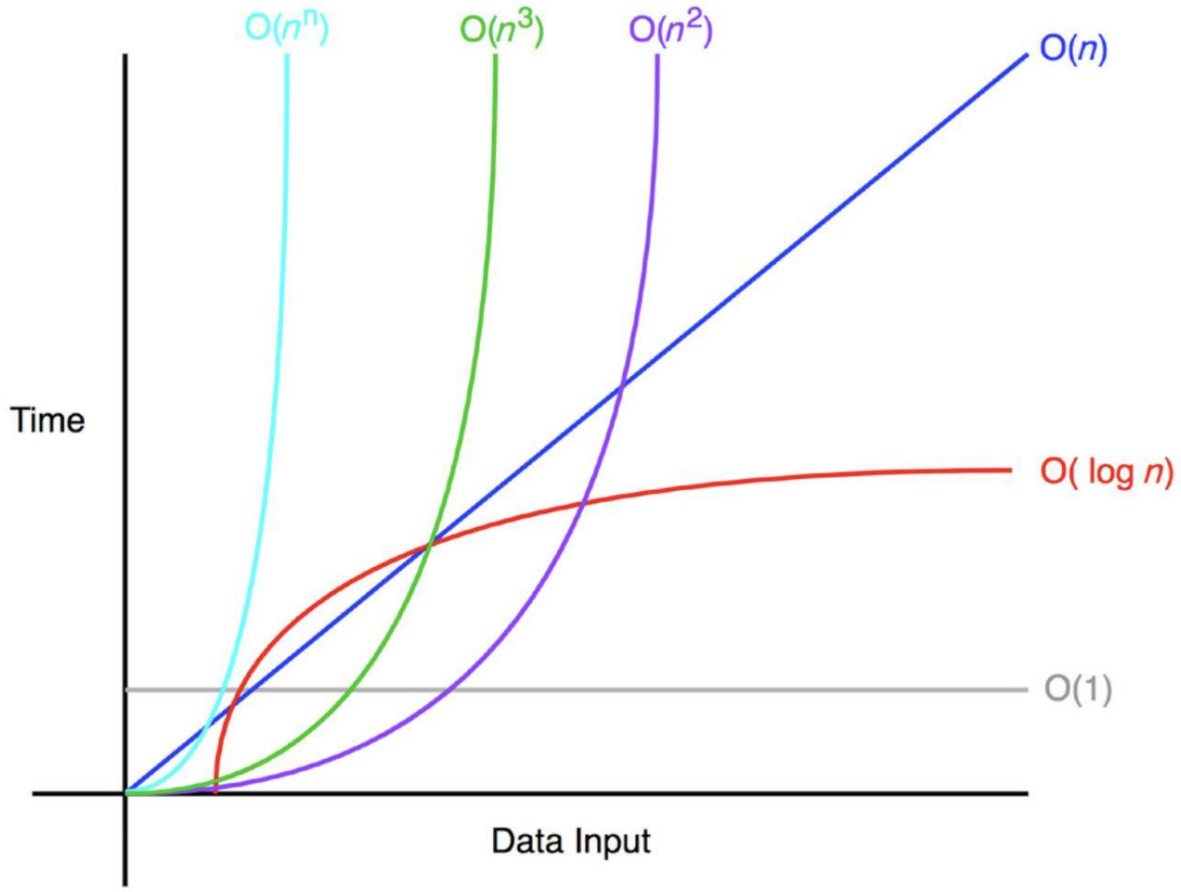
Algorithm & Data Structures



Semester 3 C Programming

1_ADS_Intro_Algo_Structures

Oct 30, 2022.

Student: Andre Sanao

Course: Technology

Table of Contents

Introduction

The purpose of this assignment is to research 3 different sort of sorting algorithms. After researching the sorting algorithm, they will be tested and compared to see which is faster in different scenarios.

Research

In this section, 3 different sorting algorithms will be implemented with 3 number (N) of elements. These elements are:

1. N = 10
2. N = 1000
3. N = 10000

These elements will be tested with the chosen algorithms. The sorting algorithm will be classified by their time complexity which are best- and worst-case behavior.

Sorting Algorithms

The following sorting algorithm are chosen for their time complexity and their popularity among others. While there are better examples, these algorithms are used as an example for some Big O notation for their performance.

