General Pattern Matching on Regular Collage System

Jan Lahoda and Bořivoj Melichar

Dept. of Computer Science and Engineering
Faculty of Electrical Engineering
Czech Technical University
Karlovo náměstí 13
121 35 Prague 2
Czech Republic

e-mail: {lahodaj,melichar}@fel.cvut.cz

Abstract. This paper presents a brand new approach to the general pattern matching on regular collage systems. Our approach provides $\mathcal{O}(||D|| + |S| + E)$ (where E is the preprocessing cost) worst-case time complexity. It is based on fact that a deterministic finite automaton is able to distinguish only a limited number of strings.

Keywords: pattern matching in compressed text, pattern matching, text compression, finite automata, collage systems

1 Introduction

In the concurrent world, each year more and more data are to be stored and processed. It also seems that the amount of data to be stored and processed grows faster than the data storage media and processing appliances.

The pattern matching in compressed text helps in both directions: the data are compressed in order to consume less space, and then an algorithm for the pattern matching in compressed text is employed to simplify the data processing.

In this paper, we provide a new approach for general (regular expression) pattern matching over collage systems. Collage systems are means of representing several compression methods in a unique way, and our approach uses finite automata as unique approach for solving many pattern matching problems.

2 Basic notions and notations

Let us denote *pref*(P), *fact*(P) and *suff*(P) set of all prefixes, factors and suffixes (respectivelly) of string P. Let us denote *LSpref*(w, P) and *LSfact*(w, P) longest suffix of w that is concurrently a prefix (factor) of P.

2.1 Collage systems

Collage systems [KST+99, KMT+01] are means of representing several compression methods in a unique way. A collage system is a pair (D, S), where:

D (called dictionary) is a sequence of assignments in form $X_1 = expr_1$; $X_2 = expr_2$; $\cdots$; $X_l = expr_l$ where the expression for assignment X_k is constructed in one of these forms:

a	for any $a \in (A \cup \{\varepsilon\})$,	*(primitive assigment)*
X_iX_j	for $i, j < k$,	*(concatenation)*
${}^{[j]}X_i$	for $i < k$ and an integer j,	*(prefix truncation)*
$X_i^{[j]}$	for $i < k$ and an integer j,	*(suffix truncation)*
$(X_i)^j$	for $i < k$ and an integer j.	*(j times repetition)*

S (called sequence) is a sequence of assignments defined in D in form

$$S = X_{i_1}, X_{i_2}, \ldots, X_{i_n}$$

Let us denote u_i string representing assigment X_i and $u = u_{i_1}u_{i_2}\ldots u_{i_n}$ string representing the collage system.

[KST+99] describes how to express various compression methods using collage systems.

Several types of collage systems were defined in [KST+99]. The two most important types in our case are regular and simple collage systems. The dictionary of a regular collage system can contain assignments only in form of a or X_iX_j. The simple collage systems are such regular collage systems, where for each assigment in form of X_iX_j holds either $X_i = a$ or $X_j = a$ for some $a \in A$.

For example, let us consider the following collage system: $D = \{X_1 = a, X_2 = b, X_3 = X_1X_2, X_4 = X_3X_3\}$, $S = \{X_4X_4\}$. Then the assignments represent the following strings $u_1 = a$, $u_2 = b$, $u_3 = ab$, $u_4 = abab$ and the whole collage system represents string $u = abababab$.

3 Previous work

The collage systems were defined in [KST+99] as generalization of several compression methods. In this paper, an algorithm for exact one pattern matching was provided. In [KMT+01], an algorithm for exact multiple pattern has been provided. For a given collage system and pattern(s) of total length m, these algorithms have time complexity $\mathcal{O}(||D|| + |S| + m^2 + r)$ and space complexity $\mathcal{O}(||D|| + m^2)$.

