Practical 9: Advanced Sorting

Thi Van Anh DUONG Student ID: 90023112

Diploma of Information Technology, Curtin College

DSA1002: Data Structures and Algorithms

Coordinator: Khurram Hameed

31 August 2022

Student Declaration of Originality

☒	This assignment is my own original work, and no part has been copied from another student’s work or source, except where it is clearly attributed.
☒	All facts/ideas/claims are from academic sources are paraphrased and cited/referenced correctly.
☒	I have not previously submitted this work in any form for THIS or for any other unit; or published it elsewhere before.
☒	No part of this work has been written for me by another person.
☒	I recognise that should this declaration be found to be false, disciplinary action could be taken and penalties imposed in accordance with Curtin College policy.

Electronic Submission:

☒	I accept that it is my responsibility to check that the submitted file is the correct file, is readable and has not been corrupted.
☒	I acknowledge that a submitted file that is unable to be read cannot be marked and will be treated as a non-submission.
☒	I hold a copy of this work if the original is damaged, and will retain a copy of the Turnitin receipt as evidence of submission.

DISCUSSION ON THE PERFORMANCE OF 4 ADVANCED SORTING METHODS:

Here is the table which records my testing against 4 sorting algorithms:

Method Array size	Merge Sort	Quick Sort		
		Leftmost pivot	Median-of-3 pivot	Three-way
10	0.0058s	0.0033s	0.0036s	0.0035s
100	0.0361s	0.0323s	0.0308s	0.0307s
1000	0.3646s	0.3658s	0.3004s	0.3202s
10000	2.0697s	1.9535s	1.8253s	1.9401s

Table 1. Time record to sort of an array size-n (random values)

Method Array size	Leftmost pivot	Median-of-3 pivot	Three-way
10	0.0014s	0.0013s	0.0014s
100	0.0053s	0.0048s	0.0054s
1000	0.041s	0.0474s	0.0573s
10000	0.3352s	0.313s	0.0435s

Table 2. Time record to sort an arrays (repeating values)

- With random arrays:
  + *merge sort*: worst case time complexity $O(N \log_2 N)$
  ➔ depend on the increasing of array sizes
  ➔ is consistent regardless of the array case
  ➔ not an in-place sort, therefore it requires more memory from the computer.
  ➔ can work well with large datasets
  ➔ stable
  + *3 kinds of quick sort*: worst case time complexity $O(N^2)$
  ➔ time taken are slightly better than merge sort
  ➔ the performance of quick sorts with arrays containing repetitive values excels that of the random ones, which should have been the opposite in theory(should have time complexity at $O(N^2)$).
  ➔ can not work well with large datasets
  ➔ does not require any additional storage
  ➔ unstable

  Overall, all of these advanced sorting algorithms reduce the overall amount of time needed in compare to bubble sort, selection sort and insertion sort. However, merge sort is more efficient and works faster than quick sort in case of larger array size or datasets. Whereas, quick sort is more efficient and works faster than merge sort in case of smaller array size or datasets.