Faculty of Engineering & Technology

Computer Science Department

Data Structures and Algorithms

COMP2421

Project Report

Prepared by:

Mohammad Abu Shams 1200549

Instructor: Dr. Ahmad AbuSnaina

Section: 1

Date: 23-2-2022

Table of Contents

List of Figures

List of Tables

Gnome Sort

Gnome Sort is a sorting algorithm with a time complexity of O (N^2) that involves swapping adjacent elements that are out of order in order to sort the entire list. This technique is similar to Insertion Sort, but instead of shifting elements, adjacent elements are swapped.

The algorithm is named after the traditional Dutch Garden Gnome, who sorts flower pots in a similar manner. The gnome follows these steps:

1. The gnome advances one step if the pots before and after are arranged correctly.
2. If the order is off, the gnome switches the pots and goes back one step.
3. The gnome begins to move ahead when he reaches the beginning of the pot line. When he reaches the end, the list is sorted.

Gnome Sort is distinct from Bubble Sort or Insertion Sort, and is in fact the simplest sorting algorithm.

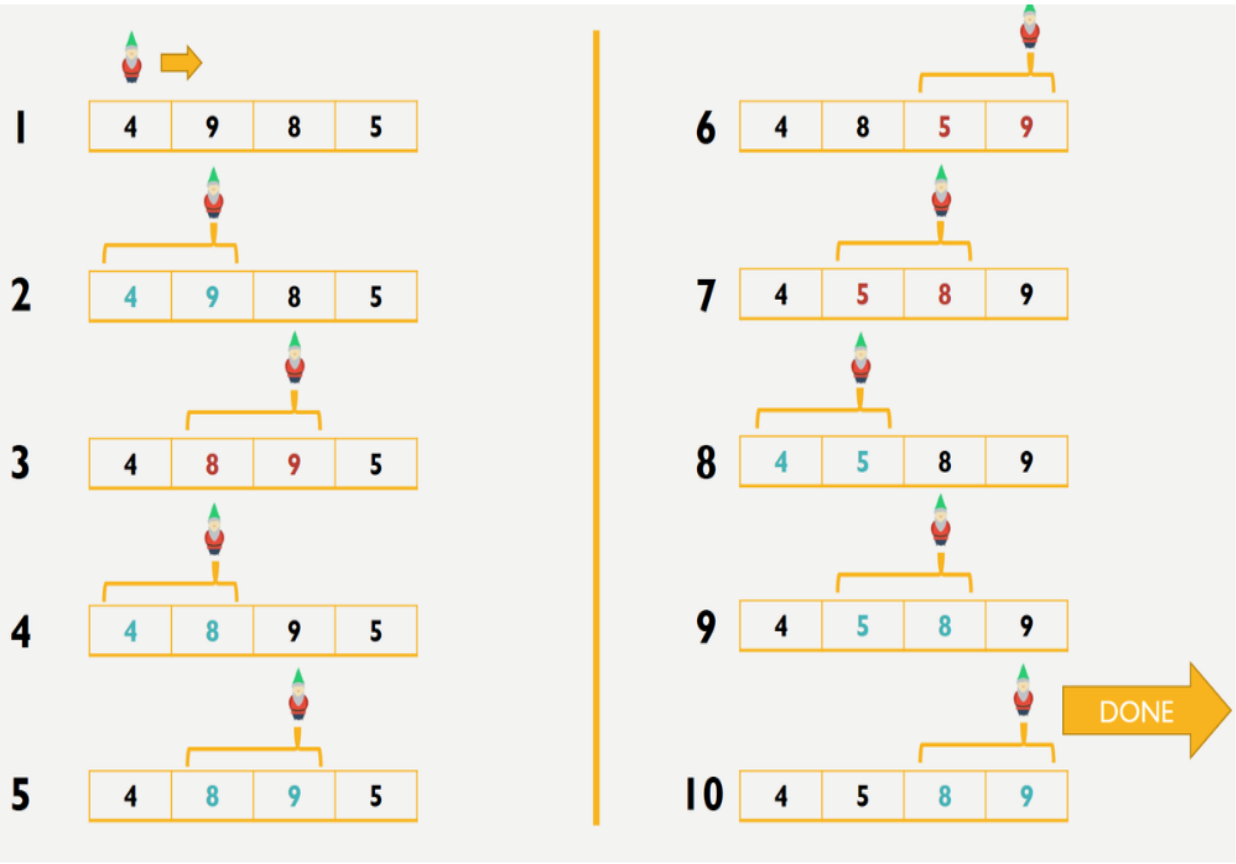

Figure 1: Gnome Sort [1]

