Boran Yıldırım
21401947
CS202-01

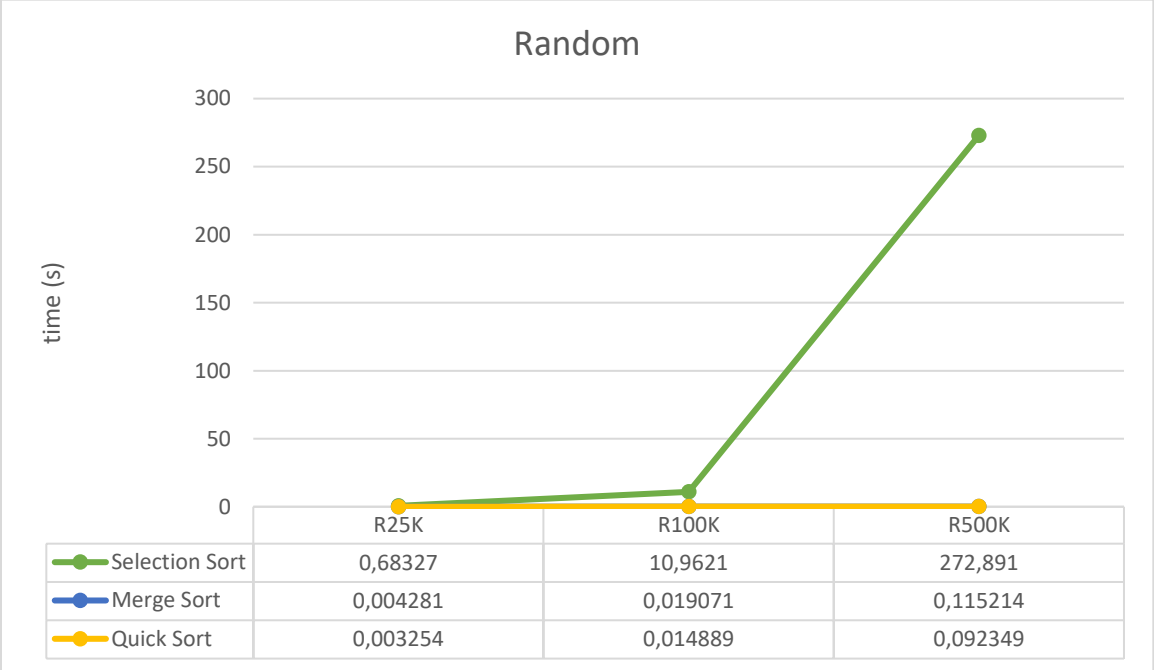
Summary

Selection Sort, Merge Sort and Quick Sort algorithms were tested with different datas and different sizes. Merge Sort performed $O(n \log n)$ and Selection Sort performed as $O(n^2)$ in all experiments. Quick Sort is performed both $O(n \log n)$ and $O(n^2)$. The reason of differences in efficiency of Quick Sort is partition, in ascending and descending order datas one side of parts has 1 integer and the other side has $(n-1)$ integers. This problem directly affect the efficiency of Quick Sort. However, ascending order and descending order datas may not be considered as real world problem as randomly distributed datas and the performance of Quick Sort is very well in randomly distributed datas. If the input is unknown and the time is very important, Merge Sort can be used without any risks.

Detailed Anlyze starts from page 2.

Boran Yıldırım
21401947
CS202-01

Detailed Analyze



	R25K	R100K	R500K
Selection Sort	0,68327	10,9621	272,891
Merge Sort	0,004281	0,019071	0,115214
Quick Sort	0,003254	0,014889	0,092349

Sorting algorithms work to sort in descending order. Selection sort is very inefficient for comparing with Quick and Merge Sort so below only talked about Quick and Merge Sort algorithms.

R25K represents 25000 random numbers from 1 to 25000. The construction of 3 arrays take 0.00037 seconds. Each comparison is enumarated and the results are 312723074, 726325 and 682853 respectively. Quick Sort is 0.0011 seconds faster than Merge Sort and Merge Sort makes 43472 more comparison than Quick Sort. Therefore, Quick Sort is the winner of Random 25K with both time and compare efficiency.

