ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2013
Seria: MATEMATYKA STOSOWANA z. 3 Nr kol. 1899

Marcin WOŹNIAK, Zbigniew MARSZAŁEK
Institute of Mathematics
Silesian University of Technology

ON SOME PROPERTIES OF BUBBLE SORT WITH LOGIC CONTROL OF ORDER FOR LARGE SCALE DATA SETS

Summary. Sorting is a one of very important topics for computer science. In the modern computing, computers must operate on bigger and bigger amounts of data. Therefore we try to analyze modified version of bubble sort algorithm and its properties for large data sets. Article aims to show and analyze the possible behavior of bubble sort with logic control of order for large scale data sets.

O WŁASNOŚCIACH ALGORYTMU SORTOWANIA BĄBELKOWEGO Z FUNKCJĄ KONTROLI UŁOŻENIA ELEMENTÓW DLA DUŻYCH ZBIORÓW DANYCH

Streszczenie. Sortowanie jest jednym z bardzo ważnych tematów współczesnej informatyki. We współczesnej informatyce komputery muszą pracować na coraz większej liczbie danych, dlatego staramy się analizować jeden z podstawowych algorytmów sortowania i jego właściwości dla dużych zbiorów danych. Artykuł ma na celu przeanalizowanie możliwego zachowania badanej wersji algorytmu z funkcją logicznej kontroli ułożenia dla dużych zbiorów danych.

2010 Mathematics Subject Classification: 68W40, 68Q25, 68P01, 68P10.
Corresponding author: M. Woźniak (Marcin.Wozniak@polsl.pl).
Received: 14.05.2013 r.

1. Introduction

In the literature [1,2,10,11,16] are presented some interesting versions of classic bubble sort algorithm. However classic interpretation of bubble sort algorithm does not have the possibility of intelligent control over the arrangement. Control of the content accelerates the process of sorting. To control the order was introduced a special boolean variable. Having information about ordering, the process will be shorter. The analysis presented in Section 3. will answer how it contributes to the acceleration of the speed of the algorithm and how it determines it's impact on the complexity, which is important for large data sets. Let us now present examined version of algorithm with logic control of order.

2. Bubble sort with logic control of order

Modifications of this method and the references to its interesting derivatives are also described in [1,2,10,11,16]. However, the authors of [5,9,12,14,18] show the possibility of constructing adaptive algorithms to improve sorting and data mining. At the same time in [3–7, 12] are presented interesting solutions for special data structures. Bubble sort algorithm is converting elements in the following pairs. The procedure swaps compared elements. The biggest one is moved to the end of sequence in following iterations. Therefore, in subsequent iterations, each time we consider decreased sequence. A special bubble sort with logic control of order is also known from [1, 2, 10, 11, 16]. Many interesting implementations of sorting methods and their derivatives for parallel systems are presented by the authors of [5,8,12–15,18]. Let us now present the examined bubble sort algorithm wit logic control.

Our implementation of examined version was programmed in CLR standard for MS Visual Studio. In Figure 2 is presented a block diagram of examined algorithm.

2.1. Time complexity

Let us now talk about theoretical time complexity of bubble sort with logic control of order. If we sort worst-case, which is the reverse order of arrangement of the elements, at each step we swap all possible elements. In this case theoretical time complexity is equal to classic version described in [17]. We can also talk about the complexity of the best case, in which we only compare $n - 1$ times. Time

complexity is then $\vartheta(n)$. However most interesting is average time complexity for all input sequences. Let us denote the number of comparisons by $t = n - k$, where k ranges from 1 to $n - 1$. If the sorting stop after checking all the elements then $k = n - 1$. It is important to remember that there is exactly one such sequence.

```
   Start
   Load data
   Set starting index i
1  Set logic variable t1 = true
2  if loaded sequence is sorted then
   |  Return to 6
   else
   |  Set lgic variable t1 = false
   |  Return to 3
   end if
3  if check if element of index i is last one then
   |  Return to 6
   else
   |  Return to 4
   end if
4  Start sorting
   Take next element with index j to compare
   if element with index j is not the last one then
   |  if next element is bigger then
   |  |  Swap elements
   |  |  Set logic variable t1 = true
   |  else
   |  |  Return to 5
   |  end if
5  |  Increase index j++
   |  Return to 2
   else
   |  Increase index i++
   |  Return to 1
   end if
6  Write sorted sequence
   Stop
```