Time Complexity

Table 1: Time Complexity of Gnome Sort

Worst Case	Average Case	Best Case
O(N^2)	O(N^2)	O(N)

Space Complexity

The Space Complexity is O (1).

Stability

It is a stable sorting algorithms.

Space

It is an out place sorting algorithms.

Sorting

Sorted (ascending): If the input data array is already sorted in ascending order, Gnome Sort will have the best case running time, which is O (N).

Sorted (descending): If the input data array is already sorted in descending order, Gnome Sort will have the worst case running time, which is O(N^2).

Not sorted: If the input data array is not sorted, Gnome will depend on the exact arrangement of the array's element. It will have the average case running time, which is O (N^2).

Tree Sort

The Tree sort algorithm sorts a list by utilizing a Binary Search Tree data structure. The first step involves constructing a binary search tree from the input list or array. Then, by performing an in-order traversal of the binary search tree, the elements are extracted and sorted.

Algorithm:

To use Tree sort, start by gathering the input elements and storing them in an array. Then, insert the elements from the array into a Binary Search Tree. Finally, obtain the sorted list of elements by performing an in-order traversal of the tree.
Tree sort is advantageous in various applications, including:

• Online sorting: In situations where the input elements arrive gradually over time, and the sorting algorithm needs to maintain a continuously sorted list, Tree sort is a popular option. By updating the binary search tree with each new element and obtaining the sorted list using an in-order traversal, a continuously sorted list can be achieved.

• Adaptive sorting: When a splay tree is used as the binary search tree, the resulting algorithm is called "splay sort." Splay sort is an adaptive sorting algorithm, which implies that it has a faster running time than O (N log N) for partially sorted input arrays. As a result, splay sort is an efficient alternative for sorting partially sorted or almost sorted data.

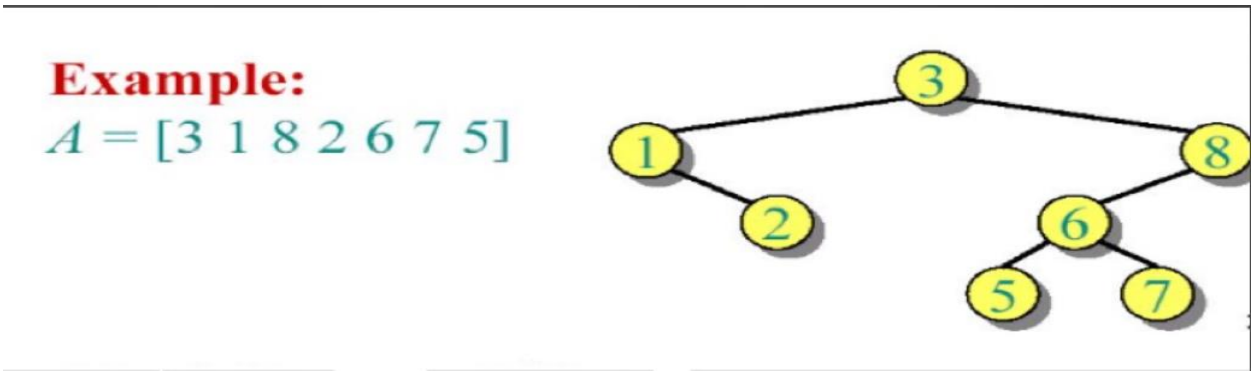

Figure 2: Tree Sort [2]

The output is : [1,2,3,5,6,7,8]

Time Complexity

Table 2: Time Complexity of Tree Sort

Worst Case	Average Case	Best Case
O(N^2)	O(N log N)	O(N log N)

Space Complexity

The Space Complexity is O (N).

