REPORT-ASSIGNMENT-5

Implementing Graph Coloring Algorithm using locks

ES19BTECH11003

We take all the input from input.txt in which first row contains no of partitions and no of vertices and later on consists of the adjacency matrix of the given graph

Low-Level Design of the Program

In both Coarse-grained and fine-grained we use semaphores for the locking system in which coarse grain we use only 1 semaphore whereas in fine-grained we use n semaphores where n is the total vertex count.

In both the codes we initialise all the semaphores with 1 in the main function itself

Later we call the partition function to make partitions of all the vertices and store the starting and ending of the partition vertex id in the TH 2-D array according to the index number.

Later we call the input_matrix function to store the adjacency matrix in the global 2-D array defined and we also randomly shuffle the elements 1-D array so that each partition may contain different vertices randomly

Here we start time to be measured and we start creating K threads and each thread executes the testCS function and the testCS is different for both the codes.

TestCs in coarse-grained

Here we use an assign 1-D bool array to find whether the colour is available for the vertex or not.

Initially we assign all elements of color array to -1 and all elements of assign array to false.here colour array is global array and assign array is local array defined in the function itself.

Now,we use for loop to iterate over each vertex of the partition and while iterating if we found that given vertex in the partition is an external vertex using the find_LM function we sem-wait function for block and after coming out from that we find edges with the current external vertex and mark all its neighbour vertex index value of assign to true so that it make sure that these colors should not be used to color the present external vertex.

Later we release the resources using sem-post function for block .and after finding the least possible colour for the given vertex we again use sem-wait and after passing through this we assign colour for the given vertex and again finding the vertices that are linked to present vertex and we change the assign array values to false.

And at last we again use the sem-post function to release the resources at the end of the testCS function.

Remaining Main

And later using the no_of-colours function and print function we print the total no of colours used and the colour of each vertex.

TestCs in fine-grained

Here we use an assign 1-D bool array to find whether the colour is available for the vertex or not.

Initially we assign all elements of color array to -1 and all elements of assign array to false.here colour array is global array and assign array is local array defined in the function itself.

Now,we use for loop to iterate over each vertex of the partition and while iterating if we found that given vertex in the partition is an external vertex using the find_LM function we sem-wait function for all the vertices which have an edge with the current external vertex and along with that we also lock the current external vertex too in the ascending order of the vertex ID

Later after finding the edges and their colours and with respect to that we make changes in the assign array to true for the colours that this vertex neighbours have.

Later we release the resources for all the vertices which have an edge with the current external vertex and along with that we also release the current external vertex too in the ascending order of the vertex ID using sem-post function for block .and after finding the least possible colour for the given vertex we again use sem-wait and all the vertices which have an edge with the current external vertex and along with that we also lock the current external vertex too in the ascending order of the vertex ID

after passing through this we assign colour for the given vertex and again finding the vertices that are linked to present vertex and we change the assign array values to false.

And at last we again use the sem-post function to release the resources of all the vertices which have an edge with the current external vertex and along with that we also release the current external vertex too in the ascending order of the vertex ID at the end of the testCS function.

