CMPUT 379

Assignment 3

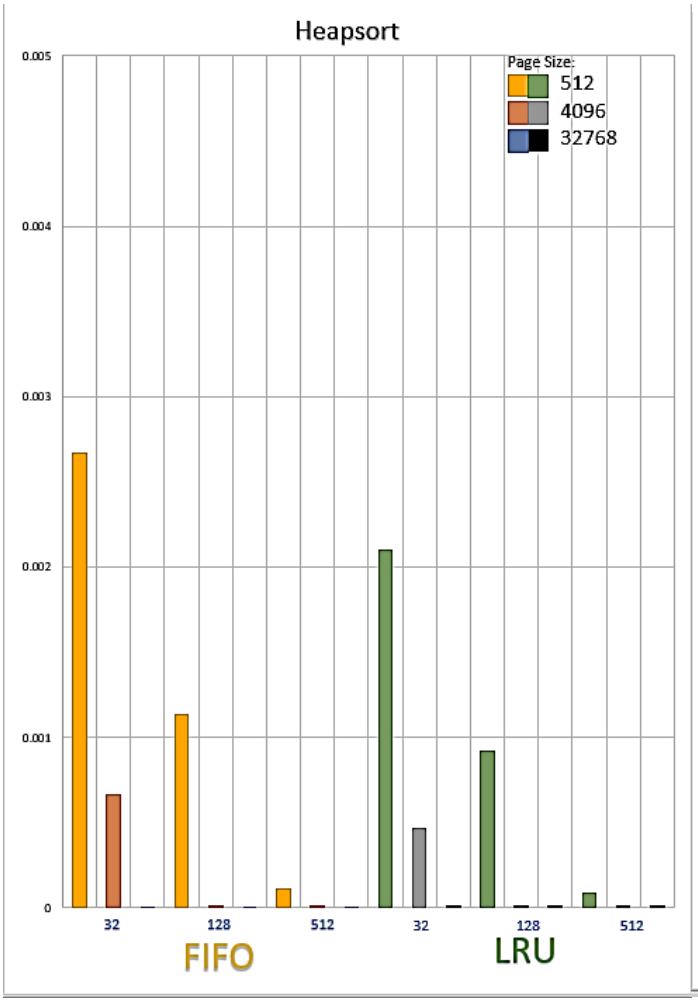
For this assignment, we studied TLB behavior through building a TLB simulator using two policies: FIFO and LRU to exam the output of valgrind for different programs and different sizes of instances.

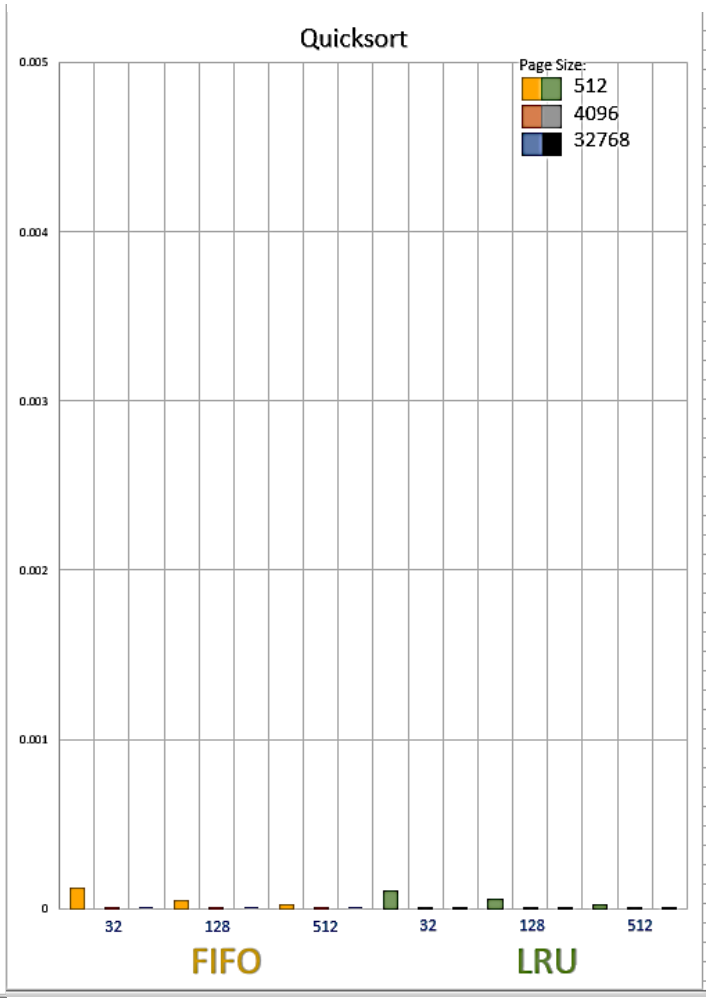
Through running our TLB simulator we can get 162 groups of data. We consider three sorting program, two policies, three instance sizes, three page-sizes, and three tlb-sizes as the variables. No flush period, no -i option here. We collected the result and plot them in following bar charts.

