Paweł DYMORA, Mirosław MAZUREK, Dominik STRZAŁKA

Politechnika Rzeszowska, Zakład Systemów Rozproszonych

Long-range dependencies in quick-sort algorithm

Streszczenie. *Sortowanie jest jednym z najczęstszych wykorzystywanych typów przetwarzania w systemach komputerowych. W prezentowanym podejściu sortowanie będzie rozważane jako wprowadzenie porządku w przetwarzanym zadaniu wejściowym oraz algorytm jako fizyczny system (odpowiedzialny za obliczenia). Zazwyczaj analiza zachowania dowolnego algorytmu jest realizowana w kontekście klasycznej złożoności obliczeniowej. W niniejszej pracy istnienie zależności długoterminowych w dynamice przetwarzania jest wyznaczane w oparciu o współczynnik Hurst'a. (**Zależności długoterminowe w algorytmie quick-sort**).*

Abstract*. Sorting is one of the most frequently used types of processing in computer systems. In presented approach sorting will be considered as an introduction of order into processed input task and algorithm as a physical system (responsible for computations). This analysis shows how the dependencies in processed tasks can influence the behavior of algorithm (or equivalently Turing machine). Normally, analysis of any algorithm behavior is done in terms of classical computational complexity. In this paper the rate of existence of long-term correlations in processing dynamics is calculated basing on Hurst coefficient.*

Słowa kluczowe: zależności długoterminowe, algorytm szybkiego sortowania, współczynnik Hurst'a, samopodobieństwo.
Keywords: long-range dependence, quick-sort algorithm, Hurst factor, self-similarity.

doi:10.12915/pe.2014.01.35

Introduction

It is widely believed that computing is a mathematical science. This view is somewhat justified, since the foundation of most of the considerations in computer science is the idea of a Turing machine [1]. This is a purely mathematical concept that was presented as one of the possible answers to the problem posed by David Hilbert (other answers were given by K. Gödel, A. Church and S. Kleene). During years this model was also regarded as a point of reference for many theoretical aspects in computer science [2]. This machine has nothing to do with physical device and description of some of its features and properties is quite difficult in terms of physics, especially the infinite length of tape (i.e. memory) and zero energy consumption during processing. This is a model that is used for algorithmic processing and whenever someone says 'algorithm' always has in mind the model of Turing machine. However, on the other hand, it should be also noted that any implementations of this machine (simply: computers) are physical devices, which are subject to many restrictions: tape (memory) always has finite length and also during operation machine consumes energy and produces entropy [3]. Many aspects of the practical applications of the theory developed in computer science are carried out by computer engineering relating to various types of systems: computers, computer networks, software (including operating systems), databases, graphical interfaces, etc. Most of their description is implemented in terms of simple systems, but it seems that modern computer systems have evolved towards increasing software and hardware complexity and their description needs a new perspective: complex systems.

From the physical point of view computer processing is done in computer systems (it takes a place in machine) and is a transformation of energy into useful work (implemented calculations, performed algorithms) and entropy (part of the energy that is wasted: for example, heat generated from the fan, but not only - this will be explained a bit later). This problem was pointed out by Charles Bennett, who stated that [3]: *computers may be thought of as engines for transforming free energy into waste heat and mathematical work*, but somehow his approach was not pursued. In this paper this observation is a main point of reference and as an example of considerations that can be carried the problem of sorting will be presented. In theoretical computer science it is assumed that it does not matter how the implementation of the Turing machine is done (it even may be a steam engine), but it seems that there is a need to introduce among mathematical considerations some important and necessary physical aspects. Justification for this statement can be found among many voices saying that the computer science is not necessary a mathematical science but rather a physical one [2]. The main problem is a fact that the level of usage of various statistical methods and analysis in theoretical considerations is very small; generally it is assumed that mathematical considerations are sufficient. However, some areas of computer science where this isn't enough can be shown. For example:
static and dynamic hazard in combinatorial systems (digital devices) [4], a problem of states encoding in asynchronous sequential circuits (critical and non-critical races) [4], statistical self-similarity that causes limited bandwidth of queues [5], limited scalability of distributed systems [6], communication costs that influence the efficiency of distributed and parallel systems [7], rapid decline in the performance of systems with limited resources [8].

Most of these problems have a physical background and cannot be understand in terms of mathematical (theoretical) approach.

