On Binary Jumbled Pattern Matching

Gabriele Fici

I3S, CNRS & Université Nice Sophia Antipolis

27 Nov 2012
Marne-la-Vallée

Joint work with: Zs. Lipták, F. Cicalese, P. Burcsi, S. Kroon and G. Badkobeh

Text s over $\{a, b\}$ of length $|s| = n$.

Binary Jumbled Pattern Matching Problem: Given $(x, y) \in \mathbb{N} \times \mathbb{N}$, decide whether a substring occurs in s with x a's and y b's.

Example

$s = aababab baaabbaabbb$.
For $(2, 2)$ the answer is yes; for $(1, 4)$ is no.

Text s over $\{a, b\}$ of length $|s| = n$.

Binary Jumbled Pattern Matching Problem: Given $(x, y) \in \mathbb{N} \times \mathbb{N}$, decide whether a substring occurs in s with x a's and y b's.

Example

$s = aababab baaabbaabbb$.
For $(2, 2)$ the answer is yes; for $(1, 4)$ is no.

It is a kind of approximate string matching in which anagrams are allowed.

Simple Solution: Slide a window of size $x + y$ along the text and count the number of a's. $O(n)$ time (optimal), on-line.

Simple Solution: Slide a window of size $x + y$ along the text and count the number of a's. $O(n)$ time (optimal), on-line.

Question: Can a preprocessing reduce the query time? (Useful when many queries are expected.)

Simple Solution: Slide a window of size $x + y$ along the text and count the number of a's. $O(n)$ time (optimal), on-line.

Question: Can a preprocessing reduce the query time? (Useful when many queries are expected.)

Our approach: Build an index on the text s.

First solution: [Naive] Compute and store the Parikh Set of s (i.e., the set of Parikh vectors of all the substrings of s).

- $O(n^2)$ preprocessing time,
- $O(n^2)$ index size,
- $O(\log n)$ query time.

First solution: [Naive] Compute and store the Parikh Set of s (i.e., the set of Parikh vectors of all the substrings of s).

- $O(n^2)$ preprocessing time,
- $O(n^2)$ index size,
- $O(\log n)$ query time.

New goal: Reduce index size and preprocessing time.

Lemma (Cicalese, F., Lipták, PSC 2009)

If (x, y) *and* $(x + k, y - k)$ *both occur in* s*, then so does* $(x + i, y - i)$ *for any* $0 \leq i \leq k$*.*

Lemma (Cicalese, F., Lipták, PSC 2009)

If (x, y) *and* $(x + k, y - k)$ *both occur in* s*, then so does* $(x + i, y - i)$ *for any* $0 \leq i \leq k$*.*

Theorem

To answer BJPM queries for s*, it is sufficient to know, for every* $0 \leq m \leq n$*, the max and the min of a's in the substrings of length* m *of* s*.*

Lemma (Cicalese, F., Lipták, PSC 2009)

If (x, y) *and* $(x + k, y - k)$ *both occur in* s*, then so does* $(x + i, y - i)$ *for any* $0 \leq i \leq k$*.*

Theorem

To answer BJPM queries for s*, it is sufficient to know, for every* $0 \leq m \leq n$*, the max and the min of* a*'s in the substrings of length* m *of* s*.*

So we define:

$F_a(m) = \max\{x : (x, y) \text{ occurs in } s,\ x + y = m\}$

$f_a(m) = \min\{x : (x, y) \text{ occurs in } s,\ x + y = m\}$

Example

$s = aababab baaabbaabbb$.

m	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
F_a	0	1	2	3	3	3	4	5	5	5	6	7	7	7	8	9	9	9	9
f_a	0	0	0	0	1	2	2	2	3	4	5	5	5	6	6	7	7	8	9

Let $(x, y) = (1, 4)$. To answer the query check whether $f_a(5) \leq 1 \leq F_a(5)$.