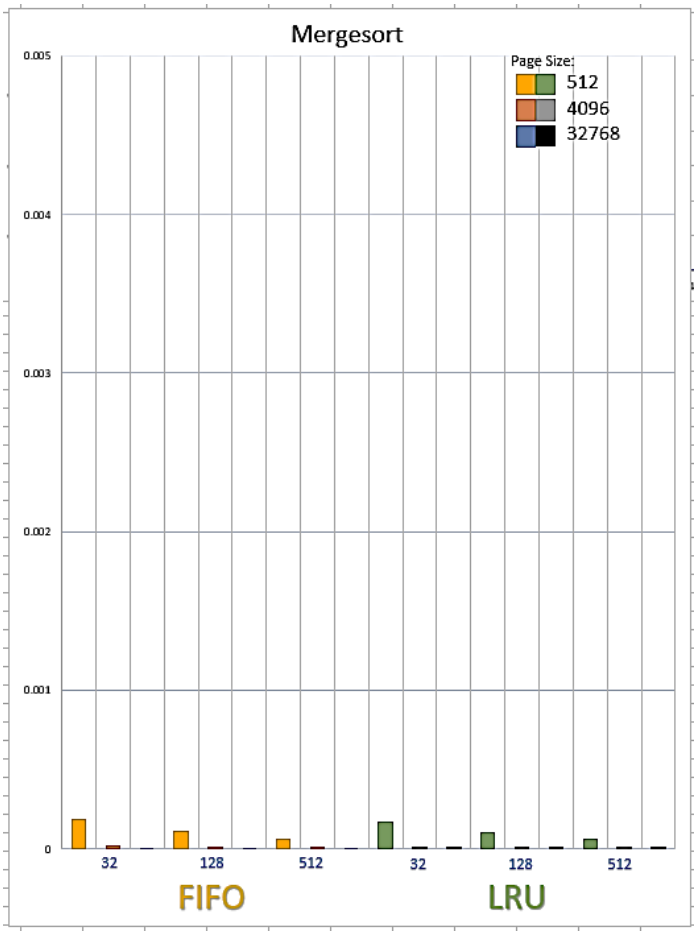
The y value of the graphs is miss rate and we used the same y-scale for each graph and the bars are divided in groups by different policies, page size and tlb size, representing on x-axis

The bar charts are posted below:

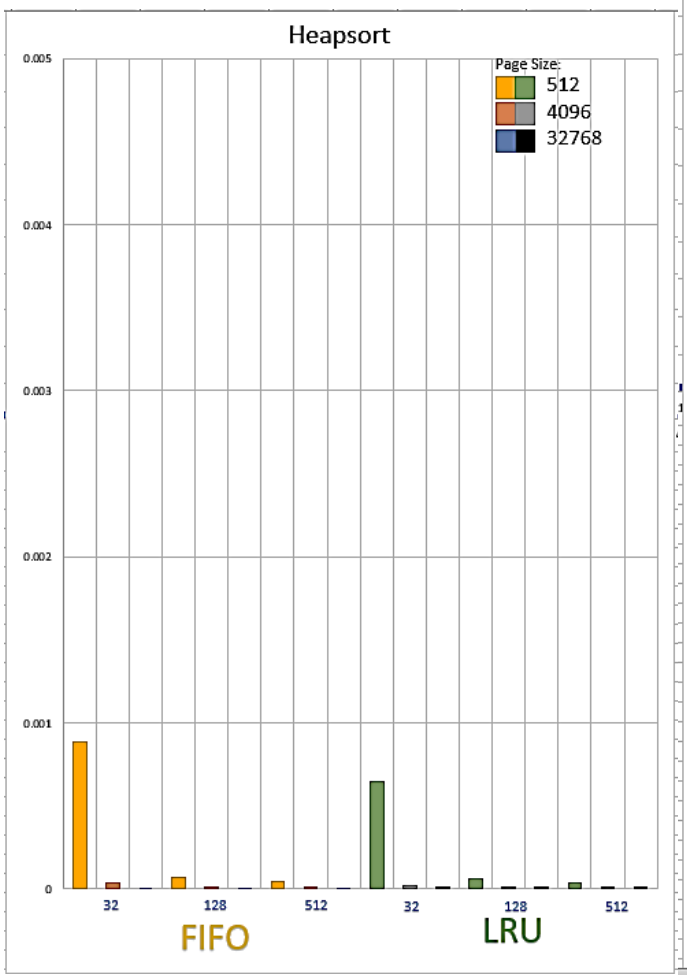
For large instance size (100000):

