CME 2204 ASSIGNMENT - I

ALGORITHM ANALYSIS

*I used the algorithm of heapSort, dual PivotQuickSort and shellSort in my homework.* *I observed the  
runtimes of the sorting algorithms and thought which algorithm is better and where it can be used.  
During homework,* *the StackoverflowError occurred when the array of Increasing and decreasing numbers was over 25k.In my opinion, because it was too recursive, the memory became full and such an error occurred.* *To solve the StackoverflowError, I increased the default memory of eclipse from arguments section.(command: -Xss).*

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | EQUAL INTEGERS | | | RANDOM INTEGERS | | | INCREASING INTEGERS | | | DECREASING INTEGERS | | |
|  | 1000 | 10000 | 100000 | 1000 | 10000 | 100000 | 1000 | 10000 | 100000 | 1000 | 10000 | 100000 |
| *heapSort* | 1.0ms | 2.0ms | 4.0ms | 0.0ms | 4.0ms | 18.0ms | 0.0ms | 1.0ms | 11.0ms | 0.0ms | 1.0ms | 12.0ms |
| *dualPivot QuickSort* | 0.0ms | 1.0ms | 15.0ms | 1.0ms | 2.0ms | 18.0ms | 0.0ms | 30.0ms | 2748.0ms | 1.0ms | 29.0ms | 2712.0ms |
| *shellSort* | 0.0ms | 5.0ms | 16.0ms | 1.0ms | 7.0ms | 19.0ms | 0.0ms | 0.0ms | 3.0ms | 0.0ms | 1.0ms | 4.0ms |

*Runtime Performance****My assumptions;  
Algorithms of Sorting***

1. *ShellSort (Best case: O(nlogn))*
2. *HeapSort (Best case: O(nlogn))*
3. *DPQuickSort (Best case: O(nlogn))*

*The worst case of the DP Quick Sort is O(n^2) when the array is already sorted in an increasing or decreasing order.  
DP Quick sort can sometimes pass shell sort in "equal and random integers", but quick sort works very unstable.  
I think the shell sort is performing well because it's space complexity O (1).  
In my opinion,Shell sort algorithm is bit faster than Heap Sort algorithm.  
Heap sort algorithm is better than other algorithms in Equal and Random integers.  
Shell sort algorithm is better than other algorithms in Increasing and Decreasing integers.  
Some values could not be displayed because they are too low. (etc. "0.0ms")*

***Shell Sort Runtime:****Best case: O(nlogn) , Average Case: O(n(logn)^2) , Worst Case: O(n(logn)^2) , Space Complexity: O(1)* ***Heap Sort Runtime:****Best case: O(nlogn) , Average Case: O(nlogn) , Worst Case: O(nlogn) , Space Complexity: O(n)* ***Dual Pivot Quick Sort Runtime:****Best case: O(nlogn) , Average Case: O(nlogn) , Worst Case: O(n^2) , Space Complexity: O(nlogn)*

***Question:***

*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:***

*According to the students' scores and preferences, I think the most suitable Shell Sort algorithm is appropriate because when looking random integers, runtime performance is the best Shell sort Algorithm.* *If Shell Sort algorithm is not used, I definitely prefer the Heap Sort algorithm because the Heap Sort algorithm performs almost close to the Shell Sort algorithm and the Heap Sort algorithm is more efficient than the DP Quick Sort algorithm. The worst case of the DP Quick Sort algorithm is O (n ^ 2) (increasing and decreasing numbers). So DP Quick Sort should not be preferred.*

*Note:* *If the number increases, the Shell Sort algorithm is always better than other algorithms.(* *Array size has been increased and tried.)*