Text Searching

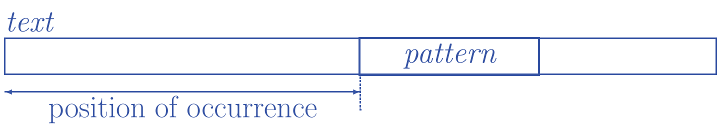
Maxime Crochemore

King's College London

Maxime.Crochemore@kcl.ac.uk

http://www.dcs.kcl.ac.uk/staff/mac/

Pattern matching



⋆ **Problem**

Find all the occurrences of pattern x of length m
inside the text y of length n

⋆ **Two types of solutions**

– Fixed pattern — preprocessing time $O(m)$
Use of combinatorial properties — searching time $O(n)$

– Static text — preprocessing time $O(n)$
Solutions based on indexes — searching time $O(m)$

Searching for a fixed string

- ⋆ **String matching**
  given a pattern x, find all its locations in any text y
- ⋆ **Pattern**: a string x of symbols, of length m

  t a t a
- ⋆ **Text**: a string y of symbols, of length n

  c a c g t a t a t a t g c g t t a t a a t
- ⋆ **Occurrences** at positions 4; 6, 15:

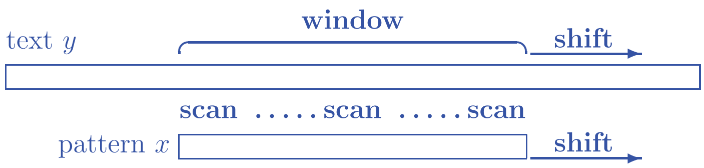
```
c a c g t a t a t a t g c g t t a t a a t
        t a t a               t a t a
            t a t a
```

- ⋆ **Basic operation**: symbol comparison ($=, \neq$)

Interest

- ⋆ **Practical**
  basic problem for
  - search for various patterns
  - lexical analysis
  - approximate string matching
  - comparisons of strings—alignments
  - ...
- ⋆ **Theoretical**
  - design of algorithms
  - analysis of algorithms—complexity
  - combinatorics on strings
  - ...

Sliding window strategy



- Scan-and-shift mechanism

---

put window at the beginning of text
while window on text **do**
    scan: **if** window = pattern **then** report it
    shift: shift window to the right and
    memorize some information for use during next scans and shifts

---

Naive search

c a c g t a t a t a t g c g t t a t a a

t a t a

Principles

- No memorization, shift by 1 position

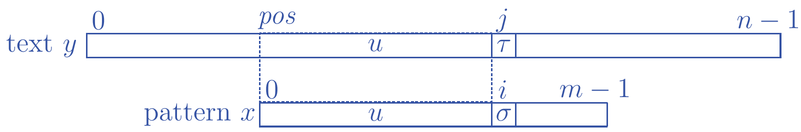
Complexity

- $O(m \times n)$ time, $O(1)$ extra space

Number of symbol comparisons

- maximum $\approx m \times n$
- expected $\approx 2 \times n$
  on a two-letter alphabet,
  with equiprobablity and independence conditions

Naive string-searching algorithm



NAIVE_SEARCH(string x, y; integer m, n)
    $pos \longleftarrow 0$
    while $pos \leq n - m$ **do**
        $i \longleftarrow 0$
        while $i < m$ **and** $x[i] = y[pos + i]$ **do**
            $i \longleftarrow i + 1$
        if $i = m$ **then** output('x occurs in y at position ', pos)
        $pos \longleftarrow pos + 1$

Acceleration by hashing

- Hash function: $h : \mathbf{\Sigma}^m \longrightarrow \mathbf{N}$

ACCELERATED_SEARCH(string x, y; integer m, n; h)
    $h_x \longleftarrow h(x)$
    for $pos \longleftarrow 0$ **to** $n - m$ **do**
        if $h_x = h(y[pos \,..\, pos + m - 1])$ **then**
            $i \longleftarrow 0$
            while $i < m$ **and** $x[i] = y[pos + i]$ **do**
                $i \longleftarrow i + 1$
            if $i = m$ **then** output('x occurs in y at position ', pos)

- Uses arithmetic operations in addition to symbol comparisons
- **What hash function to speed-up the algorithm?**
- **Goal:** $h(u) = h(v)$ **if it is very likely that** $u = v$

Hash function

- ⋆ Hash Function: $h : \mathbf{\Sigma}^m \longrightarrow \mathbf{N}$
- ⋆ Principle: do as if symbols are integers
  similar to number representation but with approximation
- ⋆ Parameters: integers d (like a base) and q (for the modulo)
- ⋆ Definition:

$$\begin{aligned} h_{pos} &= h(y[pos\,..\,pos+m-1]) \\ &= (y[pos]d^{m-1} + y[pos+1]d^{m-2} + \cdots + y[pos+m-1]) \ \mathbf{mod}\ q \\ &= ((\ldots(y[pos]d + y[pos+1])d + \cdots + y[pos+m-2])d \\ &\qquad\qquad\qquad + y[pos+m-1]) \ \mathbf{mod}\ q \end{aligned}$$

