Dictionary Matching

Maxime Crochemore

King's College London

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

Matching several strings

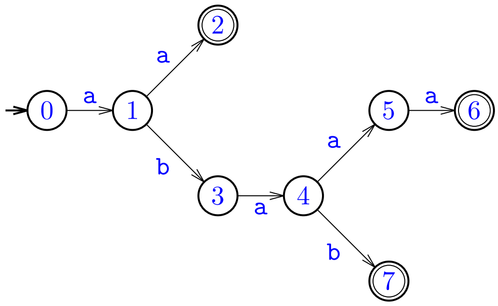
- **Dictionary**: set of finite strings, $X = \{x_0, x_1, \ldots, x_{k-1}\}$ (empty string not in X)
- **Matching**: locate occurrences of the strings in any text y

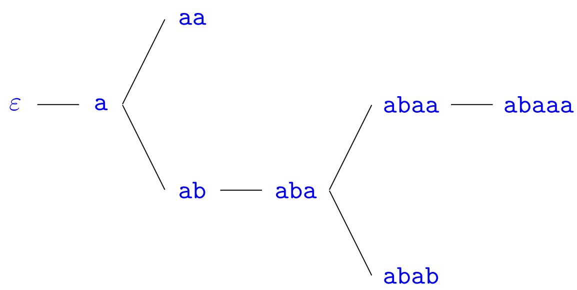
Note that each position on y can be the position of several strings, yielding a possible quadratic output (e.g. $X = \{\mathtt{a}, \mathtt{aa}, \ldots, \mathtt{a}^k\}$ and $y = \mathtt{a}^n$)

- **Output**: list of positions on y that are end-positions of some string in X
- **Standard method**: based on the Dictionary Matching Automaton (Aho-Corasick automaton), an extension of the String Matching Automaton
- **Other method**: extension of the right-to-left window scanning strategy (Boyer-Moore technique)

Trie of the dictionary

- **Trie**: $\mathcal{T}(X)$, digital tree whose branches are labelled by strings of X
  As an automaton, the language it accepts is X
- **Example** : $\mathcal{T}(\{\mathtt{aa}, \mathtt{abaaa}, \mathtt{abab}\})$

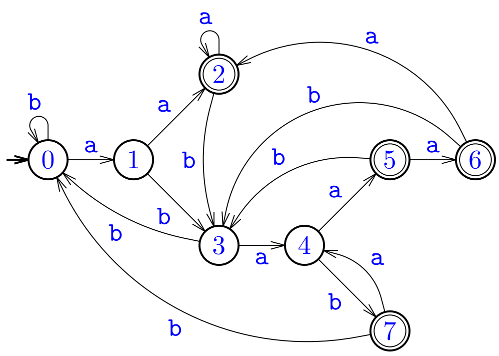




- Nodes are all prefixes of strings in X
- $\mathcal{T}(X)$ is the basis of the Dictionary Matching Automaton

Dictionary Matching Automaton

- **Dictionary Matching Automaton**, $\mathcal{D}(X)$
  it accepts the language A^*X and is defined by:
  - set of states is $\mathrm{Pref}(X)$; initial state is the empty string
  - set of terminal states is $\mathrm{Pref}(X) \cap A^*X$
  - arcs are of the form $(u, a, h(ua))$
    where $h(ua)$ = longest suffix of ua that belongs to $\mathrm{Pref}(X)$
- **Example** : $\mathcal{D}(\{\mathtt{aa}, \mathtt{abaaa}, \mathtt{abab}\})$ with alphabet $A = \{\mathtt{a}, \mathtt{b}\}$



Searching with DMA

⋆ **Searching**

Det-search-by-failure(X, y)

$M \leftarrow$ DMA-by-failure(X)
$r \leftarrow initial[M]$
for each letter a of y, sequentially **do**
    $r \leftarrow$ Target$(,)(r, a)$
    Output-if$(terminal[r])$

⋆ **Implementation** of $\mathcal{D}(\{\texttt{aa}, \texttt{abaaa}, \texttt{abab}\})$ with a transition table

Next state table	0	1	2	3	4	5	6	7
a	0	2	2	4	5	6	2	4
b	1	3	3	0	7	3	3	0

⋆ **Searching time**: $O(|y|)$

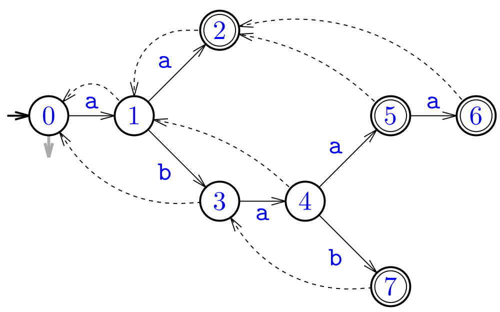
j		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...
$y[j]$		c	d	a	b	b	a	b	a	a	b	a	b	a	b	b	a	a	...
state	0	0	0	1	3	0	1	3	4	**5**	3	4	**7**	4	**7**	0	1	**2**	...

Implementation by failure function

⋆ **Aim**: reduction of the implementation to $O(\Sigma\{|x| : x \in X\})$
independent of the alphabet

⋆ **Failure function** (u nonempty string)
$f(u)$ = the longest proper suffix of u that belongs to Pref(X)

⋆ **Implementation** of $\mathcal{D}(\{\texttt{aa}, \texttt{abaaa}, \texttt{abab}\})$ with successor lists and f



state	successors
0	(a,1)
1	(a,2), (b,3)
2	
3	(a,4)
4	(a,5), (b,7)
5	(a,6)
6	
7	

⋆ **Searching**

j		0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	...
$y[j]$		c	d	a	b	b	a	b	a	a	b	a	b	a	b	b	a	a	...
state	0	0	0	1	3	0,0	1	3	4	**5**	2,1,3	4	**7**	3,4	**7**	3,0,0	1	**2**	...