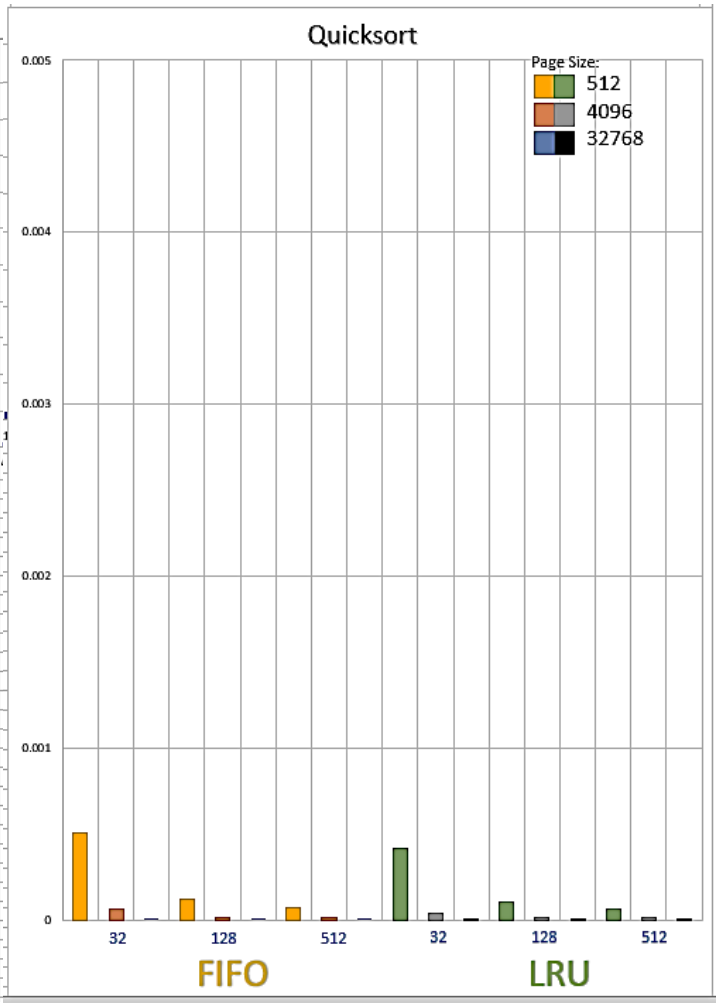


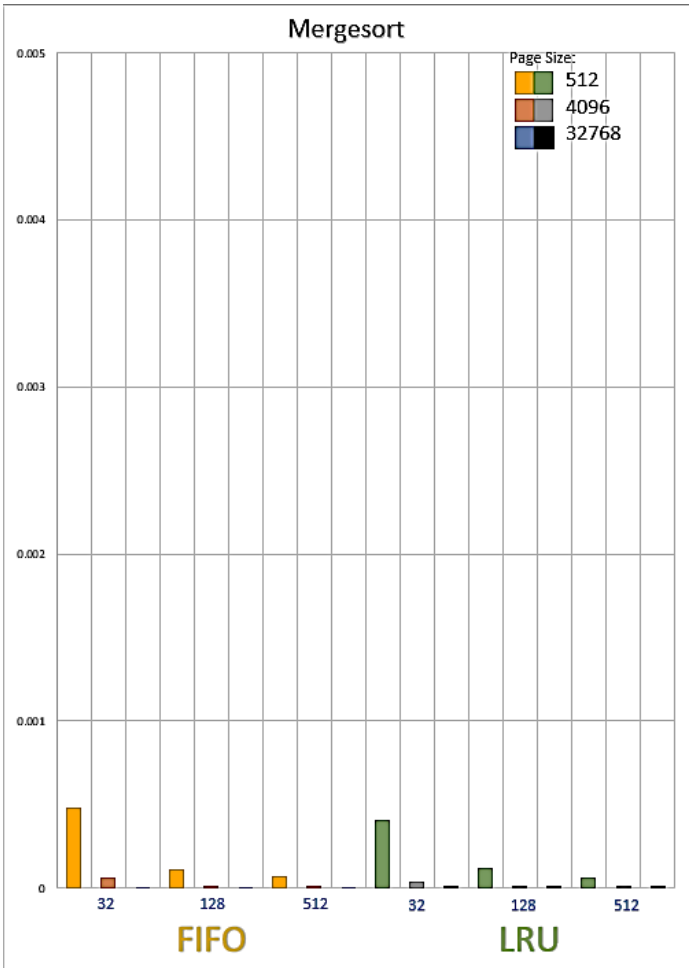




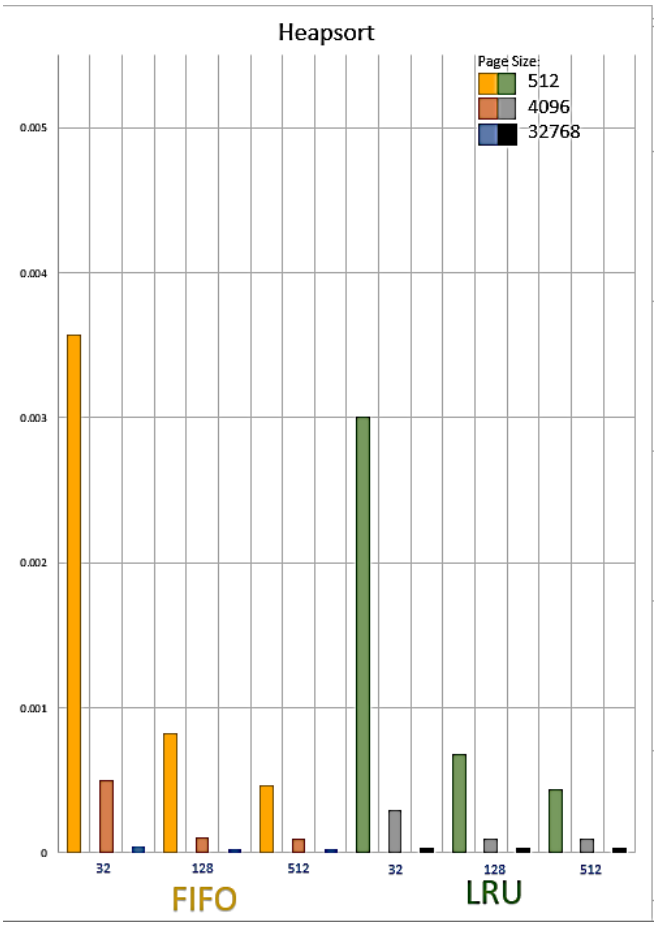
For medium instance size (10000):