The following sorting algorithms are:

Insertion Sort

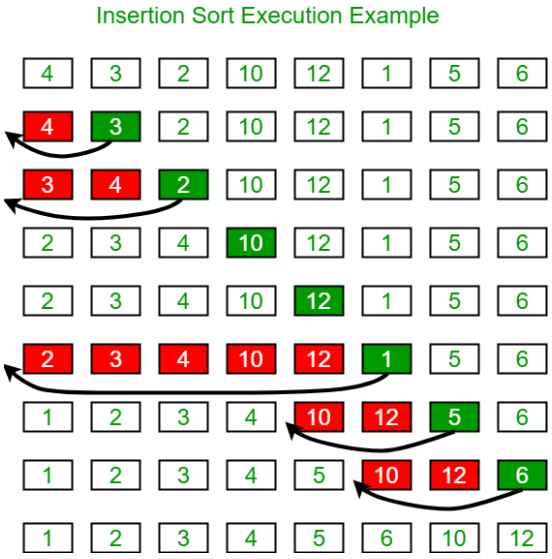


Figure 1 Visual example of insertion sort

Insertion sort is one of the most popular sorting algorithms. It is a simple algorithm that works like how you sort a card from your hand. The array is split into 2 sorted and unsorted part. Elements from the unsorted part are picked and placed at the correct position in the sorted part.

Time complexity

The time complexity for insertion sorting is best in O(N) and worst in O(N^2). It takes the maximum time to sort if the elements are sorted in reverse order however it takes minimum time when the elements are already sorted.

Quick Sort

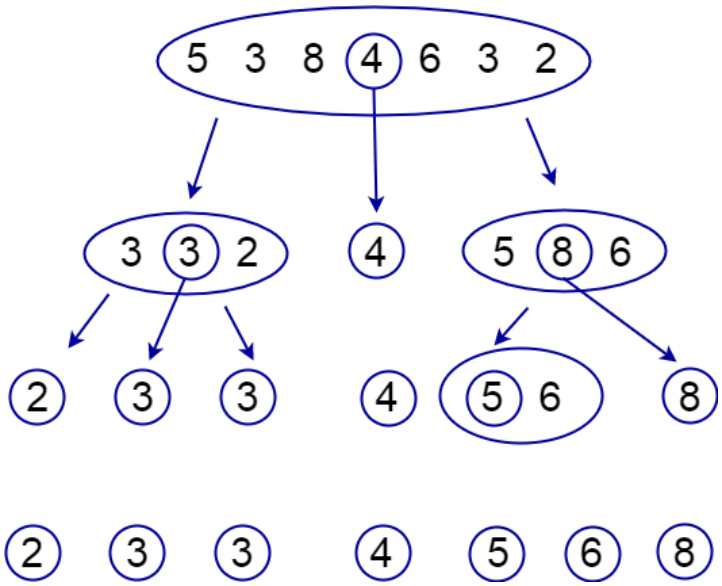


Figure 2 Visual example of quick sort

Quick sort uses a divide and conquer method. Quick sort picks an element from the array and this will be called the pivot element. After assigning a pivot, the unsorted array is divided into 2 with values less than the pivot will be the first sub array while values greater than the pivot will be the second sub array. This step is also known as the partition operation and will recursively repeat assigning and dividing of the sub arrays of the smaller and greater values separately.

Time complexity

The time complexity for quick sorting is best & average O(N lg N) and worst in O(N^2). The best situation in this case is when the partition process always chooses the middle element as the pivot. The worst case is when the partition process always picks the greatest or the smallest value as the pivot. If it is considered that the partition strategy where the last element is always picked as a pivot, the worst case would occur when the array is already sorted in increasing or decreasing order.

Heap Sort

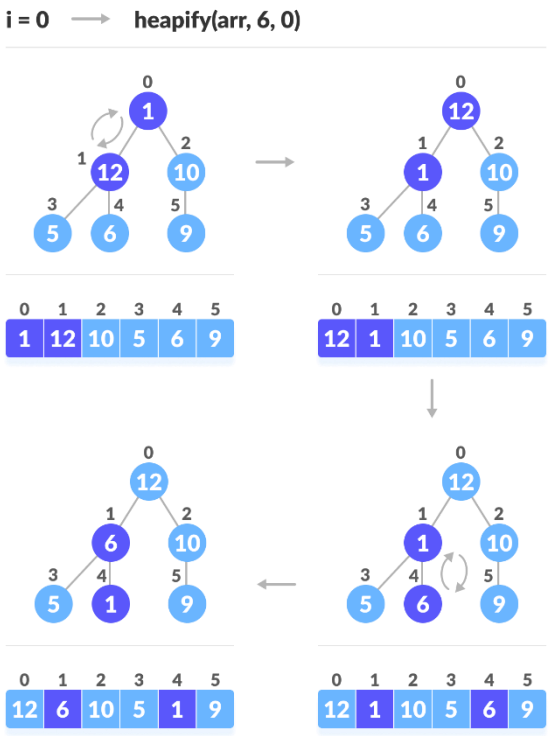


Figure 3 Visual example of heap sort

Heap sort is a comparison-based sorting algorithm based on Binary Heap data structure. It has similarity to Selection sort where the minimum element is chosen and placed at the beginning and the process is repeated for the remaining elements. The heap sort algorithm has limited uses because Quick sort are better in practice. It uses a process called “heapify” where a heap data structure is created from a binary tree represented using an array. It is used to create Min-Heap or Max-Heap.

