International Journal of
Emerging Research in Management &Technology
ISSN: 2278-9359 (Volume-4, Issue-4)

Research Article | **April 2015**

Learning System for Various Sorting Algorithm and Their Comparison

Divesh Khemani
B.E 8th Semester, CSE,
Chhattisgarh Swami Vivekanand Technical Universty, Bhilai,
Chhattisgarh, India

Akanksha Pandey
B.E 8th Semester, CSE,
Chhattisgarh Swami Vivekanand Technical Universty, Bhilai,
Chhattisgarh, India

Siddharth S. Shukla
Associate Professor, Department of CSE
Shri Shankaracharya Institute Professional Management & Technology, Raipur,
Chhattisgarh, India

Abstract:

Most Pragmatic applications in computer programming require the output to be arranged in a sequential order. Sorting is rearranging information or data into either ascending or descending order. An algorithm is any well-defined procedure or set of instructions, that takes some values as input, processes them and gives some output in the form of some values. Lots of sorting algorithms has been developed to raise the performance in terms of computational complexity, memory and other factors. The goal of this paper is too reviewed on various sorting algorithms and compares the various performance factors among them.

***Keywords**: selection sort, insertion sort, bubble sort, quick sort, time complexity*.

I. INTRODUCTION

Sorting is the rearrangement of items in a list into their correct abecedarian order, while algorithm is a stepwise way of achieving solution for a craved result. There are a number of sorting algorithms like quick sort, heap sort, merge sort; selection sort, insertion sort etc. There are two classes of sorting algorithms comparison based and non-comparison based sort.

Algorithms have a essential and key role in solving the abstract problems, colloquially an algorithm is a well-defined procedure that takes input and gives output. Algorithm is a tool or a sequence of steps to solve the abstract problems. There are various methodologies and techniques based on which various different kinds of algorithms are contrived. Out of various problem solving algorithms, let us discuss about the sorting algorithms. In case of sorting, it is required to arrange a sequence of numbers or array into a acknowledged order, mostly non-decreasing. In computer science, a sorting algorithm is that which puts elements of a list into a definite order. The orders that are mainly used are numerical order and abecedarian order. There are two conditions enlisted that output must satisfy. These conditions are:

1. The output is in non-decreasing order (each element is no smaller than the previous element according to the desired total order)
2. The output is a permutation or reordering of the input.

Further, the data is often taken to be in an array, which allows random access, rather than a list, which only allows sequential access, though often algorithms can be applied with suitable modification to either type of data.

Sorting algorithms are prevalent in introductory computer science classes, where the abundance of algorithms for the problem provides a gentle introduction to a variety of core algorithm concepts, such as
big O notation, divide and conquer algorithms, data structures, randomized algorithms, best, worst and average case analysis, time-space tradeoffs, and lower bounds.

II. SORTING ALGORITHM

A. Selection Sort

Selection sort is one of the simplest sorting algorithms. It is actually an improvement in performance of bubble sort. This algorithm is called selection sort because it works by selecting a minimum element in each step of the sort. This algorithm, iterate through a list of n unsorted items, has a worst-case, average-case, and best-case run- time of $O(n^2)$, assuming that comparisons can be done in constant time. The Selection sort spends most of its time trying to find the minimum element in the "unsorted" part of the array. The brief algorithm for selection sort is given below.

Algorithm 1:
SELECTION_SORT (A)
1: for i ← 0 to n-1 do
2: min ← i;
3: for i ← 0 to n-1 do
4: If A[j] < A[min] then
5: min ← j
6: swap(A[i] ,A[min])
Total run time Cost == $O(n^2)$ asymptotically

It works as follows:
1. Find the smallest element using a linear scan.
2. Swap the element with the element in first position.
3. Find the second smallest element in the remaining array.

B. Insertion Sort

Insertion sort iterates, consuming one input element each repetition, and growing a sorted output list. Each iteration, insertion sort removes one element from the input data, finds the location it belongs within the sorted list, and inserts it there. It repeats until no input elements remain. Sorting is typically done in-place, by iterating up the array, growing the sorted list behind it. At each array-position, it checks the value there against the largest value in the sorted list (which happens to be next to it, in the previous arrayposition checked). If larger, it leaves the element in place and moves to the next. If smaller, it finds the correct position within the sorted list, shifts all the larger values up to make a space, and inserts into that correct position. The resulting array after *k* iterations has the property where the first *k* + 1 entries are sorted ("+1" because the first entry is skipped). In each iteration the first remaining entry of the input is removed, and inserted into the result at the correct position.

Algorithm 2:
INSERTION (DATA, N)
1. Set A[0] = -∞
2. Repeat Steps iii to v for K = 2 to N
3. Set TEMP = DATA[K] and PTR = K-1
4. Repeat while TEMP < DATA[PTR]
a. Set DATA[PTR+1] = A[PTR]
b. Set PTR =PTR – 1
5. Set DATA[PTR+1] = TEMP
6. Exit.

C. Bubble Sort

In this task, the goal is to sort an array of elements using the bubble sort algorithm. The elements must have a total order and the index of the array can be of any discrete type. For languages where this is not possible, sort an array of integers. The bubble sort is generally considered to be the simplest sorting algorithm. Because of its simplicity and ease of visualization, it is often taught in introductory computer science courses. Because of its abysmal $O(n^2)$ performance, it is not used often for large (or even mediumsized) datasets.

