**Amdahl’s law says if a fraction r of a program cannot be parallelizd , the maximum speedup is 1/r , Determine the maximum speed up ?**

maximum speed up is 1/log(n) as the sorting process in the quicksort can’t be parallelized , but the partitioning portion can be , doing O(n) work in O(log(n)) time , causing the complexity to be O(log^2(n)) by parallelization instead of O(nlogn) in serial , this if we assume correct choice of pivots , which will not introduce too much overhead

**Can we run the for loops in main in parallel ? That is, we could farm out each iteration to a different processor, if we had enough ?**

Since the for loops are found in the partitioning portion of the algorithm they could be run in parallel

**Will running the parallelized version on 16 cores (assuming an array with billions of elements) would be worthwhile ?**

When we parallelize the function the size of n would equal n/p making the runtime O(log^2(n/p)) = O(log^2(n/16)) , I think it would improve the algorithm and introduce speed up significantly

**Will running the parallelized version on 1024 cores (assuming an array with billions of elements) would be worthwhile ?**

When we parallelize the function the size of n would equal n/p making the runtime O(log^2(n/p)) = O(log^2(n/1024)) , I think it would not introduce enough effect to care about using this number of cores, as the increase in speed comes from changing O(nlog(n)) into O(log^2(n)) not from the number of processors