http://dx.doi.org/10.35596/1729-7648-2020-18-6-33-40

Original paper

UDC 004.043; 004. 048; 004.415.532; 004.422.614; 004.422.63; 004.424.64

STRING PROCESSING MODEL FOR KNOWLEDGE-DRIVEN SYSTEMS

VALERIAN P. IVASHENKO

Belarusian State University of Informatics and Radioelectronics (Minsk, Republic of Belarus)

Submitted 24 May 2020

Abstract. The purpose of the work is to confirm experimentally theoretical estimates for time complexity of operations of the string processing model linked with the metric space for solving data processing problems in knowledge-driven systems including the research and comparison of the operation characteristics of these operations with the characteristics of similar operations for the most relevant data structures. Integral and unit testing were used to obtain the results of the performed computational experiments and verify their correctness. The C \ C++ implementation of operations of the string processing model was tested. The paper gives definitions of concepts necessary for the calculation of metric features calculated over strings. As a result of the experiments, theoretical estimates of the computational complexity of the implemented operations and the validity of the choice of parameters of the used data structures were confirmed, which ensures near-optimal throughput and operation time indicators of operations. According to the obtained results, the advantage is the ability to guarantee the time complexity of the string processing operations no higher than $O\left(\ln^k(n)*\sqrt{n}\right)$ at all stages of a life cycle of data structures used to represent strings, from their creation to destruction, which allows for high throughput in data processing and responsiveness of systems built on the basis of the implemented operations. In case of solving particular string processing problems and using more suitable for these cases data structures such as vector or map the implemented operations have disadvantages meaning they are inferior in terms of the amount of data processed per time unit. The string processing model is focused on the application in knowledge-driven systems at the data management level.

Keywords: strings processing, lists processing, string operation, concatenation, string splitting, searching by key.

Conflict of interests. The author declares no conflict of interests.

For citation. Ivashenko V.P. String processing model for knowledge-driven systems. Doklady BGUIR. 2020; 18(6): 33-40.

Foreword

Knowledge-driven systems encompass a number of management levels [1], each of them correlated to a set of problems to be solved at this level. The level of data management is responsible for processing data that are presented as per the data representation model in a specified language. The texts in these languages are strings. Thus, the problems of data processing can be reduced to those of string processing[1].

In knowledge-based systems that include knowledge-driven systems, denotation semantics, including model-theoretic semantics in logical knowledge representation models and partially

[1] Smith B. *Methods and algorithms of string computations*. Moscow: OOO “I.D. Williams”; 2006.

operational semantics are expressed through mathematical abstractions, considering such fundamental ones as sets. Moreover, in the theory of formal languages the strings themselves are meant to be sets, and thus such systems that deal with strings as the fundamental form of receiving represented data require implementation of set processing operations at the syntactic and semantic levels. Therefore, it is crucial to consider the general model allowing effective performance of string operations for the sake of effective implementation of set-theoretic knowledge processing operations, including those of metric characteristics.

The presence of such general model has a special demand in the systems oriented at the integration of various types of knowledge and problem-solving models. Efficiency is considered primarily not in terms of performance, but in terms of the maximum delays that occur when performing operations that are closely related to the minimum throughput. Minimization of the delays and increasing minimum throughput in light of scalability of system may be more prominent that total problem time for the systems requiring adaptation to the surrounding medium in real-time close conditions.

String processing problems are solved in compliance with the proposed model[2]. As the primary string processing model, the formal data processing model is considered[3], containing the following string processing operations: addition of an element in the string end (push), deletion of an element from the string end (pop), deletion of string elements (popall), concatenation of strings (reduction operation) (concat), string splitting (split), obtaining of a string element by index (valueat), start of iterating (start), iterating by string forward and backward (increment and decrement), break of iterating (break), checking for an empty symbol (over), obtaining of a current string symbol (value), keeping of an element (keep), memorizing by (name) number (memorize), remembering by (name) number (remember), losing by (name) number (lose). A special feature is that the model considers operations of the entire life cycle, including operations that allow destroying data structures.