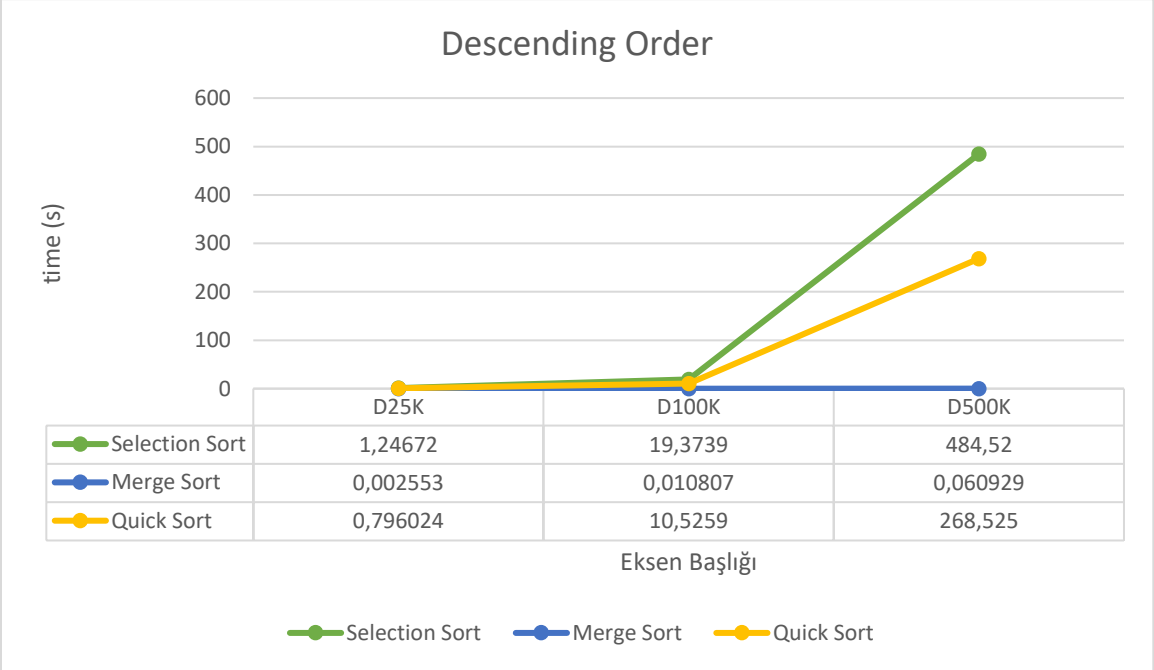
R100K represents 100000 random numbers from 1 to 100000. The construction of 3 arrays take 0.00147 seconds which is 0.0011 seconds more the array construction of R25K. Each comparison enumarated and the results 5000993811, 3305228 and 3205927 respectively. If the results are compared with R25K; comparison count of Selection Sort 4688270737 increased, comparison count of Merge Sort 2578903 increased and finally comparison time of Quick Sort 2523074. Quick Sort is 0.005 seconds faster than Merge Sort and Merge Sort makes 99301 more comparison than Quick Sort. Thus, Quick Sort is the winner of random 100K with both time and compare efficiency.

R500K represents 500000 random numbers from 1 to 500000. The construction of 3 arrays take 0.006445 seconds which is 0.004975 seconds more than R100K. Each comparison is enumarated and the results are 125006512734, 18812797 and 19471924 respectively. If the results are compared with R100K; comparison count of Selection Sort 120005519523 increased, comparison count of Merge Sort 15507539 increased and finally comparison time of Quick Sort 16265997 increased. Quick

Boran Yıldırım
21401947
CS202-01

Sort is 0.022865 seconds faster than Merge Sort but Quick Sort makes 659127 more comparison than Merge Sort. Thus, Quick Sort is the winner of random 500K with time efficiency, but Merge Sort is the winner with comparison efficiency.

In conclusion, altough size of datas are huge numbers, Quick Sort and Merge Sort performed very well. However, time can be considered as money so Quick Sort should be used rather than Merge Sort if datas are randomly distributed and have huge sizes.



Sorting algorithms work to sort in descending order so the array input is same as output in this experiment. Selection and Quick Sort is very inefficient for comparing with Merge Sort. However, Quick Sort is better than Selection Sort so below only talked about Quick and Merge Sort.

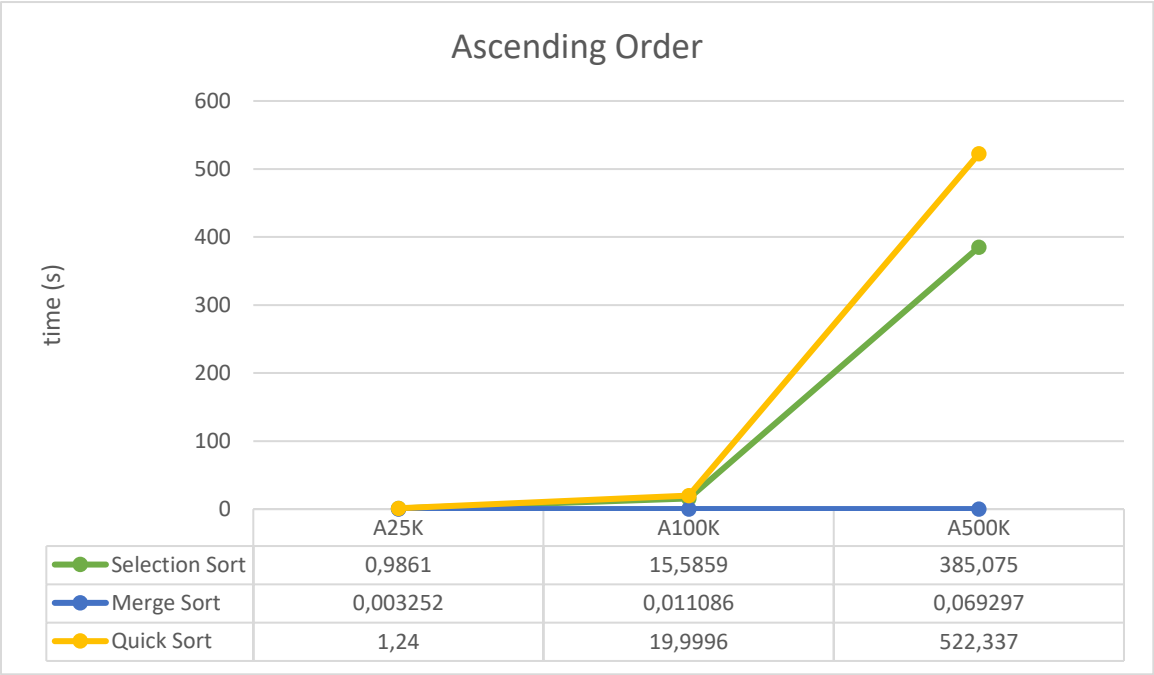
D25K represents 25000 decreasing order numbers from 25000 to 1. The construction of 3 arrays take 0.000235 seconds. Each comparison is enumarated and the results are 624999999, 580707 and 312512499 respectively. Merge Sort is 0.793471 seconds faster than Quick Sort and Quick Sort makes 311931792 more comparison than Merge Sort. Therefore, Merge Sort is the winner of decresing order 25K with both time and compare efficiency.