4 Main result

In this section we will present algorithm for pattern matching on collage systems. In order to make the pattern matching algorithm run in time $\mathcal{O}(||D|| + |S| + \tau)$ (where τ represents preprocessing time) it is necessary to use only $\mathcal{O}(1)$ time for each item in the sequence S and each expression in dictionary D. Therefore, in this section we present a new approach to compute "descriptions" of each $X_i \in D$ so we are able to

create and update description for each X_i in $\mathcal{O}(1)$ time and to process each item in sequence S in $\mathcal{O}(1)$ time.

Our solution is based on the fact that there is only a very limited number of strings that behave "differently" in a given pattern matching automaton (see Definition 1). So the main idea is to find a (shorter) string (so called representant string), from a limited set of predefined strings, for each dictionary item, that will behave in the same way in the pattern matching automaton. As will be shown later, the representant string of concatenation of two representant strings of two dictionary items can then be computed in $\mathcal{O}(1)$.

Definition 1. *Let $M = (Q, A, \delta, q_0, F)$ is a pattern matching automaton (deterministic finite automaton) and δ^* is a transitive reflexive closure of the transition function δ. Then relation $\sim_M$ (shortcut $\sim$ will be used in the future when the automaton M is clear from the context) is defined as follows: for each two strings $u, v \in A^*$ holds that $u \sim v$ if and only if for each $q \in Q$ holds:*

1. *$\delta^*(q, u) = \delta^*(q, v)$*
2. *exactly one of the following is true:*
   - *there exist strings $u' \in A^*$, $u' \in pref(u)$ and $v' \in A^*$, $v' \in pref(v)$ such that $\delta^*(q, u') \in F$ and $\delta^*(q, v') \in F$,*
   - *for all strings $u' \in A^*$, $u' \in pref(u)$ and $v' \in A^*$, $v' \in pref(v)$ holds that $\delta^*(q, u') \notin F$ and $\delta^*(q, v') \notin F$,*

Definition 2. *For a given deterministic finite automaton M, let us suppose that an ordering is given on the set of states Q, so we can enumerate the states in an order.*

For each string $u \in A^*$, let us define signature $\mathcal{S}(u)$ as a vector of pairs $\mathcal{S}(u) = ((q'_1, f_1), \ldots, (q'_{|Q|}, f_{|Q|}))$ of length $|Q|$, where a pair (q'_i, f'_i), $q'_i \in Q$ and $f_i \in \{true, false\}$. On position i is computed as $q'_i = \delta^*(q_i, u)$ and boolean f_i is true if and only if there is a prefix u' of u such that $\delta^*(q_i, u') \in F$, false otherwise.

Example 3. *Let us consider finite deterministic automaton shown in Figure 1. For order of states: (A, B, C), the signatures are as follows:*

u	$\mathcal{S}(u)$		
	A	B	C
ε	(A, f)	(B, f)	(C, f)
a	(B, t)	(B, f)	(B, f)
ab	(C, t)	(C, t)	(C, t)
aba	(B, t)	(B, t)	(B, t)
abc	(A, t)	(A, t)	(A, t)
b	(A, t)	(C, t)	(A, f)
ba	(B, t)	(B, t)	(B, f)
bc	(A, t)	(A, t)	(A, f)
c	(A, f)	(A, f)	(A, f)

Theorem 4. *For a given deterministic finite automaton M and for each two strings $u, v \in A^*$ holds that $u \sim v$ if and only if $\mathcal{S}(u) = \mathcal{S}(v)$.*

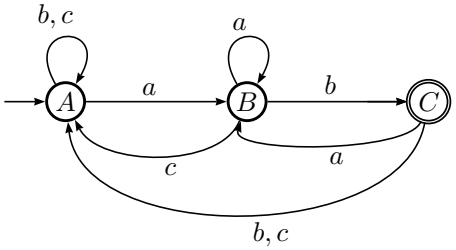

Figure 1: Example pattern matching automaton

Proof. Leads directly from Definitions 1 and 2. □

Example 5. *Let us consider finite deterministic automaton shown in Figure 1 and the order of states:* (A, B, C). *Then the signature of strings aba and cccaba is the same:* $\mathcal{S}(aba) = \mathcal{S}(cccaba) = ((B,t),(B,t),(B,t))$. *Therefore it holds that* $aba \sim cccaba$.