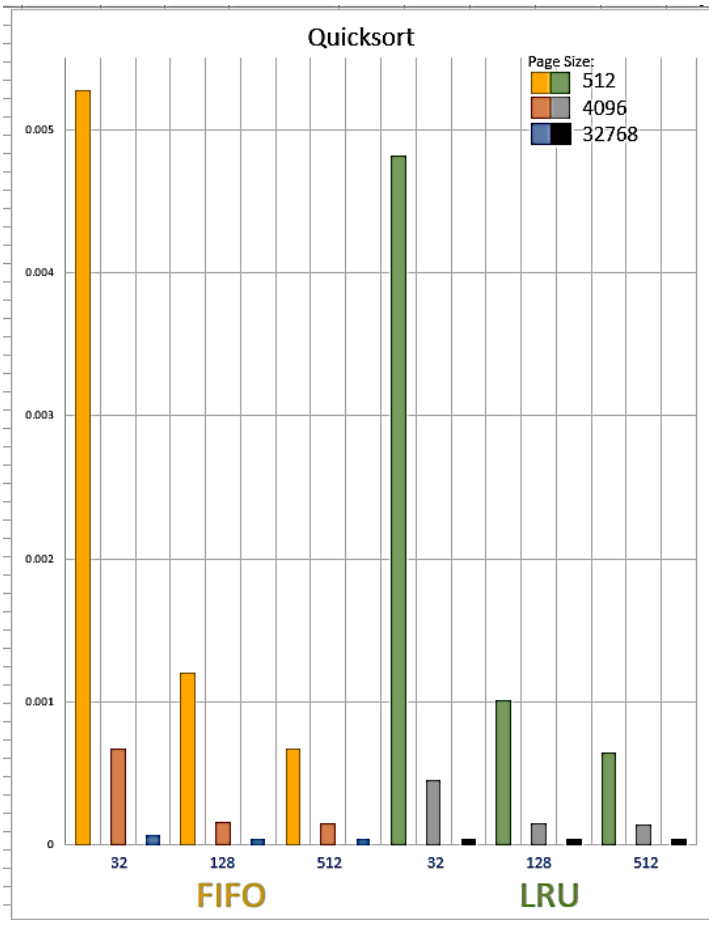


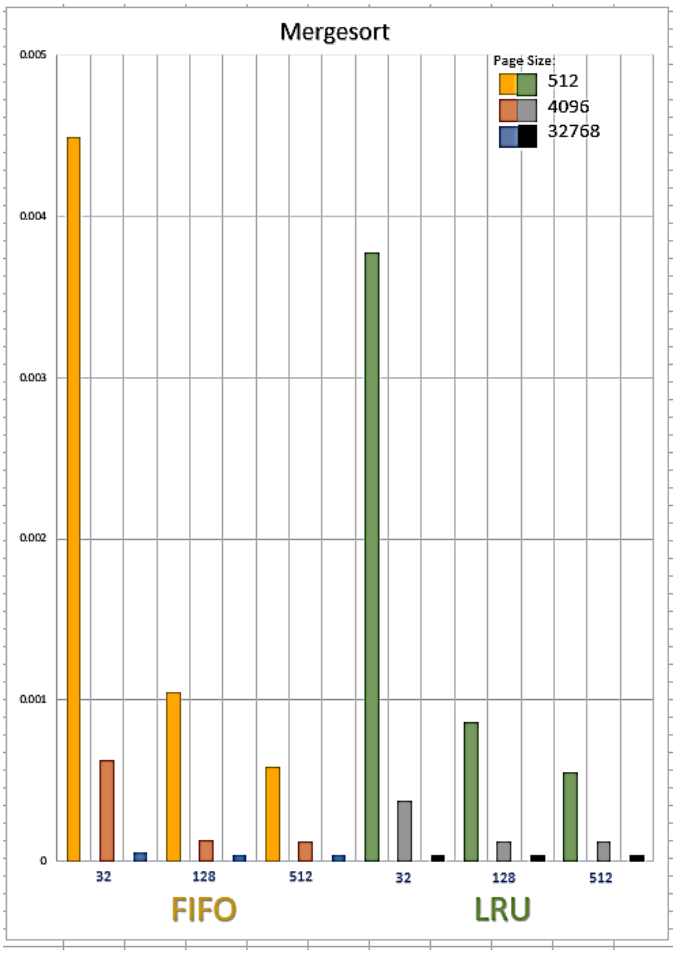




For small instance size (1000):

