Final Observations

In the following assignment I have performed graph coloring using three algorithms:
a) Sequential Algorithm
b) Coarse Locked
c) Fine Locked

The number of vertices (n) and the number of partitions (k) are given as user input. The adjacency list is also given as user input. I have taken an adjacency list as input for the graph. I have created a graph class to store the graph in the form of an adjacency list. This graph is then partitioned into k partitions. Each partition is handled by a separate thread. After partitioning the graph, the vertices are divided into two parts:
a) Internal vertices: Internal vertices are those whose adjacent vertices are in the same partition.
b) External vertices: These vertices have one or more adjacent vertices in another partition.

There are no synchronization issues to color the internal vertices since all the adjacent vertices belong to the same partition.

Extra care needs to be taken to color the external vertices since one or more adjacent vertices belong to other partitions.
The threads can be synchronized in two ways:

a) Coarse Locks: In this approach, all the external vertices are given a common lock. So when one external vertex accesses the lock, i.e, calls the wait function, other external vertices need to wait till the lock is released. The synchronization issues are solved in this manner.

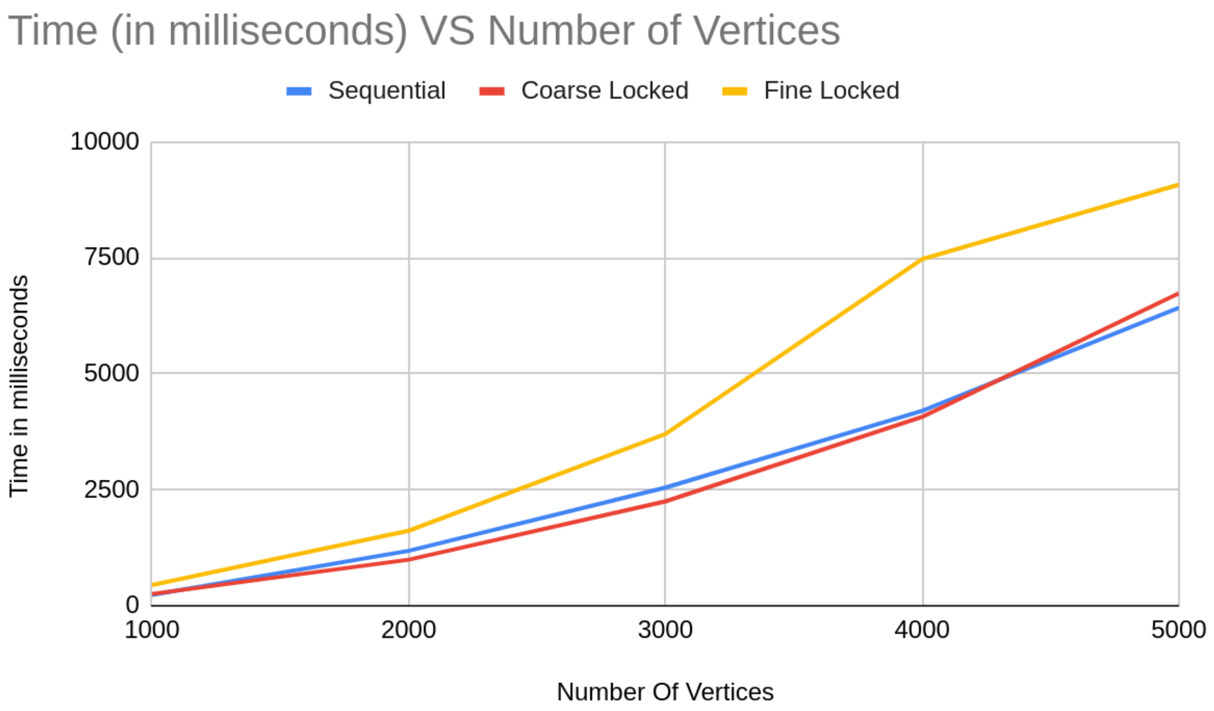
b) Fine Locks: In this approach, all the vertices present in the graph are assigned locks. When an external vertex is to be colored, that vertex accesses the lock of its adjacent vertices and its own lock in increasing order of value of vertices. In this way, the synchronization problems and the possibility of deadlock is resolved. The locks are then released in the decreasing order of values of vertices.