- ⋆ Hörner's rule for the first hash value

Next hash value

- ⋆ From h_{pos} to h_{pos+1}:

$$\begin{aligned} h_{pos} &= (y[pos]d^{m-1} + y[pos+1]d^{m-2} + \cdots + y[pos+m-1]) \ \mathbf{mod}\ q \\ h_{pos+1} &= (y[pos+1]d^{m-1} + y[pos+2]d^{m-2} + \cdots + y[pos+m]) \ \mathbf{mod}\ q \\ &= ((h_{pos} - y[pos]d^{m-1})d + y[pos+m]) \ \mathbf{mod}\ q \end{aligned}$$

- ⋆ **It requires a fixed number of arithmetic operations**
- ⋆ **. . . then executes in constant time**

Karp-Rabin string searching

★ Typical parameters: $d = 2^k$ and q is a prime number

KR(string x, y; integer m, n, d, q)
  $(h_x, h_y, D) \longleftarrow (0,\ 0,\ d^{m-1}$ **mod** $q)$
  for $i \longleftarrow 0$ to $m-1$ **do**
    $h_x \longleftarrow (h_x d + x[i])$ **mod** q
    $h_y \longleftarrow (h_y d + y[i])$ **mod** q
  for $pos \longleftarrow 0$ **to** $n-m$ **do**
    if $h_x = h_y$ **then**
      $i \longleftarrow 0$
      while $i < m$ **and** $x[i] = y[pos+i]$ **do**
        $i \longleftarrow i+1$
      if $i = m$ **then** output('x occurs in y at position ', pos)
    if $pos < n-m$ **then**
      $h_y \longleftarrow ((h_y - y[pos]D)d + y[pos+m])$ **mod** q

---

Complexity of the problem

pattern x of length m
text y of length n $(n > m)$

Theorem 1 *The search can be done optimally in time $O(n)$ and space $O(1)$.*

[Galil and Seiferas, 1983]

Theorem 2 *The search can be done in optimal expected time $O(\frac{\log m}{m} \times n)$.*

[Yao, 1979]

Theorem 3 *The maximal number of comparisons done during the search is $\geq n + \frac{9}{4m}(n-m)$, and can be made $\leq n + \frac{8}{3(m+1)}(n-m)$.*

[Cole *et alii*, 1995]

Known bounds on symbol comparisons

Lower bounds	Upper bounds	

★ **access to the whole text**

n		[Folklore]
	$2n-1$	**[Morris and Pratt, 1970]**
$\frac{4}{3}n$		**[Zwick and Paterson, 1992]**

★ **search with a sliding window of size** m

	$\frac{3}{2}n$	**[Apostolico and Crochemore, 1989]**
$\frac{4}{3}n$	$\frac{4}{3}n$	**[Galil and Giancarlo, 1991]**
$\frac{4}{3}n - \frac{1}{3}m$		**[Galil and Giancarlo, 1992]**
	$n + \frac{4\log m+2}{m}(n-m)$	**[Galil and Breslauer, 1993]**
$n + \frac{9}{4m}(n-m)$	$n + \frac{8}{3(m+1)}(n-m)$	**[Cole *et alii*, 1995]**

Known bounds on symbol comparisons (followed)

Lower bounds	Upper bounds	

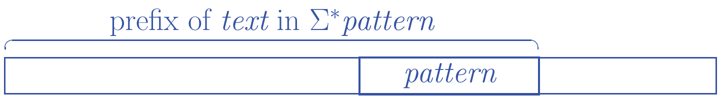
★ **search with a sliding window of size** 1

	$2n-1$	**[Morris and Pratt, 1970]**
$(2-\frac{1}{m})n$	$(2-\frac{1}{m})n$	**[Hancart, 1993]**
		[Breslauer *et alii*, 1993]

★ **delay = maximum number of comp's on each text symbol**

	m	**[Morris and Pratt, 1970]**
	$\log_\Phi(m+1)$	**[Knuth, Morris and Pratt, 1977]**
	$\min\{\log_\Phi(m+1), \mathrm{card}\Sigma\}$	**[Simon, 1989]**
$\min\{1+\log_2 m, \mathrm{card}\Sigma\}$	$\min\{1+\log_2 m, \mathrm{card}\Sigma\}$	**[Hancart, 1993]**
	$\log\ \min\{1+\log_2 m, \mathrm{card}\Sigma\}$	**[Hancart, 1996]**

Methods



- **sequential searches** (window size = one symbol)
  - ↪ adapted to telecommunications
  - ↪ based on efficient implementations of automata

  [Knuth, Morris, Pratt, 1976], **[Simon, 1989]**,
  [Hancart, 1993], **[Breslauer, Colussi, Toniolo, 1993]**

Methods (followed)

- **time-space optimal searches**
  - ↪ mainly of theoretical interest
  - ↪ based on combinatorial properties of strings

  [Galil, Seiferas, 1983], **[Crochemore, Perrin, 1991]**,
  [Crochemore, 1992], **[Gąsieniec, Plandowski, Rytter, 1995]**,
  [Crochemore, Gąsieniec, Rytter, 1997]

- **practically-fast searches**
  - ↪ used in text editors, data retrieval software
  - ↪ based on combinatorics + automata (+ heuristics)

  [Boyer, Moore, 1977], **[Galil, 1979]**,
  [Apostolico, Giancarlo, 1986], **[Crochemore *et alii*, 1992]**