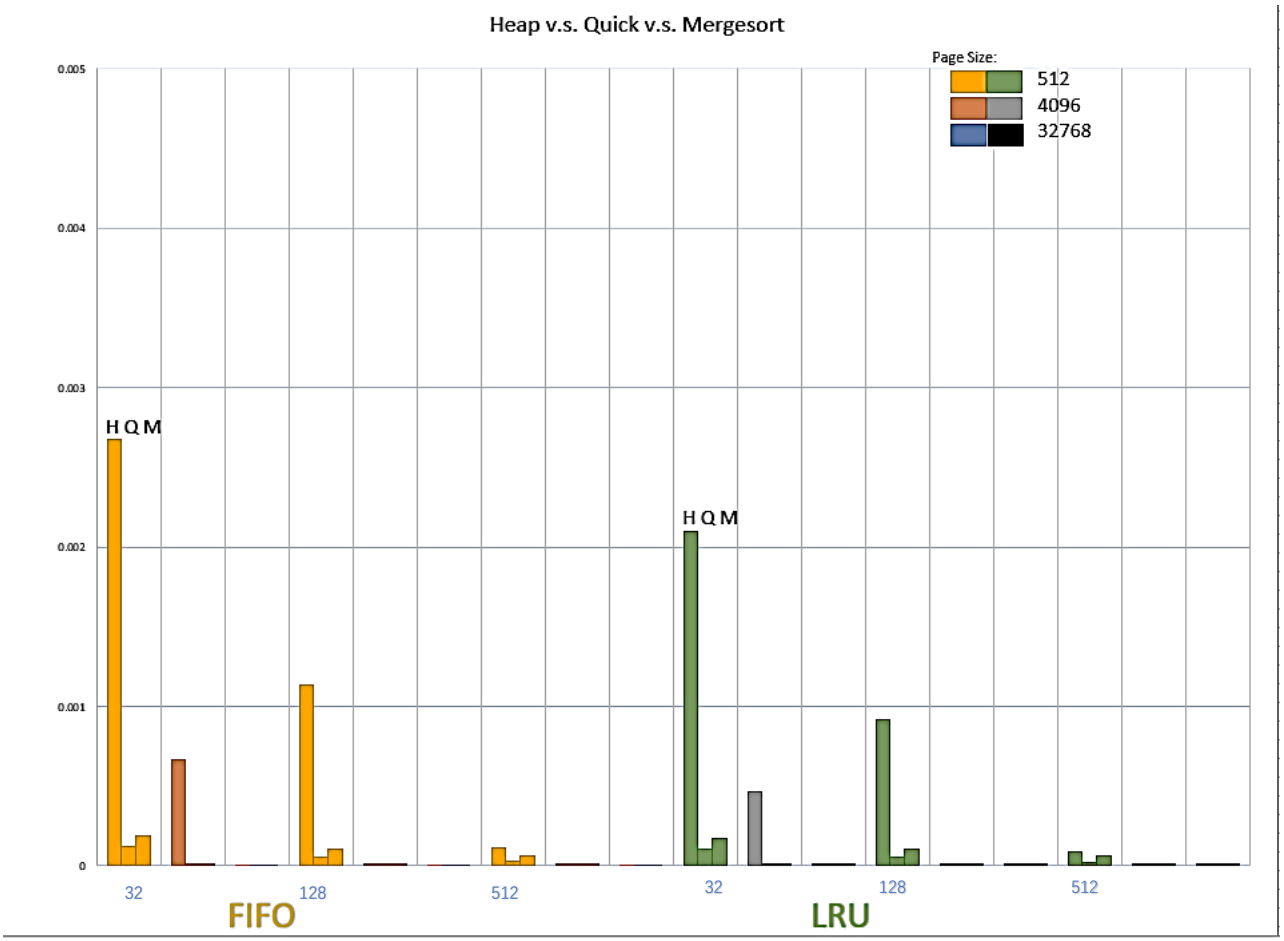




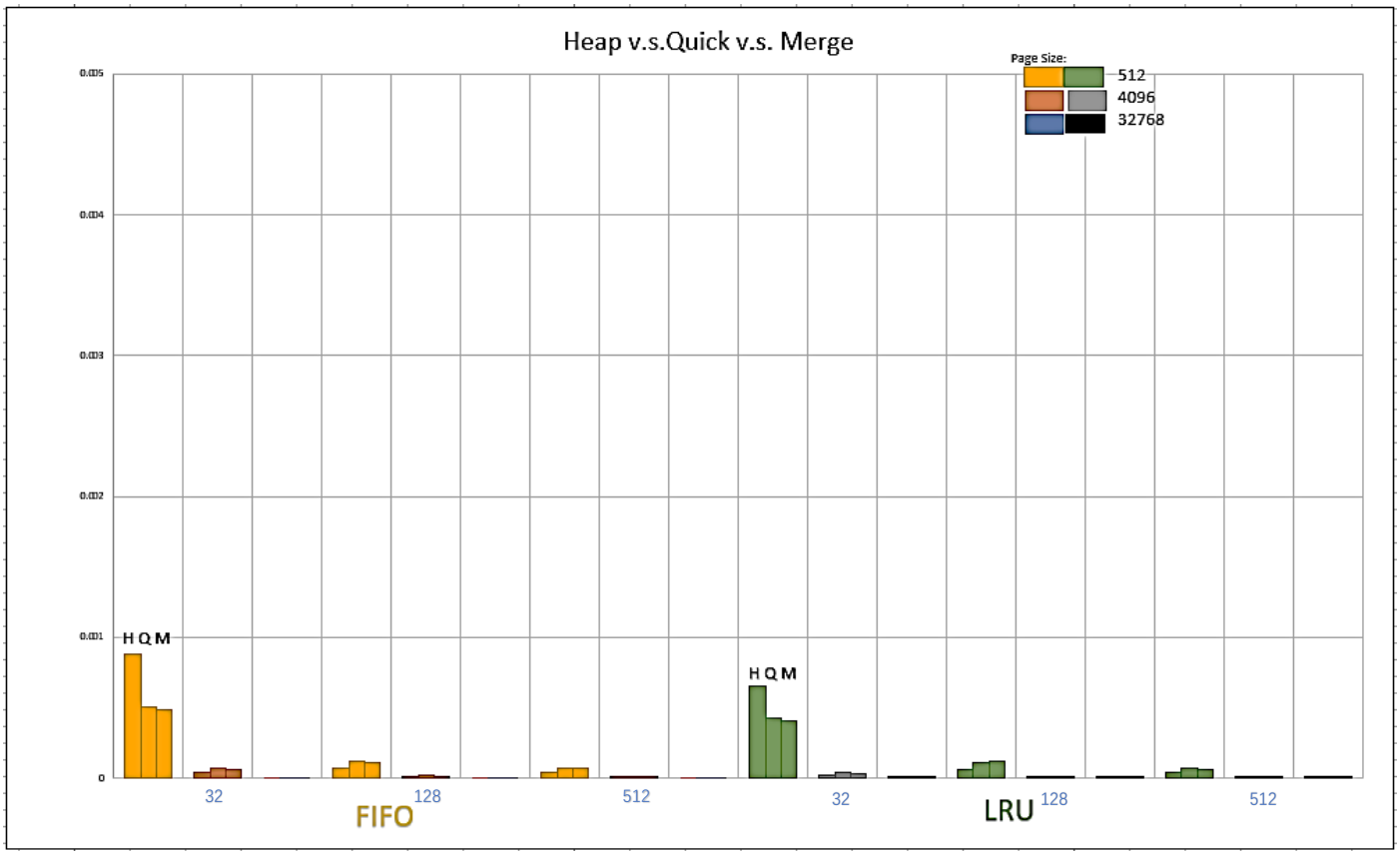


Then, we put the data of different