So we have:

Second solution: [Cicalese, F., Lipták, PSC 2009] Compute and store the tables of F_a and f_a.

- $O(n^2)$ preprocessing time,
- $2n$ index size,
- $O(1)$ query time.

So we have:

Second solution: [Cicalese, F., Lipták, PSC 2009] Compute and store the tables of F_a and f_a.

- $O(n^2)$ preprocessing time,
- $2n$ index size,
- $O(1)$ query time.

Question: Is it possible to reduce the preprocessing time?

So we have:

Second solution: [Cicalese, F., Lipták, PSC 2009] Compute and store the tables of F_a and f_a.

- $O(n^2)$ preprocessing time,
- $2n$ index size,
- $O(1)$ query time.

Question: Is it possible to reduce the preprocessing time?

Best bound known:
- $O(n^2/\log n)$ (Burcsi, Cicalese, F., Lipták, FUN 2010 & Moosa, Rahman, IPL 2010)
- $O(n^2/\log^2 n)$ in the RAM model (Moosa, Rahman, JDA 2012)
- $O(n^{1+\epsilon})$ randomized Monte Carlo algorithm (Cicalese, Laber, Weimann, Yuster, CPM 2012)

Alternatively, one can define:

$G_a(i) = \min\{\#b\text{'s in a substring containing } i \ a\text{'s}\}$

$g_a(i) = \max\{\#b\text{'s in a substring containing } i \ a\text{'s}\}$

Alternatively, one can define:

$G_a(i) = \min\{\#b\text{'s in a substring containing } i\ a\text{'s}\}$

$g_a(i) = \max\{\#b\text{'s in a substring containing } i\ a\text{'s}\}$

Example

$s = aababababbaaabbaabbb$.

i	0	1	2	3	4	5	6	7	8	9
$G_a(i)$	0	0	0	0	2	2	4	4	6	6
$g_a(i)$	3	3	5	5	5	7	8	9	9	9

Let $(x, y) = (1, 4)$. To answer the query check whether $G_a(1) \leq 4 \leq g_a(1)$.

Example

$s = aababababbaaabbaabbb$.

i	0	1	2	3	4	5	6	7	8	9
$G_a(i)$	0	0	0	0	2	2	4	4	6	6
$g_a(i)$	3	3	5	5	5	7	8	9	9	9

Remark: It is sufficient to store the points where the function changes! So we define the (oredered) sets:

$$L_G = \{(3,0),(5,2),(7,4),(9,6)\}$$

$$L_g = \{(0,3),(2,5),(5,7),(6,8),(7,9)\}$$

We call $L = (L_G, L_g)$ the Corner Index of s.

Computation of the Corner Index

Definition

We say that (x, y) dominates (x', y'), denoted $(x, y) \triangleright (x', y')$, if $(x, y) \neq (x', y')$, $x \geq x'$ and $y \leq y'$.

Computation of the Corner Index

Definition

We say that (x, y) dominates (x', y'), denoted $(x, y) \rhd (x', y')$, if $(x, y) \neq (x', y')$, $x \geq x'$ and $y \leq y'$.

Algorithm: For L_G: Compute the Parikh vectors of substring starting with the ith a-run and spanning k a-runs. If no element of L_G dominates (x, y), then it is added to L_G, and all elements of L_G which (x, y) dominates are removed from the list.

$s = aababababbaaabbaabbb$

a	2	1	1	3	2
b	1	1	2	2	3

L_G : ~~(2, 0)~~, (3, 0), ~~(4, 2)~~, (5, 2), ~~(6, 4)~~, (7, 4), (9, 6)

k	
1	(2, 0)(1, 0)(1, 0)(3, 0)(2, 0)
2	(3, 1)(2, 1)(4, 2)(5, 2)
3	(4, 2)(5, 3)(6, 4)
4	(7, 4)(7, 5)
5	(9, 6)

Let ρ be the length of the Run-Length Encoding of s. We have:

Third solution: [Badkobeh, F., Kroon, Lipták, 2012] Compute and store the Corner Index.

