DOKUZ EYLUL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CME 2204
Algorithm Analysis

Comparison of Sorting Algorithms

Lecturers

Instructor Dr. Zerrin Işık

Prepared By

2016510040 - Elif Karataş

19.04.2020 - Sunday

	EQUAL INTEGERS			RANDOM INTEGERS			INCREASING INTEGERS			DECREASING INTEGERS		
	1.000	10.000	100.000	1.000	10.000	100.000	1.000	10.000	100.000	1.000	10.000	100.000
Heap Sort	0,8518	1,2006	1,5598	0,8374	1,5261	5,2394	0,7958	1,638	3,9495	1,2057	1,3864	4,3246
Dual Pivot Quick Sort	0,0709	0,5265	1,8217	0,0469	1,3608	2,4221	0,121	2,3576	29,1559	0,0845	2,2693	26,4098
Shell Sort	0,0879	0,4362	2,4946	0,0707	1,2555	4,799	0,0538	0,7417	2,7704	0,0495	0,7091	2,9808

Table.1: Inputs/Sorting Algorithm Run Time in ms

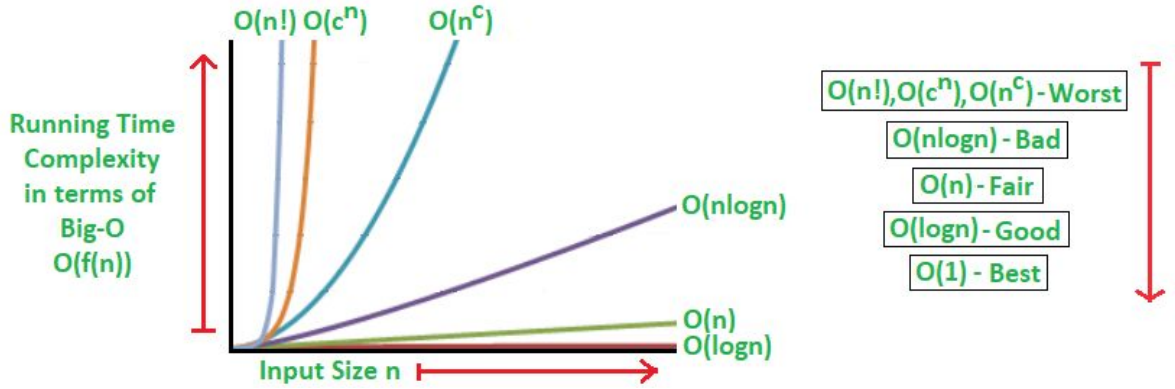
Analysis of Time Complexity for Sorting Algorithms

Heap Sort: Heap Sort has **O(nlog n)** time complexities for all the cases (best case, average case, and worst case). **Heapify** costs **O(logn)** and **Build-Heap** makes **O(n)** such calls. Best case for **equal** integers.

Dual Pivot Quicksort: Time complexity is **O(nlnn)**. Best case for **random** integers. Worst case for **increasing** and **decreasing** integers.

Shell Sort: Worst case complexity is **$O(n^2)$**. Best case complexity is **O(n*log n).** Best case for **increasing** integers.

Comparison and Selection of Time Complexities for Sorting Algorithms



If there isn't any information about integers, We should use shell sort. Because it is the best choice unless there are too many equal integers.

If there are too many equal integers, We should use dual pivot quicksort but if the array size

is too high this will change. Heap sort is best if there are too many integers and also it is an in-place algorithm.

Example Scenario:

We aim to place students at universities according to their central exam grades and preferences. If there are millions of students in the exam, which sorting algorithm would you use to do this placement task faster?

Answer:

Have three different sorting algorithms (heapSort, dualPivotQuickSort and shellSort). So, We must consider time complexities:

Heap Sort: O(nlgn)
Dual Pivot QuickSort: O(nlnn)
Shell Sort: $O(n^2)$

If we compare time complexities of heap sort and dual pivot quick sort, there is a little difference between run times but when array size be larger, difference will increase. So, best choice for this scenario is **dual pivot quicksort**.

References:

Heap Sort:

https://www.geeksforgeeks.org/heap-sort/

https://www.geeksforgeeks.org/time-complexity-of-building-a-heap/?ref=rp

Dual Pivot Quicksort:

https://dev.to/s_awdesh/double-pivot-quick-sort--javas-default-sorting-algorithm-1m4

https://www.quora.com/What-is-the-complexity-of-dual-pivot-Quicksort

https://cs.stackexchange.com/questions/24092/dual-pivot-quicksort-reference-implementation

http://hg.openjdk.java.net/jdk8/jdk8/jdk/file/tip/src/share/classes/java/util/DualPivotQuicksort.java

Shell Sort:

https://www.geeksforgeeks.org/shellsort/