Software often uses different data structures for different problems and operations that are optimized for these problems. However, when solving problems that require operations for different data structures, which is typical for the systems, focused on knowledge integration, data conversion from one structure to another is inefficient. In addition, the use of multiple data structures in some cases complicates implementation and makes it difficult to analyze. The problem is to create a model for processing strings with operations of the main set with minimal delays, focused on the implementation of set processing. The purpose of the work is to confirm, based on computational experiments, theoretical estimates of the time complexity of operations of the string processing model, considered as elements of the metric space, for solving data processing problems in knowledge-driven systems.

This paper considers a sequential implementation of string processing operations and does not address parallel implementation.

Routine of experiment

As the result of the experiment, it is expected to obtain data to study a dependence of the operation time on the size of the string in order to compare it with the theoretical one. Also, based on experimental data, it is supposed to get the dependencies of the throughput of operations and compare them with the throughput of the corresponding operations for vector/map structures. It is expected that the obtained practical data will correspond to the theoretical dependencies, and the minimum throughput of the implemented operations will exceed that for the vector/map data structures. To estimate throughput, both the string elements and the strings themselves, which are accessible from other strings using link elements, are taken as a unit of data. This nesting of strings makes it possible to implement complex structures of knowledge representation languages at a higher level, including semantic networks.

[2] Ivashenko V.P., editors: Shilin L.Yu. *Model for processing strings and lists of data for knowledge-driven systems*. Minsk: BSUIR; 2018:106-107.

[3] Ivashenko V.P., Belchikov A.S., Eremeev A.P., editors: Shilin L.Yu. *Models of information processing in intelligent systems based on semantic technologies*. Minsk: BSUIR; 2016: 106-107.

Traditional ways of representing strings (arrays) cannot meet the requirements, so string processing operations are implemented on the basis of memory allocation and reallocation operations [1] and string representation in form of a sequence tree [2], which is a type of B-tree with the structures on its basis used in file systems and various relational DBMS. However, this paper examines the work of not only the operations characteristic of such trees, but also the operations of their connection and separation. The operations of the string processing model were implemented using JavaScript[4] [a] and C \ C++. To verify the implemented operations and conduct computational experiments, Unit [3, 4], regression and integration testing [5] were used. Unit tests were designed to check the correctness of operations, while integration tests implemented fragments of the main scenarios for working with strings. The results of multiple measurements of the integration test running time were averaged and the average values were taken as experimental data.

Results and discussion

The results of computational experiments showed a partial superiority of the implemented operations in comparison with operations on JavaScript arrays. Computational experiments were also performed (Windows 7, AMD A6-3400M APU, 4GB DDR3-666) and some of their results are represented by the graphs in Fig. 1 and 2, which show that in comparison with operations on vector\map structures, the implemented operations of the string processing model on string sizes up to 1000000 elements are close in operation time and provide higher throughput (at least 1000 elements per second). Also, the analysis of graphs demonstrates that the nature of dependencies for push, pop, increment, and keep operations is close to linear, which corresponds to the theoretical estimate of the operating time for a single data element: $O\big(h(r,q,n)+\log_q(n)\cdot g(n)\big)$ ($g(n)\sim 1$; $q\sim\log(n_{\max})\cdot\sqrt{n_{\max}}$; $n\le n_{\max}$; $h(r,q,n)\sim 1$), where n is the string length, q is the number of descendants of the tree node. Some deviations from the linear nature of the dependence can be explained by the influence of memory and cache mechanisms. For graphs of split and concat operations, the nature of the dependence per data unit is close to power-law, which corresponds to the theoretical estimate $O\big(h(r,q,n)+\log_q(n)\cdot\big(g(n)+f(n)\cdot q\big)\big)$ ($f(n)\sim 1$). Similar conclusions can be drawn from the graphs of other operations: $O\big(h(r,q,n)+\log_q(n)\cdot g(n)+\big(\log_2^2(n)+\log_q(n)\cdot q\big)\cdot f(n)\big)$ – theoretical estimate.