sorting in the same chart to make comparasion:

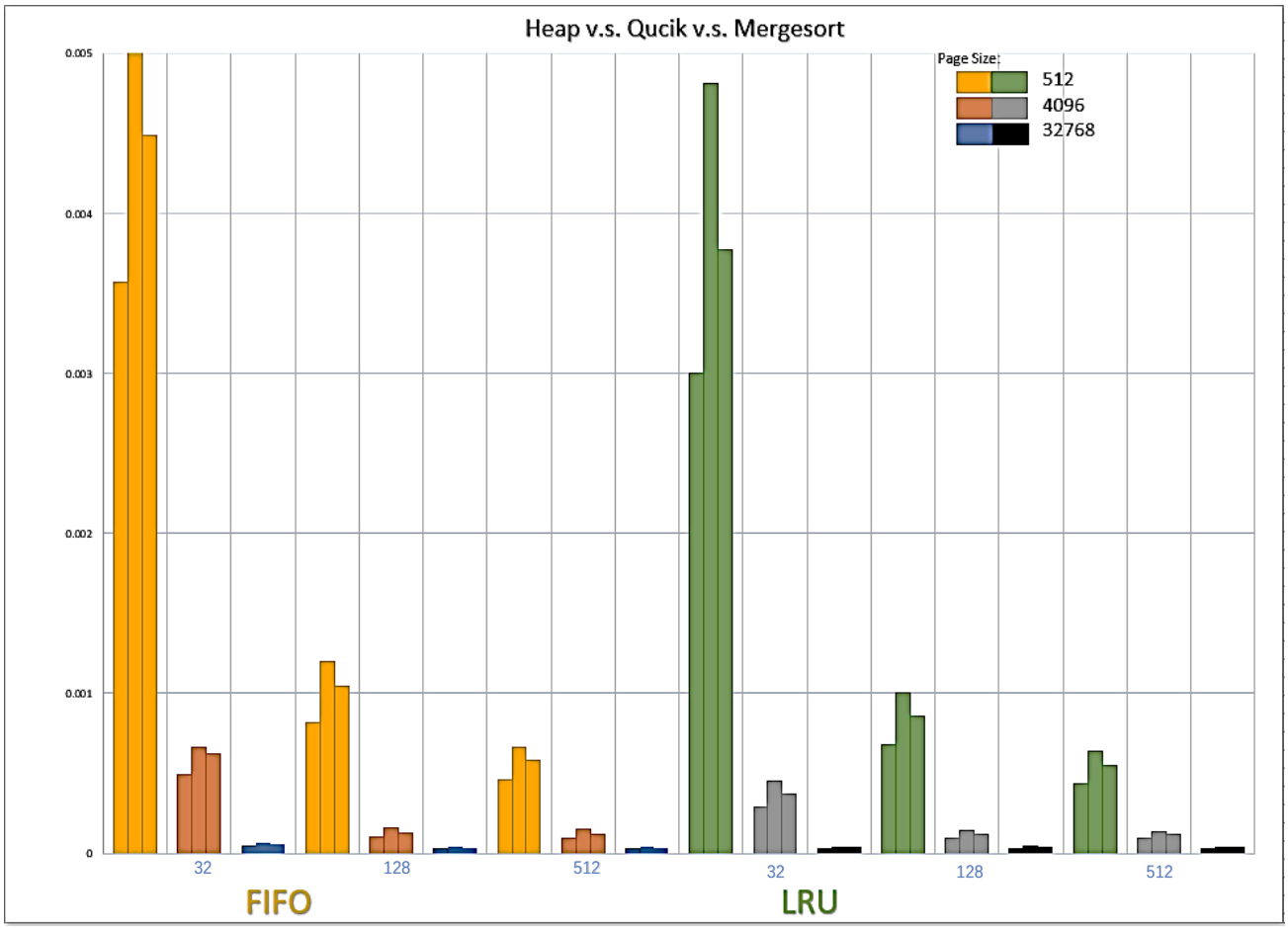
For large instance size (100000):



