IRSTI 50.13.13

Zh. Mamatnabiyev[1], M. Zhaparov[2]
[1,2]Suleyman Demirel University, Kaskelen, Kazakhstan

COMPARATIVE ANALYSIS OF SORTING ALGORITHMS USED IN COMPETITIVE PROGRAMMING

Abstract. Sorting is one of the most used and fundamental operation in computer science since the first time it had been tried to arrange list of items in some sequence. A large number of sorting algorithms were designed in order to have best performance in terms of computational complexity, memory usage, stability, and methods. In addition, in algorithmic problem solving, not all algorithms have same efficiency. This paper makes comparison research and discusses four different types of sorting algorithms: bubble sort, quick sort, insertion sort, and merge sort. The algorithms are tested and compared for data usage and time spent for sorting given amount of data.

Keywords: sorting algorithms, competitive programming, bubble sort, insertion sort, quick sort, merge sort.

***

Аңдатпа. Сұрыптау – белгілі бір тізбектегі элементтердің тізімін реттеуге арналған компьютерлік ғылымда ең жиі қолданылатын операциялардың бірі. Көптеген сұрыптау алгоритмдері есептеу күрделілігі тұрғысынан ең жақсы өнімділік, жад, тұрақтылық және басқа да әдістермен пайдалану үшін құрастырылған. Сонымен қатар, алгоритмдік есептерді шешу кезінде барлық Алгоритмдер бірдей тиімділікке ие емес. Бұл мақалада салыстырмалы зерттеулер жүргізіледі және сұрыптау алгоритмдерінің төрт түрлі түрлері талқыланады: көпіршікті сұрыптау, жылдам сұрыптау, кірістіру арқылы сұрыптау және біріктіру арқылы сұрыптау. Алгоритмдер белгілі бір деректер көлемін сұрыптауға жұмсалатын уақыты мен қолданылу жағынан тестіленеді және салыстырылады.

Түйін сөздер: сұрыптау алгоритмдері, бәсекелік программалау, көпіршікті сұрыптау, енгізу арқылы сұрыптау, жылдам сұрыптау, біріктіру арқылы сұрыптау.

***

Аннотация. Сортировка является одной из наиболее используемых и фундаментальных операций в компьютерной науке с тех пор, как впервые была предпринята попытка упорядочить список элементов в некоторой последовательности. Большое количество алгоритмов сортировки было разработано, чтобы иметь лучшую производительность с точки зрения

вычислительной сложности, использования памяти, стабильности и методов. Кроме того, при решении алгоритмических задач не все алгоритмы имеют одинаковую эффективность. В этой статье проводится сравнительное исследование и рассматриваются четыре различных типа алгоритмов сортировки: пузырьковая сортировка, быстрая сортировка, вставка и сортировка слиянием. Алгоритмы тестируются и сравниваются на предмет использования данных и времени, затрачиваемого на сортировку заданного объема данных.

Ключевые слова: алгоритмы сортировки, конкурентное программирование, пузырьковая сортировка, быстрая сортировка, сортировка вставок, сортировка слиянием.

Introduction

Sorting is a process of placing any kind of items in a list in certain order or sequence, i.e. either ascending or descending order. Since first time people tried to write some algorithms in order to sort unsorted order of elements, there were built many algorithms. Through the years, problems needed less time to sort arranged data. For example, bubble sort analyzed in early 1956s [1]. It requires linearithmic time - $O(n^2)$ – in worst case, for better performance it is possible in O(n) time. Quick sort is found more advanced for in-place data, which is initiated by C.A.R. Hoarein 1961 and it is considered as the best algorithm for decades [2]. After years another types of sorting algorithms were built, i.e. quick sort, merge sort, binary search tree that require less time in asymptotic Big-O notation. By 21st century asymptotically optimal algorithms also have been invented, with now widely used Timsort dating to 2002, and the Library sort being first published in 2006 [1]. Therefore, sorting algorithms have always attracted a great deal of research.

There are two types of sorting techniques such as comparison-based and non-comparison sorting technique. A comparison-based sort is a type of sorting that reads list of elements and then decides which element must be placed first and which one is next to make order items of the list at the final state. Bubble, selection, insertion, etc. can be example for this type of sorting. Non-comparison sorting technique sorts elements by their type and composition without compering. Bucket sort and count sort can be example for this category of sorting algorithm [2]. In this work, comparison-based sorting types are included.