Theorem 6. *The relation* $\sim$ *is equivalence, and moreover, it is right congruence.*

Proof. To prove that relation $\sim$ is an equivalence, we need to prove that it is reflexive, symmetric and transitive (for each $u, v, w \in A^*$):

reflexivity it is obvious that $\mathcal{S}(u) = \mathcal{S}(u)$,

symmetry if $u \sim v$, then $\mathcal{S}(u) = \mathcal{S}(v)$, and also $\mathcal{S}(v) = \mathcal{S}(u)$, and therefore $v \sim u$,

transitivity if $u \sim v$ and $v \sim w$, then $\mathcal{S}(u) = \mathcal{S}(v)$ and $\mathcal{S}(v) = \mathcal{S}(w)$, then $\mathcal{S}(u) = \mathcal{S}(w)$ and therefore $u \sim w$.

To prove that relation $\sim$ is a right congruence, it is necessary to prove that for all $\alpha, \beta, \gamma \in A^*$, such that $\alpha \sim \beta$ holds $\alpha\gamma \sim \beta\gamma$. Let us prove this by contradiction: let us suppose there is an automaton M, and strings $\alpha, \beta, \gamma \in A^*$, such that $\alpha \sim \beta$, but not $\alpha\gamma \sim \beta\gamma$. This means that there exists a state $q \in Q$ such that at least one of there is true:

1. $\delta(q, \alpha) = \delta(q, \beta)$, but $\delta(q, \alpha\gamma) \neq \delta(q, \beta\gamma)$,

2. no $\alpha' \in \mathit{pref}(\alpha)$ and $\beta' \in \mathit{pref}(\beta)$ exists such that $\delta(q, \alpha') \in F$ and $\delta(q, \beta') \in F$ and there exists $(\alpha\gamma)' \in \mathit{pref}(\alpha\gamma)$ such that $\delta(q, (\alpha\gamma)') \in F$ and there is no $(\beta\gamma)' \in \mathit{pref}(\beta\gamma)$ such that $\delta(q, (\beta\gamma)') \in F$ (or vice versa).

The first variant is not possible, because $\delta(q, \alpha) = \delta(q, \beta) = q'$ and $\delta(q', \gamma) = q''$, and therefore $q'' = \delta(q, \alpha\gamma) = \delta(q, \beta\gamma)$.

The second variant is not possible, because obviously: $\delta(q, \alpha) = \delta(q, \beta) = q'$, $|(\alpha\gamma)'| > |\alpha|$ and therefore there must be $\alpha\gamma' = (\alpha\gamma)'$ and so if $\delta(q, \alpha\gamma') \in F$ then $\delta(q', \gamma') \in F$ and $\delta(q, \beta\gamma') \in F$, and so exists $\beta\gamma' \in \mathit{pref}(\beta\gamma)$ which breaks this condition.

Therefore, such automaton M and string α, β, γ cannot exist, and therefore the equivalence $\sim$ is a right congruence. □

The equivalence defined in the Definition 1 defines strings that behave "in the same way" in the pattern matching automaton (they lead for a given state $q \in Q$ into the same state q' and they remember whether or not they passed through a final state).

Definition 7. *Let $W \subset A^*$ be a set of class representatives for partition $A^*/\sim$ of A^*, such that for each $w \in W$ holds that there does not exist any w' such that $w \sim w'$ and $|w'| < |w|$.*

Theorem 8. *For a given automaton $M = (Q, A, \delta, q_0, F)$, the corresponding set W is finite and moreover has at most $(2|Q|)^{|Q|}$ elements.*

Proof. As for each pair of strings $u, v \in A^*$ holds that $u \sim v$ if and only if $\mathcal{S}(u) = \mathcal{S}(v)$, it is therefore clear that there cannot be more classes of equivalence in partition $A^*/\sim$ than is the number of distinct vectors. The number of different vectors is $(2|Q|)^{|Q|}$ for a given automaton (each tuple of the vector can contain $2|Q|$ distinct values, and $|Q|$ tuples are independently combined into a vector). □

Although the size of set of representatives W is overwhelming, for most practical purposes the size of this set is much smaller. Section 5 analyses these cases.