For medium instance size (10000):



For small instance size (1000):

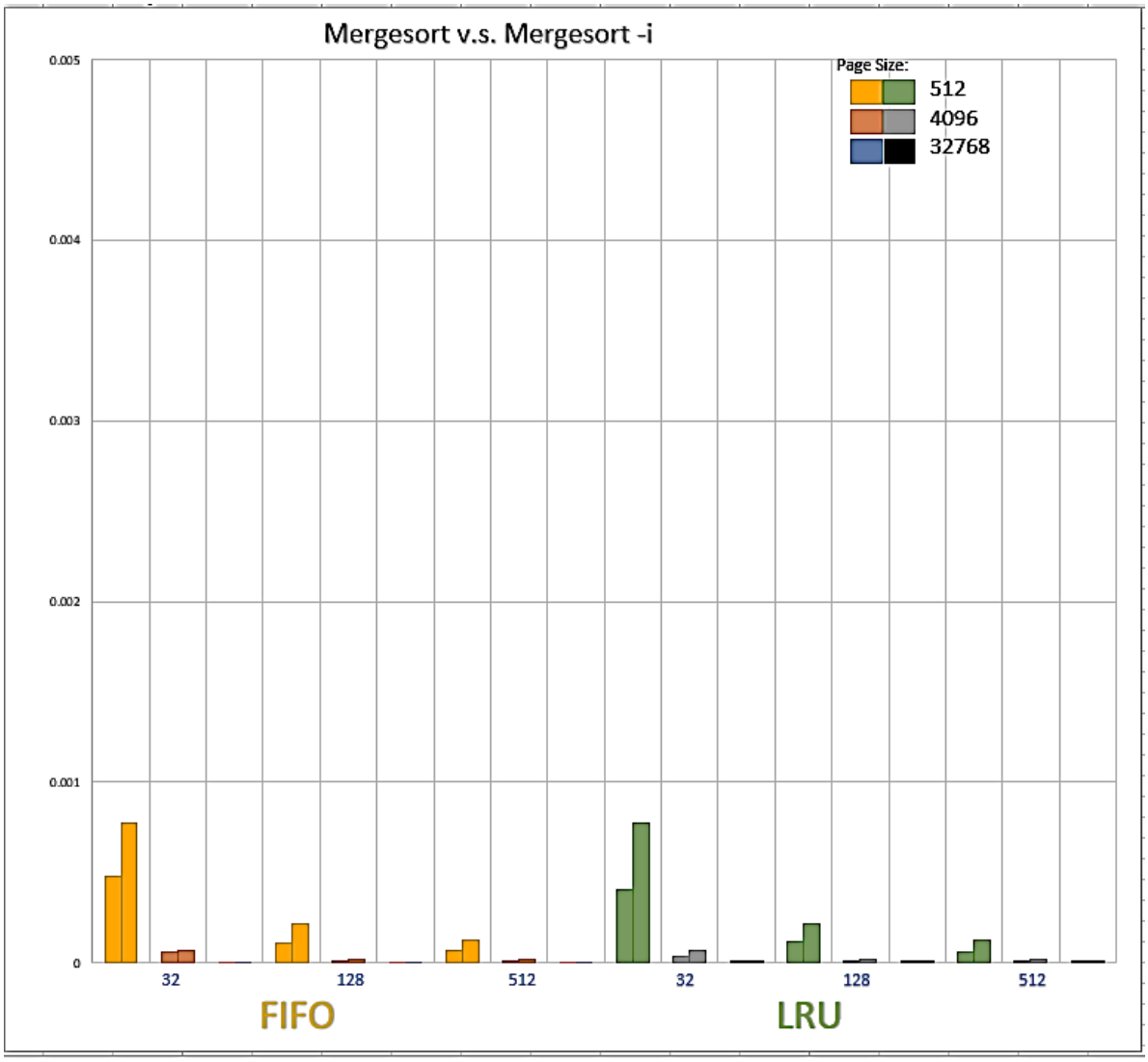


From the comparing graphs above, firstly we can find LRU is the policy which is most suitable for TLB. The miss rate of LRU for different groups are all lower than that of FIFO. The reason for that is the mechanism of LRU is similar with TLB: storing the address used most often to improve the efficiency. If some addresses are used repeatedly, LRU can highly reduce the miss rate. Besides, larger page size and larger tlb size will also improve the TLB simulator's performance through reducing the miss rate, since they can store more address to increase the rate of hitting. Larger page size can reduce the miss rate more efficiently than larger tlb size. When the tlb size is increased to certain extent, such as from 128 to 512, the improvement is not as effective as the extending from 32 to 128 since the storage of TLB is close to store all the amount of address used at 128, and 512 will not make too much change.

