JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Study on Sorting Algorithms in Field of Data Structure

Sneha Unnarkar, Dr.Viral Parekh, Dr.Harsha Padheriya
snehaunnarkar1991@gmail.com, viralparekh@sal.edu.in, hpadheriya@gmail.com

SALITER, Assistant Professor, SAL College of Engineering

Abstract: *Based on a comparison operator on the elements, a Sorting Algorithm rearranges the items of an array or list. The comparison operator is used to determine the new order of elements in the respective data structure.Sorting algorithms are one of the most fundamental aspects of computer science. Its objective is to make record finding, insertion, and deletion easier. Five sort algorithms were described to summarise the time and space complexity: bubble, select, insert, merger, and quick.*

Keywords: Sorting Algorithm, Data Structure, Record Finding, Computer Science, Bubble, Insert, Merge,Quick

Introduction

Arranging or sorting objects or items is a time-consuming activity. Its origins can be dated back to the seventh century BCE. The King of Assyria, according to Abdul Wahab and O.Issa (2009), used the idea of sorting clay tablets for the Royal Library by shape[1]. Sorting isn't a quantum leap, but it has progressed in lockstep with the evolution of human cognition. In computer science, sorting is the process of alphabetising, arranging, categorising, or organising data components in an ordered sequence based on similar qualities. Data sorting is essential since it maximises the data's utility. In our daily lives, we may see many examples of sorting, such as how we can easily discover important items in a shopping mall or utility store because the items are organised categorically. Because all of the terms are presented in alphabetical order, finding a word in a dictionary is not difficult. Similarly, the advantages of sorting make finding a phone number, name, or address in a telephone directory a breeze. Sorting is one of the most important fundamental operations in computer science because of its numerous applications. Prioritisation and scheduling the shortest jobs first are examples of sorting procedures. We need to understand sorting algorithms and how they might be used in different situations. The CPU time consumption of five sorting algorithms (Bubble, Quick, Insertion, Selection, and Merge) on a given input will be examined in the best, worst, and average circumstances in this study.

Insertion Sort:

Insertion sort is a straightforward sorting algorithm that works similarly to how you arrange cards in your hands. The array is divided into two halves, one sorted and the other unsorted.
The values from the unsorted component are selected and placed in the sorted part in the proper order.
Unsorted items are relocated and placed into the sorted sublist using the Insertion Sort algorithm, which searches progressively (in the same array). Because the average and worst case complexity of this technique are O(n2), where n is the number of items, it is not suitable for huge data sets.

The essential principle of insertion sort is:

Consider the first two members of the array data. If they're out of order, swap them. Consider the third component and where it should go among the first three.
Consider the fourth item; arrange it among the first four elements in the appropriate spot, then sort the array until it is complete.

Algorithm:

Step 1: If it's the first element, it's already sorted. Step 1: Go back to the first step;
Step 2: Choose the next element
Step 3: Compare all elements in the sorted sub-list that are greater than the sorted value.

Step 4 – In the sorted sub-list, shift those elements that are greater than the value to be sorted. Step 5: Enter the value.
Step 6: Continue sorting the list until it is complete.

Selection Sort

Selection sort is a basic sorting algorithm. This categorisation approach is an in-place comparison-based algorithm.The list is divided into two parts: a sorted segment at the top and an unsorted section at the bottom.On the left is the sorted piece, and on the right is the unsorted portion. The sorted area is initially empty, whereas the unsorted section contains the entire list.
The smallest element of the unsorted array is picked and swapped with the leftmost element, making that element a part of the sorted array. This operation keeps moving the unsorted array boundary one element to the right. This technique is not suitable for large data sets since its average and worst case complexity are O(n2), where n is the number of elements.

The selection sort algorithm's fundamental concept is to find the smallest element in the data list. Place this item at the very top of the list.Find the smallest element in the list.Start at the second spot on the list and work your way down until all of the data is sorted.[5]

Algorithm:

In the first step, set MIN to 0.
Step 2: Scrutinise the list for the tiniest item.
Step 3: Substitute the new value for the value at MIN.
Step 4: To point to the next element, increase the value of MIN. Step 5: Continue sorting the list until it is complete.

Merge Sort

The divide and conquer strategy is the foundation of the sorting technique known as merge sort. With a worst-case time complexity of O, it is one of the most renowned algorithms (n log n). Merge sort divides the array into equal halves, joining the parts in a sorted method.
Following is the algorithm for merging sorts.Make two n/2-length segments out of array A.uses the merge sort method to repeatedly sort each component.Combine the two sorting elements to get a single sorted list.
Algorithm:

Step 1: Return if there is just one element in the list because it has already been sorted. Step 2: Recursively split the list in half. till it cannot be divided any longer.
Step 3:Merge the smaller lists into a new list in sorted order in step three.