Algorithm 4 shows how to construct the set of representatives W for a given automaton M.

Algorithm 4 Construction of the set of representatives W

Require: Deterministic finite automaton M
Ensure: Set of representatives W correspoding to the automaton M

```
1: U = {ε}
2: while U is not empty do
3:     remove a w from U such that there is no w' ∈ U such that |w| < |w'|.
4:     if S(w) is not in S_M then
5:         W = W ∪ {w}
6:         for each a ∈ A put wa into U
7:         S_M = S_M ∪ {S(w)}
8:     end if
9: end while
```

To solve the pattern matching problem on the (regular) collage system in $\mathcal{O}(||D|| + |S|)$ time, it is necessary to compute $w \in W$ corresponding to each item in $\mathcal{O}(1)$ time. For the simple assignment (like $X_i = a$), it is trivial. In order to compute representant string for $X_k = X_i X_j$ from representant strings of X_i and X_j, a characteristic automaton M_H is defined.

Definition 9. *For a given deterministic finite automaton M and correspoding set of representatives W,* characteristic *automaton $M_H = (Q_H, W, \delta_H, q_{H0}, \emptyset)$ is defined in the following way:*

Q_H: $Q_H = W$,

$\delta_H : Q_H \times W \to Q_H$: $\delta_H(q_H, w) = u$, *where $w, u \in W$ such that $q_H w \sim u$,*

q_{H0}: $q_{H0} = \varepsilon$.

The automaton M_H has obviously space complexity $\mathcal{O}(|W|^2)$ for regular collage systems. For simple collage systems, simplified characteristic automaton can be employed which space complexity is only $\mathcal{O}(|W||A|)$ (only expressions in form $X_k = X_i a$ or $X_k = aX_i$, where $a \in A$ are allowed).

Another problem to solve is the match detection. This can be done using a special final markers table $\mathcal{F}$.

Definition 10. *For a given deterministic finite automaton M and correspoding set of representatives W, the final markers table $\mathcal{F}$ is defined for each $w \in W$ and $q \in Q$ such that $\mathcal{F}[w, q] = true$ if and only if there exists a $w' \in pref(w)$ such that $\delta(q, w') \in F$.*

Algorithm 5 Pattern Matching on Regular Collage Systems

▷ Preprocessing phase
for the given pattern P and pattern matching problem $\mathcal{P}$ construct pattern matching automaton M, characteristic automaton M_H and final markers table $\mathcal{F}$.
compute representative for each dictionary item from the dictionary D
▷ Pattern matching phase
$q = q_0$
$j = 0$
for all X from $S = \{X_{i_1}, X_{i_2}, \ldots, X_{i_n}\}$ **do**
    let $w \in W$ is the representant string corresponding to X
    if $\mathcal{F}[w, q]$ is true **then**
        report occurence(s) between positions j and $j + |X.u|$
    end if
    $q = \delta(q, w)$
    $j = j + |X.u|$
end for

5 On the Size Of W

Although the worst-case size of the set of representatives W for a given automaton M is overwhelming (up to $(2|Q|)^{|Q|}$), for many practical cases the size of this set is much smaller. In this section, a proof that for exact one pattern matching of an aperiodic pattern of length m, the size of the set W is $\mathcal{O}(m^2)$. Moreover, results of pratical experiments for commonly used patterns and pattern matching problem are discussed.

5.1 Exact One Pattern Matching

In this section, we prove that for each deterministic finite automaton constructed to solve exact one pattern matching for an aperiodic pattern of length m (see Definition 14), the size of set W is $\mathcal{O}(m^2)$.