Fig. 1. Bubble sort algorithm with logic control of order

Rys. 1. Algrytm sortowania bąbelkowego z funkcją kontroli ułożenia elementów

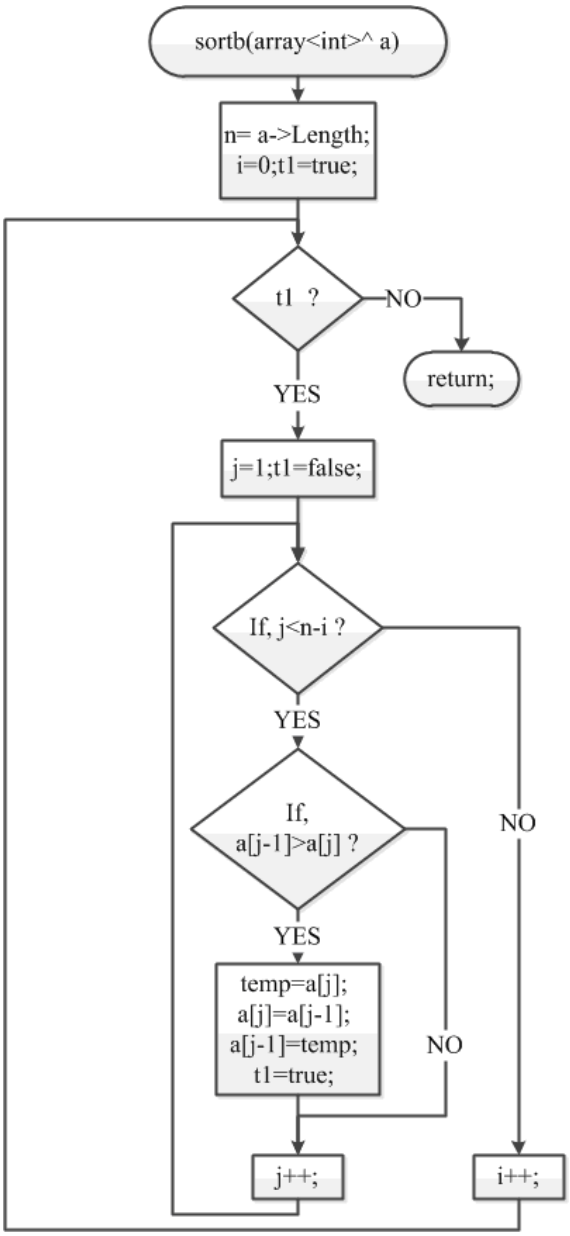

Fig. 2. The block diagram of implemented bubble sort algorithm with logic control of order

Rys. 2. Schemat blokowy zaimplementowango algrytmu sortowania bąbelkowego z funkcją kontroli ułożenia elementów

Therefore similarities of elements permutations can be approximated by a special linear function. In Figure 3 there is shown theoretical approximation of a linear function.

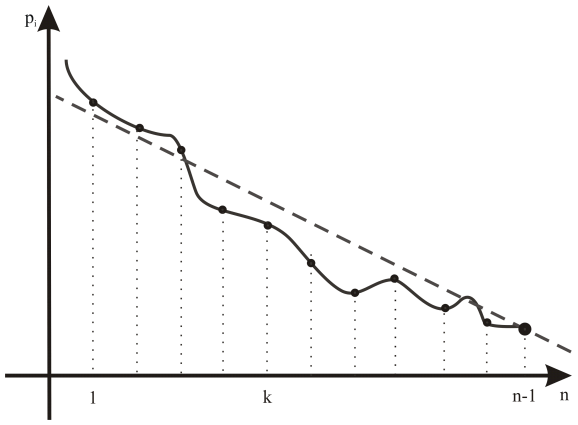

Fig. 3. Possible approximations of probability permutations of the elements in the control function

Rys. 3. Poglądowy rysunek przybliżenia wartości prawdopodobieństwa permutacji elementów w trakcie działania funkcji kontrolnej

Let us explain, origin for the approximation in Figure 3.