All operations are performed on a structure of the same type, i. e. the cost of copying data from one structure to another (from vector to map) and back is eliminated. This allows you to use the proposed implementation to create systems with minimal delays for operating conditions close to real-time modes. The confirmed characteristics allow us to implement sequential destructive operations of intersection and joining of sets with subquadratic time complexity. Further improvement is possible due to parallel processing.

A study was made of the dependence of the operation time for various parameter values (the size of the non-leaf node and the size of the leaf node of the sequence tree) of structures used for data representation. Dependency graphs are shown in Fig. 3. Based on the research results, the parameters were selected that provide the best operating time.

Within the framework of a given string processing model, such tasks as searching for the largest (maximum) common subsequence of two strings, searching for the largest (maximum) common substring [6, 7], calculating the value of metrics on strings [8–11], and others [7] are solved. To calculate metrics and construct a metric space based on a given model, additional concepts are introduced that allow calculating the number of characters that can be changed, rearranged, generated, duplicated, absorbed, and destroyed.

[4] Ivashenko V. P., editors: Shilin L.Yu. *Model for processing strings and lists of data for knowledge-driven systems*. Minsk: BSUIR; 2018:106-107.

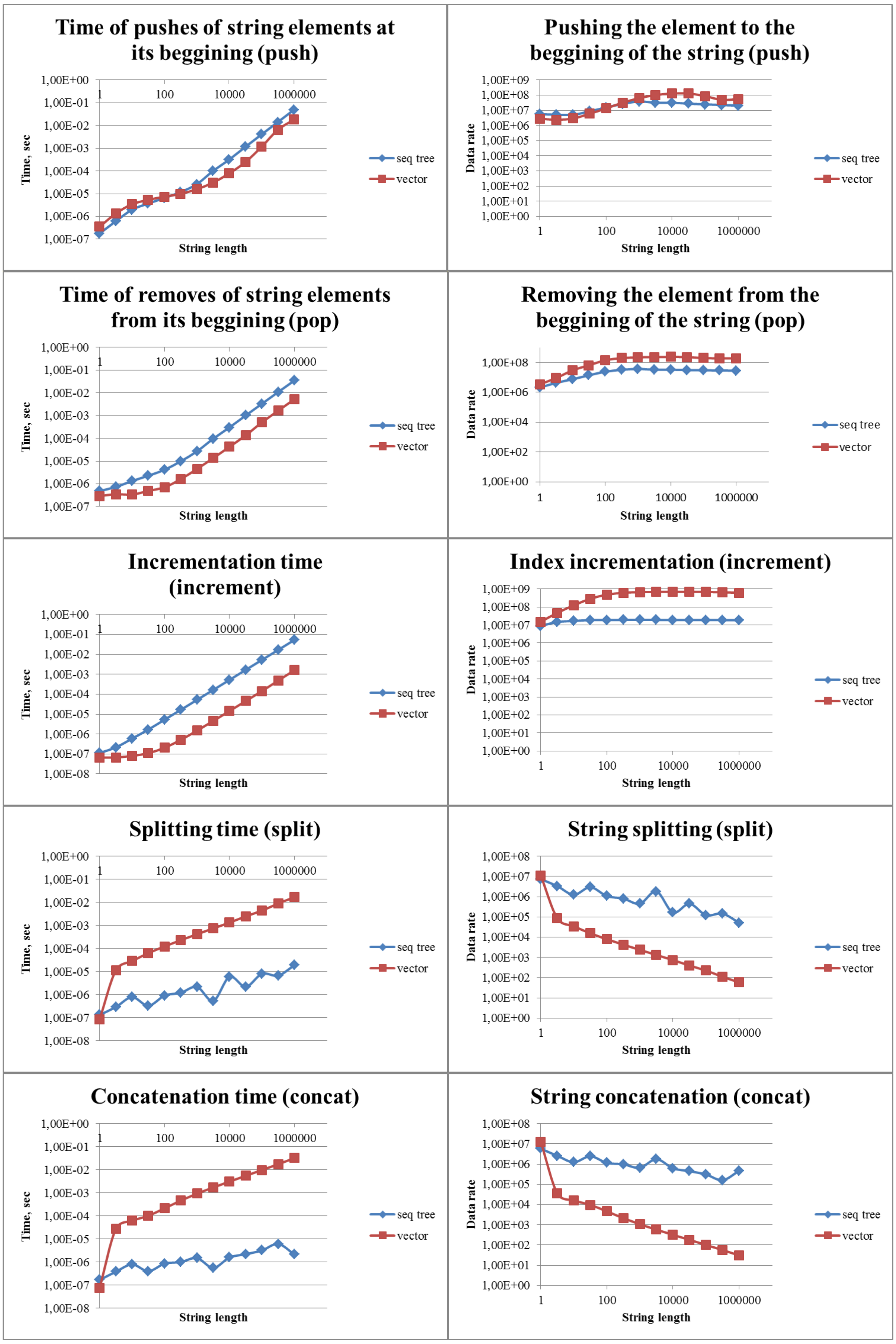

Fig. 1. The results of the computational experiments comparing to vector data structure operations

