An Experimental Study of Online Colouring First-Fit Graph Algorithm on Triangle-Free Graphs

COMP 6651 Winter 2023, Concordia University, Canada

Amrita Awasthi (40203422), Vignesh Pugazhendhi (40230262), Rishab Kumar (40199196),

Jimi Mukeshchandra Mehta (40225526), Saketh Oppula (40221013), Vaibhav Verma (40195571)

Github:

https://github.com/jimimehta/COMP-6651-Project/

Abstract

We study the triangle-free graphs and its generation using the triangle-free graph generator algorithm. An assignment of a colour to each vertex in online fashion of such graphs such that no two adjacent vertices of a generated graph are the same in colour. We performed a first-fit graph colouring algorithm in triangle free graph colouring in online fashion. And analysed the number of colours assigned by the algorithm with respect to the number of vertices in the graph.

1. INTRODUCTION

In recent years, it is challenging in online graphs to assign a minimum number of colours to the vertices in such a way that no two adjacent vertices are assigned with the same colour. In small graphs this problem is easy to tackle but in the real world, due to its potential use in several areas, including wireless networks, scheduling, and resource allocation, map colouring etc. online graph colouring algorithms have attracted increasing attention in recent years. In order to do this, academics have put forth several online graph colouring algorithms with varying time complexity, accuracy, and adaptability to dynamic graph settings trade-offs. There are a number of proposed algorithms for graph-colouring i.e., greedy algorithm which is a very straightforward algorithm that functions by sequentially giving the vertices different colours. The method chooses a colourless vertex at each step, giving it the smallest colour that is not utilised by any of its neighbours. Backtracking Algorithm that assigns colours to vertices in a recursive manner, reversing the assignment if it results in a conflict. Until a proper colouring is discovered, the programme investigates every potential colour assignment. The Welsh-Powell algorithm is a greedy one that orders the vertex degrees in decreasing order before successively colouring each one. For some categories of graphs, this technique can generate the best possible results. Saturation Degree Algorithm that determines the number of colours that have already been utilised by each vertex's neighbours, or the saturation degree of each vertex. The most saturated vertex is chosen by the algorithm, and it is given a colour that none of its neighbours are using. The Tabu Search method is a metaheuristic method that iteratively enhances the existing colouring by making little adjustments and assessing the new results. In order to maintain track of recent motions and prevent cycling, the algorithm uses a tabu list.

In this report, we present Triangle free graph generator algorithm and First Fit graph colouring algorithm for online graphs. A systematic experimental technique is used in an experimental study of online colouring graph algorithm “First-Fit” to give a thorough evaluation and find its dependency on the number of vertices in a graph. The algorithm will be tested on multiple graphs of the same graph and varying number of vertices in a graph as part of the project, and their performance will be analysed. The knowledge gathered from this study can assist practitioners in deciding which online graph colouring algorithm is best for their particular application domain and in pinpointing potential areas for algorithmic advancement. Overall, the research can advance the state-of-the-art in online graph colouring algorithms and the usefulness of those techniques.

2. BACKGROUND INFORMATION

Online Graph: In a general graph, we have a complete set of vertices and edges of a graph. But, in online graphs, we do not have access to the complete graph beforehand. The vertices and edges are introduced in an online manner i.e., one by one vertex and associated edges are revealed, complete graph is not known at once. Now, in such a scenario we have to assign the colour to the vertices arriving.

Graph Colouring: The objective of "graph colouring" is to colour each vertex of a graph differently so that no two neighbouring vertices have the same colour.

First Fit Algorithm: In this algorithm, as the graph’s vertices are incrementally revealed without having access to the complete graph beforehand, in accordance with the placement of the vertex in a graph, the algorithm allocates colours to vertices one at a time, colour is assigned to the vertex in such a way that the colour is not matched with its adjacent neighbours and the allotted colour is the the smallest natural number that fits, that is, the smallest integer that hasn't already been used to colour its neighbours.

3. PROBLEM DESCRIPTION

The goal of this study is to create triangle-free graphs with n vertices and determine the average number of colours utilised by the First Fit algorithm on N triangle-free graphs, as well as how this number varies with n. Picking two parameters for the study is required, specifically n = number of vertices in each online graph and N = number of online graphs. For the code to handle them, n and N must both be suitably large. On each of the generated N instances of k-colorable online networks with n vertices, the First Fit algorithm is implemented and applied. This method is repeated while increasing n. Since the chromatic number of triangle-free graphs is unknown and difficult to compute, it is not possible to determine the algorithm's competitive ratio. Therefore, the study's main objective is to calculate the average number of colours used by First Fit on N triangle-free graphs and analyse how the First-Fit's average colour usage depends on the number of vertices.