Theorem 1. *Average bubble sort time with logic control of order is*

$$\vartheta_{avg}(n) = \sum_{i=1}^{n-1} p_n(k) \cdot id_n(k), \tag{1}$$

where n means the number of sorted elements, $id_n(k)$ – number of actions we do to sort a sequence, $p_n(k)$ – probability to stop sorting in k-th iteration.

Proof. Formula for approximation by a linear function in general form is

$$y = ax + b. \tag{2}$$

In examined example important are assumptions (3) and (4)

$$f(n-1) = 1, \tag{3}$$

$$\sum_{n-1}^{i-1} f(i) = n!. \tag{4}$$

Substituting (3) into (2) we have formula (5) for coefficient b

$$b = 1 - a \cdot (n-1). \tag{5}$$

Let us now set formula for coefficient a. The total amount of all the events breaking bubble sort algorithm with logic control of order is equal to the number of permutations in n sequence according to the formula

$$\sum_{n-1}^{i-1}(a \cdot i + b) = n!. \tag{6}$$

Substituting value of b from formula (5) into (6), we have

$$\sum_{n-1}^{i-1}(a \cdot i + 1 - a \cdot (n-1)) = n!. \tag{7}$$

Then performing further mathematical operations described by formulas (8)–(10), we can determine the final form of a factor

$$\sum_{n-1}^{i-1}(a \cdot i + 1 - a \cdot n + a)) = \sum_{n-1}^{i-1}(-a \cdot (n-i-1) + 1). \tag{8}$$

Thus, there is the following equality

$$-a \cdot \sum_{n-1}^{i-1}(n-i-1) + \sum_{n-1}^{i-1} 1 = n!. \tag{9}$$

Setting the sum in formula (9), we obtain the general form

$$-a \cdot \frac{(n-2) \cdot (n-1)}{2} = n! - n + 1. \tag{10}$$

Thus, we can finally write a formula for the coefficient of linear approximation

$$a = \frac{2 \cdot (n! - n + 1)}{(n-2) \cdot (n-1)}. \tag{11}$$

At the same time finally putting the formula (11) into equation (5), we obtain the form factor b designated as an approximation

$$b = 1 + \frac{2 \cdot (n! - n + 1)}{(n-2) \cdot (n-1)}. \tag{12}$$

After some mathematical operations the final form of coefficient b is

$$b = \frac{2 \cdot n! - n}{n-2}. \tag{13}$$

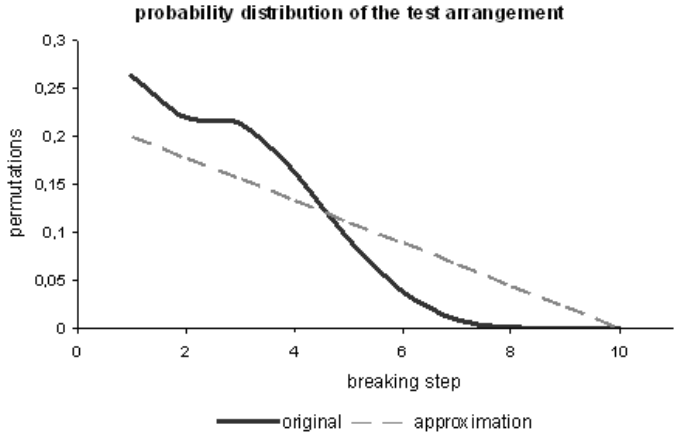

Fig. 4. Approximation of probability of permutations in the control function

Rys. 4. Przybliżenie wartości prawdopodobieństwa permutacji elementów w trakcie działania funkcji kontrolnej

Approximation of stop probability for $n = 11$ elements sequence shows Figure 4.