After heapifying the array, the elements are one by one deleted from the root node of the Max-Heap and replace it with the last node in the heap and then heapify the root of the heap. The process is repeated until the size of the heap is greater than 1. The heapify procedure can only be applied to a node if its children nodes are heapified. So the heapification requires to be performed in the bottom-up order.

Time complexity

The time complexity for heap sorting is Best/Avg/Worst in O(N lg N) . During heapifying the elements whose sub nodes are already Max-Heaps, it is needed to keep comparing the element with its left and right children push it downwards until it reaches a point where both its children are smaller than it.

In worst cases, it will need to move an element from the root to the leaf node (bottom end nodes) making a multiple of lg(N) comparison and swaps. Only in smaller array sizes can heap sort be more efficient where its underlying data structure, heap, can be efficiently used if we want to extract the smallest (or largest) from the list of items without the overhead of keeping the remaining items in the sorted order.

Conclusion

In this section will be the conclusion of the chosen sorting algorithm. These 3 will be compared to how quick they can sort.

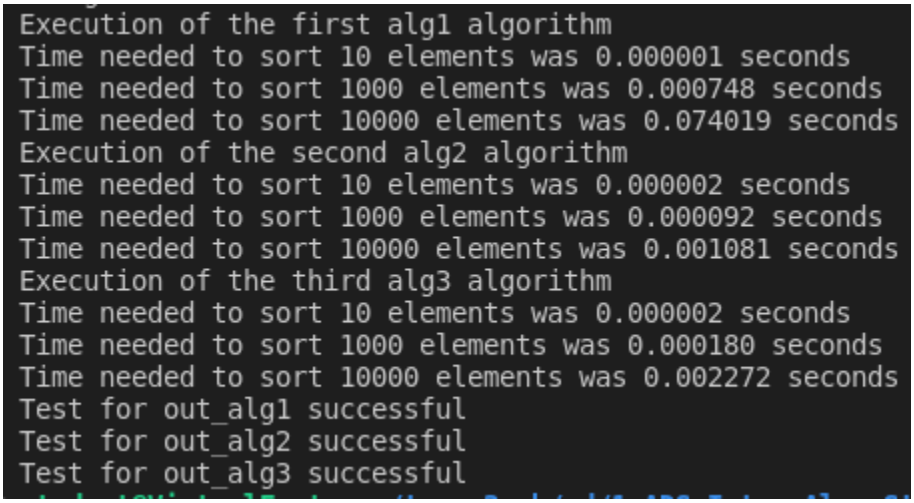


Figure 4 Execution speed of chosen algorithms

For the first algorithm(alg1) is Insertion sort, second algorithm(alg2) is Quick sort and lastly third algorithm(alg3) is Heap sort.

An array size of 10 elements Insertion and Heap sort is tied while Quick sort is only a bit slower. In array size of 1000 elements, Quick sort out speed the other 2 and is twice as fast as Heap sort. Lastly an array size of 10000 elements, Quick sort seems to be the fastest.

Quick sort seems to be faster the greater the size of the array while Insertion sort for smaller array. Heap sort is an all-rounder where it is not a disadvantage.

Bibliography

Sorting algorithm (2022) *Wikipedia*. Wikimedia Foundation. Available at: https://en.wikipedia.org/wiki/Sorting_algorithm#Classification (Accessed: October 30, 2022).

Sorting algorithms (no date) *BetterExplained*. Available at: https://betterexplained.com/articles/sorting-algorithms/ (Accessed: October 30, 2022).

Das, S. (2022) *10 best sorting algorithms you must know about*, *Crio Blog*. Crio Blog. Available at: https://www.crio.do/blog/top-10-sorting-algorithms/ (Accessed: October 30, 2022).

Choudhary, V. (2018) *Big-O notation explained with examples*, *Developer Insider*. Developer Insider. Available at: https://developerinsider.co/big-o-notation-explained-with-examples/ (Accessed: October 30, 2022).