Quick Sort

The foundation of the very effective sorting algorithm known as "quick sort" is the division of a large data array into smaller arrays. A huge array is divided into two arrays, one of which contains values less than the pivot value—the value on which the division is made—and the other of which contains values higher than the pivot value. Quick sort divides an array into two subarrays, then calls itself twice in recursive fashion to sort them. Due to its average and worst-case complexity being O(n2), where n is the number of elements, this approach is extremely effective for huge data sets.
This is how the partition algorithm functions. The pivot value is A[p] = x.
Elements smaller than x can be found in A [p...q - 1].
The elements in A [q + 1...s - 1] are greater than or equal to x. There are elements in A[s...r] that are currently unknown.
Algorithm:

Step 1:Pick the highest index value that has a pivot.
Step 2:Remove the pivot and move two variables to the left and right of the list. Step 3:Left indicates a low index.
Step 4:Right indicates a high.
Step 5:move right while value at left is smaller than the pivot Step 6:move left while value at right is bigger than pivot Step 7:If steps 5 and 6 do not match, switch the left and right Step 8:If left > right, the intersection is the new pivot point.

Bubble Sort

A straightforward sorting algorithm is bubble sort. Each pair of adjacent elements in this comparison-based sorting algorithm is compared to each other, and if they are not in the correct order, the elements are swapped. Due to its average and worst-case complexity being O(n2), where n is the number of elements, this approach is not appropriate for huge data sets.

The following are the steps in the bubble sort:
Up until the largest item reaches the end of the array, surrounding elements are exchanged. For the remainder of the array, repeat the previous step.

Algorithm:

If list[i] > list[i+1],

begin BubbleSort(list) for all items of the list. If list[i] > list[i+1],

end if end for return list end BubbleSort.

COMPARISON BETWEEN ALGORITHMS

The best, worst, and average cases of a given algorithm in computer science express the minimum, maximum, and average resource utilisation, respectively. The resource that is typically taken into account is running time, or time complexity, although it may alternatively be memory or another resource. The worst-case execution time is frequently of special relevance in real-time computing since it is crucial to understand how much time would be required in the worst scenario to ensure that the algorithm will always finish on time. [6]

Worst scenario complexity comparison table

Algorithm	**Time Complexity**	**Space Complexity**
Quick Sort	O(n^2)	O(log(n))
Merge Sort	O(n log(n))	O(n)
Bubble Sort	O(n^2)	O(1)
Insertion Sort	O(n^2)	O(1)
Selection Sort	O(n^2)	O(1)

Table 1 Multi-class classifications [21]

Conclusion

In this paper, I've covered popular sorting algorithms. Depending on the kind and volume of data, various algorithms must be chosen, compared for how quickly they run, and tried to be analysed from a broad perspective. These algorithms' analyses are based on the same information and computer.Gnome sort has been demonstrated to be the fastest method for previously sorted data, but selection sort outperforms Gnome and Bubble on unsorted data. The elements can be switched around using gnome sort or bubble sort. However, selection sort keeps sorting even if the items have already been sorted. Since no single algorithm can absolutely solve any problem, additional comparisons between more distinct sorting methods are necessary.The findings demonstrate the effectiveness of Quick sort for both small and large numbers. In comparison to other O(n log n) algorithms, quick sort is considerably faster in practise. Because each element will only be compared to nearby components and exchanged if they are out of order, the Bubble sort performs the most swaps. Large arrays are sorted much more slowly by insertion sort than small arrays.

Reference

[1] Wahab, A., Issa, O.A. Fundamentals of Library & Information Sciences, (1st ed.). Cataloguing-in-Publication Data, Ilorin (2009).

[2] Cormen, T.H., Leiserson, C.E., & Rivest, R.L. Introduction to Algorithms (2nd ed.). Prentice Hall of India private limited, New Delhi-110001 (2001).

[3] Aho A., Hopcroft J., and Ullman J. The Design and Analysis of Computer Algorithms, Addison Wesley (1974).

[4]www.cse.iitk.ac.in/users/cs300/2014/home/~rahume/c s300A/ techpaper-review/5A.pdf

[5]T. H. Cormen, C. E. Lieserson, R. L. Rivest and S. Clifford,
!Introduction to Algorith", 3rd ed., The MIT Press Cambridge, Massachusetts London, England 2009.

[6] https://en.wikipedia.org/wiki/Best,_worst_and_average_ case

[7] Kazim Ali, International Journal of Advanced Research in Computer Science, 8 (1), Jan-Feb 2017, 277-280

[8] Chhajed. N, U. Imran, Simarjeet. S., B., A Comparison Based Analysis of Four Different Types of Sorting Algorithms in Data Structures with Their Performances, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 2, February 2013.