Problem of sorting

In this paper we present a different perspective on the problem of sorting analysis. It is one of the most important forms of processing in computer systems. According to Donald Knuth words from his book *The Art of Computer Programming*, even 70% of operations performed by computer system are sorting. This problem is algorithmically closed, i.e., algorithms are as efficient as it is assumed by theoretical considerations (class O (n log n)) and this is one of the first issues raised on algorithms and programming courses. They are used to explain a question: "What is the computational complexity?". This concept was introduced around 1967 by Hartmanis and Stearns [9], who wanted to show how the solutions of some problems can be difficult and there may be different ways (more or less efficient) to solve computational tasks. Theirs proposed approach made important assumptions that the algorithm performance is measured by [10]:
1) The analysis of the number of dominant operations requests, which may include comparison, arithmetic operations, loop calls, etc. This diversity is justified by the fact that in most types of computers (but not all), each such an operation is performed during the same time.
2) The measure is calculated independently on the input set properties (instances) and the so-called worst case (the largest number of required dominant operations) is taken (in

the sorting problem usually the opposite arrangement of elements is considered). This condition implies the independence between a set of input data (task) and a Turing machine (hardware).

3) The complexity is expressed in mathematical notations: O, Ω, Θ

Donald Knuth says [11] that instead of sorting one should talk about ordering, therefore, in terms of physics this is a process of introduction of order into processed set according to the accepted relation (usually ≤). Sorting reduces the level of entropy in input set – in fact this entropy is moved outside the computer and thus the production of entropy in the computer system denotes not only the waste of energy, but also can be related to the specific types of performed processing.

Insertion-sort as a background

There are many different methods of sorting of various complexities. As an introduction to these considerations a sort algorithm based on insertion will be shown It is a very simple algorithm, but quite informative. The idea of its work reflects the behavior of bridge player who sorts cards. This algorithm is quite slow, because for the worst-case (a reverse order of the elements in the input set) its complexity is $O(n^2)$ – the number of dominant operations increases with the square of the input size of the task (n is the number of processed elements – keys – in the input set), but on the other hand in the optimistic case its complexity is $\Omega(n)$ - the number of dominant operations grows linearly. This algorithm is quite often used as a didactical example [10], but it also very interesting because it has a different complexity for extreme cases and one can ask how it works when the assumption of independence between input data and Turing machine is abandoned. Theoretical approach is trying to deal with this by analyzing the expected (average) computational complexity, and again for insertion-sort the result is shown by the notation O (again it's $O(n^2)$), but one can go step beyond and wonder whether the dynamics of this algorithm can be described using the methods associated with statistical analysis of complex systems. This can be done – see for example [12] and [13] where it has been shown that the number of dominant operations is also dependent on the properties of input data set. This showed that since insertion-sort algorithm has its own Turing machine, and it can be implemented in a physical machine, it may be also interesting to determine whether the amount of consumed energy and produced entropy has some kind of dependencies usually associated with the widely understood complex systems. In presented paper other important algorithm will be analyzed, i.e., quick sort.

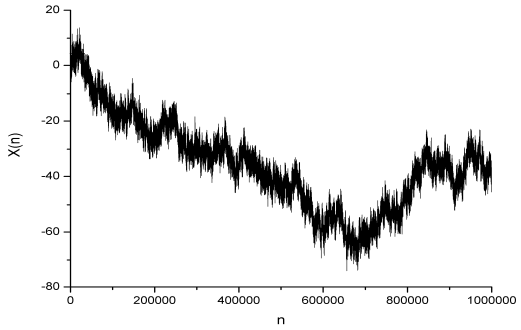
Simulation tests

For simulation tests a set of special input data was taken - trajectories of fractional Brownian motion with different values of the index H. This type of trajectories has clearly visible downward and upward trends which during sorting can be considered as local optimistic and pessimistic cases of the pre-sorting data. H index is taken in order to change the level of long-term dependencies and memory effect. The example of such sets is presented in Fig. 1 and Fig. 2.

A quick-sort algorithm is structured differently than the insertion-sort algorithm. It was proposed by Ch. Hoare and it uses the divide and conquer method. In the algorithm the whole task is divided into smaller tasks then they are solved by recurrence approach again by quick-sorting. It is a recursive algorithm - repeatedly calls itself.