The most important thing in sorting algorithms is their time complexity and memory usage while using them. During competitive programming, most problems require definite time and memory for running and executing the solution written for the given problem. So, during comparison, most frequently used sorting algorithms will be compared for by these two features.

Competitive programming is based on developing programming skills by solving problems and tasks. Most competitive problems require their participants to some tasks by its features such as data usage, memory and most often time for

running their solution to problem. For this purpose, most people prefer to use comparison-based technique of sorting algorithm. Actually, this type of sorting is simpler than another one and it recommends different kinds of sorting algorithms that require less code by strength. As most often used sorting algorithms in competitive programming, four types of sorting algorithms will be discussed and analyzed.

Working procedure

A. Bubble sort

Basic idea of Bubble sort to compare two values and interchange them if they are not in proper order is each element is compared with its adjacent element to check which element is bigger and which element is smaller [3].

Analysis: Bubble is generally considered as the simplest sorting algorithm, and for this purpose it is one of the most often taught sorting algorithm while learning sort types in introductory courses [1]. Programming code strength is less and can be completed easily. Time complexity of bubble sort is as follows: $O(n^2)$ for worst and average case whereas O(n) for already sorted array. Effective for sorting small amount of data. Let's consider bubble sort on example (Figure 1).

B. Insertion sort

As its name says, in insertion sort works by inserting each item in its proper place in the final list. The algorithm works by comparing first two elements and swap it for required order, then increase it to by one element and repeat first step until final sequence [3].

Analysis: Like bubble sort it is efficient for on small lists. In worst case its time complexity $O(n^2)$. If data is already sorted, then its best case is O(n) [2]. Programming code can be written in a few lines of code and easy to understand as shown in the Figure 2.

C. Quick sort

Quick sort woks by partitioning, the pivot element is chosen and if given element is bigger or equal to pivot, it is placed to right side, otherwise it is put to left part of the pivot element. This operation is recursively called until all given elements are sorted [5].

Quick sort is the best example of sorting that is based on divide and conquers strategy. It is one of the fastest sorting algorithms. That is why is used in many libraries of programming languages [4].

Analysis: Running time of quick sort mostly depends on the selection of pivot element. If pivot is selected randomly, then its runtime is O (n log n). For the worst case its running time is O (n^2) which happens rarely. It is not stable algorithm but it is an in-place [4]. An example of work process is shown in the Figure 3.