Thus, probability of stop in k-th step of n sorted elements can be described

$$p_n(k) = \frac{a \cdot k + b}{n!}. \tag{14}$$

Moreover the number of actions is

$$id_n(k) = \frac{(n - 1 + k) \cdot (n - k)}{2}. \tag{15}$$

Thus, the average sort time and thus theoretical time complexity of the bubble sort with logic control of order describes formula

$$\vartheta_{avg}(n) = \sum_{k=1}^{n-1} p_n(k) \cdot id_n(k), \tag{16}$$

this ends our discussion. □

Theoretical assumptions were examined and the results are presented in Section 3.

3. Experimantal results

Bubble sort with logic control of order was examined for the level of 1000000 elements. Above this number of elements, discussed method can take a considerable

amount of time, even if we apply the control function. Research results were plotted in Figures 5–6.

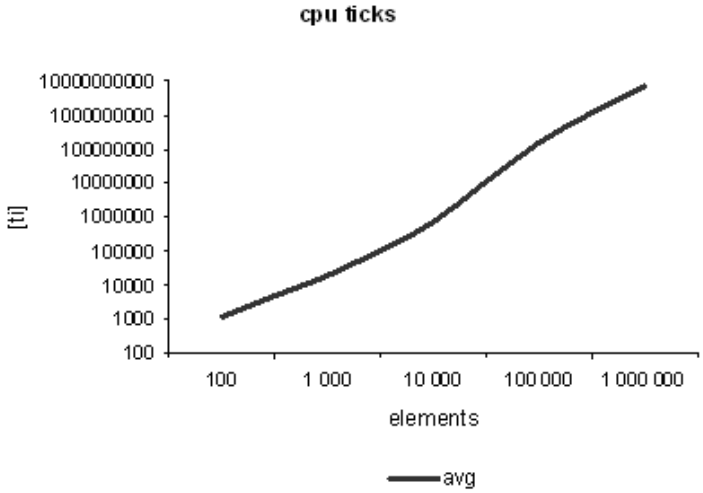

Fig. 5. The average CPU clock cycles of bubble sort with logic control of order
Rys. 5. Średnia liczba cykli zegarowych procesora w trakcie sortowania bąbelkowego z funkcji kontroli ułożenia elementów

The average number of CPU clock cycles has shown that it increases slower than for the classic version shown in [17].

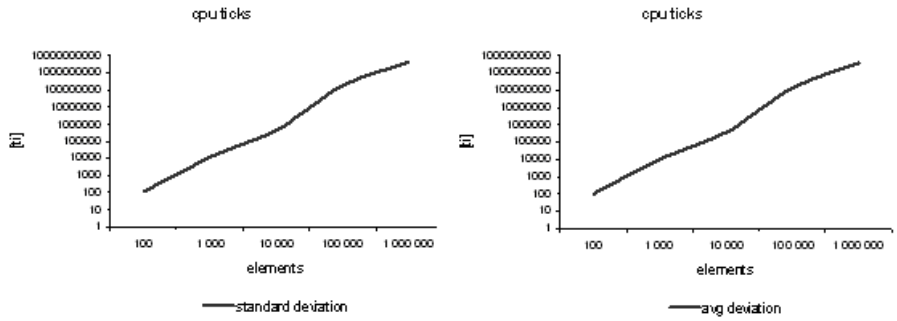

Fig. 6. Charts of the standard deviation and the average deviation of results
Rys. 6. Wykres odchylenia standardowego oraz odchylenia średniego wyników

Average number of CPU clock cycles is shown in Figure 5 is related to the standard deviation and average deviation shown in Figure 6. The size of these

characteristics shows that the control function allows speeding up the algorithm for large data sets. Let us see other time statistics. Research results were plotted in Figures 7–8.

