OS 2

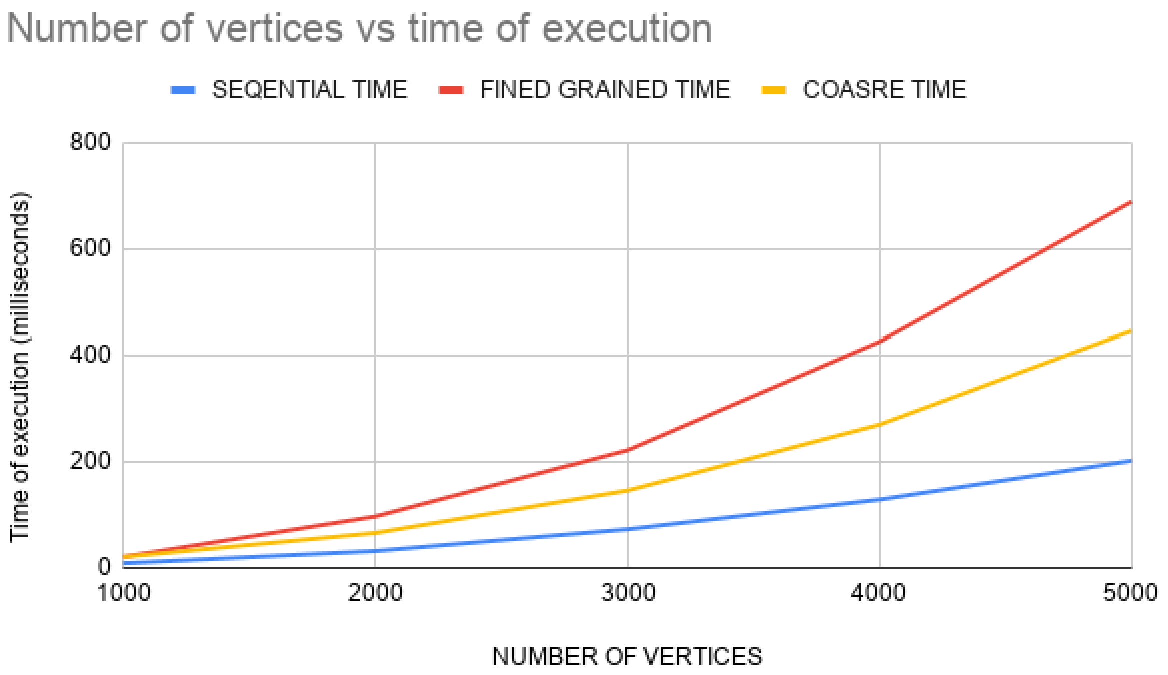
Name:Rithvik kondapalkala
Roll no:CS19BTECH11038
GREEDY ALGORITHM

Graph coluring sequential
* Color is variable determining colour of node
* ColourUsed is array of all colours .could be used for verifiying all colours used by all its adjacent vertices
*colour used for a vertex
    ColourUsed[i]=true means i colour is used
*Intialize all colors of vertices to -1 means not colured yet
* Make all colurs unused colorUsed=false
*for each vertice
    Vertexcolor()
        - checking all colours used by adjacent vertices using colourused of a colour is true
        - checking from start if any colur is unused then colour is given to the vertex
        - making colurused false for next vertice verification
*Fine-Grained Locks:
Mutex locks are used

Creating partions
    Partionlist is a class
        -addnode
        -assigninout for assigning internal and external vertice of a parttion
        -display
        -find for finding a vertex in the partition
    Randomly assign a vertex to a partion
        * Color is variable determining colour of node
* ColourUsed is array of all colours .could be used for verifiying all colours used by all its adjacent vertices
*colour used for a vertex
    ColourUsed[i]=true means i colour is used
*Intialize all colors of vertices to -1 means not colured yet
* Make all colurs unused colorUsed=false
*for each partion
    create a thread
    make all internal vertices
    -assign a colour using vertex colour
    for externsl vertices
    -make a lock all its adjacent vertices inincreasing order
    -vertexcolour of externalvertice
    - release all locks
Vertexcolor()
        - checking all colours used by adjacent vertices using colourused of a colour is true
        - checking from start if any colur is unused then colour is given to the vertex
        - making colurused false for next vertice verification
Coarse-Grained Lock:
Mutex locks are used

