


FACULTY OF ENGINEERING
COMPUTER ENGINEERING DEPARTMENT
Tınaztepe Yerleşkesi, Buca-Kaynaklar, Dokuz Eylül Üniversitesi, İZMİR, TÜRKİYE
30.03.2024
Assignment: CME 2204 | Assignment 1 | Comparing Dual Pivot Quick Sort and Shell Sort Prepared by: İrem Tekin

1. Introduction
In the domain of algorithmic sorting techniques, discerning the most efficient method for various datasets is paramount. This project undertakes a comparative analysis between Shell Sort and DualPivotQuickSort to elucidate their respective efficiencies across different scenarios.

2. Overall Assessment of The Assignment
Shell sort, like quicksort, is a comparison-based sorting algorithm. However, it differs in its approach. Instead of breaking the array into smaller sub-arrays as quicksort does, Shell sort works by repeatedly sorting subsets of elements far apart from each other and then progressively reducing the gap between elements to be compared. This makes Shell sort more efficient than simple insertion sort, especially for larger arrays.

Dual-pivot quicksort, on the other hand, is an enhancement of the traditional quicksort algorithm. Instead of selecting just one pivot element, dual-pivot quicksort selects two pivot elements, dividing the array into three sections: elements less than the smaller pivot, elements between the two pivots, and elements greater than the larger pivot. This approach helps to improve the efficiency of quicksort, particularly for arrays with many duplicate elements.

2.1 Time Complexity

2.1.1 DualPivotQuick Sort
The time complexity of DualPivotQuick Sort is O(n log n) for the best and average cases, and O(n^2) for the worst case, which is an improvement over the worst-case scenario of traditional quicksort.

2.1.2 Shell Sort
While the exact worst-case time complexity of Shell Sort depends on the gap sequence and the specific input, it is generally considered to be more efficient than the worst-case scenario of DualPivotQuick Sort, which is O(n^2) when a poor pivot selection strategy is employed.

2.2 Space Requirement
2.2.1 Shell Sort
Shell Sort typically requires only a constant amount of additional space, regardless of the size of the input array. This additional space is primarily used for temporary storage and swapping elements during the sorting process. The space complexity of Shell Sort is therefore O(1), meaning it is constant and does not grow with the size of the input array.

2.2.2 DualPivotQuick Sort
DualPivotQuick Sort is an in-place sorting algorithm, similar to traditional Quicksort. This means that it doesn't require additional memory proportional to the size of the input array beyond a constant amount. The space complexity of DualPivotQuick Sort is O(1), indicating constant space usage regardless of the size of the input array.

3. Comparison Table

	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
dualPivot QuickSor	0.2676 ms	1.1687 ms	6.9495 ms	0.3171 ms	1.6378 ms	11.2611 ms	8.1035 ms	15.2375 ms	713.5747 ms	1.8986 ms	14.0979 ms	766.0307 ms
shellSort	0,2741 ms	2.1856 ms	8.4433 ms	0.4794 ms	3.4651 ms	12.9542 ms	0.2171 ms	2.2643 ms	6.3672 ms	0.3034 ms	2.048 ms	5.8378 ms

4.Master Method:

4.1 DualPivotQuickSort:
4.1.1 Best Case and Average Case:
$T(n)=3T(n/3)+\Omega(n)$

a = 3 ,b = 3 ,d = 1
$\log_b a$ vs d
$\log_3 3$ = d 1=1 case 2: O($n^d \log n$) ➔ Θ(n logn)

4.1.2 Worst Case:
$T(n)=T(n-1)+O(n)$
$=T(n-2)+O(n-1)+O(n)$

$T(n)=T(n-1)+O(n)$. The next step is to stop when $n-k=1$,because when $k=n-1$, we get $T(1)$.When $k=n-1$, $O(kn-(k-1))=O(n^2)$.
Thus, $T(n)=O(n^2)$.

This implies that the worst-case time complexity of the given recurrence relation is $O(n^2)$.

4.2 ShellSort:
It is not appropriate to analyze the time complexity of Shell Sort with Master Theorem because Shell Sort does not typically have a recursive structure.

5.Scenario

5.1 (A)

The goal is to place students at universities according to their central exam (YKS) grades and preferences. If there are 3 million of students in the exam, which sorting algorithm would you use to do this placement task faster?

Answer: The two sorting algorithms provided in your code are Dual Pivot Quick Sort and Shell Sort. Both of these algorithms have average time complexities of O(n log n) or slightly worse. However, Dual Pivot Quick Sort tends to perform better in most cases, especially with large datasets.Therefore, for the placement task of 3 million students, you should use the Dual Pivot QuickSort algorithm to achieve faster sorting.

5.2 (B)

You have a Turkish-English dictionary with a single word translation for each word in alphabetical order. You aim to translate it into an English-Turkish dictionary. For example; if your Tur-Eng dictionary contains [ayna : mirror, bardak : glass, elma : apple, kitap : book] then your new Eng-Tur dictionary should contain [apple : elma, book : kitap, glass : bardak, mirror: ayna]. Assume that there are thousands of words in your dictionary, which sorting algorithm do you use to perform this translation faster?

Answer: In the given context, considering that the dataset is already sorted alphabetically, it would be more appropriate to choose a stable sorting algorithm such as Shell Sort for creating the English-Turkish dictionary. Stable sorting algorithms ensure that items with the same key values retain their relative order, which is advantageous when the dataset is already sorted. This helps to minimize unnecessary processing overhead.

Therefore, in this particular scenario, Shell Sort would be a preferable choice for efficiently translating the English-Turkish dictionary.

4. References

https://www.youtube.com/watch?v=Vv0as6y-tdM
https://www.youtube.com/watch?v=zJLbQd8buxo
https://www.youtube.com/watch?v=r3a25XPf2A8&t=3s
https://www.youtube.com/watch?v=mI9lBmzzTyQ&t=3s
https://www.geeksforgeeks.org/dual-pivot-quicksort/
https://chat.openai.com/c/fa10568c-5119-4262-a18a-1c084cb02817 (for Big-O Notation)
https://stackoverflow.com/questions/12767588/time-complexity-for-shell-sort