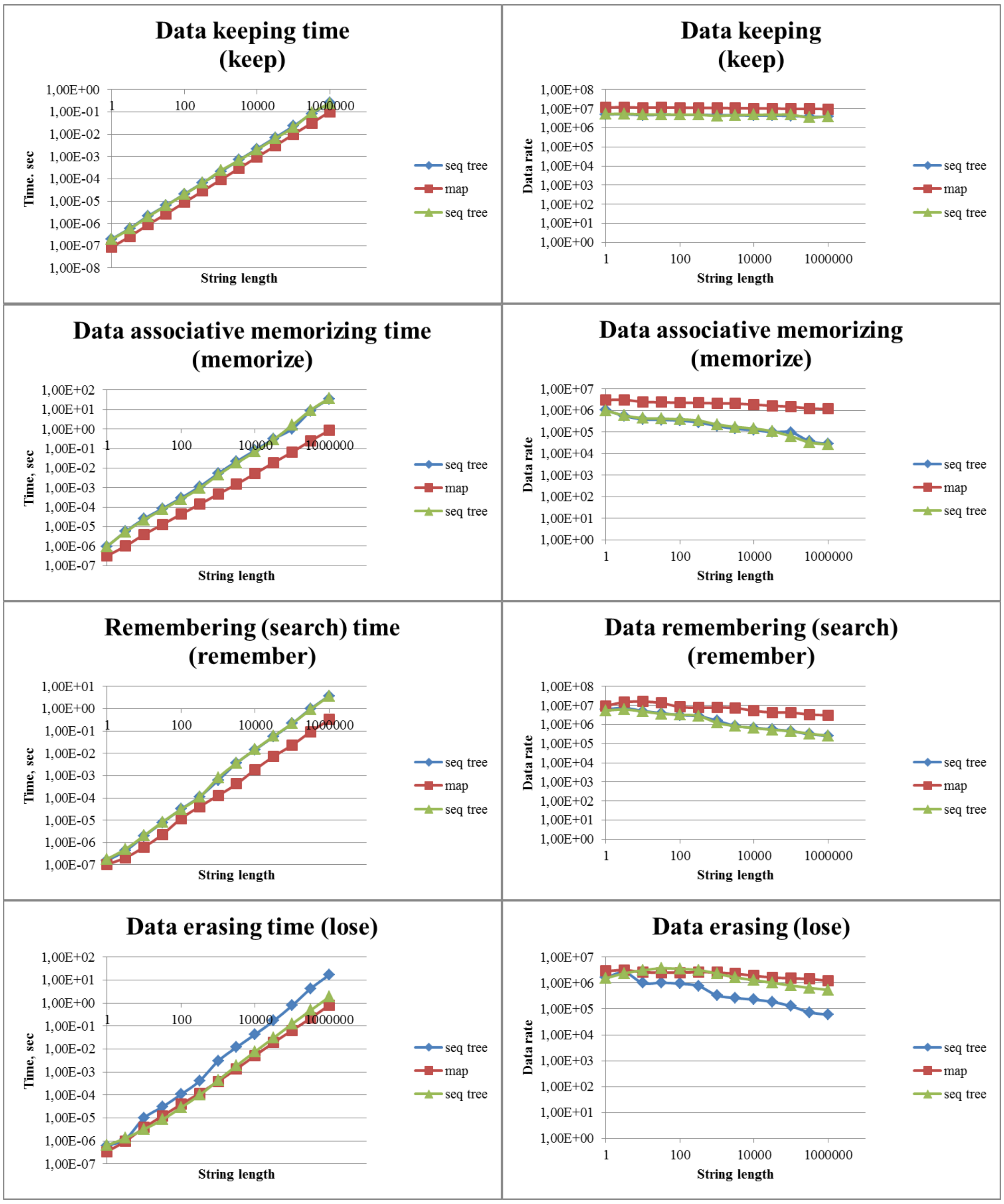

Fig. 2. The results of the computational experiments comparing to map data structure operations

Let the set of string elements χ be:

$$\delta(\chi)=\bigcap_{S}^{\chi\in S^{n}} S. \tag{1}$$

Assume $\varepsilon(\langle\alpha,\beta\rangle)$ is the editorial prescription (program):

$$\varepsilon(\langle\alpha,\beta\rangle)\in\{M,T,R,I,D\}^{\{i|(i\in(\mathbb{N}/\{0\}))\wedge(i\leq\max(\{\dim(\alpha)\}\cup\{\dim(\beta)\}))\}}, \tag{2}$$

which matches each occurrence of a component in a string with the command type: M – matching, T – interchanging, R – replacement, I – insertion (adding), D – reduction (deletion), where $\dim(\chi)$ is the χ string length, and the number of corresponding types of commands is:

$$\sigma\left(\langle\alpha,\beta,\lambda\rangle\right)=\left|\left\{i\,\middle|\,\varnothing\subset\left(\left(\{i\}\times\lambda\right)\cap\varepsilon\left(\langle\alpha,\beta\rangle\right)\right)\right\}\right|,\text{ where }\lambda\subseteq\{M,T,R,I,D,X,G,C,P,E,F\},\qquad(3)$$

$$\sigma\left(\langle\alpha,\beta,\{M\}\rangle\right)=\sigma\left(\langle\beta,\alpha,\{M\}\rangle\right).\qquad(4)$$