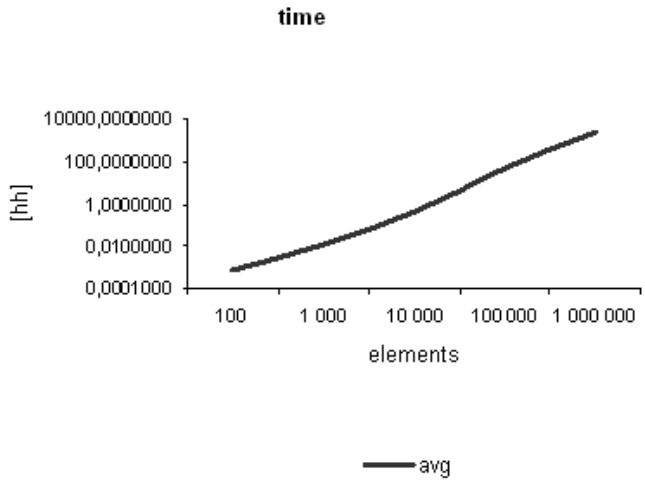

Fig. 7. The average time of bubble sort with logic control of order
Rys. 7. Średni czas sortowania bąbelkowego z funkcją kontroli ułożenia elementów

Figure 7 shows that time increases practically linear with the number of sorted elements. This phenomenon is similar to the classical form.

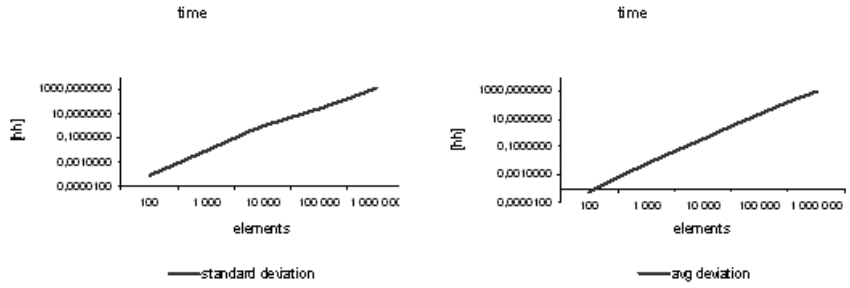

Fig. 8. Charts of the standard deviation and the average deviation of results
Rys. 8. Wykres odchylenia standardowego oraz odchylenia średniego wyników badań

The standard deviation and average deviation shown in Figure 8 indicate that the bubble sort algorithm with control of order for large sequences of data charac-

terizes lower volatility than the classic one. Coefficients of variation of the bubble sort with logic control of order are shown in Figure 9. Analysis of the coefficient of variation shows that the proposed modification is able to increase efficiency of sorting. Characteristic curves, approximated by polynomials are shown in Figure 9.

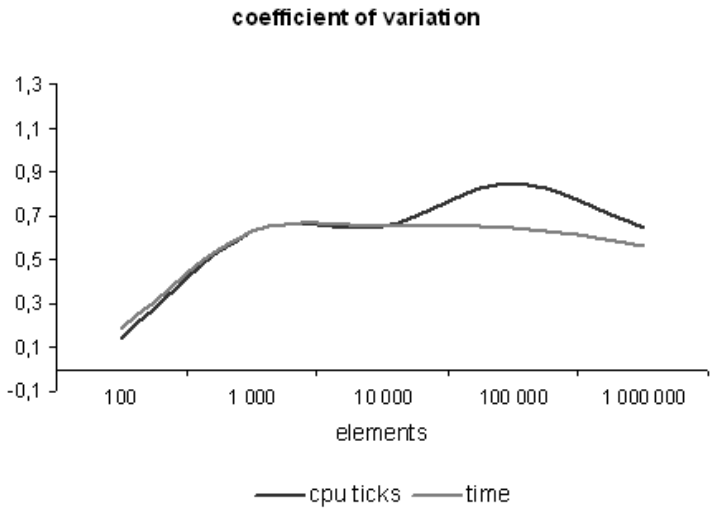

Fig. 9. Polynomial approximation of coefficient of variation for bubble sort with logic control of order

Rys. 9. Aproksymacja wielomianowa współczynnika zmienności dla sortowania bąbelkowego z funkcją kontroli ułożenia elementów

In Figure 9 we can see that high volatility characterize the CPU clock cycles of the algorithm, in particular for the collection of sequences over 100000 elements. Other characteristic of time variation is stable. This means that the nature of the operations performed in the algorithm of bubble sort with logic control of order does not affect the duration of the task. The results indicate the possibility of variation of the effective application of the algorithm in the range under 10000 elements. This is a larger number than for insertion sort algorithm described in [17] and classic bubble sort. The results, suggest that the modification increases the efficiency for large collections.