Stability

It is a stable sorting algorithms.

Space

It is an in place sorting algorithms.

Sorting

Sorted (ascending): If the input data array is already sorted in ascending order, Tree Sort will have the worst case running time, which is O(N^2).

Sorted (descending): If the input data array is already sorted in descending order, Tree Sort will have the best case running time, which is O(N log N).

Not sorted: If the input data array is not sorted, Tree Sort will have the average case running time, which is O (N log N).

Stooge Sort

The Stooge Sort is an intriguing sorting algorithm that, while not particularly efficient, is a very interesting method of sorting data. This recursive sorting method involves splitting the input array into two sections that overlap, with each section containing two-thirds of the original array. The first and last two-thirds of the array are sorted independently, and then the first two-thirds are sorted again to achieve a fully sorted array.

To perform the sorting, the Stooge Sort algorithm splits the input array into two sections and sorts them individually. Through a merge pass, the algorithm then combines the sorted sections, swapping elements between the overlapping parts of the two sections as necessary to ensure the resulting array is sorted. This pass ensures the array is left in a fully sorted state by the end of the algorithm.

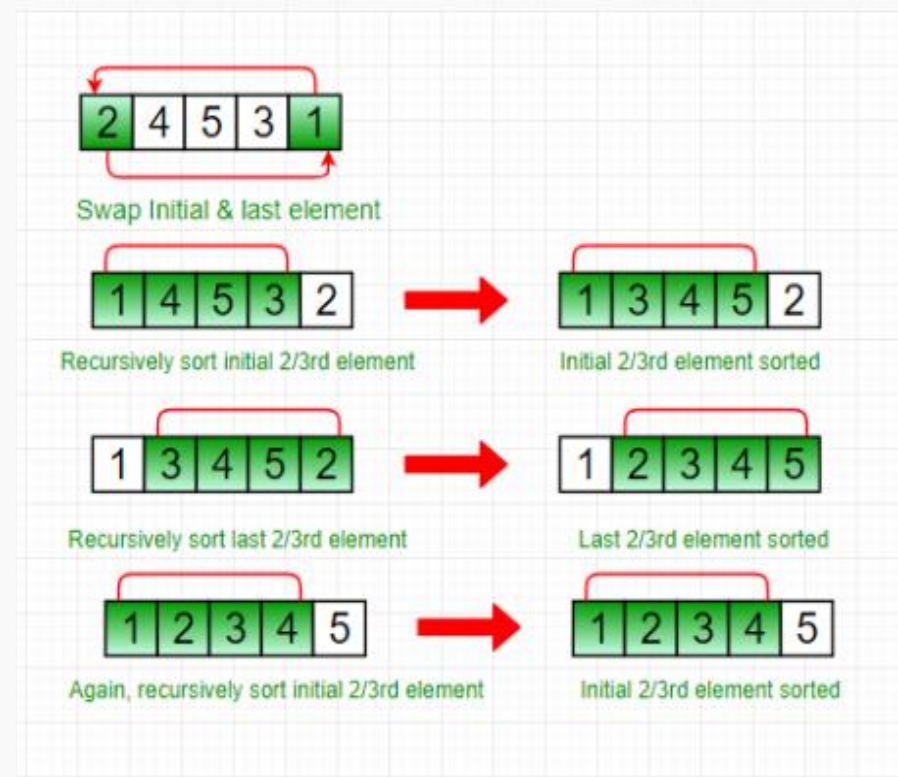

Figure 3: Stooge Sort [3]

Time Complexity

Table 3: Time Complexity of Stooge Sort

Worst Case	Average Case	Best Case
O(N^(log 3 / log 1.5))	O(N^(log 3 / log 1.5))	O(N^(log 3 / log 1.5))

Space Complexity

The Space Complexity is O (log N).

Stability

It is a stable sorting algorithms.

Space

It is an in place sorting algorithms.

