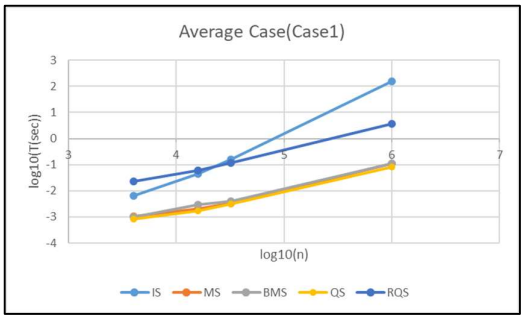
B12901070 電機二 陳世庭 PA1 REPORT

1. Runtime & Complexity

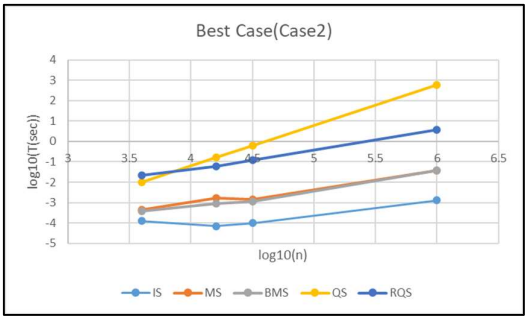
Table 1. Table of CPU time and Memory

input size	IS		MS		BMS		QS		RQS	
	CPU time (s)	Memory (KB)	CPU time (s)	Memory (KB)	CPU time (s)	Memory (KB)	CPU time (s)	Memory (KB)	CPU time (s)	Memory (KB)
4000.case2	0.125m	5904	0.435m	5904	0.365m	5972	9.93m	5968	21.836m	5904
4000.case3	11.994m	5904	0.36m	5904	0.152m	5904	10.423m	5904	20.864m	5904
4000.case1	6.28m	5904	1.048m	5904	1m	5904	0.805m	5904	22.304m	5904
16000.case2	0.069m	6056	1.605m	6056	0.888m	6676	160.183m	6688	61.818m	6056
16000.case3	85.899m	6056	1.656m	6056	0.51m	6312	139.744m	6304	62.865m	6056
16000.case1	43.552m	6056	2.064m	6056	2.945m	6056	1.707m	6056	62.2m	6056
32000.case2	0.1m	6188	1.376m	6188	1.131m	7480	617.935m	7504	118.511m	6188
32000.case3	326.73m	6188	1.594m	6188	1.107m	6744	543.738m	6744	117.984m	6188
32000.case1	163.055m	6188	3.286m	6188	3.912m	6188	3.114m	6188	119.431m	6188
1000000.case2	1.278m	12144	38.064m	13872	37.102m	39592	590552m	56844	3701.24m	12144
1000000.case3	317698m	12144	43.124m	13880	41.19m	28120	366766m	27248	3698.09m	12144
1000000.case1	158311m	12144	111.557m	13880	107.662m	12144	81.258m	12144	3741.77m	12144

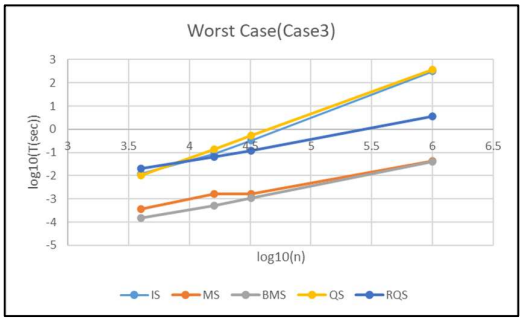
The data were tested on EDA union lab machines-port :40060



Graph1



Graph2



Graph3

Table 2. Slope of log(T)-log(n) graph

	IS	MS	BMS	QS	RQS
Case1	1.8729	0.8773	0.8564	0.8608	0.9438
Case2	0.4934	0.8057	0.8573	1.99	0.945
Case3	1.8805	0.8531	1.0261	1.8975	0.9502

(1) Insertion Sort

According to our textbook, the best-case time complexity of insertion sort is $O(n)$, while the average-case and worst-case complexities are $O(n^2)$.In Table 2, we observe that the slope of average case and the worst case are nearly 2, which is consistent with time complexity $O(n^2)$. However, the slope of the best case is 0.6, making the time complexity of best case $O(n^{0.6})$, which is inconsistent with our textbook. The reason is probably because our number of data is not large enough to make the highest degree term dominant the time complexity. In other words, the time complexity would be affected by constants due to relatively small data size.

(2) Merge Sort & Bottom-Up Merge Sort

According to our textbook, the time complexity of merge sort and bottom-up merge sort is $O(nlogn)$. In Table 2, we observe that the slopes of MS and BMS are close to 1, making the time complexities of MS and BMS $O(n)$. We can observe the following equation.

$$log(nlogn) = log\, n + log\ (logn)$$

The later term varies from 0.556 to 0.778 when the input-size varies from 4000-1000000 (we choose log10 as log), which is quite insignificant. Therefore, the relative input-size in our assignment makes time complexity look like it is $O(n)$, while it actually is $O(nlogn)$.

(3) Quick Sort and Randomized Quick Sort

According to our textbook, the average-case time complexity of quick sort is $O(nlogn)$, while the best-case and worst-case complexities are $O(n^2)$. In Table2, we observe that the slope of the best case and the worst case are nearly 2, which is consistent with time

complexity $O(n^2)$. On the other hand, the slope of other cases for quick sort and randomized quick are 1, while the time complexity is $O(nlogn)$. This is due to the same reason discussed in (2).

2. Comparison between MS and BMS

   The runtime of MS and BMS are barely different, since they both use the "method of divide and conquer" and they both have time complexity $O(nlogn)$. The major difference between merge sort and bottom-up merge sort is that the merge sort uses recursive function calls while the bottom-up merge sort uses iterative loops. The use of recursive function calls may lead to a deeper call stack for larger arrays. Therefore, the BMS would be more efficient for large arrays due to CPU cache.

3. Comparison between QS and RQS

   The runtime of RQS is much better than QS in best case and worst case since randomized selection of pivots avoid inefficient pivots in nearly sorted case. That is, it avoids constant selection of the largest or the smallest number as pivot. However, when considering average case, the runtime of QS is better than RQS since the generation of random number takes some time.

   ps. That's probably because I used mt19937 to generate random numbers.

4. Data Structure used and other findings in this assignment.

(1) Data Structure

We used vector instead of array to use many convenient manipulations such as using "data.size()" to get the input size.

(2) Constant matters

I compared my classmate's RQS with mine and found out that the runtime of my classmate's RQS is better than mine. The reason is that the way of generation of random numbers is different. This reminds me although the algorithm is the same, the difference in coding would significantly affects runtime. (My classmate used rand() instead of mt19937).

(3) Call by reference

As shown in homework 1, how we access data is sometimes critical. In PA1, we used "call by reference" to directly access the data instead of making a copy of data.

(4) Sentinel

In the lecture, we often assume sentinel is ∞. However, we cannot write infinity in our code. Therefore, we used the largest integer 2147483647 as our sentinel. (The datatype we used was "int")