Name: Mukkavalli Bharat Chandra
Roll No. : ES19BTECH11016

Report

In the following assignment I have performed graph coloring using three algorithms:
a) Sequential Algorithm (not included in the final submission)
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 synchronisation issues to colour the internal vertices since all the adjacent vertices belong to the same partition.

Extra care needs to be taken to colour the external vertices since one or more adjacent vertices belong to other partitions.
The threads can be synchronised 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 synchronisation issues are solved in this manner.

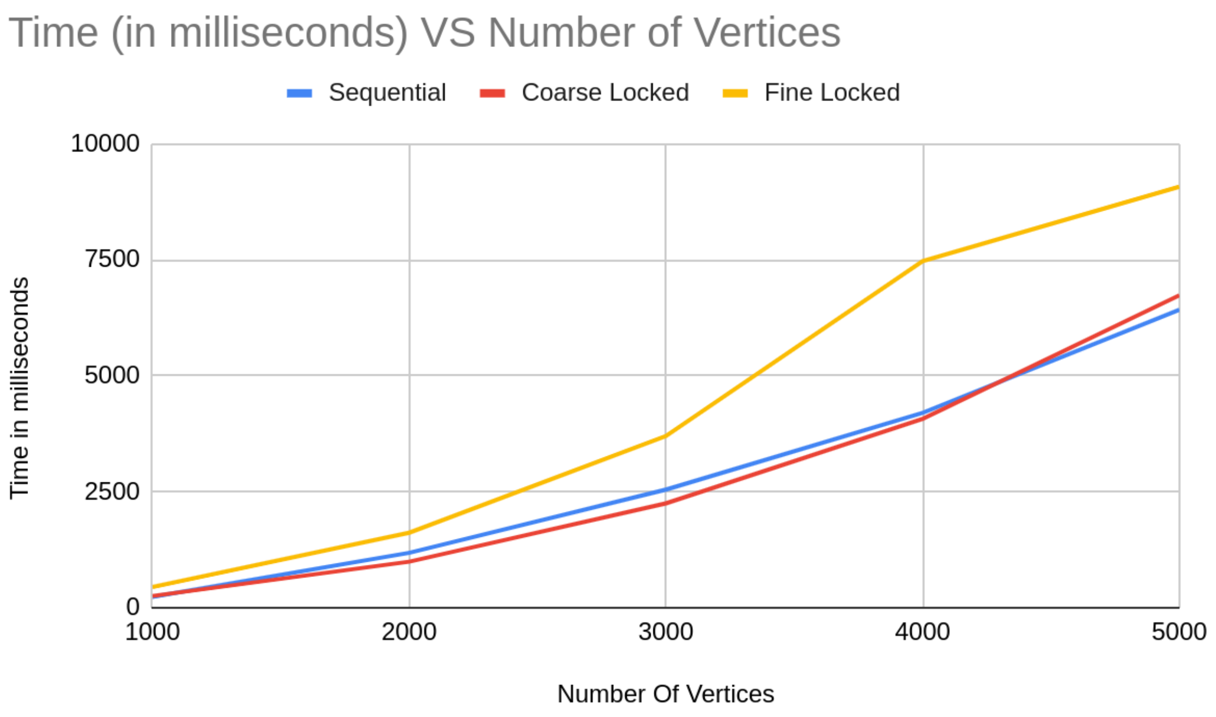
b) Fine Locks: In this approach all the vertices present in the graph are assigned locks. When an external vertex is to be colored, that vertex accesses the lock of its adjacent vertices and its own lock in an increasing order of value of vertices. In this way the synchronisation problems and the possibility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