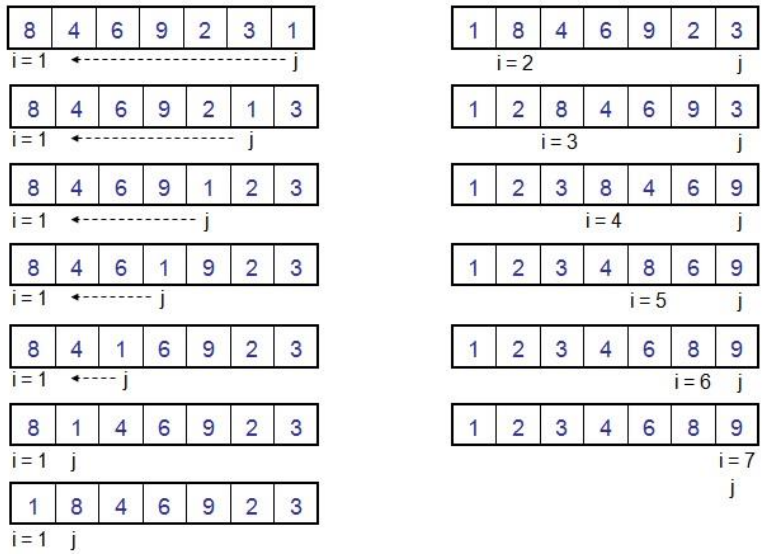

Figure 1. Bubble Sort example for sorting 7 elements in an array

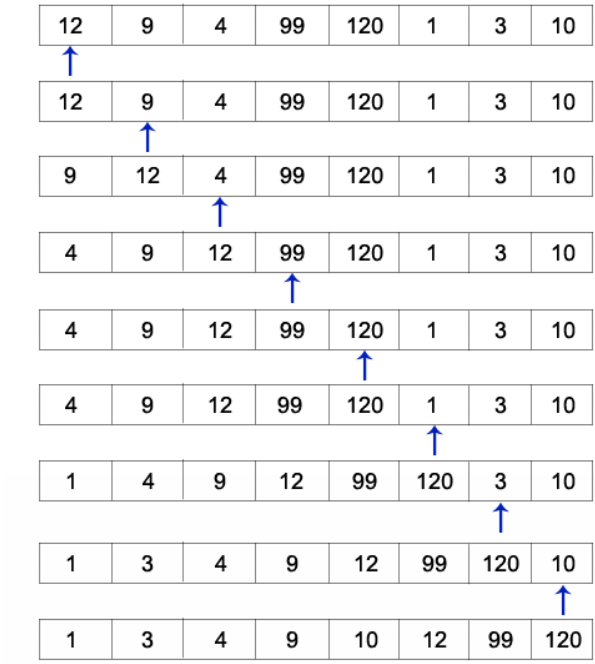

Figure 2. Insertion sort example for sorting 8 elements in an array

D. Merge sort

Main idea of the merge sort is divide and conquer strategy technique and is one of the most popular problem solving algorithm [4]. Extra array is needed when dividing array into 2 parts until it gets to 2 elements, then swap them if they are not in corresponding order. After comparing them join this array to second sorted array. This process continues until final sorted list.

Analysis: For all cases (best, average and worst) for running time is O (n log n). Unlike quick sort, it is not in-place algorithm, but it is stable [3]. The strength of programming code takes much more lines than other three sorting algorithms given above. That is why it is harder to complete algorithm. However, it is easy to understand and complete without programming code (Figure 4).

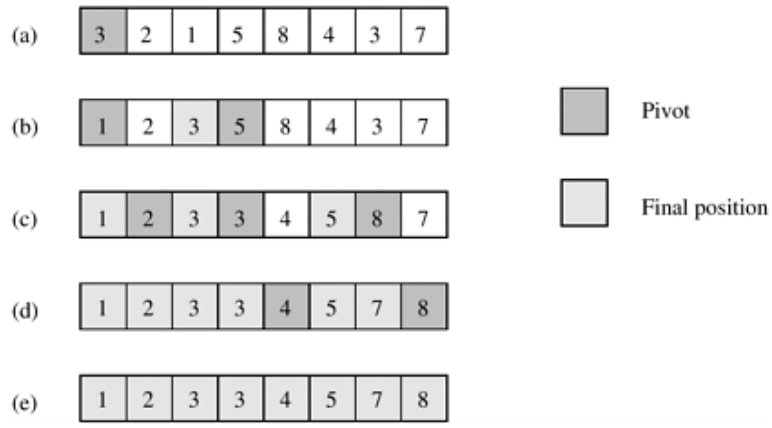

Figure 3. Quick sort example for sorting 8 elements in an array

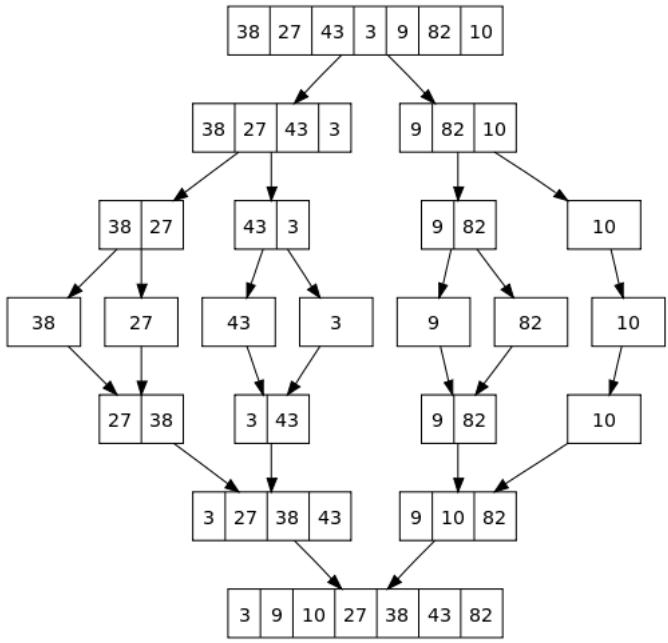

Figure 4. Merge sort example for sorting 7 elements in an array

Comparative analysis

In previous sections, learned about most often used sorting algorithms, their working process, and algorithm and for what purposes they can be used.