Bubble sort algorithm can be depicted as follows:
Algorithm 3:

```
void BubbleSort(int a[])
{
int i,j;
for (i=MAXLENGTH; --i >=0;) {
swapped = 0;
for (j=0; j<i;j++) {
if (a[j]>a[j+1]) {
Swap[a[j],a[j+1]);
swapped=1;
}
}
if (!swapped) return;
}
}
```

D. Quick Sort

Quick sort is a comparison sort developed by Tony Hoare. Also, like merge sort, it is a divide and conquer algorithm, and just like merge sort, it uses recursion to sort the lists. It uses a pivot chosen by the programmer, and passes through the sorting list and on a certain condition, it sorts the data set.

Quick sort algorithm can be depicted as follows:
Algorithm 4:
QUICKSORT (A)
1: step←m;
2: while step > 0
3: for (i←0 to n with increment 1)
4: do temp←0;
5: do j←i;


```
6: for (k←j+step to n with increment step)
7: do temp←A[k];
8: do j←k-step;
9: while (j>=0 && A[j]>temp)
10: do A[j+step]=A[j];
11: do j←j-step;
12: do Array[j]+step←temp;
13: do step←step/2;
```

III. LITERATURE SURVEY

Robert Sedgewick is the author of a well-known book series *Algorithms*, published by Addison-Wesley. The first edition of the book was published in 1983 and contained code in Pascal. Subsequent editions used C, C++, Modula-3, and Java. With Philippe Flajolet he wrote several books and preprints which promoted analytic combinatorics, a discipline which relies on the use of generating functions and complex analysis in order to enumerate combinatorial structures, and to study their asymptotic properties. As explained by Knuth in *The Art of Computer Programming*, this is the key to perform average case analysis of algorithms. [1].

Bentley received a B.S. in mathematical sciences from Stanford University in 1974, and M.S. and Ph.D in 1976 from the University of North Carolina at Chapel Hill; while a student, he also held internships at the Xerox Palo Alto Research Center and Stanford Linear Accelerator Center. After receiving his Ph.D., he joined the faculty at Carnegie Mellon University as an assistant professor of computer science and mathematics. At CMU, his students included Brian Reid, John Ousterhout, Jeff Eppinger, Joshua Bloch, and James Gosling, and he was one of Charles Leiserson's advisors. Later, Bentley moved to Bell Laboratories. [2][3].

Dr. Baecker is an expert in human-computer interaction ("HCI") and user interface ("UI") design. His research interests include work on electronic memory aids and other cognitive prostheses; computer applications in education;computer-supported cooperative learning, multimedia and new media; software visualization; groupware and computersupported cooperative work; computer animation and interactive computer graphics; computer literacy and how computers can help us work better and safer; and entrepreneurship and the management of small business as well as the stimulation of innovation. Baecker is also interested in the social implications of computing, especially the issue of responsibility when humans and computers interact. [9].

IV. CONCLUSION

This paper discusses comparison based sorting algorithms and there learning technique. It analyses the performance of these algorithms for the same number of elements.

Selection sort is used when no extra space is required to sort the record. Bubble Sort is a very simple algorithm that is only suitable for small lists. There are lots of alternative sorting algorithms that are more performant than Bubble Sort. Bubble Sort is widely considered to be the least performant of all the established sorting algorithms.

The selection sort is a good one to use. It is intuitive and very simple to program. It offers quite good performance, its particular strength being the small number of exchanges needed. For a given number of data items, the selection sort always goes through a set number of comparisons and exchanges, so its performance is predictable.

After analyzing the different sorting algorithm for particular problems, we conclude that, based on the problem characteristics different algorithms are efficient for different problems. We can't say particular sorting algorithm is efficient for all kind of problems. Based on problem criteria and characteristics one of the sorting algorithm is efficient.

Table 1. Comparison of various sorting

	Worst	**Average**	**Best**
Selection Sort	$O(n^2)$	$O(n^2)$	$O(n^2)$
Insertion Sort	$O(n^2)$	$O(n^2)$	O(n)
Bubble Sort	$O(n^2)$	$O(n^2)$	O(n)
Quick Sort	$O(n^2)$	O(nlogn)	O(nlogn)

REFERENCES

[1] Algorithms in Java, Parts 1-4, 3rd edition by Robert Sedgewick. Addison Wesley, 2003.
[2] Programming Pearls by Jon Bentley. Addison Wesley, 1986.
[3] *Quicksort is Optimal* by Robert Sedgewick and Jon Bentley, Knuthfest, Stanford University, January, 2002.
[4] Dual Pivot Quicksort: Code and Discussion.
[5] *Bubble-sort with Hungarian ("Csángó") folk dance* YouTube video, created at Sapientia University, Tirgu Mures (Marosvásárhely), Romania.
[5] Wirth, N., 1976. "Algorithms + Data Structures = Programs": Prentice-Hall, Inc. Englewood Cliffs, N.J.K. Mehlhorn. Sorting and Searching. Springer Verlag, Berlin, 1984.
[6] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein."Introduction to Algorithms". MIT Press, Cambridge, MA, 2nd edition, 2001.

[7] Aho, A.V., Hopcroft, J.E., Ullman, J. D., 1974."The Design and Analysis of Computer Algorithms" Addison-Wesley.
[8] Select-sort with Gypsy folk dance YouTube video, created at Sapientia University, Tirgu Mures (Marosvásárhely), Romania.
[9] Sorting Out Sorting, Ronald M. Baecker with the assistance of David Sherman, 30 minute color sound film, Dynamic Graphics Project, University of Toronto, 1981. Excerpted and reprinted in SIGGRAPH Video Review 7, 1983. Distributed by Morgan Kaufmann, Publishers.
[9] A. Aho, J. Hopcroft, J. Ullman, "Data Structures and Algorithms", Pearson India reprint, 2000
[10] Wikipedia: *Sorting Algorithm*, Retrieved from http://en.wikipedia.org/wiki/Sorting_algorithm