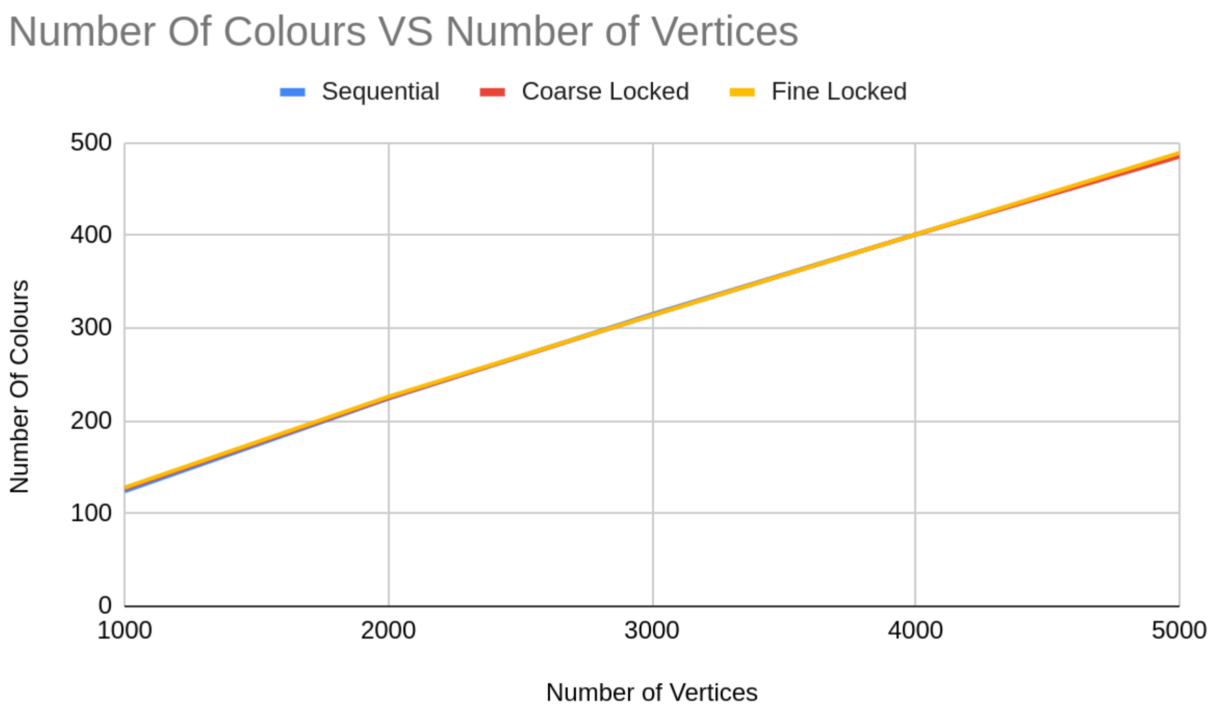
After partitioning the graph, the next step is to identify the internal and external vertices. I used maps for this purpose. Each partition is stored in a map. So I can cross check the map with the adjacency list of a vertex present in a partition. In this way I can assign whether a vertex present in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

To colour the vertices in a partition I used a map, since searching in a map takes O(logn) time. I used a greedy approach to colour the vertices. Initially all the vertices are given a colour -1 which means they are not coloured. Then I store the colours of the adjacent matrices in a map. I start iterating the map in a while loop searching for an available colour. I used the variable color for this purpose. It is initialised 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 colour 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).

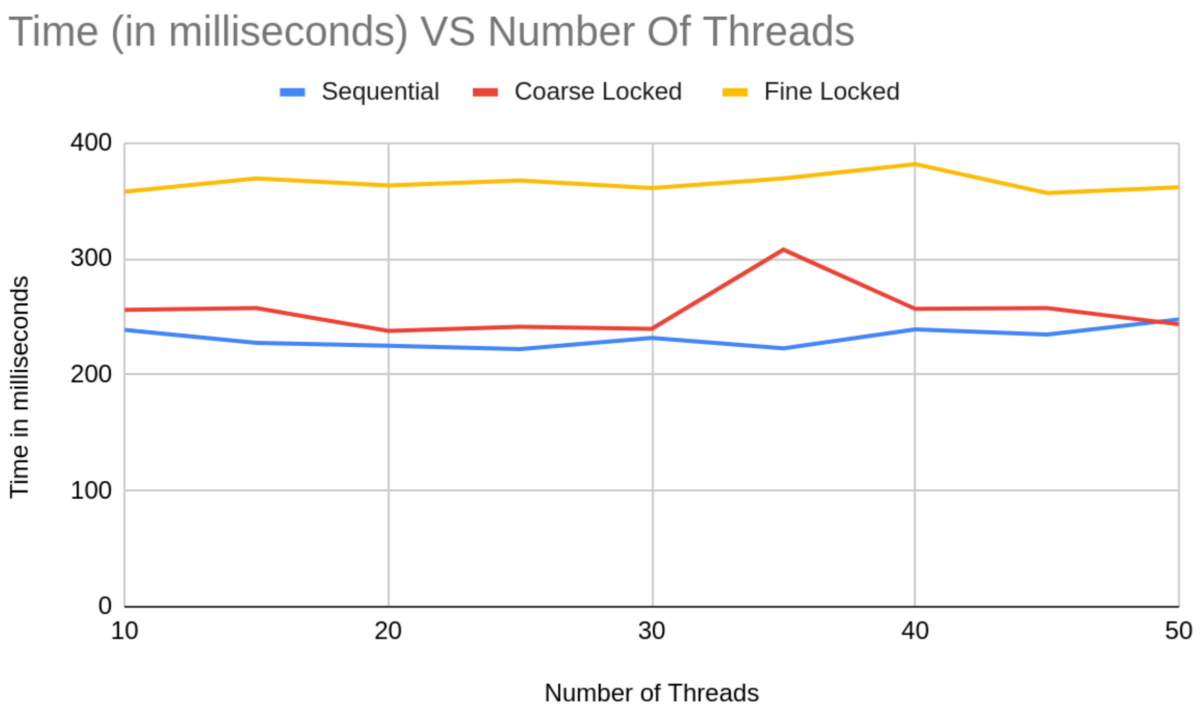
The following graphs depict the performance of the three algorithms: Sequential, Coarse Locked and Fine Locked