Moreover, our experiments have shown, that the aperiodic pattern is the worst-case with regard to the size of set W. We have created automata and sets W for all patterns of length 6, and none of these patterns performed worse than the aperiodic pattern.

Definition 11. *Let automaton $M = (Q, A, \delta, q_0, F)$ be a pattern matching automaton for exact one pattern matching of pattern P.*

Then for each state $q \in Q$ exists exactly one string $u \in A^*$ such that $u \in pref(P)$, $\delta(q_o, u) = q$ and there is no shorter prefix with the same property. Let us define function corr, which for each state q has value of the appropriate string u.

Lemma 12. *For a given automaton M, let us define set W' which fulfills these properties:*

1. *$\varepsilon \in W'$*
2. *for each $a \in A$ and $w' \in W'$ exists $u' \in W'$ such that $u' \sim w'a$*

Then for each $w \in A^*$ exists a $w' \in W'$ such that $w \sim w'$.

Proof. (by induction by the length of w)

1. For $w = \varepsilon$, $w \in A^*$, there clearly exists $w' = \varepsilon$ (the first condition on set W'), such that $w \sim w'$.
2. Let us suppose that the claim holds for all $w \in A^*$, $|w| \leq k$. Than for each $a \in A$ and $w \in W$, $|w| \leq k$ holds: there exists $w' \in W'$ such that $w \sim w'$. There also exists $u' \in W'$ such that $u' \sim w'a$. As the equivalence $\sim$ is a right congruence, it also holds that $wa \sim u'$. The claim therefore also holds for all $|wa| \leq k + 1$.

□

Corollary 13. *For set W' defined in Lemma 12 holds that $|W| \leq |W'|$.*

Proof. (by contradiction)Let us suppose there exists such automaton M, correspoding set of class representatives W and a set W' such that $|W| > |W'|$. But then there must be two $w_1, w_2 \in W$, $w_1 \neq w_2$ such that there exists $w' \in W'$, $w_1 \sim w'$, $w_2 \sim w'$. As $\sim$ is equivalence, it is clear that $w_1 \sim w_2$, and that means that strings w_1 and w_2 are in the same class of equivalence, and therefore set W is not set of class representatives, which is the contrandiction with the assumptions. Therefore such automaton M and sets W and W' cannot exist. □

Definition 14. *Let us call pattern $P = a_1 a_2 \dots a_m$ of length m such that for each two $i, j \in < 1, m >$, $i \neq j$ holds $a_i \neq a_j$ aperiodic.*

Lemma 15. *For an aperiodic pattern $P = a_1 a_2 \cdots a_m$ of length m, alphabet $A = a_1, \dots, a_m, x$, and corresponding automaton M, construct set $W' = \{w' : w' \in fact(P)$ or $w' = sxp, s \in suff(P), p \in pref(P)\}$. The set W' fulfills requirements defined in Lemma 12.*

Proof. As $\varepsilon \in fact(P)$, it is clear that $\varepsilon \in W'$.

Let as for each factor $f \in fact(P)$ denote a_n the symbol in A for which $fa \in fact(P)$. Note that there is no a_n for all $f \in suff(P)$.

For each $w' \in W'$ and $a \in A$, let us analyse all possibilities (for each combination of w' and a, only the topmost step is valid):