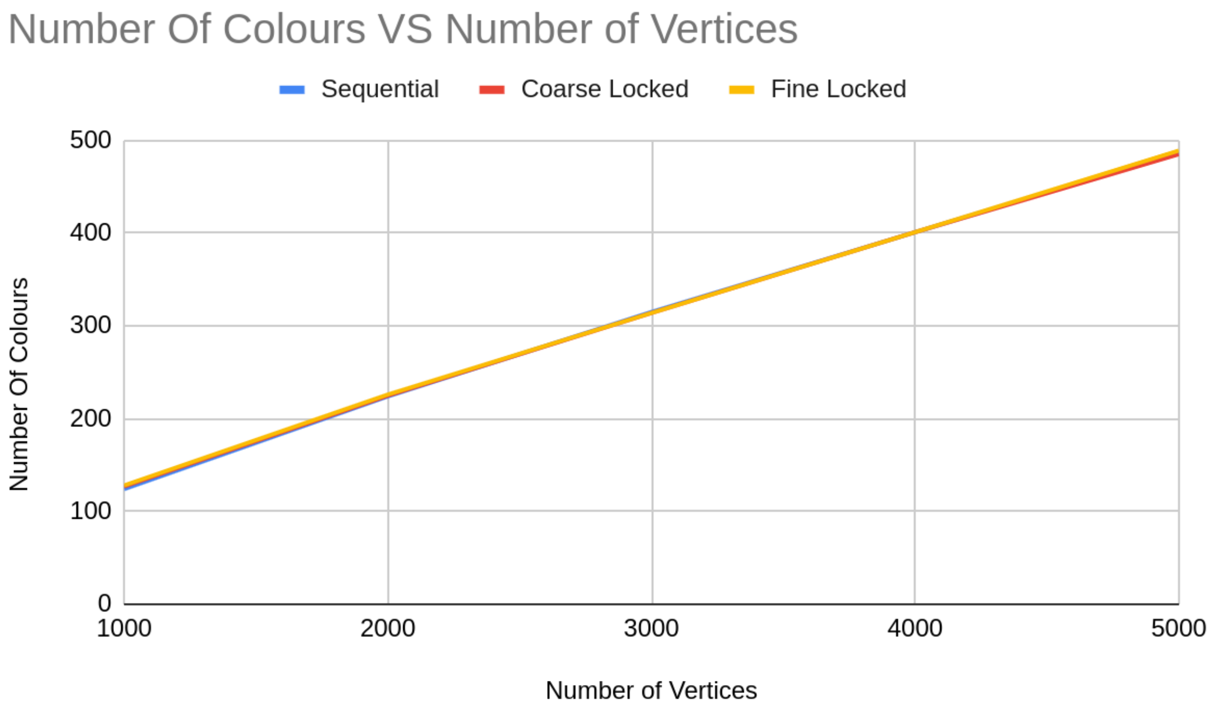
After partitioning the graph, the next step is to identify the internal and external vertices. I used maps for this purpose. Each partition is stored in a map. So I can cross-check the map with the adjacency list of a vertex present in a partition. In this way, I can assign whether a vertex present in the partition is external or internal.

I used a map to color the vertices in a partition, since searching in a map takes O(logn) time. I used a greedy approach to color the vertices. Initially, all the vertices are given a color -1 which means they are not colored. Then I store the colors of the adjacent matrices in a map. I start iterating the map in a while loop searching for available color. I used the variable color for this purpose. It is initialized to zero. This variable iterates through the map to check if the color is already present, if it is present color increments by 1, else the value of color is assigned as the color to the vertex. This approach is common for both internal and external vertices.

The output is written in the file output.txt (not included in the final submission).

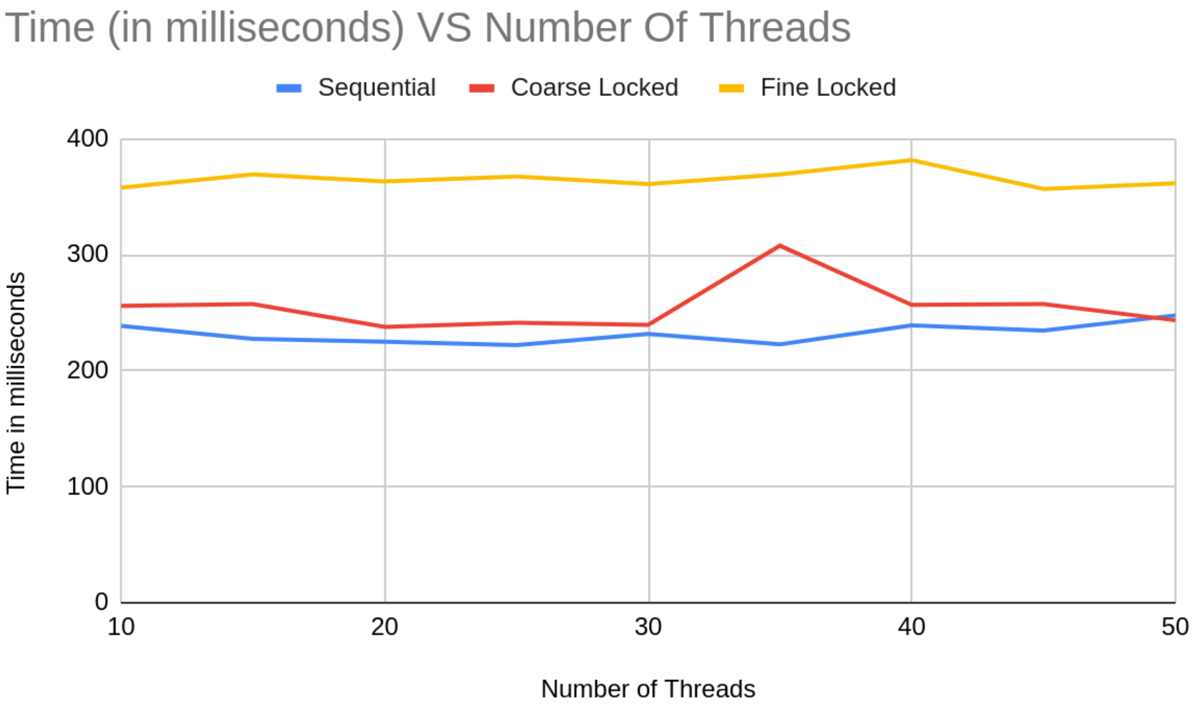
The following graphs depict the performance of the three algorithms: Sequential, Coarse Locked, and Fine Locked