Sorting

Sorted (ascending): If the input data array is already sorted in ascending order, Stooge Sort will have the best case running time, which is O(N^(log 3 / log 1.5)).

Sorted (descending): If the input data array is already sorted in descending order, Stooge Sort will have the worst case running time, which is also O(N^(log 3 / log 1.5)).

Not sorted: If the input data array is not sorted, Tree Sort will have the average case running time, which is also O(N^(log 3 / log 1.5)).

Burst Sort

It is a sorting algorithm developed by Tony P. Hoare and Charles M. Payne in 1997, is a unique approach that combines the quicksort and radix sort algorithms. This combination enables Burst Sort to achieve faster sorting times than either of its parent algorithms.

Burst Sort's sorting process is divided into two phases. In the first phase, quicksort is used to sort the elements in the list based on their first character. In the second phase, radix sort is used to sort the remaining characters in each element. By merging these two algorithms, Burst Sort can complete sorting tasks faster than quicksort or radix sort alone.
Burst Sort is especially effective at sorting strings, making it a great choice for applications that require fast sorting of large amounts of string data. Due to its efficiency, Burst Sort is widely used in sorting applications such as databases and text processors.

For example, we have an array with elements {1, 7, 3, 8, 9, 5, 2}.

1- We split the elements into two sub-arrays

{1, 7, 3}
{8, 9, 5, 2}

2- We sort the two sub-arrays

{1, 3, 7}
{2, 5, 8, 9}

3- Merge the two-sub arrays.

{1, 2, 3, 5, 7, 8, 9}

Time Complexity

Table 4: Time Complexity of Burst Sort

Worst Case	Average Case	Best Case
O(N^2)	O(N log N)	O(N log N)

Space Complexity

The Space Complexity is O (N).

Stability

It is not a stable sorting algorithms. It depends on the two sorting:
Quick sort and it is not stable algorithms, and radix sort it is a stable algorithms, so the burst sort is not stable.

Space

It is an out place sorting algorithms. It depends on the two sorting:
Quick sort and it in place algorithms, and radix sort it is out place algorithms, so the burst sort is out place.

Sorting

Sorted (ascending): If the input data array is already sorted in ascending order, Burst Sort will have the best case running time, which is O(N log N).

Sorted (descending): If the input data array is already sorted in descending order, Burst Sort will have the worst case running time, which O(N^2).

Not sorted: If the input data array is not sorted, Burst Sort will have the average case running time, which is O(N log N).

Sleep Sort

This sorting algorithm leverages multithreading to achieve its desired outcome. For each element in the input array, a corresponding thread is created, which sleeps for a duration proportional to the element's value.

The threads with the shortest sleep times awaken first, and their elements are printed in order. This process is repeated until all elements are printed in sorted order, with the largest element taking the longest time to awaken and be printed. The multithreading process is managed by the operating system in the background, which makes the algorithm appear "mysterious" since we have no visibility into its internal processes.