- $w' = \varepsilon$: it clearly holds $a \in \mathit{fact}(P)$ or $a = x$, and so $a \in W'$
- $w' \in \mathit{fact}(P)$, $a = a_n$: it clearly holds $fa \in \mathit{fact}(P)$ and so $fa \in W'$
- $w' \in \mathit{fact}(P)$, $w' \notin \mathit{suff}(P)$, $a = a_1$: as for each state $q \in Q$ holds that $\delta(q, a_1) = q_1$, and that there is no such $q \in Q$ and $w'' \in \mathit{pref}(w')$ such that $\delta(q, w'') \in F$, it holds that $w'a \sim a_1$.
- $w' \in \mathit{fact}(P)$, $w' \notin \mathit{suff}(P)$, $a \neq a_1$, $a \neq a_n$: as the pattern is not periodic, the longest suffix of $w'a$ that is prefix of P is ε, and that there is no such $q \in Q$ and $w'' \in \mathit{pref}(w')$ such that $\delta(q, w'') \in F$, it holds that $w'a \sim \varepsilon$.
- $w' = P$, $a = a_1$: as for each state $q \in Q$ holds that $\delta(q, P) = q_m \in F$, $\delta(q_m, a_1) = q_1$, it holds that $Pa \sim Pxa$.
- $w' = P$, $a \neq a_1$: as for each state $q \in Q$ holds that $\delta(q, P) = q_m \in F$, $\delta(q_m, a) = q_0$, it holds that $Pa \sim Px$.
- $w' \in \mathit{suff}(P)$, $a = a_1$: as for one state $q_s \in Q$ holds that $\delta(q_s, w') = q_m$, and for all other $q \in Q$ holds that $\delta(q, w') = q_0$, it holds that $w'a \sim w'xa_1$.
- $w' \in \mathit{suff}(P)$, $a \neq a_1$: as for one state $q_s \in Q$ holds that $\delta(q_s, w') = q_m$, and for all other $q \in Q$ holds that $\delta(q, w') = q_0$, it holds that $w'a \sim w'x$.
- $w' = sxp$ for some $s \in \mathit{suff}(P)$, $p \in \mathit{pref}(P)$, $a = a_n$ (a_n regarding prefix p), $pa_n \neq P$: it clearly holds that: $w'a \in W'$.
- $w' = sxp$ for some $s \in \mathit{suff}(P)$, $p \in \mathit{pref}(P)$, $a = a_n$ (a_n regarding prefix p), $pa_n = P$: it clearly holds that: $w'a \sim P$.
- $w' = sxp$ for some $s \in \mathit{suff}(P)$, $p \in \mathit{pref}(P)$, $a = a_1$, $p \neq P$: as $\delta^*(q, pa_1) = q_1$ for all $q \in Q$, and there is no $p'' \in \mathit{pref}(p)$ such that $\delta(q, p'') \in F$, it holds that $w'a \sim sxa_1$.
- $w' = sxp$ for some $s \in \mathit{suff}(P)$, $p \in \mathit{pref}(P)$, $a = a_1$, $p = P$: as $\delta^*(q, Pa_1) = q_1$ for all $q \in Q$, and $\delta(q, P) \in F$, it holds that $w'a \sim Pxa_1$.
- $w' = sxp$ for some $s \in \mathit{suff}(P)$, $p \in \mathit{pref}(P)$, $p \neq P$: as $\delta^*(q, pa) = q_0$ for all $q \in Q$, and there is no $p'' \in \mathit{pref}(p)$ such that $\delta(q, p'') \in F$, it holds that $w'a \sim sx$.
- $w' = sxp$ for some $s \in \mathit{suff}(P)$, $p \in \mathit{pref}(P)$, $p = P$: as $\delta^*(q, Pa) = q_0$ for all $q \in Q$, and $\delta(q, P) \in F$, it holds that $w'a \sim Px$.

□

Lemma 16. *For an aperiodic pattern $P = a_1a_2 \cdots a_m$ of length m, alphabet $A = a_1, \ldots, a_m, x$, and corresponding automaton M for exact one pattern matching, the set W has at most $\mathcal{O}(m^2)$ items.*

Proof. As the set defined in the Lemma 15 has at most $\mathcal{O}(|Q|^2)$ elements, and according to Corollary 13, the set W has at most $\mathcal{O}(|Q|^2)$ elements.

