DOKUZ EYLUL UNIVERSITY

ENGINEERING FACULTY

DEPARTMENT OF COMPUTER ENGINEERING

CME2204 ALGORITHM ANALYSIS

ASSIGNMENT - I

Comparison of Heapsort, dualPivotQuicksort and Shellsort

by

Vedat Özcan

2016510083

IZMIR

05.04.2020

Contents

1. INTRODUCTION

In this study, we will try to calculate how long the three different sorting algorithms (**Heap Sort, Dual Pivot Quick Sort and Shell Sort**) sort the inputs with different values and lengths. We will implement these three algorithms and try to calculate their runtime.

1.1 Problem Definition

Heap sort is a comparison based sorting technique based on Binary Heap data structure. It is similar to selection sort where we first find the maximum element and place the maximum element at the end. We repeat the same process for remaining element.

As we know, the Dual Pivot Quick Sort takes two pivot as left and right. Then partitioning the array from these pivots, so that all elements are left to the left pivot are less than to the left pivot, and all elements that are between the left and right pivot thats are greater than or equal the left pivot and smaller than or equal the right pivot.All the remaining elements are larger than to the right pivot.

Shell sort is an algorithm that first sorts the elements far apart from each other and successively reduces the interval between the elements to be sorted. It is a generalized version of insertion sort. In shell sort, elements at a specific interval are sorted. The interval between the elements is gradually decreased based on the sequence used. the performance of the shell sort depends on the type of sequence used for a given input array.

2. RUNNING TIME TABLES

	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
heapSort	0.107	0.709	2.590	0.406	1.759	12.508	0.648	1.650	10.794	0.630	1.723	9.993
dualPivot QuickSort	0.4708	1.985	13.262	0.648	3.379	22.546	4.596	64.554	745.461	3.207	38.299	1055.218
shellSort	0.201	2.124	8.776	0.394	3.114	15.580	0.202	2.207	7.158	0.300	2.416	7.809

Table 1: Running time table of algoritms (milliseconds)

```
Total running time of heapSort for equal integers is: 0.1064 milliseconds
Total running time of dualPivotQuickSort for equal integers is: 0.5461 milliseconds
Total running time of shellSort for equal integers is: 0.1966 milliseconds
Total running time of heapSort for random integers is: 0.1421 milliseconds
Total running time of dualPivotQuickSort for random integers is: 0.1672 milliseconds
Total running time of shellSort for random integers is: 0.4131 milliseconds
Total running time of heapSort for increasing integers is: 0.1196 milliseconds
Total running time of dualPivotQuickSort for increasing integers is: 3.3662 milliseconds
Total running time of shellSort for increasing integers is: 0.1944 milliseconds
Total running time of heapSort for decreasing integers is: 0.1243 milliseconds
Total running time of dualPivotQuickSort for decreasing integers is: 1.5568 milliseconds
Total running time of shellSort for decreasing integers is: 0.298 milliseconds
```

Table 2: A running time result for 1k array size (millideconds)

3. RESULT

Heap Sort is one of the best sorting methods being in-place and with no quadratic worst-case running time.Its runtime is the same all cases.Time increasing depanding on the size of array but it does not matter how the array is assigned.Its runtime is O(nlogn) for all situations as seen in the running time table above.

Dual Pivot Quick Sort is quite efficient for large-sized data sets as its average and worst-case complexity are O(nlogn) respectively. It's pretty easy to see why it should be O(nlogn) in the average case according to running time table. The worst case is still O(n^2), of course. For example, the worst case occurs if you choose the first and last elements as pivots and the array is increase or dicrease order. It is clearly seen in the above working time table that it exceeds 800 milliseconds.

```
public void dualPivotQuickSort(int[] arr, int p, int r) {
    try {
        if (p < r) {
            int lp = partition(arr, p, r, p); // left pivot lp = partition(arr, p, 0, p);
            int rp = partition(arr, p, r, lp); // right pivot

            dualPivotQuickSort(arr, p, lp - 1); // recursively sorts the left,
            dualPivotQuickSort(arr, lp + 1, rp - 1); // middle, and right sub-arrays
            dualPivotQuickSort(arr, rp + 1, r); // created for algorithm
        }
    } catch (StackOverflowError ex) {
        System.gc();
    }
}
```

*At first I was getting stackoverflow error in dual PivotQuickSort 10k+ increasing,decreasing states(worst case**) According to my research there is a problem with Eclipse max. RAM usage.Despite my efforts to increase max. RAM, the existing problem could not be solved.. Then I researched this problem to solve it in different ways. Finally, I looked at how garbage collector was used in java and I solved the problem by using it. After using garbage collector, I saw that dual PivotQuickSort's time in smaller array sizes has decreased.*

Shell Sort is like insertion sort best case time complexity is when the array is increment order. The would mean that the inner if statement will never be true, making the inner while loop a constant time operation. Using the bounds you've used for the other loops gives O(nlogn). The best case of O(n) is reached by using a constant number of increments. As seen in the running time table (Table 1*), this algorithm approaches the worst case (O(nlogn)^2) depending on the array size and selection of gap.

4. SCENERIO

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?

First of all, it is very important to consider the number of exam results and how they are distributed in this scenerio. It is clearly that in this scenerio there are millions of students exam and this exam randomly distributed.When we look at the runtime table of algorithms we see that the running time of Heap Sort does not change according to the situations, Shell Sort runs faster in small data size, and Dual Pivot Quick Sort runs slower than Heap Sort in the big and randomly data set. Based on all these inferences, I would used the Heap Sort algorithm and place student according to exam result.

5. REFERENCES

[1] ***Wikipedia, URL: https://en.wikipedia.org/wiki/Sorting_algorithm (Access time; March, 23, 2020).***

[2] ***Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest Clifford Stein, Introduction to Algorithms Third Edition, The MIT Press Cambridge: Massachusetts London, England.***

[3] ***Geeks for Geeks, URL: https://www.geeksforgeeks.org/heap-sort/ (Access time; March, 23, 2020).***

[4] ***Geeks for Geeks, URL: https://www.geeksforgeeks.org/dual-pivot-quicksort/ (Access time; March, 25, 2020).***

[5] ***Geeks for Geeks, URL: https://www.geeksforgeeks.org/shellsort/ (Access time; March, 25, 2020).***

[6] ***Stack Overflow, URL: https://stackoverflow.com/questions/10639322/how-can-i-specify-the-default-jvm-arguments-for-programs-i-run-from-eclipse (Access time; March, 26, 2020). (For java.lang.StackOverFlowError)***

[7] ***GeeksforGeeks, URL: https://www.geeksforgeeks.org/time-complexities-of-all-sorting-algorithms/ (Access time; March, 26, 2020).***

[8] ***Stack Overflow, URL: https://stackoverflow.com/questions/2679330/catching-java-lang-outofmemoryerror (Access time; March, 29, 2020). (For java.lang.StackOverFlowError)***