University of Science

Sorting Algorithms - An Overview

Data Structures and Algorithms

Huynh Minh Tuan - 20120024@student.hcmus.edu.vn

December 2021

Abstract

Sorting is nothing but alphabetizing, categorizing, arranging, or putting items in an ordered sequence. It is a key fundamental operation in the field of computer science. It is of extreme importance because it adds usefulness to data. In this report, I have compared eleven common sorting algorithms (Selection Sort, Insertion Sort, Bubble Sort, Shaker Sort, Shell Sort, Heap Sort, Merge Sort, Quick Sort, Counting Sort, Radix Sort, and Flash Sort). I have developed a program in C++, Python and experimented with several input sizes 10,000, 30,000, 50,000, 100,000, 300,000, and 500,000 elements. The performance and efficiency of these algorithms in terms of CPU time consumption as well as the number of comparisons that have been recorded and presented in tabular and graphical form.

1. Introduction

Sorting is not a leap but it has emerged in parallel with the development of the human mind. In computer science, alphabetizing, arranging, categorizing, or putting data items in an ordered sequence on the basis of similar properties is called sorting. Sorting is of key importance because it optimizes the usefulness of data. We can observe plenty of sorting examples in our daily life, e.g. we can easily find required items in a shopping mall or utility store because the items are kept categorically.

The items to be sorted may be in various forms i.e. random as a whole, already sorted, very small or extremely large in numer, sorted in reverse order etc. There is no algorithm that is best for sorting all types of data. We must be familiar with sorting algorithms in terms of their suitability in a particular situation.

In this paper, I am going to compare eleven common sorting algorithms (Selection Sort, Insertion Sort, Bubble Sort, Shaker Sort, Shell Sort, Heap Sort, Merge Sort, Quick Sort, Counting Sort, Radix Sort, and Flash Sort) for their CPU time consumption and number of compared operations on four different data arrangements (Sorted data (in ascending order), Nearly sorted data, Revherse sorted data, and Randomized data).

2. Algorithm presentation

2.1. Selection Sort

Idea

The Selection Sort is based on the idea of finding the minimum element in an unsorted array and then putting it in its correct position in a sorted array.

Pseudo code

Algorithm 2.1: Selection Sort

```
Input: $a_1, a_2, ..., a_N$
Output: $a_1, a_2, ..., a_N$ (in sorted)
for $i \leftarrow 1$ to $N$ do
    $minIndex \leftarrow i$
    for $j \leftarrow i + 1$ to $N$ do
        if $a_{minIndex} > a_j$ then
            $minIndex \leftarrow j$
        end
    end
    swap($a_{minIndex}$, $a_i$)
end
```

Complexity

Best case time complexity: $O(N^2)$
Worst case time complexity: $O(N^2)$
Worst case space complexity: $O(1)$

2.2. Insertion Sort

2.3. Bubble Sort

2.4. Shaker Sort

2.5. Shell Sort

2.6. Heap Sort

2.7. Merge Sort

2.8. Quick Sort

2.9. Counting Sort

2.10. Radix Sort

2.11. Flash Sort