Blaz Pocrnja
Shawn Sydia

Heapsort Results

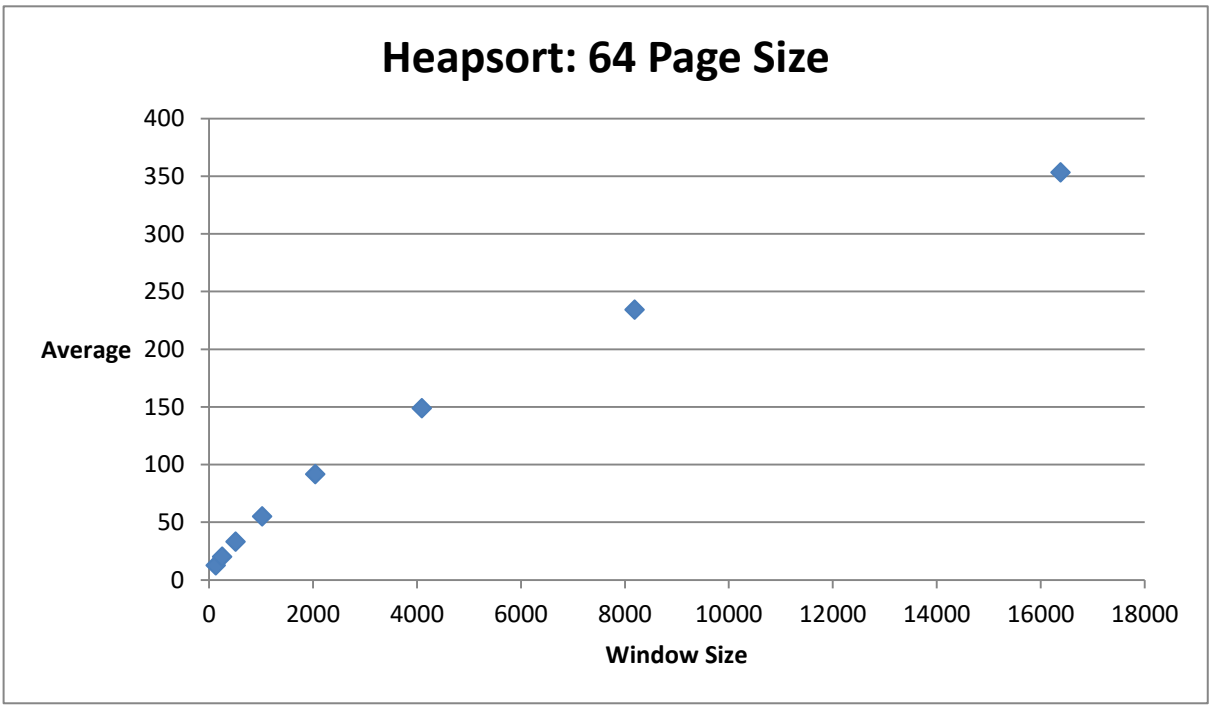

Figure 1.1

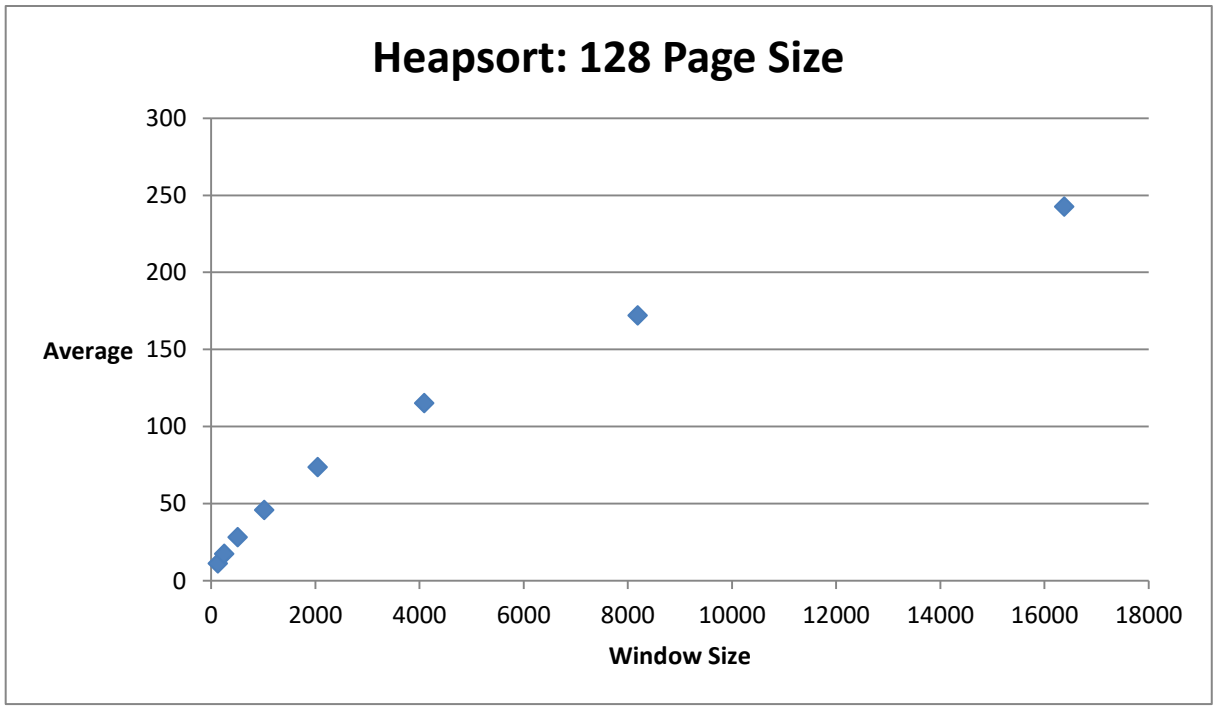

Figure 1.2

Blaz Pocrnja
Shawn Sydia

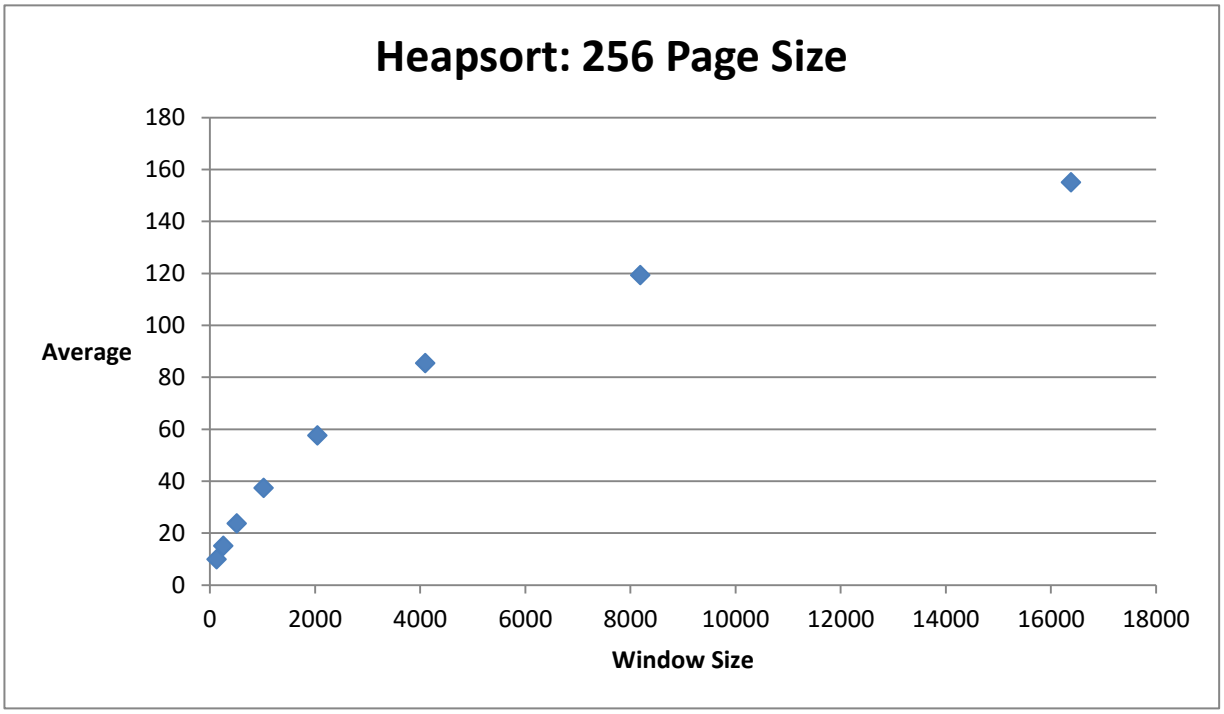

Figure 1.3

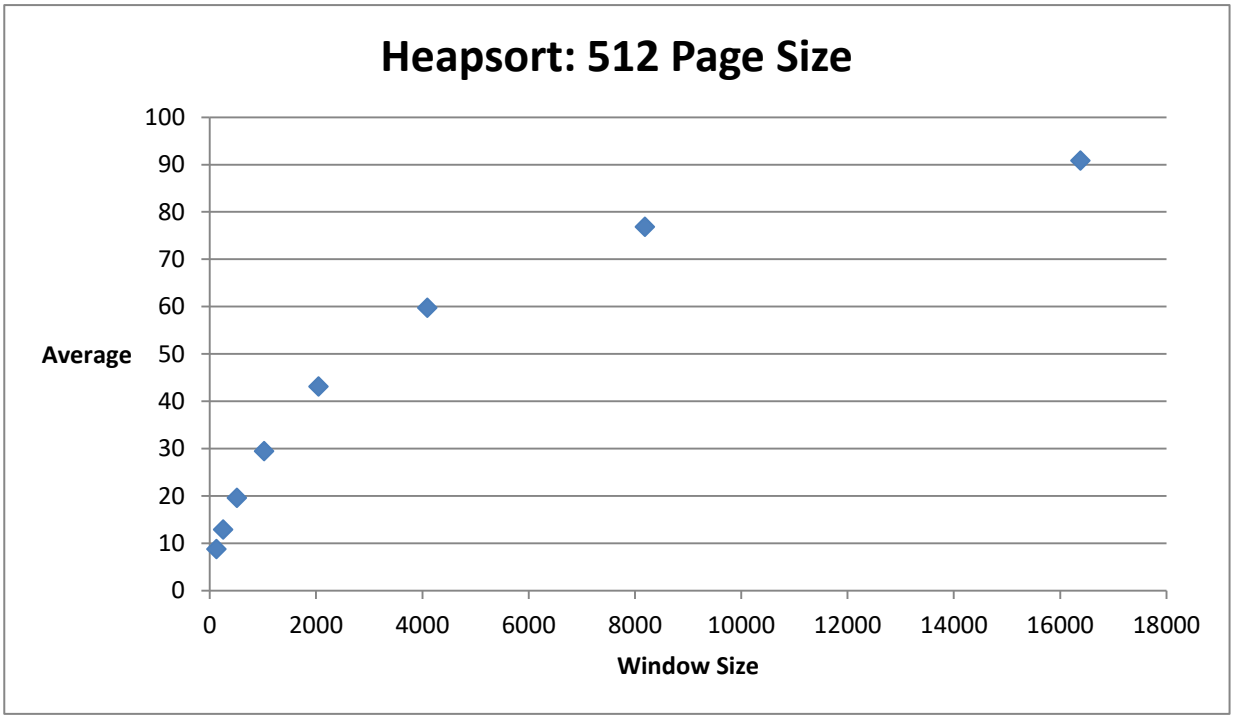

Figure 1.4

Blaz Pocrnja
Shawn Sydia

Quicksort Results

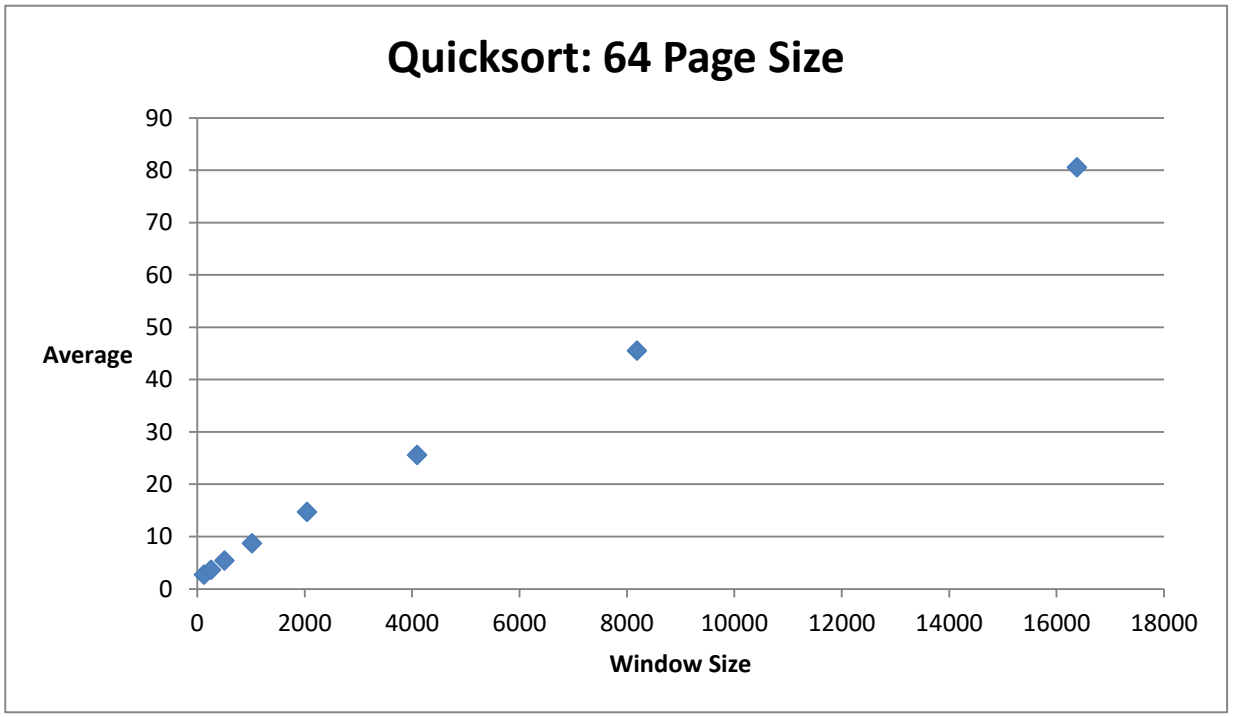

Figure 2.1