Assume the minimum total multiplicity $\pi\left(\langle\alpha,\beta,\chi\rangle\right)$ of χ occurrence in α and β and multiplicity $\kappa\left(\langle\alpha,\beta,\chi\rangle\right)$ of exceeded occurrence of χ in α as compared to β:

$$\pi\left(\langle\alpha,\beta,\chi\rangle\right)=\min\left(\left\{\sigma\left(\langle\alpha,\gamma,\{M\}\rangle\right)\middle|\gamma\in\{\chi\}^{\dim(\alpha)}\right\}\cup\left\{\sigma\left(\langle\beta,\gamma,\{M\}\rangle\right)\middle|\gamma\in\{\chi\}^{\dim(\beta)}\right\}\right);$$

$$\kappa\left(\langle\alpha,\beta,\chi\rangle\right)=\max\left(\left\{\sigma\left(\langle\alpha,\gamma,\{M\}\rangle\right)\middle|\gamma\in\{\chi\}^{\dim(\alpha)}\right\}\right)-\pi\left(\langle\alpha,\beta,\chi\rangle\right).\qquad(5)$$

Then one can find the number of transpositions $\sigma\left(\langle\alpha,\beta,\{P\}\rangle\right)$ as:

$$\sigma\left(\langle\alpha,\beta,\{P\}\rangle\right)=\left(\sum\nolimits_{\chi}^{\chi\in\delta(\alpha)}\pi\left(\langle\alpha,\beta,\chi\rangle\right)\right)-\sigma\left(\langle\alpha,\beta,\{M\}\rangle\right),\qquad(6)$$

with

$$\sigma\left(\langle\alpha,\beta,\{P\}\rangle\right)=\sigma\left(\langle\beta,\alpha,\{P\}\rangle\right);\ \sigma\left(\langle\alpha,\beta,\{M,P\}\rangle\right)=\sigma\left(\langle\alpha,\beta,\{M\}\rangle\right)+\sigma\left(\langle\alpha,\beta,\{P\}\rangle\right).\qquad(7)$$

