Construction of $SMA(x)$

- ⋆ **Unwinding arcs**
- ⋆ From $SMA($`abaa`$)$ . . .



- ⋆ . . . to $SMA($`abaab`$)$



Construction algorithm

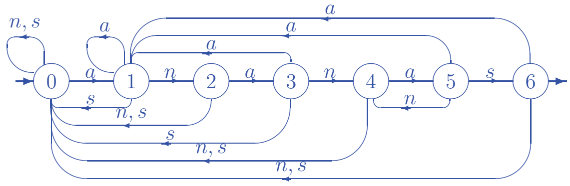
⋆ Unwind the appropriate arc

---

automaton SMA(string x)
  let *initial* be a new state
  $Q \longleftarrow \{initial\}$
  $terminal \longleftarrow initial$
  for all σ **in** Σ **do** $\delta(initial, \sigma) \longleftarrow initial$
  while not end of x **do**
    $\tau \longleftarrow$ next symbol of x
    $r \longleftarrow \delta(terminal, \tau)$
    add new state s to Q
    $\delta(terminal, \tau) \longleftarrow s$
    for all σ in Σ **do** $\delta(s, \sigma) \longleftarrow \delta(r, \sigma)$
    $terminal \longleftarrow s$
  return $(Q, \Sigma, initial, \{terminal\}, \delta)$

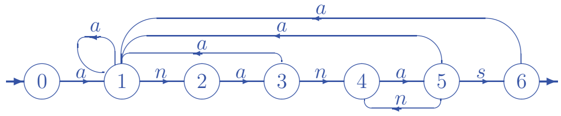
Significant arcs

⋆ Complete SMA(`ananas`)



⋆ **Forward arcs**: spell the pattern

⋆ **Backward arcs**: arcs going backwards without reaching the initial state



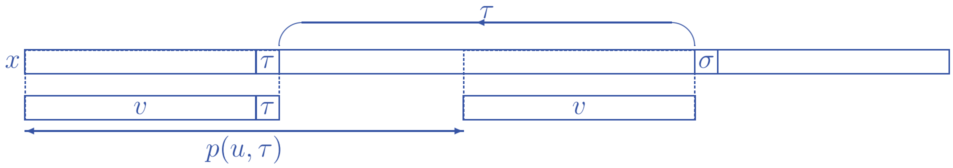
Lemma 1 *$SMA(x)$ contains at most $|x|$ backward arcs.*

⋆ Consequence: the implementation of $SMA(x)$ can be done in $O(|x|)$ time and space, independently of the alphabet size

Backward arcs in SMA

- States of $SMA(x)$ are identified with prefixes of x
  A backward arc is of the form $(u, \tau, v\tau)$ (u, v prefixes of x, τ symbol) with
  - $v\tau$ longest suffix of $u\tau$ that is a prefix of x, and $u \neq v$

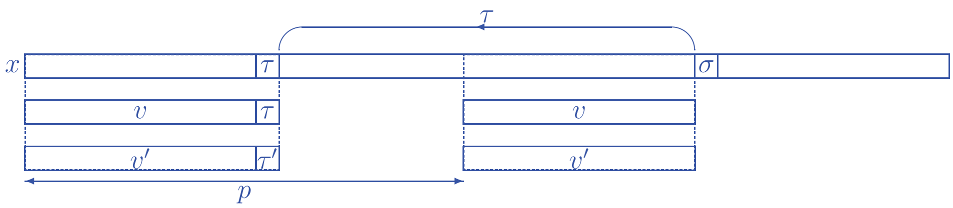
  Note: $u\tau$ is not a prefix of x
  Let $p(u, \tau) = |u| - |v|$; it is a period of u because v is a border of u



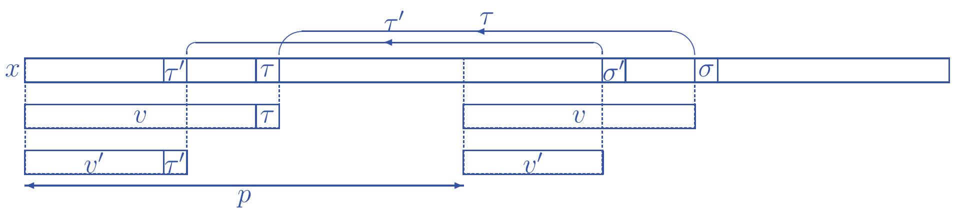
- **Backward arcs to periods: p is injective**
  Each period p, $1 \leq p \leq |x|$, corresponds to at most one backward arc, thus there are at most $|x|$ such arcs
- **A worst case**: $SMA(ab^{m-1})$ has m backward arcs ($a \neq b$)

Backward arcs (followed)

- **Proof that p is injective**
  Two backward arcs $(u, \tau, v\tau)$, $(u', \tau', v'\tau')$
  Assume $p(u, \tau) = p(u', \tau') = p$; we prove $u = u'$ and $\tau = \tau'$.



- **If** $v = v'$ then $u = u'$ and also $\tau = \tau'$



- **If** v' a proper prefix of v then $v'\tau'$ and $v'\sigma'$ are prefixes of v
  thus $\tau' = \sigma'$ a contradiction

Complexity of searching with SMA

- Pattern x of length m, text y of length n
- **With complete SMA implemented by transition matrix**

Preprocessing on pattern x	time	$O(m \times \text{card}\,\Sigma)$
	space	$O(m \times \text{card}\,\Sigma)$
Search on text y	time	$O(n)$
	space	$O(m \times \text{card}\,\Sigma)$
Delay	time	constant

- **With SMA implemented by lists of forward and backward arcs**

Preprocessing on pattern x	time	$O(m)$
	space	$O(m)$
Search on text y	time	$O(n)$
	space	$O(m)$
Delay	comparisons	$\min\{\text{card}\,\Sigma, \log_2 m\}$

- Improves on KMP algorithm