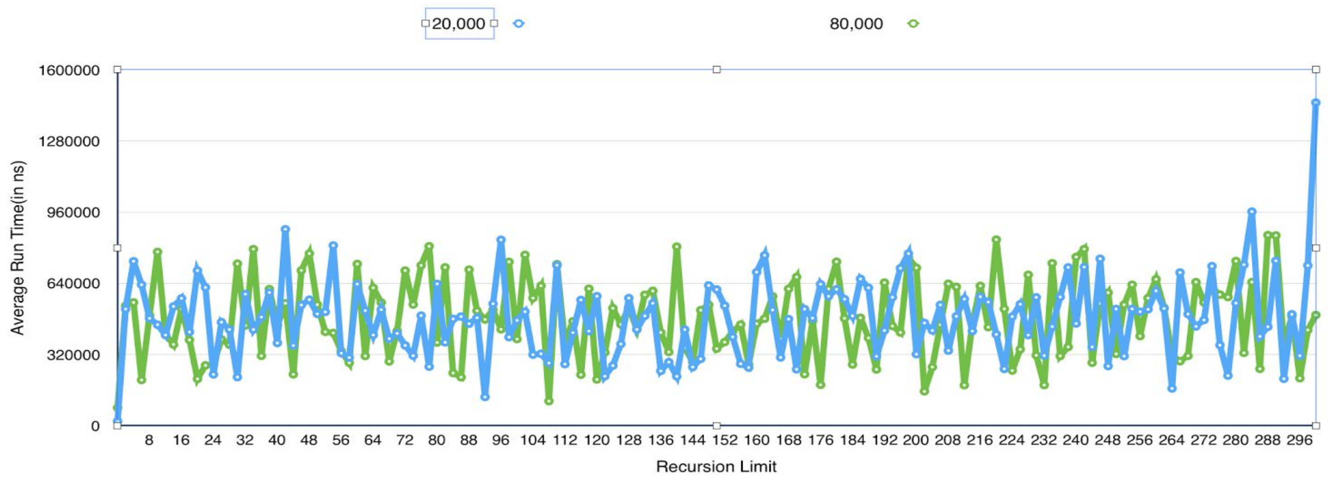
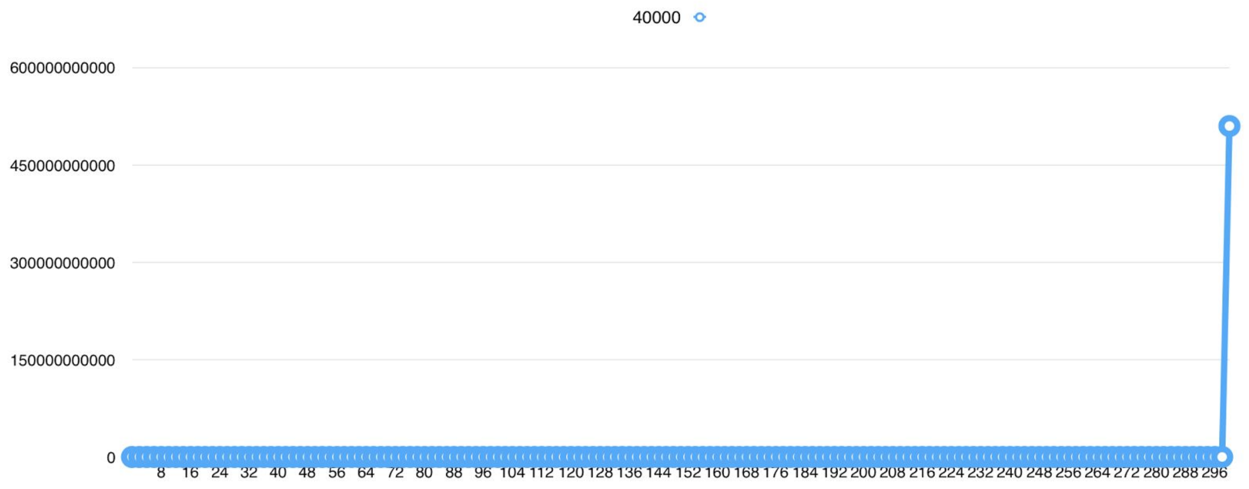


