Sorting and Classification of Sorting Algorithms

Rouf Ali
Department of Computer Application
Govt. Degree College Boys Sopore
Jammu & Kashmir, India
roufaliparrey@gmail.com

Abstract
Sorting and searching are fundamental problems in computer science that have attracted the attention of the researchers from the past. Sorting algorithms never remained the same but have evolved from the past and new approaches have been developed to solve this problem. New terminologies have been introduced by algorithmic designers to describe these algorithms. In this paper, I have studied the basis for classification of algorithms into different classes. This paper will be helpful to both researchers and learners in solving and understanding this problem.
KEYWORDS:sorting, searching, stable, unstable, adaptive,recursive,non-adaptive

Introduction

Arranging data items/elements/values/keys ($n$number of items)of an array/list in an ordered sequence is called "sorting". Shuffling is the opposite of sorting, that is, it is rearranging a sequence of items in a random or meaningless order.

Sorting can order dataitems (or elements) in an increasing, non-decreasing, decreasing, non-increasing order according to some linear relationship among the data items.A sequence of values is said to be in increasing order if the successive element is greater than the previous one. For example, -1,2, 3, 5, 6,18, 29 are in increasing order. A sequence of values is said to be in non-decreasing order, if the successive element is greater than or equal to its previous element in the sequence. This order occurs when the sequence contains similar values. For example, 1, 2,2,3,3,3, 7, 8, 9 are in non-decreasing order. A sequence of values is said to be in decreasing order, if the successive element is less than the current one. For example, 99, 18, 4, 3, 1,0,-4 are in decreasing order, as every next element is less than the previous element.A sequence of values is said to be in non-increasing order, if the successive element is less than or equal to its previous element in the sequence. This order occurs when the sequence contains duplicate values. For example, 29, 7, 6,6, 3, 3 are in non-increasing order

Sorting has two common, yet distinct meanings:

1. Ordering: arranging items in a sequence ordered by some criterion
2. Categorizing: grouping items with similar properties

Ordering items is the combination of categorizing them based on equivalent order, and ordering the categories themselves.

Datacan be sorted by text (A to Z or Z to A, that is, the alphabetical order/the lexicographic order), Numbers (smallest to largest or largest to smallest, that is, numerical order) and dates and times (oldest to newest and newest to oldest-chronological order). Sorting data is an integral part of data analysis, for example, arranging a list of names in alphabetical order or compiling a list of product cost values.Sorting data helps in quickly visualizing and

understanding data, organizing and finding the data that you want, and ultimately make more effective decisions.

Following are some of the examples of sorting in real-life applications of sorting:

1. Telephone Directory – Thetelephone directory stores the telephone numbers of people sorted by their names, so that the names can be searched easily.
2. Dictionary − the dictionary stores words in a lexicographical order so that searching of any word becomes easy.

Sorting a list of n data items can be achieved by different methods. A Sorting algorithm specifies the method to arrange data in a particular order.

There are many sorting algorithms which belong to different classes. There are particular characteristics present in a sorting algorithm that determine the class to which it belongs.

Classification of sorting algorithms

Sorting algorithms can be classified into following types:

Stable or unstable sorting algorithms

The stability of a sorting algorithm is concerned with how the algorithm treats equal (or repeated) elements. Stable sorting algorithms preserve the relative order of equal elements, while as unstable sorting algorithms don't. In other words, if a sorting algorithm, after sorting the data items, does not change the sequence of similar data items in which they appear in the input array then it is called stable sorting. If a sorting algorithm, after sorting the data items, changes the sequence of similar data in which they appear, it is called unstable sorting. Let L be a collection of unordered elements. let M be the collection of elements in L in the sorted order. consider two equal elements in L at indices i and j, that is, $L[i]$ and $L[j]$, that end up at indices m and n respectively in M. We can classify the sorting as stable if:

$i < j$ and $L[i] = L[j]$ and M[m]=L[i] and M[n]=L[j] then $m < n$

Stability is not a concern if:

1. The equal elements are indistinguishable, or
2. All the elements in the collection are distinct

Examples of stable sorts are Merge Sort, Tim sort, Counting Sort, Insertion Sort, and Bubble Sort.Examples of non-stable sorts are Quicksort, Heap sort and Selection Sort

In place or not in place (out place) sorting Algorithms

Every algorithm has memory space(storage) requirements for sorting of data items i.e. the elements of the array need to be placed in main memory before they can be sorted by the processor. If a Sorting algorithm has to sort n data items, then it at least requires n memory locations to store these data items. Some Sorting algorithms require only n locations for sorting of n data items. These sorting algorithms are called in-place sorting algorithms.

Somesorting algorithms however may require more than n storage locations (that is, buffers) to carry out sorting such algorithms are called not-in-place sorting algorithms. In particular, some sorting algorithms are "in-place". Strictly, an in-place sort needs only $O(1)$ memory beyond the items being sorted; sometimes O(log n) additional memory is considered "in-place". Out place algorithms require $O(n)$.

Examples of in place sorts are bubble sort, selection sort, insertion sort,etc.

Examples of not in place sortsare radix sort and bucket sort.

Note: Some sorting algorithms can be implemented both in place and not in place, for example, K-way merge sort etc.

Recursive or non-Recursive (iterative)

Recursion is a phenomenon in which function calls itself again and again

Sorting algorithms can be implemented as recursive algorithm or as non-recursive algorithm or both. Recursive sorting algorithms work by splitting the input array into two or more smaller subarrays and then sorting those sub arrays, it then combines the sorted subarrays into sorted output array. A non-recursive algorithm does the sorting all at once by using iterative statements/repetitive statements/loops only, without calling itself again and again. Bubble-sort is an example of a non-recursive algorithm.