Below a simple example of its work is presented (Fig. 3). The key element is called a pivot, i.e. it serves as a reference point in subsequent comparisons. There are many different implementations of this algorithm and methods of pivot selection.

a)



b)

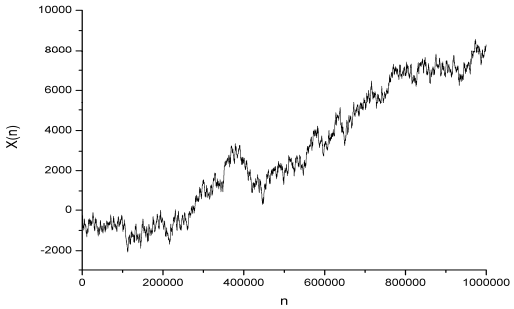


Fig. 1. Exaples of input data a) H=0.5, b) H=0.9

Regardless of how the pivot selection is done this algorithm has a mean computational complexity expressed as O ($n$log$n$), although for a well-sorted data the complexity is $O(n^2)$. Generally, a pivot value is selected in the middle (with a factor p = 0.5), but it also could be selected extremely (values of p = 0.00…1 or p = 0.00…9). In the example below, this is 7 (Fig.3).

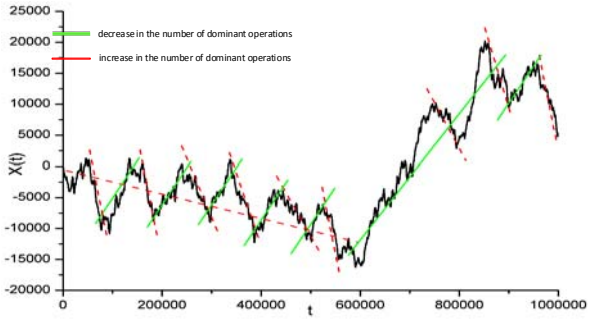


Fig. 2. Trend analyzis in the set of time series

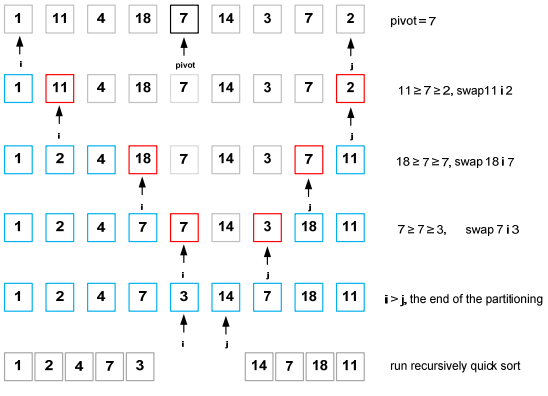
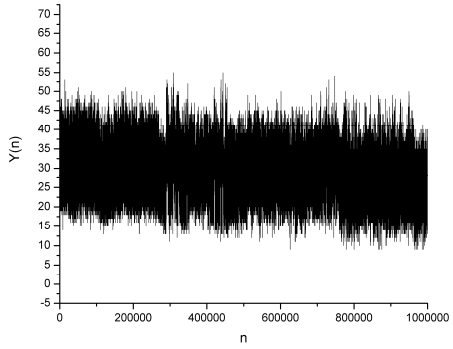


Fig. 3. Example of quick-sort processing with pivot equals 7

In subsequent steps, the elements of the input are compared to the pivot and if they are smaller than the pivot

then they are moved to the left, else when they are larger they are moved to the right (in our example this is a swap operation). After reviewing all the elements in an input set it recursively runs the same algorithm, but for the both half of input set, etc.

a)



b)

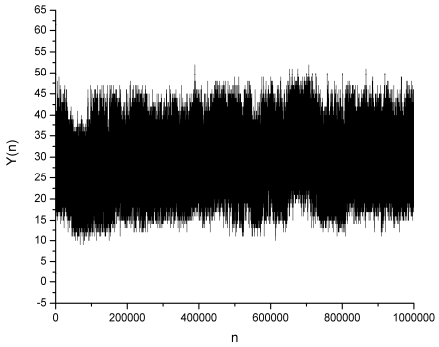
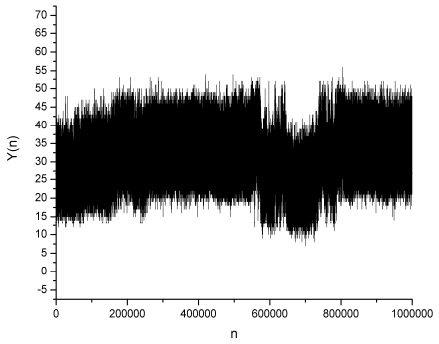


Fig. 4. Example of number of dominant operations for two input sets H = 0.1 and a) p=0.1; b) p = 0.8.

