Sequential vs Parallel Quicksort

For sorting a size 100 array:

```
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> g++ -o seq.exe .\sequential.cpp
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> ./seq.exe
Time taken by function: 5 microseconds
```

Sequential Takes 5ms

```
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> g++ -fopenmp ./openMP.cpp -o open.exe
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> ./open.exe
Time taken by function: 1068 microseconds
```

OpenMP Takes 1068ms

For sorting a size 1000 array:

```
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> g++ -o seq.exe .\sequential.cpp
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> ./seq.exe
Time taken by function: 62 microseconds
```

Sequential Takes 62ms

```
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> g++ -fopenmp ./openMP.cpp -o open.exe
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> ./open.exe
Time taken by function: 3051 microseconds
```

OpenMP Takes 3051ms

For sorting a size 10000 array:

```
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> g++ -o seq.exe .\sequential.cpp
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> ./seq.exe
Time taken by function: 2020 microseconds
```

Sequential Takes 2020ms

```
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> g++ -fopenmp ./openMP.cpp -o open.exe
PS C:\Users\iftek\OneDrive - Deakin University\SIT315-Concurrent and Distributed Programming\Quicksort> ./open.exe
Time taken by function: 32785 microseconds
```

OpenMP Takes 32785ms

As we can see that sequential is performing well for all the sizes. In the parallel openMP quicksort, when we partition the array around the initial pivot, we are giving each part to an independent thread to sort the elements. Different threads will find out pivot in each part recursively and sort it. The total number of threads will be equal to the total number of function calls to quicksort.