Examples of Recursive sorts are merge sort, quick sort

Examples of not recursive bubble sort, selection sort, insertion sort

Adaptive or Non-AdaptiveSorting Algorithms

Sorting Algorithms can be adaptive or Non-Adaptive depending on whether pre-sortedness of the input list affects or does not affect the running time

A sorting algorithm is said to be adaptive, if it takes advantage of already 'sorted' elements in the list that is to be sorted. That is, while sorting if the source list has some element already sorted, adaptive algorithms will take this into account and will try not to re-order them.

A non-adaptive algorithm is one which does not take into account the elements which are already sorted. They try to force every single element to be re-ordered to confirm their sortedness.

Examples of adaptive sorts are stranded sort, quick sort

Examples of non-adaptivesorts are Selection Sort, Merge Sort, and Heap Sort.

Computational Complexity

The algorithmic complexity of different sorting algorithms varies depending on Best, worst and average case behavior in terms of the size of the data list. For typical serial sorting algorithms, good behavior is $O(n\ log\ n)$, with parallel sort is $O(\log_2 n)$, and bad behavior is $O(n^2)$. Ideal behavior for a serial sort is $O(n)$, but this is not possible in the average case. Optimal parallel sorting is $O(log\ n)$.

Online or offline

Online sorting algorithm can sort a constant stream of input. An online algorithm is one that can process its input one by one in a serial fashion, i.e., in the order that the input is fed to the algorithm, without having the entire input available from the start of the algorithm, for example, insertion sort.

In contrast, an offline algorithm is given the whole problem data from the beginning and is required to output an answer which solves the problem at hand.

Comparison sort or Non-comparison sort

A comparison sort examines the data only by comparing two elements with a comparison operator. best complexity of comparison sort algorithm is $O(n\ log(n))$

A non-comparison algorithm sorts the data without using comparisonsit uses the internal character/nature of the values for producing sorted output. Non comparison sorts can be applied only to some particular cases which satisfy certain conditions, and requires particular values. The best complexity of non-comparison sort is $O(n)$, for example, counting sort,radix sort

Algorithm is serial or parallel

Parallel sorting algorithms are simply algorithms that allows parallel processing, that is, perform multiple operations in a given time.While serial sorting algorithms are simply algorithms that allows serial processing and don't support parallel processing.

External and Internal sorting

In internal sorting all the data to be sorted is stored in memory (main memory or RAM)at all times while sorting algorithm is being executed by a processor.In external sorting data is stored outside memory or secondary memory (like on disk) and only loaded into memory in small blocks whenever needed. External sorting is usually applied in cases when data can't fit into main memory entirely i.e. memory available is less than data to be sorted.

Conclusion

The study of Classification of sorting algorithms helps the algorithmic designers in keeping into consideration different parameters in their mind beforehand while writing the new solutions to sorting problem. In this way they can design the efficient algorithms which can be better as compared to already existing algorithm in terms of running time on processor and storage requirements. This classification can equally beneficial for the new learners of the sorting problem in introducing to them different terms and concepts they should know before studying the sorting algorithms so that develops in them better understanding of the sortingAlgorithms. In conclusion thisclassification is not final it simply provides an insight and will definitely change with evolving technology and it will as well change with the new approaches that will be developed by researchers for solving sorting problem.

Acknowledgment

I am thankful to Prof Firdous Ahmad Mala, Head, Department of Mathematics, Govt. Degree College, Sopore, J&K for providing the necessary guidance for the research work and for helping me in writing this paper. I would also like to thank Prof Joy Paulose, Head, Department of Computer Application,Christ University, Bangalore who has taught me and has been a source of inspiration.

References

[1] Donald Knuth, The Art of Computer Programming, Volume 3: Sorting and Searching, Second Edition. Addison-Wesley, 1998, ISBN 0-201-89685-0, Section 5.4: External Sorting, pp. 248–379

[2] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2009),Introduction to Algorithms (3rd ed.), Cambridge, MA: The MIT Press, pp. 171-172, ISBN 978-0262033848

[3] Mehta, Dinesh P.; Sartaj Sahni (2005). Data Structures and Applications. USA: Chapman & Hall/CRC. pp. 11-8–11-9. ISBN 1-58488-435-5.

[4] Estivill-Castro, Vladimir; Wood, Derick (December 1992). "A survey of adaptive sorting algorithms". ACM. New York, NY, USA. 24 (4): 441476.doi:10.1145/146370.146381 ISSN 0360-0300. S2CID 1540978.

[5] Petersson, Ola; Alistair Moffat (1992). "A framework for adaptive sorting". Lecture Notes in Computer Science. 621. Berlin: Springer Berlin / Heidelberg: 422–433. doi:10.1007/3-540-55706-7_38. ISBN 978-3-540-55706-7. ISSN 1611-3349.

[6] Karp, Richard M. (1992). "On-line algorithms versus off-line algorithms: How much is it worth to know the future?" (PDF). IFIP Congress (1). 12: 416–429. Retrieved 17 August 2015.
[7] Dochow, Robert (2016). Online Algorithms for the Portfolio Selection Problem. Springer Gabler.
[8] Hagerup, Torben; Jyrki Katjainen (2004). Algorithm Theory – SWAT 2004. Berlin Heidelberg: Springer-Verlag. pp. 221–222. ISBN 3-540-22339-8.