The number of exchanges (replacements) $\sigma\left(\langle\alpha,\beta,\{X\}\rangle\right)$ can be determined as:

$$\sigma\left(\langle\alpha,\beta,\{X\}\rangle\right)=\min\left(\{\dim(\alpha)\}\cup\{\dim(\beta)\}\right)-\sigma\left(\langle\alpha,\beta,\{M,P\}\rangle\right)\qquad(8)$$

with $\sigma\left(\langle\alpha,\beta,\{X,M\}\rangle\right)=\sigma\left(\langle\alpha,\beta,\{X\}\rangle\right)+\sigma\left(\langle\alpha,\beta,\{M\}\rangle\right)$.

The number of generations $\sigma\left(\langle\alpha,\beta,\{G\}\rangle\right)$ can be determined as:

$$\sigma\left(\langle\alpha,\beta,\{G\}\rangle\right)=\max\left(\{0\}\cup\left\{\left|\delta(\alpha)/\delta(\beta)\right|-\sigma\left(\langle\alpha,\beta,\{X\}\rangle\right)\right\}\right).\qquad(9)$$

The number of duplications $\sigma\left(\langle\alpha,\beta,\{C\}\rangle\right)$:

$$\sigma\left(\langle\alpha,\beta,\{C\}\rangle\right)=\left(\sum\nolimits_{\chi}^{\chi\in\delta(\alpha)}\kappa\left(\langle\alpha,\beta,\chi\rangle\right)\right)-\max\left(\{0\}\cup\left\{\sigma\left(\langle\alpha,\beta,\{X\}\rangle\right)-\left|\delta(\alpha)/\delta(\beta)\right|\right\}\right),\qquad(10)$$

with $\sigma\left(\langle\alpha,\beta,\{G,C\}\rangle\right)=\sigma\left(\langle\alpha,\beta,\{G\}\rangle\right)+\sigma\left(\langle\alpha,\beta,\{C\}\rangle\right)$:

$$\sigma\left(\langle\alpha,\beta,\{G,C\}\rangle\right)=\left(\sum\nolimits_{\chi}^{\chi\in\delta(\alpha)}\kappa\left(\langle\alpha,\beta,\chi\rangle\right)\right)-\sigma\left(\langle\alpha,\beta,\{X\}\rangle\right),\qquad(11)$$

$$\sigma\left(\langle\alpha,\beta,\{G,C\}\rangle\right)=\dim(\alpha)-\sigma\left(\langle\alpha,\beta,\{M,P\}\rangle\right)-\sigma\left(\langle\alpha,\beta,\{X\}\rangle\right).\qquad(12)$$

The number of fusions $\sigma\left(\langle\alpha,\beta,\{F\}\rangle\right)$:

$$\sigma\left(\langle\alpha,\beta,\{F\}\rangle\right)=\sigma\left(\langle\beta,\alpha,\{C\}\rangle\right).\qquad(13)$$

The number of deletions (destructions) $\sigma\left(\langle\alpha,\beta,\{E\}\rangle\right)$ can be determined as:

$$\sigma\left(\langle\alpha,\beta,\{E\}\rangle\right)=\sigma\left(\langle\beta,\alpha,\{G\}\rangle\right);\qquad(14)$$

$$\sigma\left(\langle\alpha,\beta,\{E,F\}\rangle\right)=\sigma\left(\langle\beta,\alpha,\{G,C\}\rangle\right).\qquad(15)$$

Then the distance (metrics) ρ_p between strings α and β:

$$\rho_p\left(\langle\alpha,\beta\rangle\right)=\rho_p^0\left(\langle\alpha,\beta\rangle\right);\qquad(16)$$