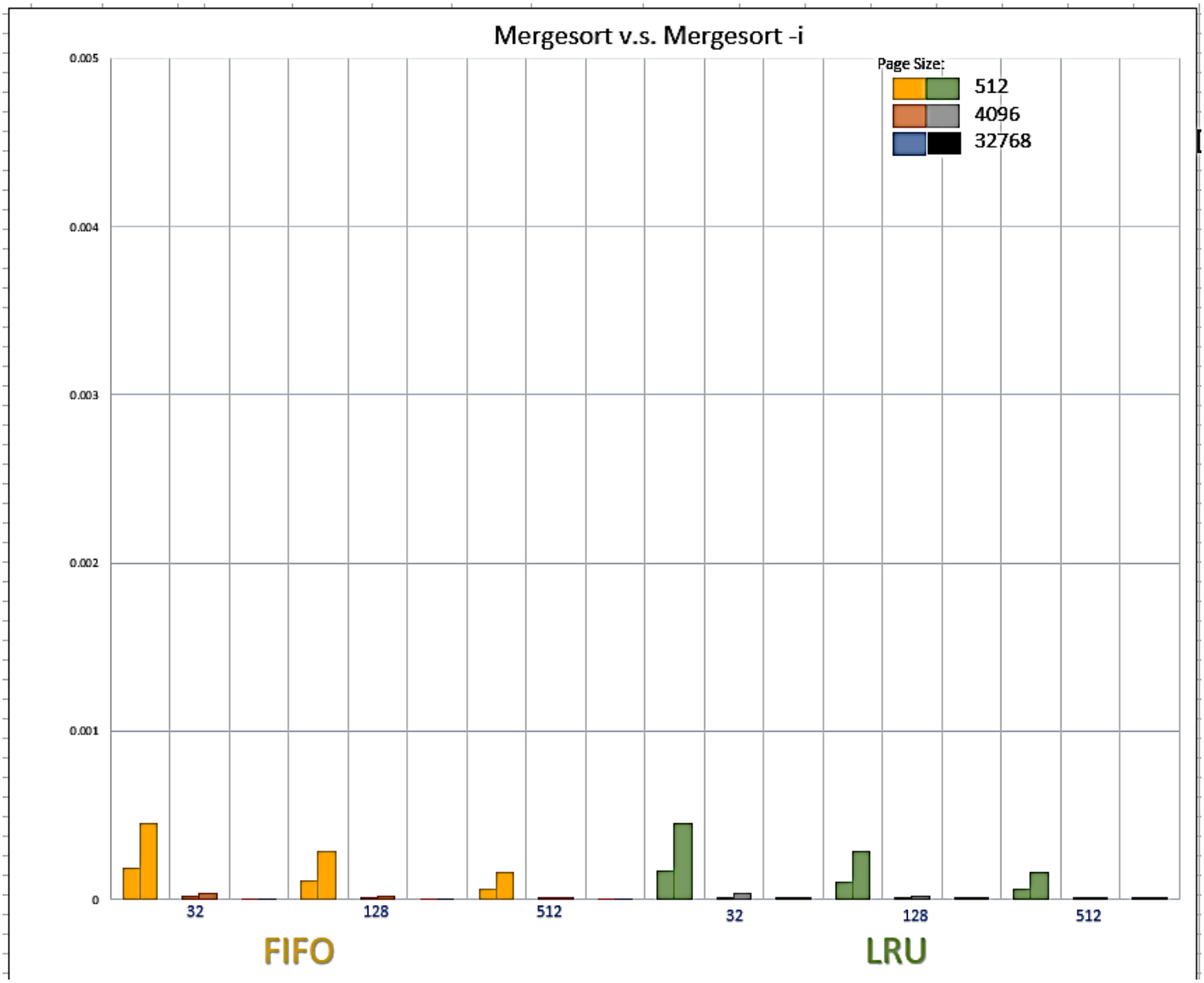
For different sorting methods, heap sort is the most unstable sorting program among three. It also needs much more memory than quick sort and merge sort. Based on its way processing, it needs to reference much more addresses thus it increases the rate of missing. Among these three programs, merge sorting is the most stable sorting program and it doesn't need extra memory or reference extra addresses and this is the reason why the miss rate of merge sort is lower than others.

Now we consider the situation of adding -i option, the comparing graph is post below.

For Merge-sorting, instance size 10000, comparing the result without "-i" and with "-i":



For Merge-sorting, instance size 100000, comparing the result without "-i" and with "-i":



If we add the -i option, the program will read input thus the amount of reference address is reduced, but the amount of miss is kept, thus the miss rate will increase.