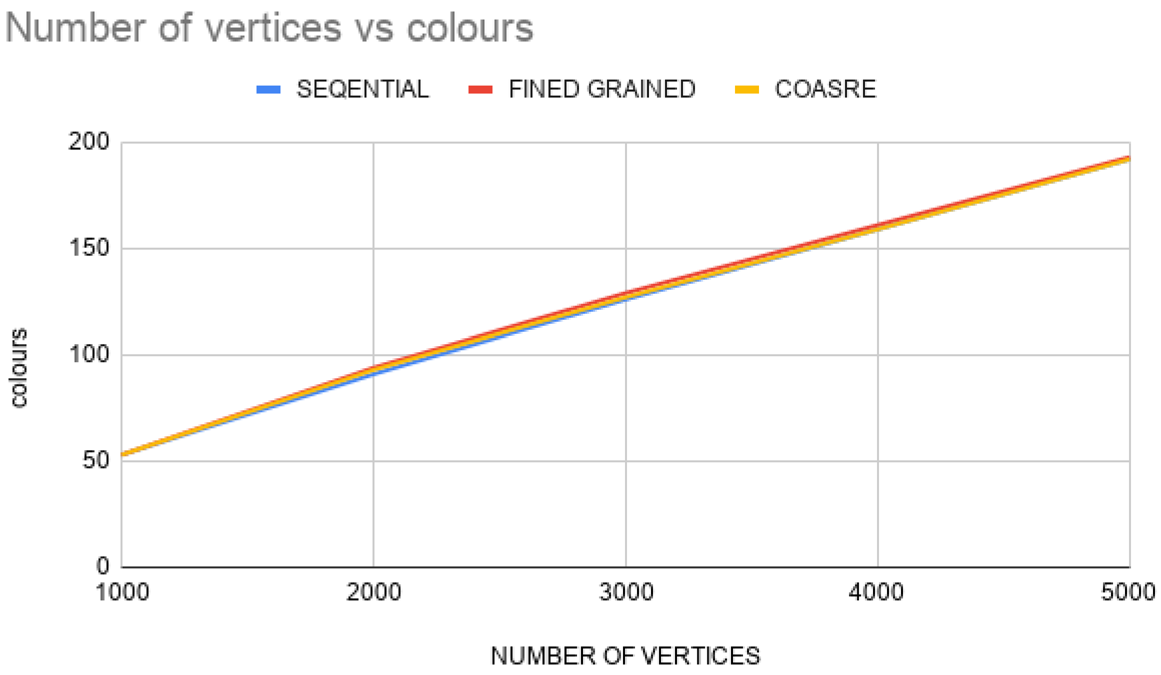
Creating partions

```
        Partionlist is a class
                -addnode
                -assigninout  for assigning internal and external vertice of a parttion
                -display
                -find for finding a vertex in the partition
        Randomly assign a vertex to a partion
                * Color is variable determining colour of node
* ColourUsed is array of all colours .could be used for verifiying all colours used by all its adjacent vertices
*colour used for a vertex
        ColourUsed[i]=true means i colour is used
*Intialize all colors of vertices to -1 means not colured yet
* Make all colurs unused colorUsed=false
*for each partion
        create a thread
        make all internal vertices
        -assign a colour using vertex colour
        for externsl vertices
        -make a common lock for all boundaries vertices
        -vertexcolour of externalvertice
        - release  lock
Vertexcolor()
                - checking all colours used by adjacent vertices using colourused of a colour is true
                - checking from start if any colur is unused then colour is given to the vertex
                -  making colurused false for next vertice verification
```