In this study an experiment is proposed in order to determine whether a quick-sort algorithm is sensitive or not on long-term dependencies in input set similarly like in the approach used for the analysis of insertion sort [13]. For each different H value (H = 0.1, H = 0.2 ... H = 0.9, H = 0.99) 100 sets of input data were generated (total number on input sets: 1000). These were the trajectories of fractional Brownian motion with $n = 10^6$ values and then each of them was sorted by quick-sort algorithm with different values of pivot (p = 0.1, 0.2 ... 0.9, 0.99) – thus the total number of quick-sort executions was 10000. For each quick-sort algorithm execution the number of dominant operations used by this algorithm for each sorted key from the input set was recorded (the total number of records of quick-sort algorithm behaviors was 10000, each such a record contains 10^6 elements). Figure 4 shows examples of such records for two input sets: one with H = 0.1 and pivot setup to p = 0.1 and p = 0.8 – top part of the figure, and second with H = 0.8 and the same pivots.

For each such a record the Hurst index H was calculated. In order to have a possibility to compare results three methods were used: DFA, R / S and the study of power spectrum. Figure 6 a,b,c shows graphs in 3D of the average (calculated on the basis of 100 time series) value of H index determined on the basis of DFA method (Fig. 6a), R / S method (Fig. 6b) and power spectrum (Fig. 6c). Because there was a possibility to change H index of input set and the value of pivot p for each quick-sort execution Figure 5a,b,c shows also values of H index (H_{out}) depending on the values of p with constant values of H in input sets (H_{in}) – left part of the figure, and also shows values of H index (H_{out}) depending on the values of H in input sets (H_{in}) with constant values of p – right part of the figure.

a)



b)

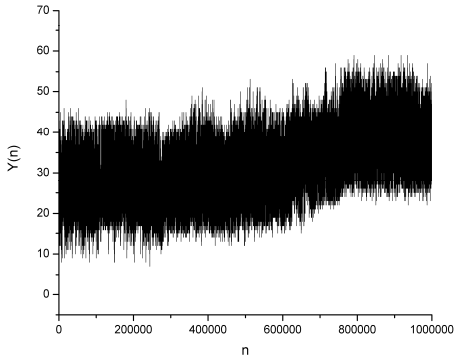
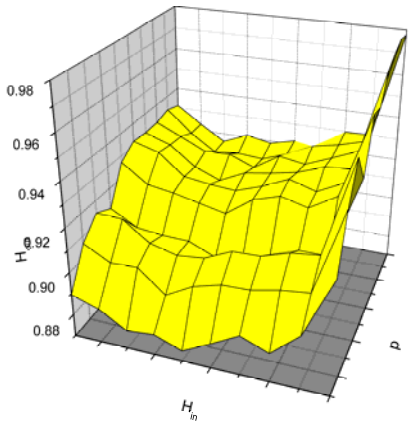
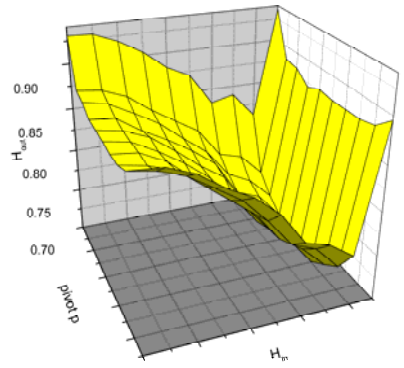


Fig. 5. Example of number of dominant operations for two input sets for H = 0.8 and a) p = 0.1; b) p=0.8.

a)



b)



c)

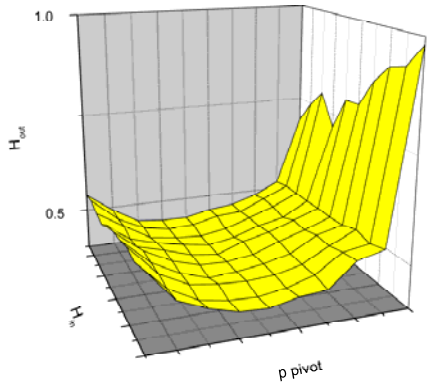


Fig. 6. Values of *H* index in output records of dominant operations for each sorted key (H_{out}) depending on the values of *H* in input set (H_{in}) and chosen pivots *p* (top part - DFA method, middle part - R/S method, bottom part - power spectrum method).