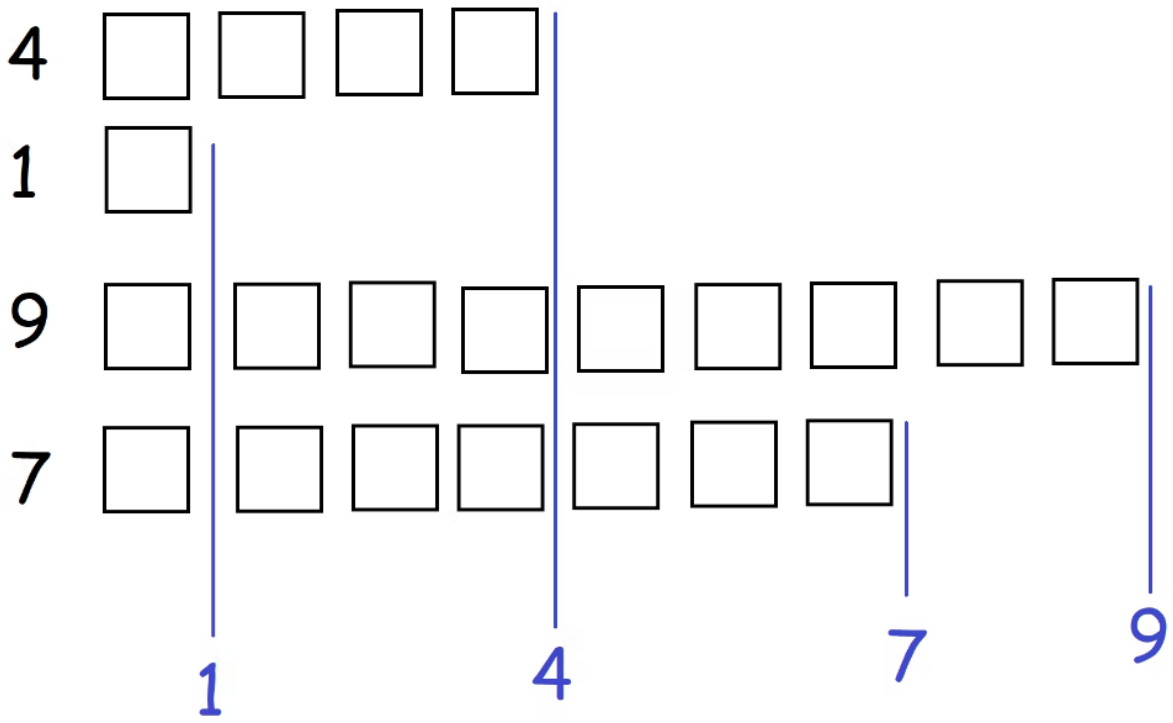

Figure 4: Sleep Sort [4]

Time Complexity

Table 5: Time Complexity of Sleep Sort

Worst Case	Average Case	Best Case
O(N^2)	O(N log N)	O(N)

Space Complexity

The Space Complexity is O (N).

Stability

It is not a stable sorting algorithms.

Space

It is an out place sorting algorithms.

Sorting

Sorted (ascending): If the input data array is already sorted in ascending order, Sleep Sort will have the best case running time, which is O (N).

Sorted (descending): If the input data array is already sorted in descending order, Sleep Sort will have the worst case running time, which is O(N^2).

Not sorted: If the input data array is not sorted, Sleep Sort will have the average case running time, which is O (N log N).

Summary

Table 6: Summary

Sort / Properties	Gnome Sort	Tree Sort	Stooge Sort	Burst Sort	Sleep Sort
Worst Case	O(N^2)	O(N^2)	O(N^(log 3 / log 1.5))	O(N^2)	O(N^2)
Average Case	O(N^2)	O(N log N)	O(N^(log 3 / log 1.5))	O(N log N)	O(N log N)
Best Case	O(N)	O(N log N)	O(N^(log 3 / log 1.5))	O(N log N)	O(N)
Space Complexity	O(1)	O(N)	O(log N)	O(N)	O(N)
Stability	Stable	Stable	Stable	Not stable	Not stable
Space	Out Place	In Place	In Place	Out place	Out place

- The sort algorithm that's have the best time complexity is Tree Sort and Burst Sort.

- The sort algorithm that's have the best space complexity is Gnome sort which is equal O (1).

- According to the space, Tree Sort and Stooge Sort are the best, because these two sorts don't use external memory to sort the given array.

- According to the stability, Tree Sort and Stooge Sort and Gnome sort are the best.

References

[1] Gnome Sort. (2023, 2 23). Retrieved from Wikimedia: https://commons.wikimedia.org/wiki/File:GnomeSort.png

[2] Tree Sort . (2023, 2 23). Retrieved from Slide Player: https://slideplayer.com/slide/9171988/

[3] Stooge Sort. (2023, 2 23). Retrieved from geeksforgeeks: https://www.geeksforgeeks.org/stooge-sort/

[4] Sleep Sort . (2023, 2 23). Retrieved from Vivian Dai's Blog : https://viviandai.hashnode.dev/esoteric-sorting-algorithms#heading-sleep-sort