As proven in [Hol00], the automaton for exact one pattern matching of pattern of length m has $m + 1$ states ($|Q| = m + 1$) and therefore $|W| = \mathcal{O}(m^2)$. □

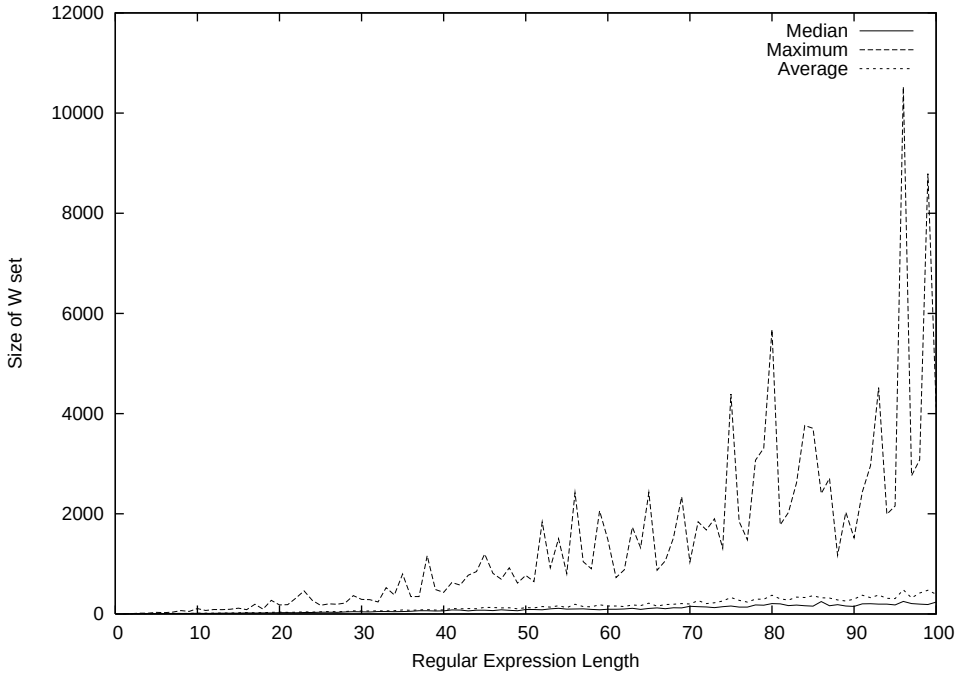

Figure 2: Size of set of representants W for a random regular expression of given length

5.2 Regular Expressions

We have constructed 100 random regular expressions for lengths 1 to 100 (therefore we tested 10000 regular expressions). We define the length of the regular expression as the number of symbols from the alphabet in the regular expression, so operators and brackets are not counted into the length of the regular expression. The regular expressions were prepended with ".*" to simulate pattern matching algorithms. The results from these experiments are summarised in Figure 2.

As can be seen from the graph, the size of the set W for our regular expressions grows much less than $|Q|^{|Q|}$ (note that $Q = \mathcal{O}(2^m)$ where m is the length of the regular expression). Therefore, it seems that the proposed algorithm may be usefull for a wide range of practical applications.

6 Conclusion

In this paper, a new method for general pattern matching on collage systems is presented. This method allows general pattern matching on the regular collage systems in linear time with respect to the size of the collage system.

Although the preprocessing time and space requirements of this method may be very high, in Section 5 is shown that for some practical applications the requirements are more acceptable. Moreover, it is possible to used here-presented approach as long as the preprocessing requirements are acceptable (gaining very fast processing time) and resort to another algorithm (decompress&search in the worst-case) otherwise.

References

[Hol00] J. Holub. *Simulation of Nondeterministic Finite Automata in Pattern Matching*. PhD thesis, Faculty of Electrical Engineering, Czech Technical University, Prague, Czech Republic, 2000.

[KMT+01] Takuya Kida, Tetsuya Matsumoto, Masayuki Takeda, Ayumi Shinohara, and Setsuo Arikawa. Multiple pattern matching algorithms on collage system. *Lecture Notes in Computer Science*, 2089:193–206, 2001.

[KST+99] T. Kida, Y. Shibata, M. Takeda, A. Shinohara, and S. Arikawa. A unifying framework for compressed pattern matching. In *procString Processing and Information Retrieval: A South American Symposium'99*, pages 89–96. IEEE CS Press, 1999.