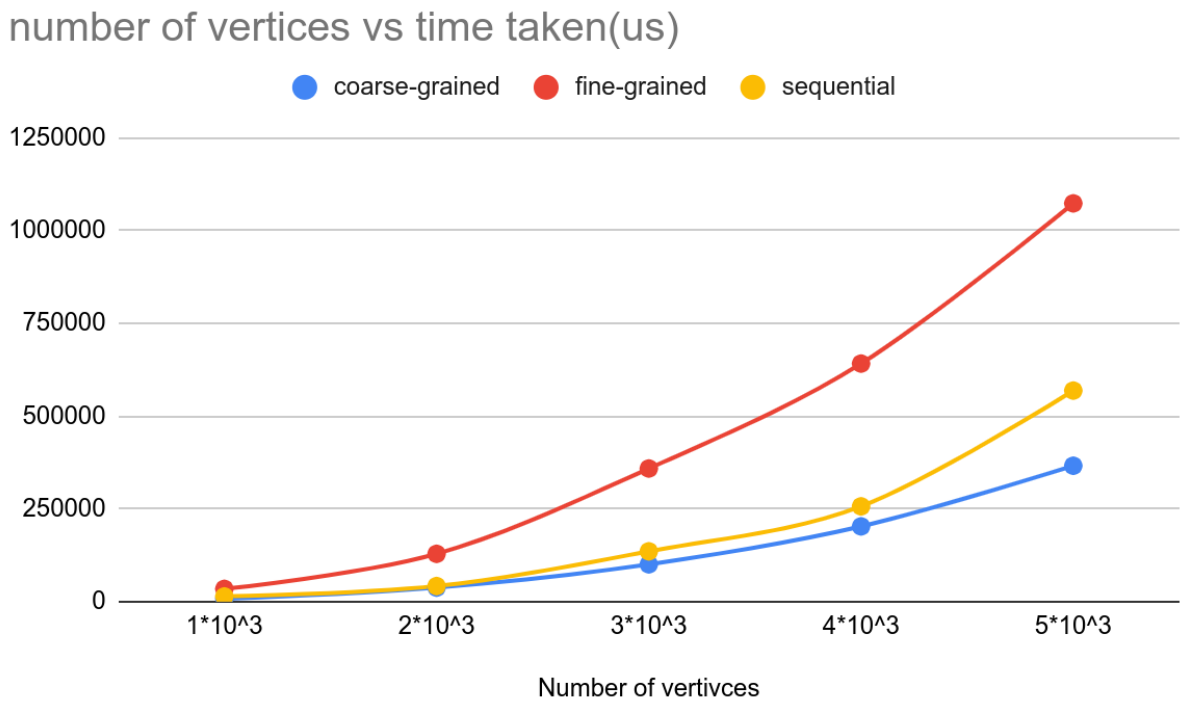
Remaining Main

And later using the no_of-colours function and print function we print the total no of colours used and the colour of each vertex.

Graph Analysis

In both graph 1&2 number of threads =100.

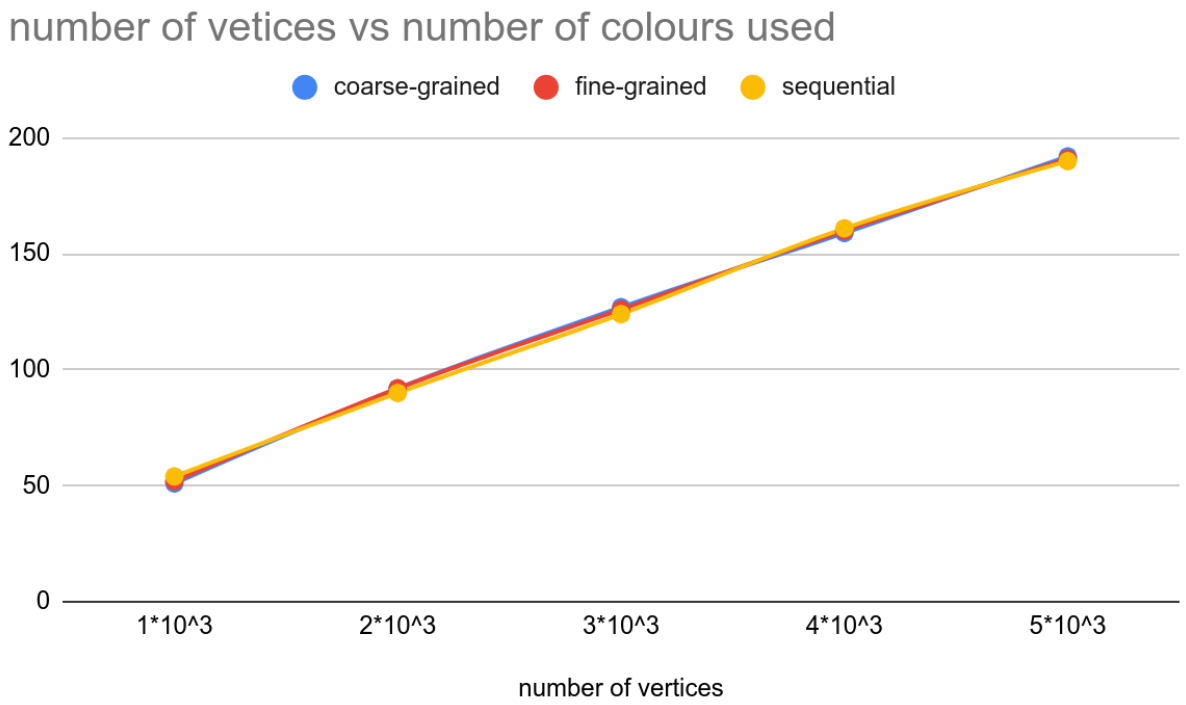
1.number of vertices in graph vs time taken by algorithm:



- From the above graph ,we see fine-grained has take large amount of time next we have and sequential and the leaset is coarse grained. . the possible explanation for graph can be as we know there are most number of of locks in fine so it would take more time as we lock and unlock them next we see our sequential, we have used threads so

we may say we should get noticeable least time for coarse but here also as we use lock and unlock.so it is not very much least

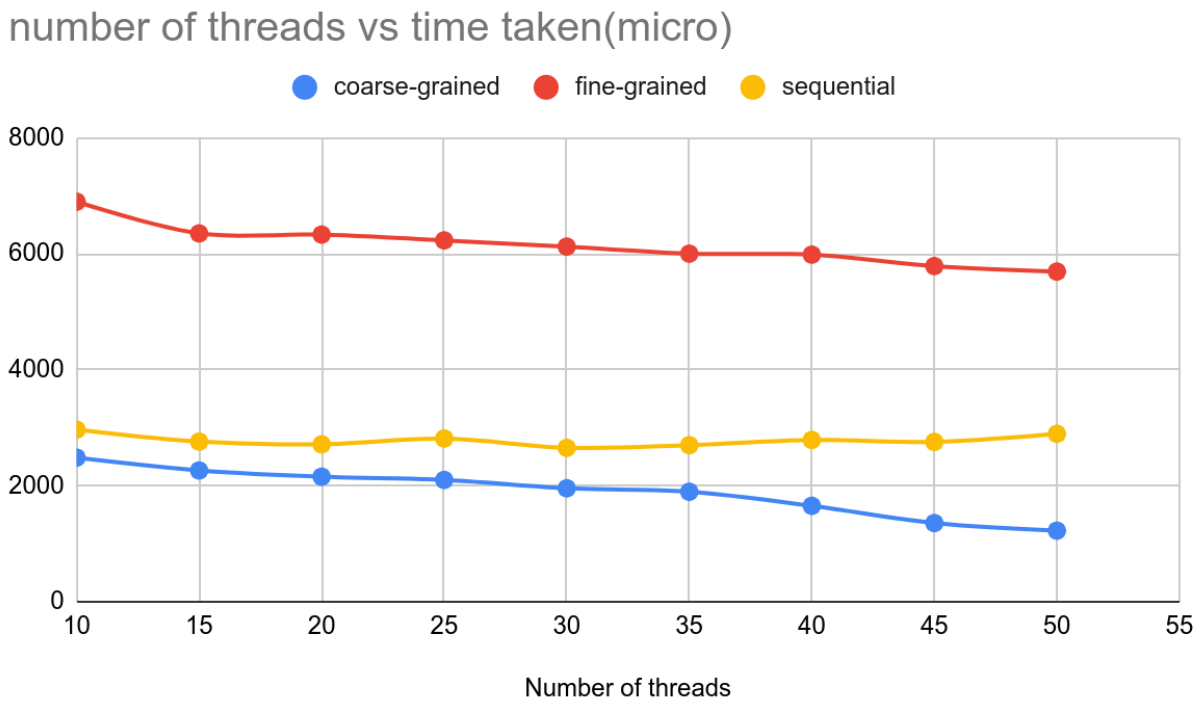
2.number of vertices in graph vs number of colours used:



➢ From the above graph we can see that as the number of vertices increases number of of colours used by all 3 also increases in all three graph. We can see it takes similare number of of colours in al 3 algorithm. There is small difference but all the 3 algorithms have same number of colours

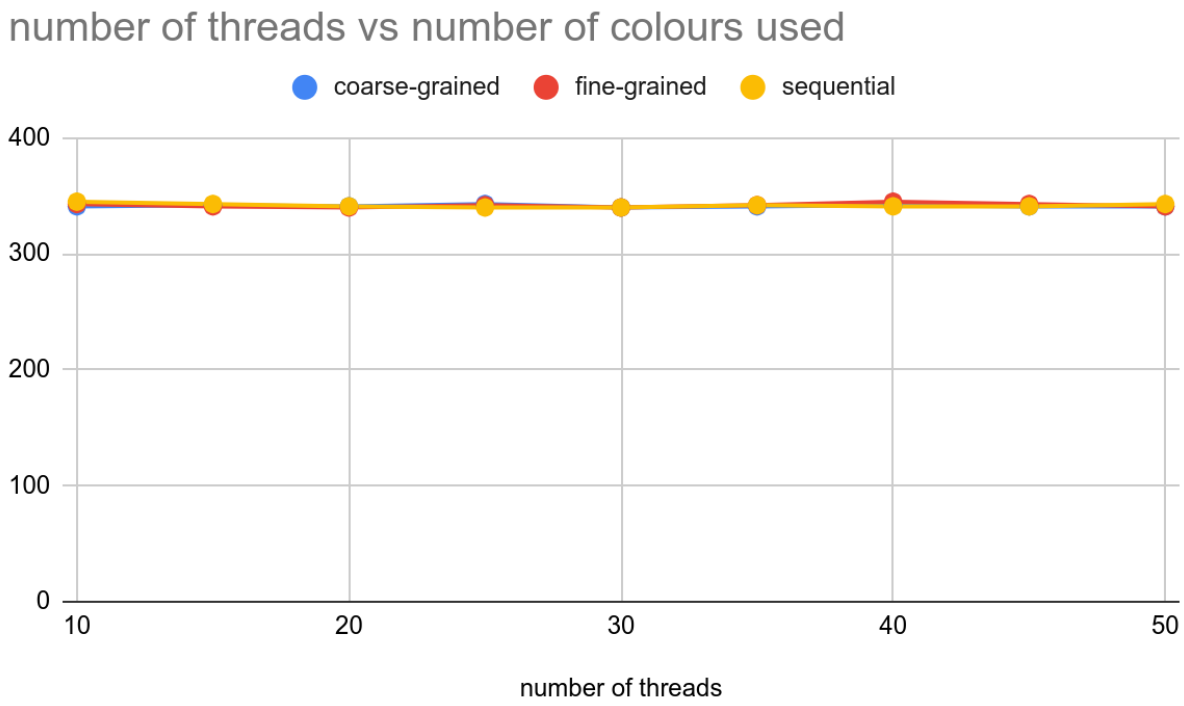
Number of vertices for below both plots are = 10000.

3.plot3:number of threads vs time taken



➢ From the above graph we can see that for coarse and fine grained number of colours are decreasing as number of threads are increasing.this is obvious because as the number of threads increases the vertices get divided and it can be done in less time but the decrement is not very much that could be due to locks used in code as locking and unlocking takes time this may be also a factor. Sequential is almost same because it has no effect with number of threads.

4.plot4:number of threads vs number of colours used



➢ From the above it is clear that almost in all cases number of colors used are same .there is no effect of thread on number of colors used. This is because number of vertices here are constant . there is almost no affect for sequential algorithm but for coarse and fine grind there may be some up and downs in graph.