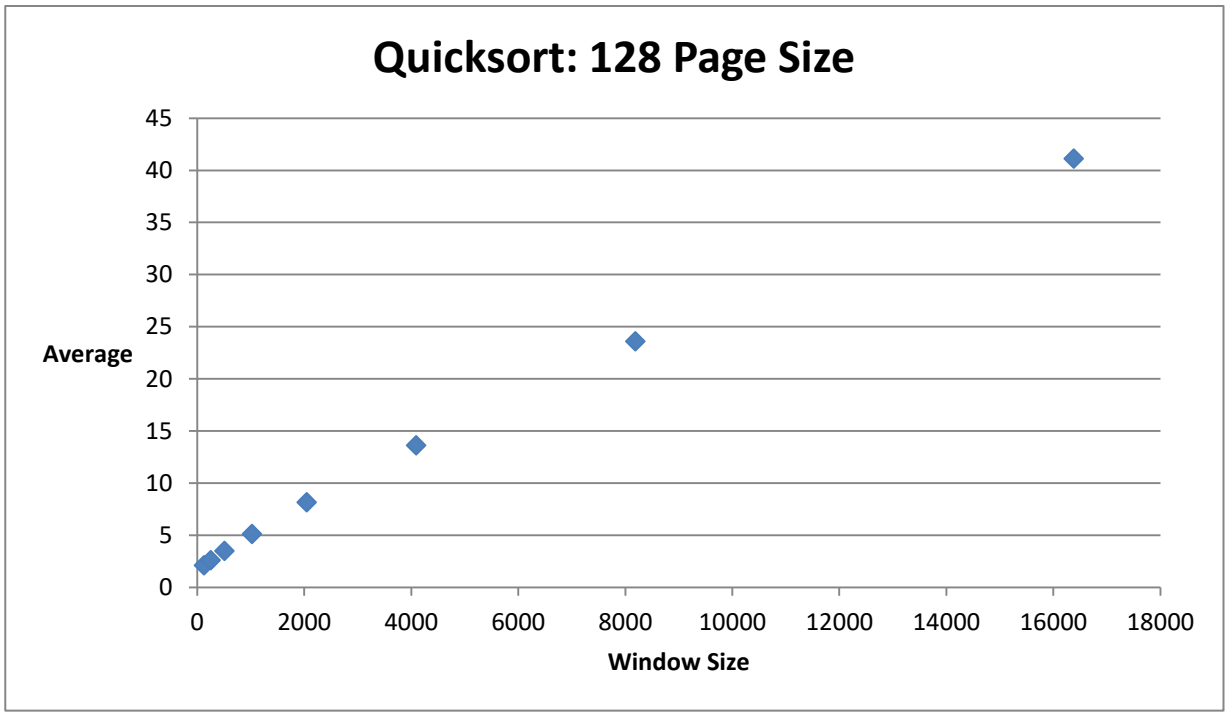

Figure 2.2

Blaz Pocrnja
Shawn Sydia

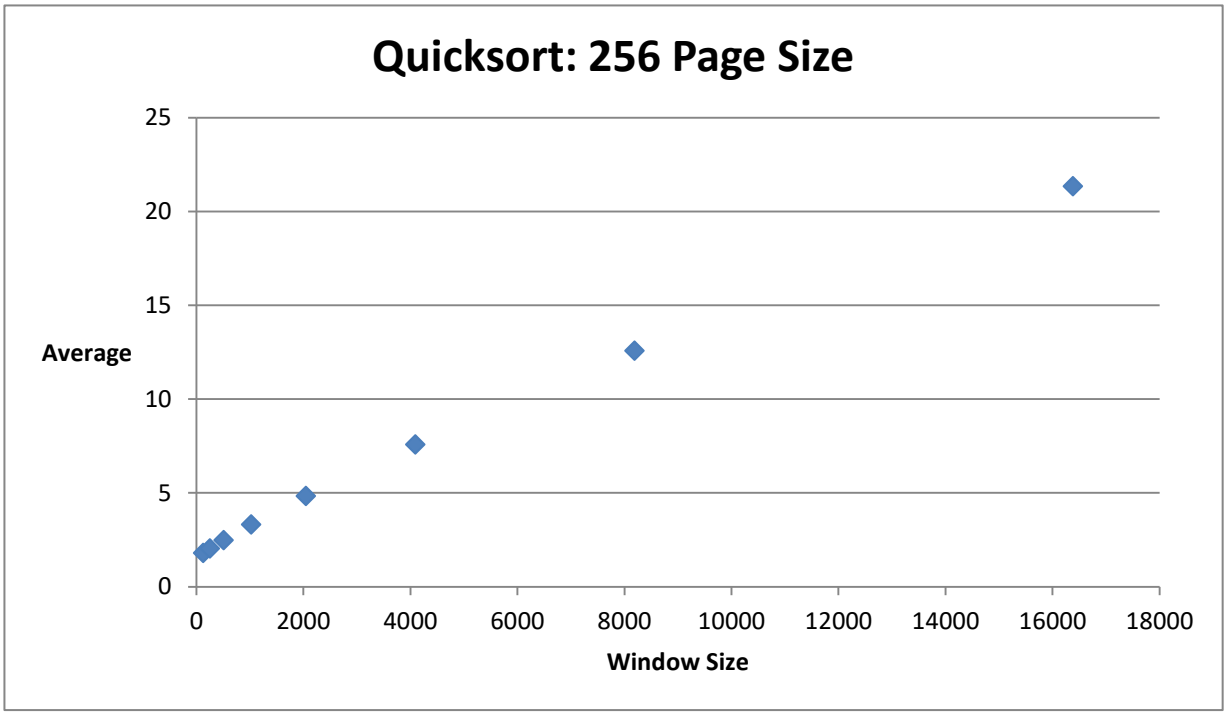

Figure 2.3

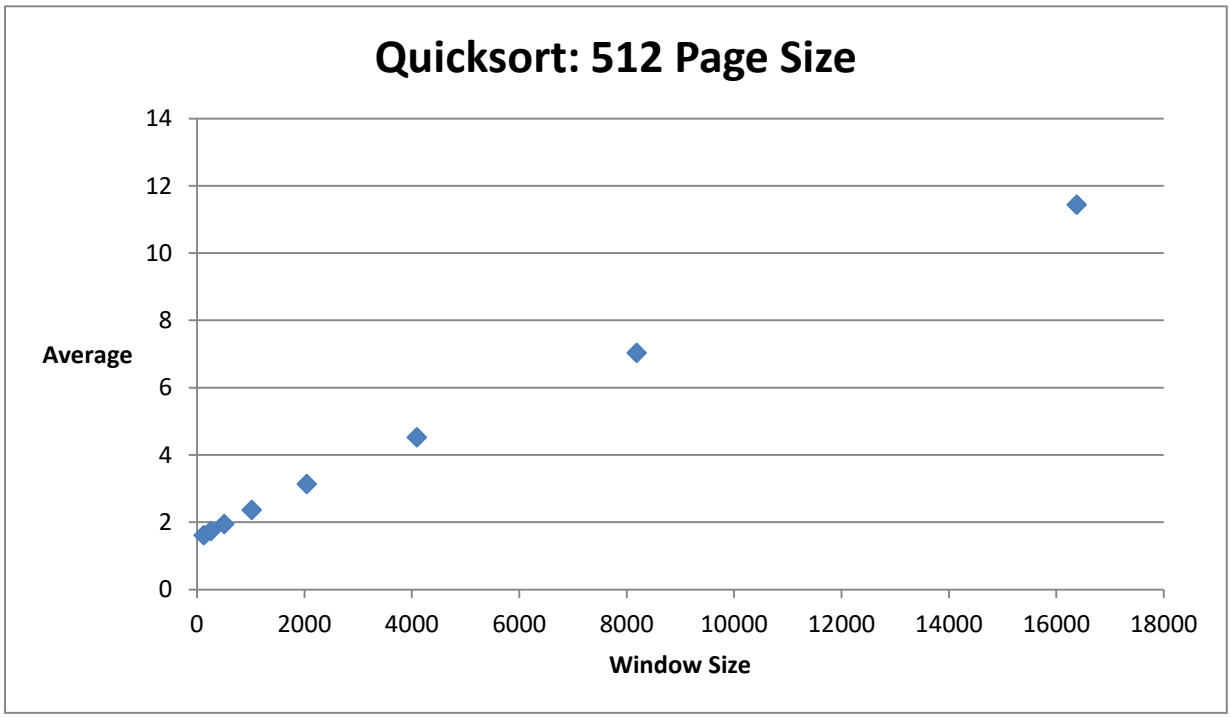

Figure 2.4

Blaz Pocrnja
Shawn Sydia

Summary of Results

FYI:
The figures above graph the average number of unique page references over all working sets of the given window size in the x-axis, for heapsort and quicksort. The page size