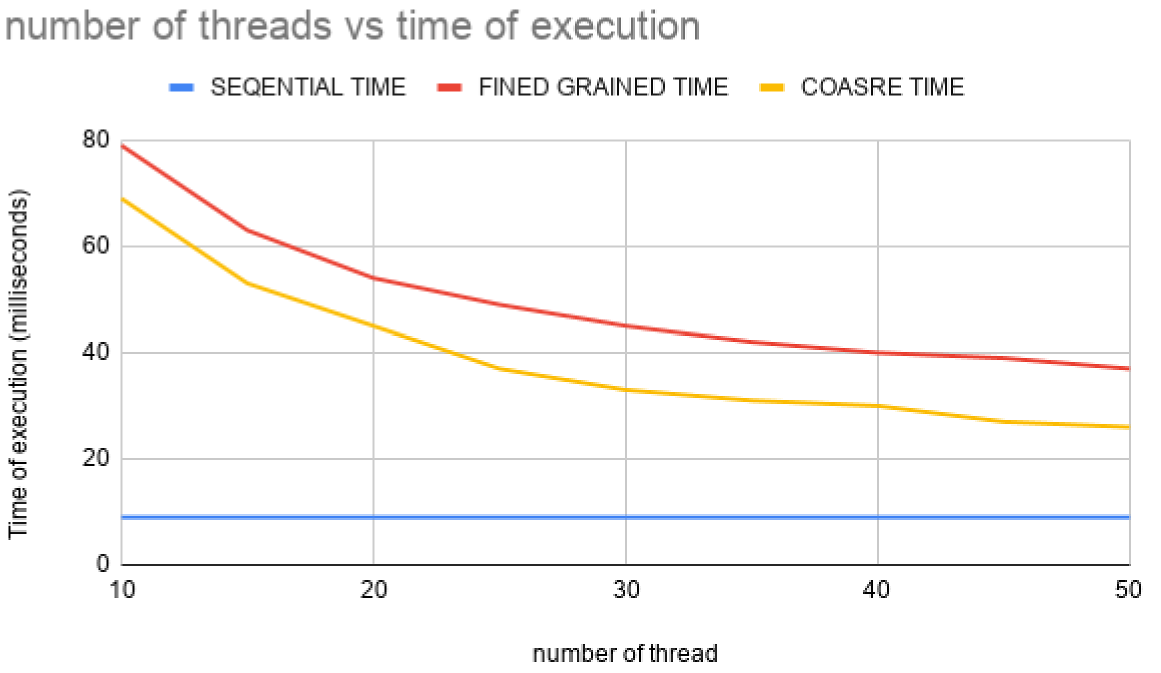


fixing number of threads checking on number of vericies
we have got the above graph
actuallly using threads the time of execution should be less for fined and coarse compared with sequential but due to the factors of thread creation and locks context switch time the fined andcoase time are more .

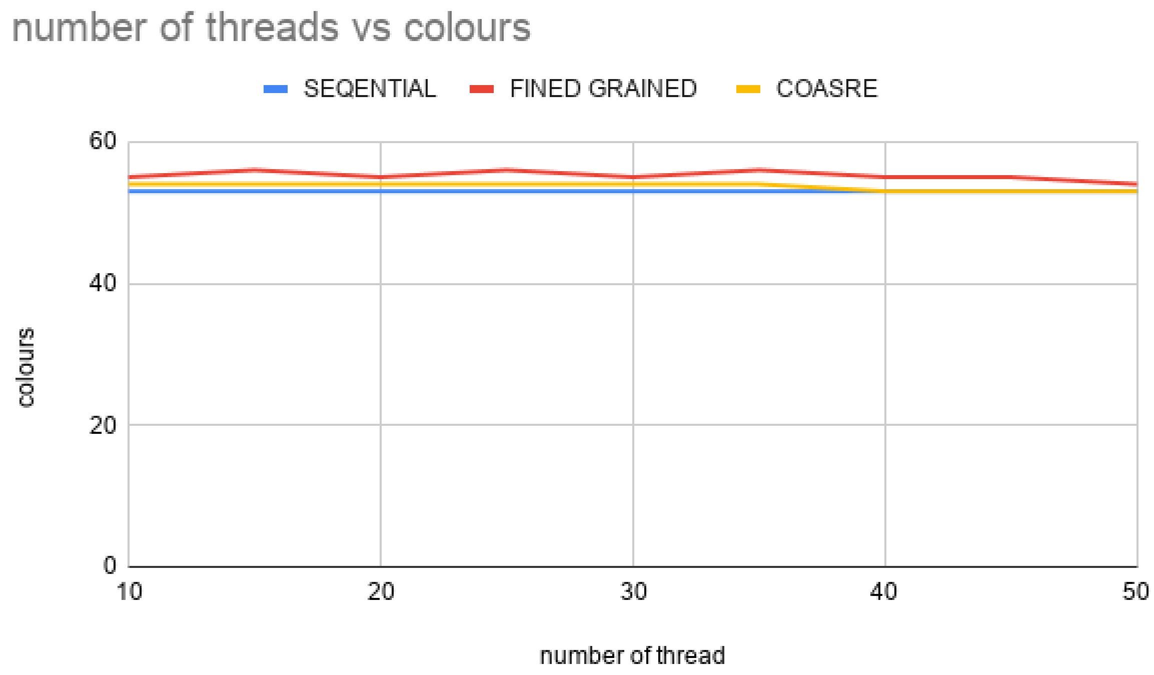
Comparing fined grained and coasre due more locks and it switching time fined grain time is more than coarse time



Number of threads were fixed and number of vertices vs colours are ploted
with increase in vertices number of colours are increasing due to more number of vertices and posssiblity of having adjacent is more so more colours are needed
fined and coasre fluctuate 1 or 2 colours here and there bit average almost same for all



fixing number of vertices to 1000 on computing graph on time of execution vs number of threads on increasing threads partions increases so computations become more paraalell and time of execution decreases for fined and coasre.



Fixing number of vertices
all three algorithm have same at all time no change in trend