Conclusions

The results of presented experiment shows that the performance of high-speed sorting may be associated with the presence of long-term processes in computer system. The first two methods used for *H* calculations, i.e., DFA and R / S, clearly indicated that, regardless of the nature of the input data (with or without long-term dependencies) and the method for selecting pivots *p* (except in the case of *p* = 0.99), the values of *H* index for the number of necessary dominant operations executed by quick-sort are extremely high. In this algorithm such an operation is a comparison of two elements (sorted keys from input set) and this process is characterized by a relatively high value of the index *H*. In the case of the study of power spectrum, the obtained values of *H* are at a slightly lower levels, but one can still see the potential effect of existence long memory. The results obtained above require further analysis.

REFERENCES

[1] Turing, A. M.: On computable numbers, with an application to the Entscheidungsproblem, Proc. of the London Mathematical Society, Series 2, 42 (1936), pp. 230–265. Errata appeared in Series 2, 43, pp. 544–546, 1937.

[2] Eberbach E., Wegner P. Beyond Turing machines, Bulletin of the European Association for Theoretical Computer Science 81, pp. 279–304, 2003.

[3] Bennett, Ch. H.: The Thermodynamics of Computation – a Review, Int. J. of Theor. Phys., vol. 21, No. 12, pp. 905–939, 1982.

[4] Ch. Roth, Fundamental of logic design, Cengage Learning, (2009)

[5] B. Strzałka, M. Mazurek, D. Strzałka: Queue Performance in Presence of Long-Range Dependencies – an Empirical Study, International Journal of Information Science, 2(4), pp. 47-53, (2012)

[6] Grabowski F., Strzałka D. Computer engineering by non-extensive statistics approach. in: New technologies in computer networks. WKiŁ, Warsaw, 2006

[7] Grabowski F., Strzałka D.: Limitations of asynchronous systems scalability, , Measurement, Automatics, Control, 53, pp. 6-9 (2007)

[8] Grabowski F. Logistic equation of arbitrary order, Physica A: Statistical Mechanics and its Applications, Vol. 389, Issue 16, pp. 3081–3093, 2010

[9] Hartmanis, J. and Stearns, R.E., On the Computational Complexity of Algorithms,. Trans. Am. Math. Soc., vol. 117 (5), 285-306, 1965

[10] T. H. Cormen, Ch. E. Leiserson, R.L. Rivest, and C. Stein, Introduction to Algorithms , MIT. Press, mcgraw- Hill Book Company, 1990

[11] D. Knuth, The art of computer programming, Addison-Wesley, 1968

[12] D. Strzałka, F. Grabowski: Towards possible non-extensive thermodynamics of algorithmic processing - statistical mechanics of insertion sort algorithm, International Journal of Modern Physics C, vol. 19 n. 9, pp. 1443 – 1458, (2008)

[13] Strzałka, D. Processes in the computer system at the interface between data and simple insertion sort algorithm in terms of nonextensive statistics, PHD. Thesis, Gliwice, 2009.

***Authors**: dr inż. Paweł Dymora, Politechnika Rzeszowska, Zakład Systemów Rozproszonych, ul. W. Pola 2, 35-959 Rzeszów, E-mail: pawel.dymora@prz.edu.pl; dr inż. Mirosław Mazurek, Politechnika Rzeszowska, Zakład Systemów Rozproszonych, ul. W. Pola 2, 35-959 Rzeszów, E-mail: miroslaw.mazurek@prz.edu.pl; dr inż. Dominik Strzałka, Politechnika Rzeszowska, Zakład Systemów Rozproszonych, ul. W. Pola 2, 35-959 Rzeszów, E-mail: strzalka@prz.edu.pl .*