Searching with failure function

⋆ **Searching**

DET-SEARCH-BY-FAILURE(X, y)

$M \leftarrow$ DMA-BY-FAILURE(X)
$r \leftarrow$ *initial*$[M]$
for each letter a of y, sequentially **do**
  $r \leftarrow$ TARGET-BY-FAILURE(r, a)
  OUTPUT-IF(*terminal*$[r]$)

⋆ **Target**: next state function using successors lists

TARGET-BY-FAILURE(p, a)

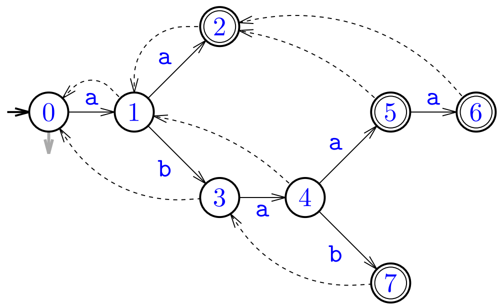
while $p \neq$ NIL and TARGET$(p, a) =$ NIL **do**
  $p \leftarrow$ *fail*$[p]$
if $p =$ NIL **then**
  return *initial*$[M]$
else **return** TARGET(p, a)

⋆ **Running time**: $O(|y| \times \log \text{card}\, A)$

Computing the failure links

⋆ $f(ua) =$ TARGET-BY-FAILURE$(f(u), a)$

⋆ **Computation via a width-first traversal of the trie**



⋆ *fail*[0], *fail*[1], *fail*[2], *fail*[3], and *fail*[4] already computed

⋆ **Computing *fail*[5] from** 4: $\delta(4, \mathtt{a}) = 5$, *fail*$[4] = 1$, and $\delta(1, \mathtt{a}) = 2$ gives *fail*$[5] = 2$

Note: since state 2 is terminal, state 5 becomes terminal

⋆ **Computing *fail*[6] from** 5: $\delta(5, \mathtt{a}) = 6$, *fail*$[5] = 2$, and $\delta(2, \mathtt{a})$ undefined but TARGET-BY-FAILURE$(2, \mathtt{a}) = 2$ gives *fail*$[6] = 2$

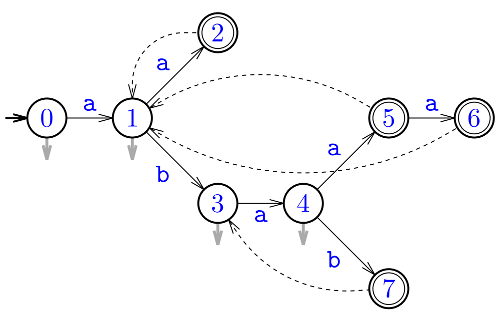
Computing the failure links—algorithm

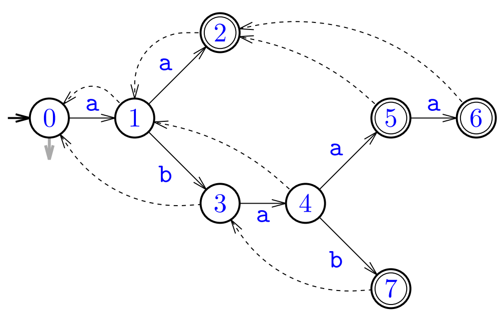
- **Failure links** *fail*[] implements the failure function f

```
DMA-BY-FAILURE(X)
M ← TRIE(X)
fail[initial[M]] ← NIL
F ← EMPTY-QUEUE()
ENQUEUE(F, initial[M])
while non QUEUE-IS-EMPTY(F) do
    t ← DEQUEUED(F)
    for each pair (a, p) ∈ Succ[t] do
        r ← TARGET-BY-FAILURE(fail[t], a)
        fail[p] ← r
        if terminal[r] then
            terminal[p] ← TRUE
        ENQUEUE(F, p)
return M
```


Optimized failure links

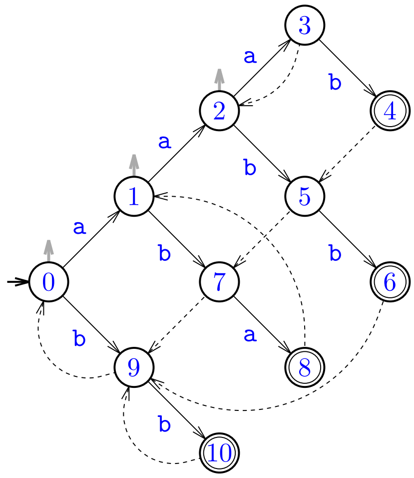
- **Next set** of state $u \in \mathrm{Pref}(X)$:
  $Next(u) = \{a : a \in A, ua \in \mathrm{Pref}(X)\}$
- **New link** defined by function f': for u nonempty $f'(u) = f^k(u)$
  where $k = \min\{\ell : Next(f^\ell(u)) \not\subseteq Next(u)\}$
- **Optimized link** **Non optimized**





* **Delay**: maximum time spent on a letter of y
  It is $\max\{|x| : x \in X\}$ even with the optimized links

* **Example**: $X = \{\texttt{aaab}, \texttt{aabb}, \texttt{aba}, \texttt{bb}\}$



* $L(m) = \{\texttt{a}^{m-1}\texttt{b}\} \cup \{\texttt{a}^{2k-1}\texttt{ba} : 1 \leq k < \lceil m/2 \rceil\} \cup \{\texttt{a}^{2k}\texttt{bb} : 0 \leq k < \lfloor m/2 \rfloor\}$