BIRZEIT UNIVERSITY

Faculty of Engineering & Technology

Department of Electrical and Computer Engineering

COMP2421

Data Structures and Algorithms

Project Four

Comparison Of Sorting Algorithms

Student Name: Ibraheem Alyan

Student ID: 1201180

Section: 2

Instructor: Dr.Ahmad Abu Snaina

Date: Tue 28 Dec 2021

Abstract

The aim of this paper is to study and compare three different sorting algorithms and discuss their properties, time complexity, space complexity, and use cases of each one. The algorithms are: TimSort, Counting Sort, Tree Sort.

This page was intentionally left blank

Contents Table

Tim Sort

I. Brief

TimSort is a hybrid sorting algorithm, it is a combination of the well known merge sort and insertion sort. TimSort is one of the most used sort algorithms (in fact the python built-in sort function is based on this algorithm) because it works better than other algorithms on real-world data.

II. How it works

Start by **dividing** the data into pieces called **Runs**, each run should be small and preferably a power of 2 (ex 64, 128), these runs are sorted each one alone using **Insertion** sort, then they are combined using the merge (combine) function of the merge sort.

III. Props and analysis

Time Complexity:

Tim Sort Takes **O(n Log(n))** in its average and worst case, and since its adaptive the best case is when the input is sorted and then it takes only **O(n)**.

Space Complexity: O (n)

Properties:

- Stable
- Comparative
- Adaptive (takes advantage of existing order)
- Offline

Counting Sort

I. Brief

Counting sort is a sorting algorithm specific for integers (with limited range) , it works (from its name) by counting occurrences of an object in the input array. Counting sort is usually not used, except as a subroutine in radix sort or when the number of distinct objects in not very far from the number of objects in the input.

II. How it works

By counting the elements that have unique values, and using those values as a prefix sum to output the elements in their right indices on the output. This clearly states that it is an online algorithm.

III. Props and analysis

Time Complexity:

Counting Sort Takes **O(n + L)** in its best, average, and worst cases. Where **n** in the number of elements and **L** is the length of the range of possible elements values.

Space Complexity: O (L)

Properties:

- Non-Stable
- Not Comparative
- Non-Adaptive
- Online

Tree Sort

I. Brief

Tree sort is a simple sorting algorithms that depends on binary search trees, meaning that its operation also depends on the type of binary search tree that is used, (self-balancing …).

II. How it works

Tree Sort Constructs a binary search tree and iterates over the input array inserting each element in the binary search tree, this approach make it an online sort algorithm. Once all the elements are added to the binary search tree, the output is simply the in-order traversal of the tree.

III. Props and analysis

Time Complexity:

Adding to a binary search tree takes on average O(log n), and performing this process for every element means the Tree Sort Takes **O(n Log n)** in its best, and average case. But depending on the binary search tree type the worst case may take up to O(square n).

Space Complexity: O (n)

Properties:

- Stable
- Comparative
- Adaptive
- Online

Conclusion

Summing up, the TimSort tends to be used the most due to it having least average complexity for most real-world data and applications and also being parallelizable since it uses merge sort, but its single point of failure is being an offline algorithm. From the previous algorithms if we needed an online algorithm we should usually use tree sort not counting sort (due to its special requirements).

References

[1] https://en.wikipedia.org/wiki/Sorting_algorithm#Stability

[2] https://en.wikipedia.org/wiki/Tree_sort

[3] https://www.geeksforgeeks.org/timsort/

[4] https://en.wikipedia.org/wiki/Timsort

[5] https://www.bigocheatsheet.com

[6] https://www.geeksforgeeks.org/counting-sort/