2020 Algorithms

#PA1

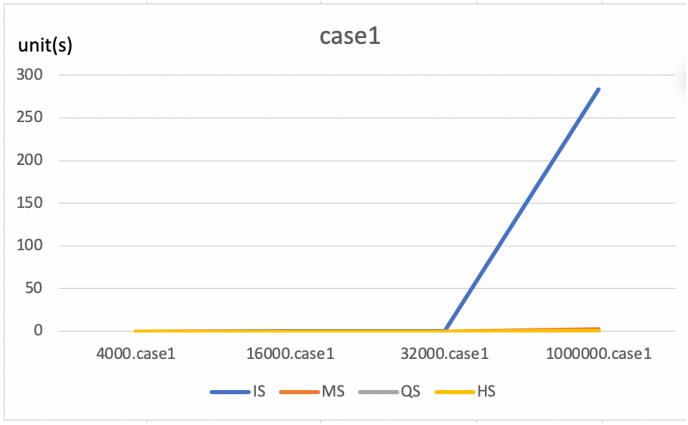
References: https://www.cs.bgu.ac.il/~ds122/wiki.files/Presentation09.pdf

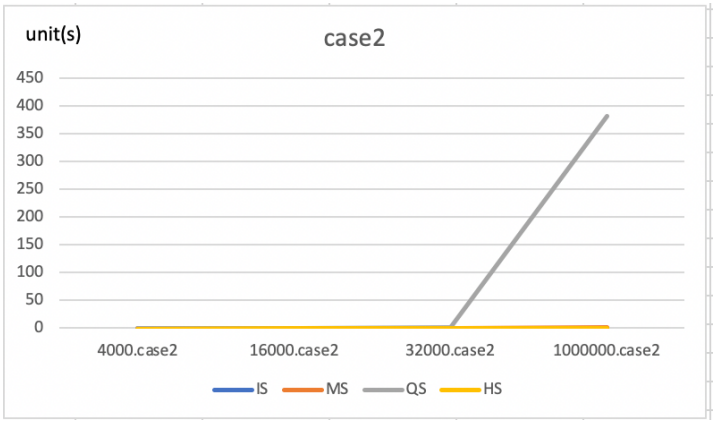
B06102020 外文四 楊晴雯

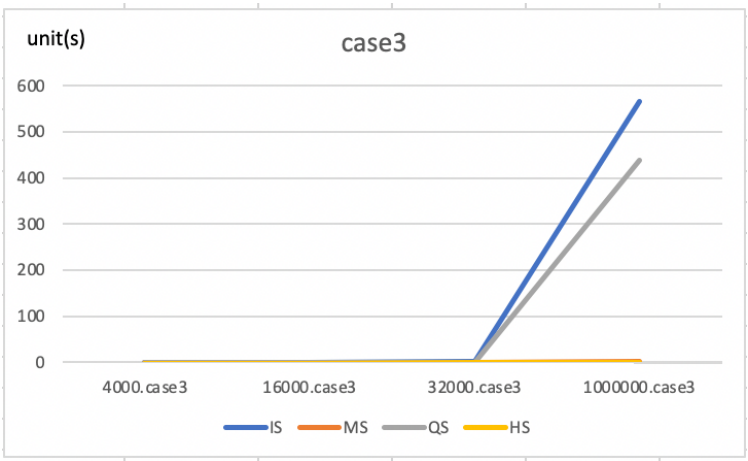
1. Table of runtime and memory usage

Input Size	IS		MS		QS		HS	
	CPU time (s)	Memory(KB)	CPU time (s)	Memory(KB)	CPU time (s)	Memory(KB)	CPU time (s)	Memory(KB)
4000.case2	0	12500	0.001	12424	0.012998	12540	0.001	12424
4000.case1	0.008999	12500	0.002	12424	0.001	12424	0.001	12424
4000.case3	0.014998	12500	0.004	12424	0.016997	12448	0.001	12424
16000.case2	0	12648	0.004	12572	0.177973	13252	0.003	12572
16000.case1	0.122981	12648	0.006	12572	0.002	12572	0.003	12572
16000.case3	0.241964	12648	0.004	12572	0.140979	12868	0.001	12572
32000.case2	0	12648	0.00699	12572	0.629904	13996	0.0399	12572
32000.case1	0.481927	12648	0.00799	12572	0.003	12572	0.0399	12572
32000.case3	0.927859	12648	0.00599	12648	0.544917	13236	0.0399	12572
1000000.case2	0.002999	18668	1.46978	20448	382.432	46092	0.13498	18592
1000000.case1	283.71	18668	2.38964	20448	0.09985	18592	0.200969	18592
1000000.case3	566.858	18668	1.53977	20448	438.845	34628	0.118982	18592

2.







3. Analysis

(The following analysis refers to the four above sorting algorithms with their acronyms: IS: Insertion Sort; MS: Merge Sort; QS: Quick Sort; HS: Heap Sort).

Case1: input in random order; average case.

IS needs $O(n^2)$ in both its average case and worst case, while the others are bounded within $O(nlgn)$.

Case2: input in sorted, ascending order.

For IS, it is the best case and requires $O(\mathrm{n})$.

For MS and HS, they don't get much affected by the way input is ordered and both runs in $O(nlgn)$ time. MS still needs to divide the arrays into n/2, n/2 subarrays and copy all the elements, the only difference is "from which subarray" it copies. For HS, in the $n-1$ times of $deleteMax()$, the last element (leaf node) in the heap is placed at the root then percolated down (to the bottom, in this case so it travels the height of the tree $O(lgn)$) following the rules of Max Heapify. The total running time for HS in this case is still $O(nlgn)$.

For QS (our QS is in earlier version), it always chooses the rightmost element as pivot, and each time the pivot is placed to its original position, which leads to a

$$\mathrm{T(n)} = \mathrm{T(n\text{-}1)} + \mathrm{n\text{-}1}$$

situation (ineffective partitioning) and by extending it we get

$T(n) = T(1) + 1 + 2 + \cdots + (n-1) = T(1)\ (\in \theta(1)) + \mathrm{(n\text{-}1)(n\text{-}2)/2}$

$T(n) \in O(n^2)$.

Case3: input in sorted, descending order.

For IS, it is the worst case and requires $O(n^2)$.

For MS and HS, they don't get much affected to the input distribution as stated above. Running time are both $O(\mathrm{nlgn})$.

For QS, it is the worst case for the same reason, that the partitioning can't work effectively. When the partitioning (almost) always results in n-1 and 0 subarrays

(the omitted 1 is the pivot), QS runs in a $O(n^2)$ runtime.

Extra Notes:

For Case2, one way to optimize MS is to compare the left subarray's last element with the right subarray's first element then decides to $merge()$ or not (this optimized MS can run in $O(\mathrm{n})$).