$$\rho_p^k\left(\langle\alpha,\beta\rangle\right)=\begin{cases}1\Big|\langle\alpha,\beta\rangle\in A\times A\\ \psi_k^{\beta}\left(\rho_p^k\left(\langle\alpha,\langle\beta\rangle\rangle\right)\right)\Big|\langle\alpha,\beta\rangle\in\left(A^{(*)}/A\right)\times A\\ \psi_k^{\alpha}\left(\rho_p^k\left(\langle\langle\alpha\rangle,\beta\rangle\right)\right)\Big|\langle\alpha,\beta\rangle\in A\times\left(A^{(*)}/A\right)\\ \varphi_p^k\left(\langle\alpha,\beta\rangle\right)\Big|\langle\alpha,\beta\rangle\in\left(A^{(*)}/A\right)\times\left(A^{(*)}/A\right)\end{cases};\ \psi_k^{\chi}(\gamma)=\upsilon_k^{\chi}*\gamma; \tag{17}$$

$$\varphi_p^k\left(\langle\alpha,\beta\rangle\right)=\sqrt[p]{\sum_{i=1}^{\dim(\alpha)}\sum_{j=1}^{\dim(\beta)}\omega_{ijk}^{\varepsilon(\langle\alpha,\beta\rangle)(i)}*\left(1_{\{M,R,X,T,P\}}^{\{\varepsilon(\langle\alpha,\beta\rangle)(i)\}}*\left(\rho_p^{k+1}\left(\langle\alpha_i,\beta_j\rangle\right)\right)^p+1_{\{I,D,G,C,E,F\}}^{\{\varepsilon(\langle\alpha,\beta\rangle)(i)\}}\right)}; \tag{18}$$

$$1_{\lambda}^{\gamma}=\begin{cases}0\big|\varnothing=(\lambda\cap\gamma)\\ 1\big|\varnothing\subset(\lambda\cap\gamma)\end{cases}, \tag{19}$$

where υ_k^{χ}, $\omega_{ijk}^{\varepsilon(\langle\alpha,\beta\rangle)(i)}$ are the weight coefficients.

Estimation for the time complexity of counting the number of transpositions, exchanges, generations, duplications, reductions, and fusions with known $\varepsilon\left(\langle\alpha,\beta\rangle\right)$ is expressed as $O\left(\left(\dim(\alpha)+\dim(\beta)\right)\cdot f\left(\dim(\alpha)+\dim(\beta)\right)\right)$, where $f\left(\dim(\alpha)+\dim(\beta)\right)$ is the time of access to one string element.