Blue: Array size 20K
Green: Array size 80K

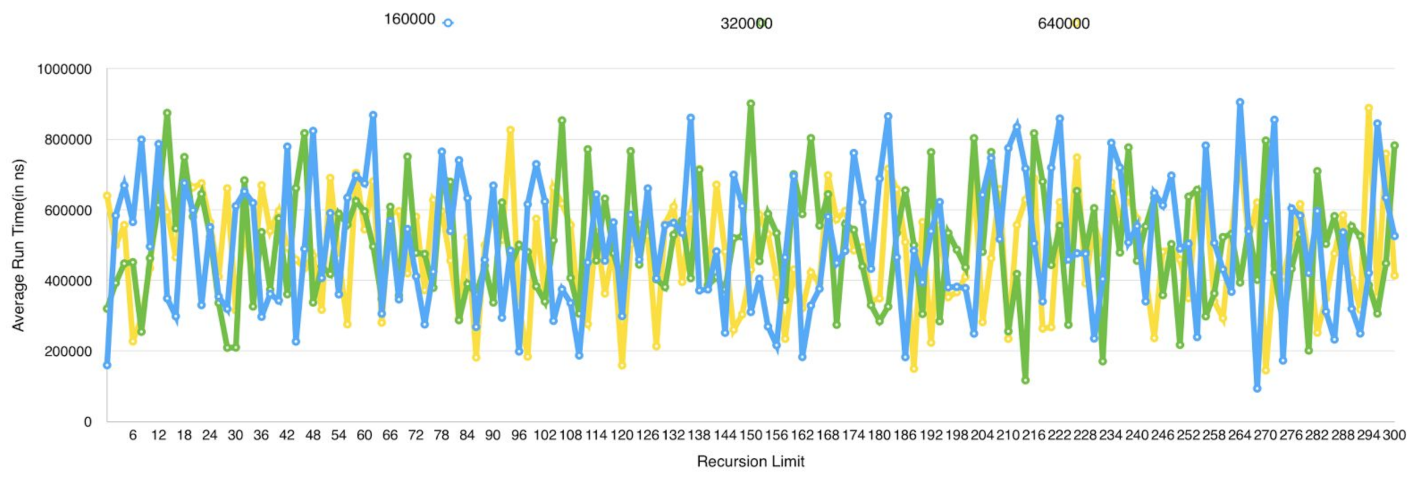
For Array Size 20K, looks like when the recursion limit was lower than 8, run time was the lowest but as recursion limit was increased, run time also increased, being highest when recursion limit was the highest. The graph is showing a lot of variations in terms of going up and down at different limits.

With comparison to array size 80K, looks like the increasing recursion limit actually decreased the run time.



Blue: Array size 40K

With array size 40K, I have got a pretty weird graph, where up till around 290 recursion limit, the average run time is not changing and it’s a flat line but suddenly when the limit is more than 290, time is highest. I think other factors might have affected this skewness of the data like other running programs, low battery etc.