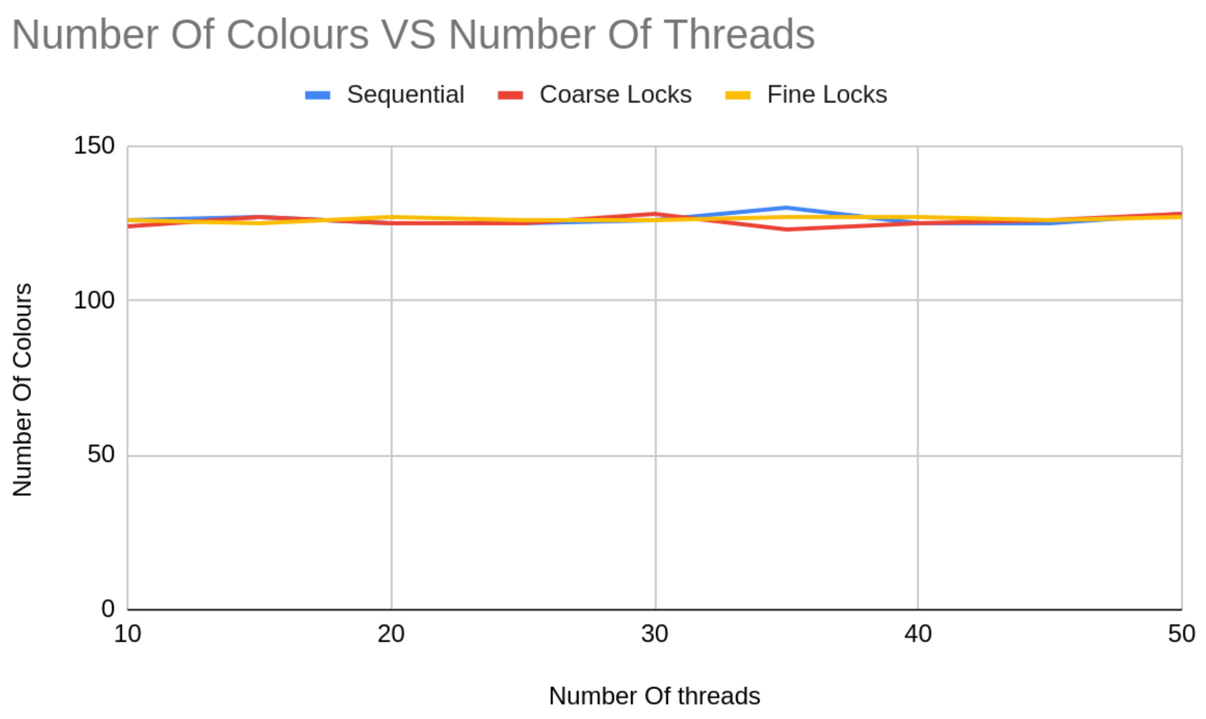




In the above two graphs the number of partitions are taken to be 500 and the number of vertices are varied.

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





In the above graphs the number of vertices is taken to be 1000 and the number of threads are varied.

- It can be seen that there is no major change in the time taken to execute for all the three algorithms. It comes to 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 colors used are almost the same for all the three algorithms. The number of colours used are independent of the number of threads. The number of colours depends only on the number of vertices. Since the number of vertices are kept constant the number of colours used won't vary a lot.