D100K represents 100000 decreasing order numbers from 100000 to 1. The construction of 3 arrays take 0.000933 seconds which is 0.000698 seconds more the array construction of D25K. Each comparison enumarated and the results 1410065407, 2622831 and 705082703. If the results are compared with D25K; comparison count of Selection Sort 785065408 increased, comparison count of Merge Sort 2042124 increased and finally comparison time of Quick Sort 392570204 increased. Merge Sort is 10.515093 seconds faster than Quick Sort and Quick Sort makes 702459873 more

comparison than Merge Sort. Thus, Merge Sort is the exact winner of decresing order 100K with both time and compare efficiency.

D500K represents 500000 decreasing order numbers from 500000 to 1. The construction of 3 arrays take 0.004023 seconds which is 0.00309 seconds more than D100K. Each comparison is enumarated and the results are 249999999999, 14758927 and 125000249999. If the results are compared with D100K; comparison count of Selection Sort 248589934592 increased, comparison count of Merge Sort 12136096 increased and finally comparison time of Quick Sort 124295167296. Merge Sort is 268.464071 seconds faster than Quick Sort and Quick Sort makes 124985491072 more comparison than Merge Sort. Thus, Merge Sort is certainly the winner of decreasing order 500K with both time and compare efficiency.

In conclusion, if the data is sorted like in this experiment, Quick Sort acts as $O(n^2)$ rather than $O(n \log n)$. Therefore, Merge Sort performed exactly better than Quick and Selection Sort.



	A25K	A100K	A500K
Selection Sort	0,9861	15,5859	385,075
Merge Sort	0,003252	0,011086	0,069297
Quick Sort	1,24	19,9996	522,337

Sorting algorithms work to sort in descending order so the array input is inverted in this experiment. Quick and Selection Sort algorithms are again very inefficient for comparing with Merge Sort. However, Selection Sort is better than Quick Sort so that Selection and Merge Sort algorithms are compared below.

A25K represents 25000 increasing order numbers from 1 to 25000. The construction of 3 arrays take 0.000249 seconds. Each comparison is enumarated and the results are 468749999, 570987 and 468762499 respectively. Merge Sort is 1.236748 seconds faster than Selection Sort and Selection Sort makes 468191512 more comparison than Merge Sort. Therefore, Merge Sort is the winner of ascending order 25K with both time and compare efficiency.

Boran Yıldırım
21401947
CS202-01

A100K represents 100000 decreasing order numbers from 1 to 100000. The construction of 3 arrays take 0.001083 seconds which is 0.000834 seconds more the array construction of D25K. Each comparison enumarated and the results 7499999999, 2012964 and 7500049999. If the results are compared with A25K; comparison count of Selection Sort 7031250000 increased, comparison count of Merge Sort 2042124 increased and finally comparison time of Quick Sort 7031287500 increased. Merge Sort is 15.577814 seconds faster than Selection Sort and Selection Sort makes 7497416048 more comparison than Merge Sort. Thus, Merge Sort is the exact winner of ascending order 100K with both time and compare efficiency.

A500K represents 500000 decreasing order numbers from 1 to 500000. The construction of 3 arrays take 0.004745 seconds which is 0.003662 seconds more than A100K. Each comparison is enumarated and the results are 187499999999, 14668207 and 187500249999. If the results are compared with A100K; comparison count of Selection Sort 180000000000 increased, comparison count of Merge Sort 12084256 increased and finally comparison time of Quick Sort 180000200000 increased. Merge Sort is 385.005703 seconds faster than Selection Sort and Selection Sort makes 187485331792 more comparison than Merge Sort. Thus, Merge Sort is definitely the winner of random 500K with both time and compare efficiency.

In conclusion, if the data is sorted like in this experiment, Quick Sort acts as $O(n^2)$ rather than $O(n \log n)$. Therefore, Merge Sort performed exactly better than Quick and Selection Sort.