According to that basic knowledge about those algorithms, they can be summarized into one (Table 1).

Algorithms were tested on Operating System Windows 8.1 64-bit, Intel (R) Pentium (R) CPU 2020M @ 2.40 GHz for 100, 1000, 10000 and 30000 elements of unsorted array. For 100 items of an array they all sorted in 0 seconds, but for larger data got different results (Figure 5).

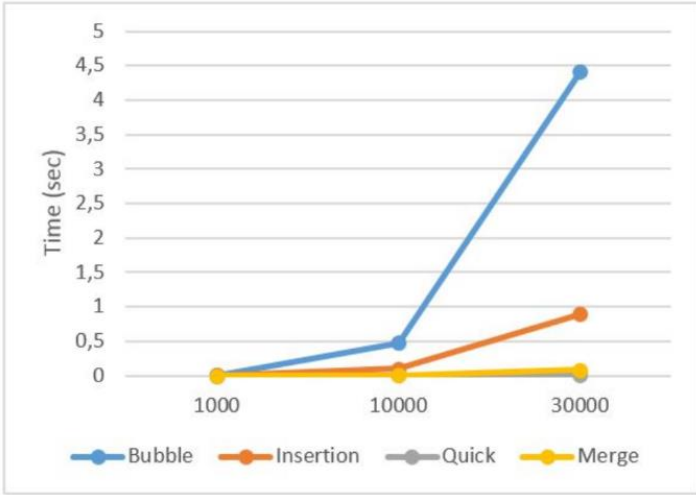

Figure 5. Time complexity tested for various amount of data

Table 1. Comparative study

	Time Complexity			**Sorting Type**	**In-place**	**Algorithm Type**	**Stability**
Sorting Algorithms	**Best case**	**Average case**	**Worst case**				
Bubble	O(N)	$O(N^2)$	$O(N^2)$	Internal	Yes	Incremental and Exchange	Yes
Insertion	O(N)	$O(N^2)$	$O(N^2)$	Internal	Yes	Incremental	Yes

Quick	O(Nlog N)	O(Nlog N)	$O(N^2)$	Internal	Yes	Divide and Conquer	Typical In Place is not stable
Merge	O(Nlog N)	O(Nlog N)	O(Nlog N)	Internal and External	No	Divide and Conquer	Yes

Conclusion

In this paper, four different types of sorting algorithms are discussed and analyzed. Bubble and Insertion sorts are simple and easy to implement. Insertion sort is faster than bubble sort. But both of them are inefficient for large data list. Quick sort is very fast and it requires very less additional space. Merge sort is also one of the fastest sort types, but it requires twice the space in memory as compared with quick sort algorithm. For huge amount of data, merge and quick sorts are preferable. If amount of data is small and definite size is already sorted, quick sort is not good choice.

References

1 Kocher, G., Agrawal, N. Analysis and Review of Sorting Algorithms. *International Journal of Scientific Engineering and Research*, Volume 2, Issue 3 (March 2014) pp. 81-84.

2 Verma, A. K., Kumar, P. List Sort: A New Approach for Sorting List to Reduce Execution Time. *Cornell University Library* (26 October 2013).

3 Pathak, A., Vajpayee, A., Agrawal D. A Comparative Study of Sorting Algorithm Based on Their Time Complexity. *International Journal of Engineering Sciences & Research Technology*, no. 3 (12) (December 2014), pp. 629-632.

4 Ali, K. A Comparative Study of Well Known Sorting Algorithms. *International Journal of Advanced Research in Computer Science*, no. 8 (1 Jan-Feb 2017): pp. 277-280.

5 Al-Jaloud, E. S., Al-Aqel, H. A., Badr G. H. Comparative Performance Evaluation of Heap-Sort and Quick-Sort Algorithms. *International Journal of Computing Academic Research,* Volume 3, no. 2 (April 2014), pp. 39-57.