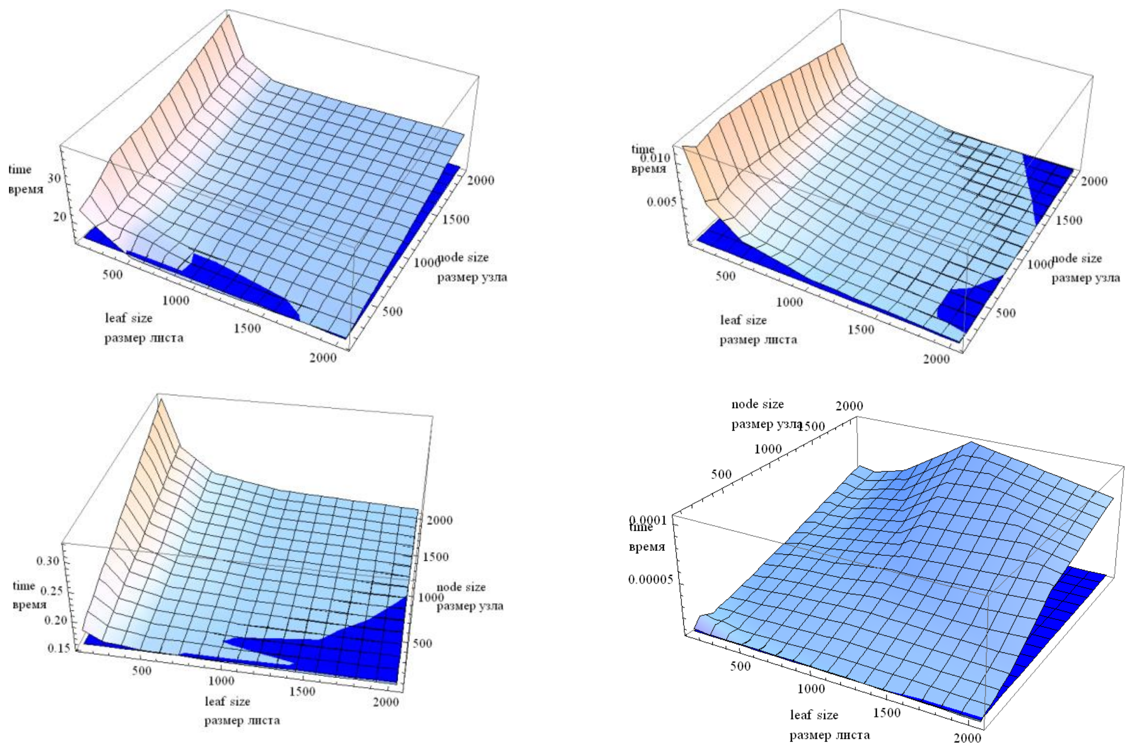

Fig. 3. Operation times (s): search by index, deletion, iteration splitting

Conclusion

Computational experiments have confirmed the theoretical estimates of the time complexity of operations of the model under consideration, which makes it possible to guarantee the time complexity of string processing operations no higher than $O\left(\ln^k(n)*\sqrt{n}\right)$. Comparison with similar operations on vector and map structures on strings up to 10^6 showed that the proposed implementation is no more than two orders of magnitude behind in sequential access and is superior to more than four orders of magnitude in the tasks of joining and splitting strings. The investigated model is proposed as a basis for creating technologies for developing intelligent systems that provide a unified representation of knowledge [12], which have compatibility, scalability, and the ability to work in real time.

References

1. Ivashenko V.P., Sintsov S.V. [Operations on management data sets in linear memory]. *Doklady BGUIR = Doklady BGUIR*. 2016;6(100):86-93. (in Russ.)
2. Bates R. Language Definition in the Schütz Semantic Editor. *JMLC 2003*. 2003;2789:229-240. DOI: 10.1007/978-3-540-45213-3_29.
3. Xie T., Taneja K., Kale Sh., Marinov D. Towards a Framework for Differential Unit Testing of Object-Oriented Programs. *Proceedings – International Conference on Software Engineering*. 2007;1:5-5. DOI:10.1109/AST.2007.15.
4. Cadar C., Koushik S. Symbolic execution for software testing: three decades later. *Communications of the ACM*. 2013;56(2):82-90.DOI: 10.1145/2408776.2408795.
5. Le Traon Y., Jéron T., Jézéquel J.-M., Morel P. Efficient Object-Oriented Integration and Regression Testing. *IEEE Transactions on Reliability*. 2000;49:12-25. DOI: 10.1109/24.855533.
6. Hirschberg D.S. A linear space algorithm for computing maximal common subsequences. *Communications of the ACM*. 1975;18 (6):341-343. DOI: 10.1145/360825.360861.
7. Gusfield D. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge: Cambridge University Press; 1997. DOI: 10.1086/420407.
8. Levenshtein V.I. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*. 1966:10(8):707-710. (in Russ.)
9. Damerau F.J. A technique for computer detection and correction of spelling errors. *Communications of the ACM*. 1964;7(3):171-176. DOI: 10.1145/363958.363994.
10. Navarro G. A guided tour to approximate string matching. *ACM Computing Surveys*. 2001;33(1):31-88. DOI: 10.1145/375360.375365.
11. Ehrenfeucht A., Haussler D.A New Distance Metric on Strings Computable in Linear Time. *Discrete Applied Mathematics*. 1988;20(3):191-203. DOI:10.1016/0166-218X(88)90076-5.
12. Ivashenko V.P. [Ontological model of space-time relations for events and phenomena in processing of knowledge] *Vestnik BrGTU*. 2017; 5(107):13-17. (in Russ.)

Information about the authors

Ivashenko V.P., PhD, Associate Professor of the Intelligent Information Technologies Department of the Belarusian State University of Informatics and Radioelectronics.

Address for correspondence

220013, Belarus,
Minsk, P. Brovki str., 6,
Belarusian State University
of Informatics and Radioelectronics
tel. +375-17-293-80-92;
e-mail: ivashenko@bsuir.by
Ivashenko Valerian Petrovich