4. Conclusions

In conclusion, we can asses that the presented version of the algorithm with logic control of order is more efficient for large data sets and allows acceleration of the sorting. Therefore, it seems appropriate to use it for large data sets. Interesting improvement for examined sorting algorithm can be multiprocessing or applying cloud computing.

References

1. Aho I.A., Hopcroft J., Ullman J.: *The design and analysis of computer algorithms*. Addison-Wesley, Indianapolis 1974.
2. Banachowski L., Diks K., Rytter W.: *Algorytmy i struktury danych*. WNT, Warszawa 1996.
3. Bentley J.L., Stanat D.F., Steele J.M.: *Analysis of a randomized data structure for representing ordered sets*. In: Proceedings of the 19th Annual Allerton Conference on Communication, Control and Computing, University of Illinois, 364–372.
4. Brown M.R., Tarjan R.E.: *Design and analysis of data structures for representing sorted lists*. SIAM J. Comput. **9** (1980), 594–614.
5. Carlsson S., Chen J.: *An optimal parallel adaptive sorting algorithm*. Inform. Process. Lett. **39** (1991), 195–200.
6. Cook C.R., Kim. D.J.: *Best sorting algorithms for nearly sorted lists*. Comm. ACM **23** (1980), 620–624.
7. Dinsmore R.J.: *Longer strings for sorting*. Comm. ACM **8** (1965), 48–65.
8. Dlekmann R., Gehring J., Luling R., Monien B., Nubel M., Wanka R.: *Sorting large data sets on a massively parallel system*. In: Proceedings of the 6th IEEE Symposium on Parallel and Distributed Processing, Dallas 1994, 29–38.
9. Estivill-Castro E., Wood D.: *A genetic adaptive sorting algorithms*. Comput. J **35** (1992), 505–512.
10. Knuth D.: *The Art of Computer Programming*, Vol. 1–3. Addison-Wesley Professional, Indianapolis 2006.
11. Knuth D.E., Greene D.H.: *Mathematics for the Analysis of Algorithms*, Birkhuser, Boston 2007.

12. Levcopolos C., Petersson O.: *A note on adaptive parallel sorting.* Inform. Process. Lett. **33** (1985), 187–191.
13. Levcopolos C., Petersson O.: *An optimal parallel algorithm for sorting presorted files.* In: Proceedings of 8th Conference on Foundations of Software Technology and Theoretical Computer Science, Lecture Notes in Comput. Sci. **338** (1988), 154–160.
14. Levcopolos C., Petersson O.: *Splitsort an adaptive sorting algorithm.* Inform. Process. Lett. **39** (1991), 205–211.
15. Sinha R., Zobe J.: *Cache-conscious sorting of large sets of strings with dynamic tries.* J. Exp. Algorithmics **9** (2004), article no. 1.5.
16. Weiss M.A.: *Data Structures and Algorithm Analysis in C++.* Prentice Hall, Indianapolis 2013.
17. Woźniak M., Marszałek Z., Gabryel M.: *The analysis of properties of insertion sort algorithm for large data sets.* Zeszyty Nauk. Pol. Śl. Mat. Stosow. **2** (2012), 45–55.
18. Zheng S.Q., Calidas B., Zhang Y.: *An efficient general in-place parallel sorting scheme.* J. Supercomput. **14** (1999), 5–17.

Omówienie

W artykule przedstawiono analizę algorytmu sortowania bąbelkowego z funkcją kontrolną ułożenia elementów. Przeprowadzone badania wykazały słuszność wprowadzenia opisanej modyfikacji zwłaszcza dla dużych zbiorów danych. Jak wykazały testy i porównania, funkcja kontrolna może pozytywnie wpływać na przyspieszenie procesu sortowania dużych zbiorów danych.