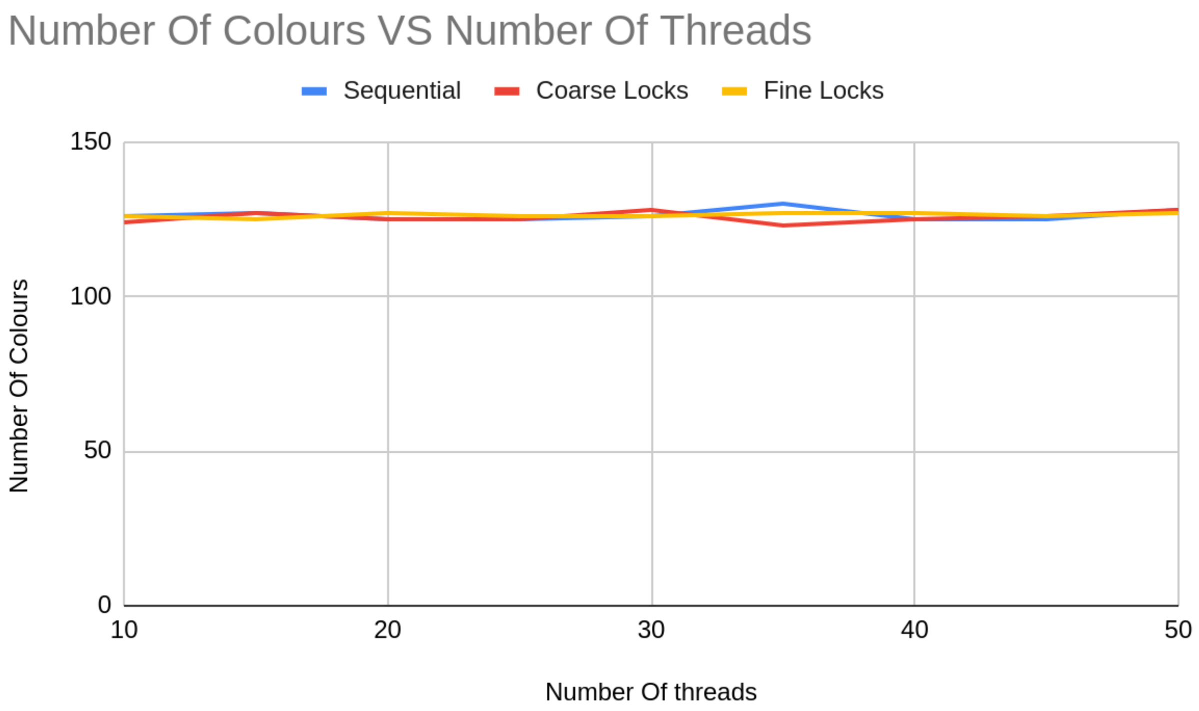




In the above two graphs, the number of partitions is taken to be 500 and the number of vertices is varied.

- It can be inferred from the above graphs that the time taken to execute increases as the number of threads increases in all three algorithms. The Fine locked algorithm takes more time compared to the Coarse locked and Sequential algorithm.
- It can be seen that for a particular number of vertices, the number of colors is almost the same for all three algorithms. And the number of colors increases as the number of vertices increases.





In the above graphs, the number of vertices is 1000 and the number of threads is varied.

- It can be seen that there is no major change in the time taken to execute for all three algorithms. It comes as a surprise that fine locks and coarse locks are taking relatively more time than the sequential algorithm. This can be attributed to the thread overhead and delay due to the locks.
- The number of colors used is almost the same for all three algorithms. The number of colors used is independent of the number of threads. The number of colors depends only on the number of vertices. Since the number of vertices is kept constant the number of colors used won’t vary a lot.