indicated in the title refers to how many 32-bit addresses constitute a page.

i.e. If window size is 3, and a page size is a single 32-bit address, then after 3 references to memory we count how many of them were unique.

{Page 1,Page 1, Page 2} Would have two unique page references.

We sum this number up with all working sets needed to sort, and divide by the number of working sets.

{1,1,2}{1,3,4}{1,2,3}
2 Unique + 3 Unique + 3 Unique = 8
8/3 = Average

Conclusion:
From looking at any graph above, it is clear that as the window size increases so does the average number of unique page references. This is obvious, as having a larger working set means there are more page refernces per set, with a greater chance of being unique.

Page size does not effect the shape of any of the graphs. The only difference being the scale of the y-axis. As page size increases the average number of unique page references per working set is decreased. This is because larger page sizes cover more addresses, so a single page may be referenced multiple times for the same number of addresses that multiple smaller pages cover.

Comparing any heapsort and quicksort graph shows that quicksort has better memory usage. If we look at figure 1.4 and 2.4, for example, we see that for page size of 512, and window size of 16384, heapsort requires over 90 unique pages on average, whereas quicksort only needs less than 12. This means that quicksort performs less page faults than heapsort, which is a relatively expensive operation, so it actually runs faster than heapsort even though they are both O(nlogn).