- $O(\rho^2 \log \rho)$ preprocessing time,
- $\leq \min(2n, \rho^2)$ index size,
- $O(\log \rho)$ query time.

Let ρ be the length of the Run-Length Encoding of s. We have:

Third solution: [Badkobeh, F., Kroon, Lipták, 2012] Compute and store the Corner Index.

- $O(\rho^2 \log \rho)$ preprocessing time,
- $\leq \min(2n, \rho^2)$ index size,
- $O(\log \rho)$ query time.

The construction time is better than all previous solutions for strings with short RLE (actually, as long as $\rho = O(n/\log n)$).

Experimental results: We generated strings consisting of r a-runs and $r-1$ b-runs (so $\rho = 2r-1$), with run-lengths chosen uniformly from the range $[1, R]$, for various choices of r and R.

Experimental results: We generated strings consisting of r a-runs and $r-1$ b-runs (so $\rho = 2r-1$), with run-lengths chosen uniformly from the range $[1, R]$, for various choices of r and R.

- 10 000 strings for each pair (r, R), with $r = 10, 100, 200, 300, 500$ and $R = 10, 2050, 100, 200, 500, 700, 1000$.

Experimental results: We generated strings consisting of r a-runs and $r-1$ b-runs (so $\rho = 2r-1$), with run-lengths chosen uniformly from the range $[1, R]$, for various choices of r and R.

- 10 000 strings for each pair (r, R), with $r = 10, 100, 200, 300, 500$ and $R = 10, 2050, 100, 200, 500, 700, 1000$.
- $0.8\rho \leq |L| \leq 3\rho$ (we had $|L| \leq \rho^2$).

Experimental results: We generated strings consisting of r a-runs and $r-1$ b-runs (so $\rho = 2r-1$), with run-lengths chosen uniformly from the range $[1, R]$, for various choices of r and R.

- 10 000 strings for each pair (r, R), with $r = 10, 100, 200, 300, 500$ and $R = 10, 2050, 100, 200, 500, 700, 1000$.
- $0.8\rho \leq |L| \leq 3\rho$ (we had $|L| \leq \rho^2$).
- In more than 99% of cases, the maximal size *MaxL* of the index during the computation never exceeded the final index size $|L|$.
  In the remaining $< 1\%$ of cases, $MaxL - |L| \leq 6$.

Experimental results: We generated strings consisting of r a-runs and $r-1$ b-runs (so $\rho = 2r-1$), with run-lengths chosen uniformly from the range $[1, R]$, for various choices of r and R.

- 10 000 strings for each pair (r, R), with $r = 10, 100, 200, 300, 500$ and $R = 10, 2050, 100, 200, 500, 700, 1000$.
- $0.8\rho \leq |L| \leq 3\rho$ (we had $|L| \leq \rho^2$).
- In more than 99% of cases, the maximal size *MaxL* of the index during the computation never exceeded the final index size $|L|$. In the remaining $< 1\%$ of cases, $MaxL - |L| \leq 6$.

Open problem 1: Can the bound $|L| = O(\rho^2)$ be reduced to $|L| = O(\rho)$?

Open problem 2: Does a bound exist on $MaxL - |L|$?

Prefix Normal Forms

Take the values of the tables $F_a(s)$ and $f_a(s)$ and write a when the value increases, b otherwise. Denote by $\mathrm{PNF}_a(s)$ and $\mathrm{PNF}_b(s)$ the words so obtained.