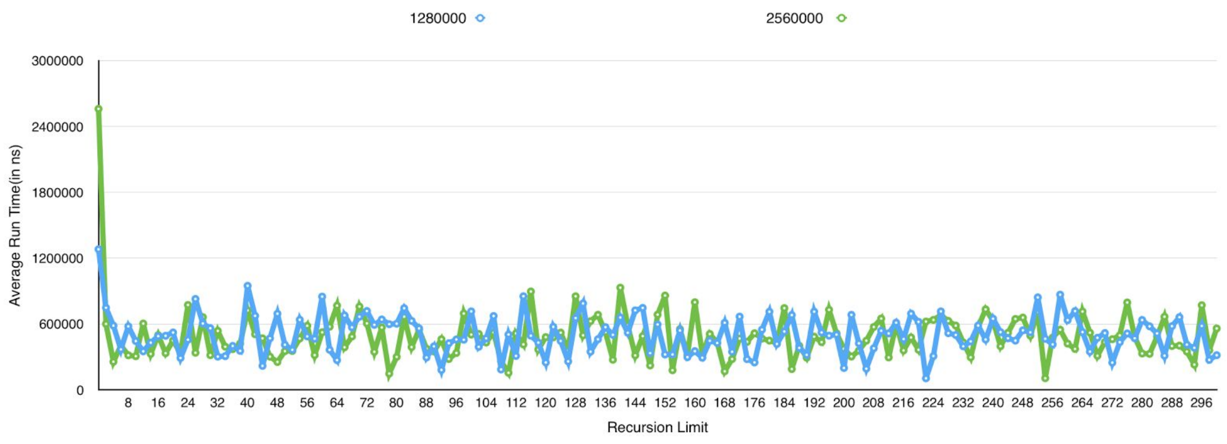


Blue: Array size 160K
Green: Array size 320K
Yellow: Array size 640K

As the size of array increases and the number of recursions increases the time is actually decreasing. But seems this trend does not hold true for all cases. For instance, from 160K to 320K they have almost overlapping graphs but if we increase it to double, the average time is decreasing and lesser with increasing limits.

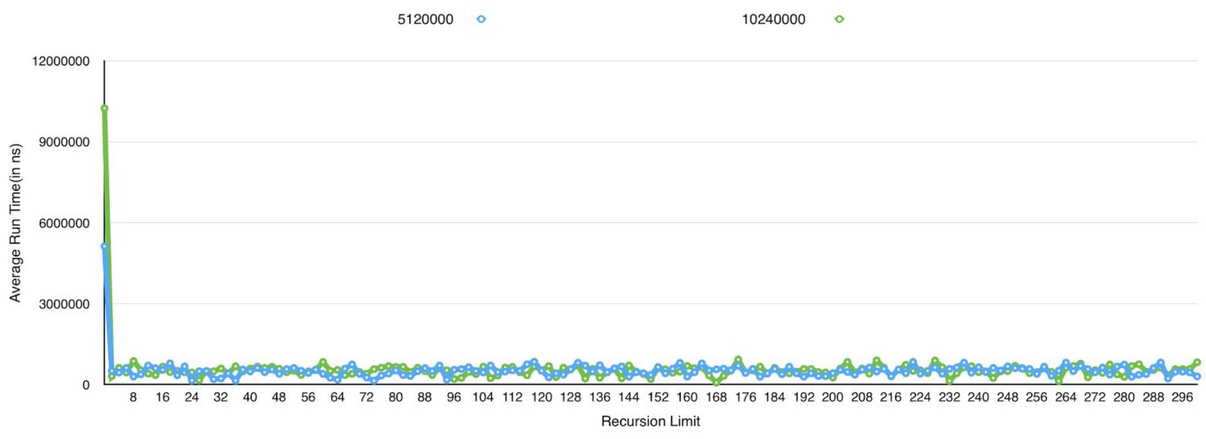
Two important observations:

- when recursion limit is too small and array size is bigger, the run time is actually more.
- As the size is increasing time is decreasing, in line with what we read in class.



Blue: Array size 1 million 2 hundred 80K
Green: Array size 2 million 5 hundred 60K

Consistent with other results, as the size increases, lowest recursion limit has the highest run time. It's hard to exactly point out an optimal limit but when the limit is highest time taken is actually less. Overall, upto a million quick sort is quicker for sorting but as it goes beyond 1 million mark, time taken increases.



Blue: Array size 5 million 1 hundred 20K
Green: Array size 10 million 2 hundred 40K

Here's the two graphs are almost overlapping, and are consistent with other results, as the size increases, lowest recursion limit has the highest run time. As green graph highlights, the bigger the size and smaller the limit, the higher the run time. So, setting an optimal recursion limit is important to make quick sort work effectively.