Example

$s = aababbaaabbaabbb$.

m	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
F_a	0	1	2	3	3	3	4	5	5	5	6	7	7	7	8	9	9	9	9
f_a	0	0	0	0	1	2	2	2	3	4	5	5	5	6	6	7	7	8	9

$$\mathrm{PNF}_a(s) = aaabbaabbaabbaabbb$$
$$\mathrm{PNF}_b(s) = bbbaabbaaabbababaa$$

Prefix Normal Forms

Take the values of the tables $F_a(s)$ and $f_a(s)$ and write a when the value increases, b otherwise. Denote by $\mathrm{PNF}_a(s)$ and $\mathrm{PNF}_b(s)$ the words so obtained.

Example

$s = aababab baaabbaabbb$.

m	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
F_a	0	1	2	3	3	3	4	5	5	5	6	7	7	7	8	9	9	9	9
f_a	0	0	0	0	1	2	2	2	3	4	5	5	5	6	6	7	7	8	9

$$\mathrm{PNF}_a(s) = aaabbaabbaabbaabbb$$
$$\mathrm{PNF}_b(s) = bbbaabbaaabbababaa$$

Question: What is the relationship between the PNFs of s and s?

Theorem (F., Lipták, DLT 2011)

$PNF_a(s)$ is the unique word having the same table F_a as s and realizing the maxima on its prefixes (i.e., for each m, no factor of $PNF_a(s)$ of length m contains more a's than the prefix of $PNF_a(s)$ of length m).

Theorem (F., Lipták, DLT 2011)

$PNF_a(s)$ is the unique word having the same table F_a as s and realizing the maxima on its prefixes (i.e., for each m, no factor of $PNF_a(s)$ of length m contains more a's than the prefix of $PNF_a(s)$ of length m).

$$\begin{aligned} s &= aababab baaabbaabbb \\ \mathrm{PNF}_a(s) &= aaabbaabbaabbaabbb \end{aligned}$$

Theorem (F., Lipták, DLT 2011)

$PNF_a(s)$ is the unique word having the same table F_a as s and realizing the maxima on its prefixes (i.e., for each m, no factor of $PNF_a(s)$ of length m contains more a's than the prefix of $PNF_a(s)$ of length m).

$$\begin{aligned} s &= aababababbaaabbaabbb \\ \mathrm{PNF}_a(s) &= aaabbaabbaabbaabbb \end{aligned}$$

Theorem (F., Lipták, DLT 2011)

Two words $u, v \in \{a, b\}^*$ have the same Parikh Set if and only if $PNF_a(u) = PNF_a(v)$ and $PNF_b(u) = PNF_b(v)$.

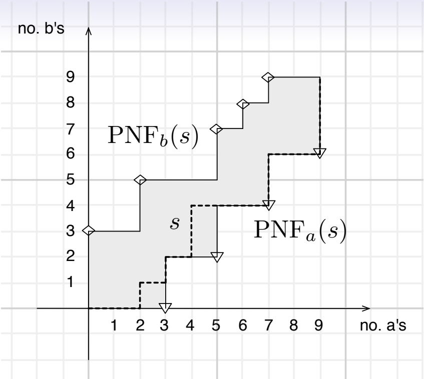

Figure: $\mathrm{PNF}_a(s) = aaabbaabbaabbaabbb$, $\mathrm{PNF}_b(s) = bbbaabbaaabbababaa$.

Recall that

$$
\begin{aligned}
L_G &= \{(3,0),(5,2),(7,4),(9,6)\} \\
L_g &= \{(0,3),(2,5),(5,7),(6,8),(7,9)\}
\end{aligned}
$$

These are the "corner points" of $\mathrm{PNF}_a(s)$ and $\mathrm{PNF}_b(s)$.

So we have:

Theorem

The size of the Corner Index is given by the lengths of the RLE of the PNFs.

So we have:

Theorem

The size of the Corner Index is given by the lengths of the RLE of the PNFs.

Open problems:

1. What is the relationship between the RLEs of s and of its PNFs?
2. What are the words with the "worst" PNFs (w.r.t. the RLE)?
3. Is it possible to compute the PNFs in $o(n^2/\log n)$ time?