4. IMPLEMENTATION

We start with the graph generation algorithm, we are considering a given graph G(V,E) with n number of vertices, vertices set V, edges empty set E, and probability P with a random value between 0 and 1. For each vertex vi, check if there exists common neighbours between vi and vj for j≠i where i ∈ {1,2,3,...n} and j ∈{2,3,4,...n}.

If no common neighbour vertex is found between the vertices vi and vj then add an edge e(vi,vj) to set with a probability of P i.e. **neighbours of vertex vi (N(vi)) and neighbours of vertex vj (N(vj))** have no element in common that confirms triangle-free graph. Finally, the graph generated will be a triangle free graph. The pseudo code is shown in Algorithm 1.

Algorithm 1 GraphGenerator algorithm for Triangle-Free Graphs

```
Procedure GraphGenerator
    Create an Empty Graph G(V,E)
    Generate and add all vertices v to set V in G
    for i = 1 to n
        for j = i+1 to n
            X ← N(vi) ∩ N(vj)
            if size(X) = 0 then
                E ← E ∪ e (vi, vj, P)
        return G(V,E)
```

e (vi, vj, P) means the Edge 'e' with probability P in between vertices vi and vj.

To perform a first-fit algorithm, we are given a graph G(V,E) with vertex set V, edge set E, and a list of available colours C in non-decreasing order. We are creating an online graph G'(V', E') with vertex set V', edge set E', and a list of colours C' from the generated triangle-free graph in Algorithm 1.

Initially, we have an empty online graph G'(V', E').

While all the vertices are not added to the Graph G' from G we do the following:

From vertex set V, a new vertex v' ∈ V arrives with its pre-neighbourhood N −(v') into G' which means we are considering the edges that have been explored till the arrival of v'. We remove all the colour values assigned to the neighbours of the v' in the neighbourhood N −(v') from the colorset C. Then assign a minimum colour available in the colour set C to v'. The process is repeated until all the vertices of graph G have not been discovered.

The pseudo code is shown in Algorithm 2.

Algorithm 2 FirstFit algorithm for online graph colouring

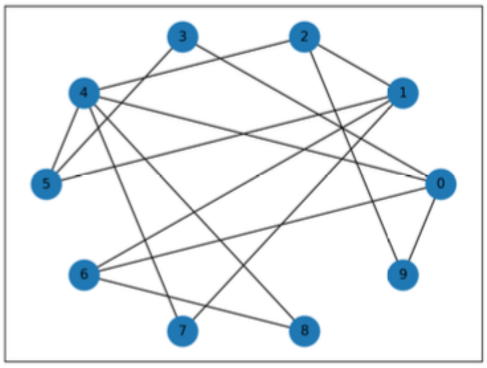
```
Procedure FirstFit
    Create an empty graph G'(V',E')
    C' ← ∅
    i ← 1
    while i ≤ n do
        A new vertex v' = σ(i) arrives with its pre-neighborhood N −(v')
        C'(v) ← min (N \ C'(N −(v')))
        i ← i + 1
    return G'(V',E')
```

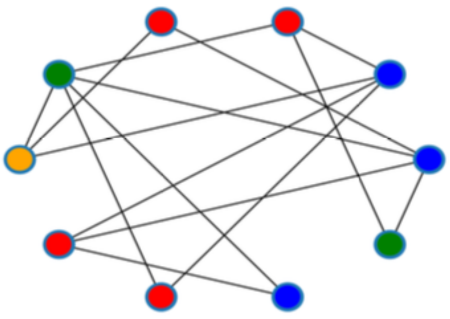
G'(V', E') is the output graph with coloured vertices with C' colour set with no two adjacent vertices having the same colour. We are explaining this by giving two examples.

In figure 4.1, a triangle-free graph with 10 vertices has been created using a graph generator algorithm shown as Graph in figure 4.1 and implemented a first-fit algorithm for the graph in online fashion.

The step by step arrival of vertices and assignment of its colour is also shown. For the online graph generated randomly with 10 vertices, the number of colours assigned is 4 and can be seen clearly in the output graph.

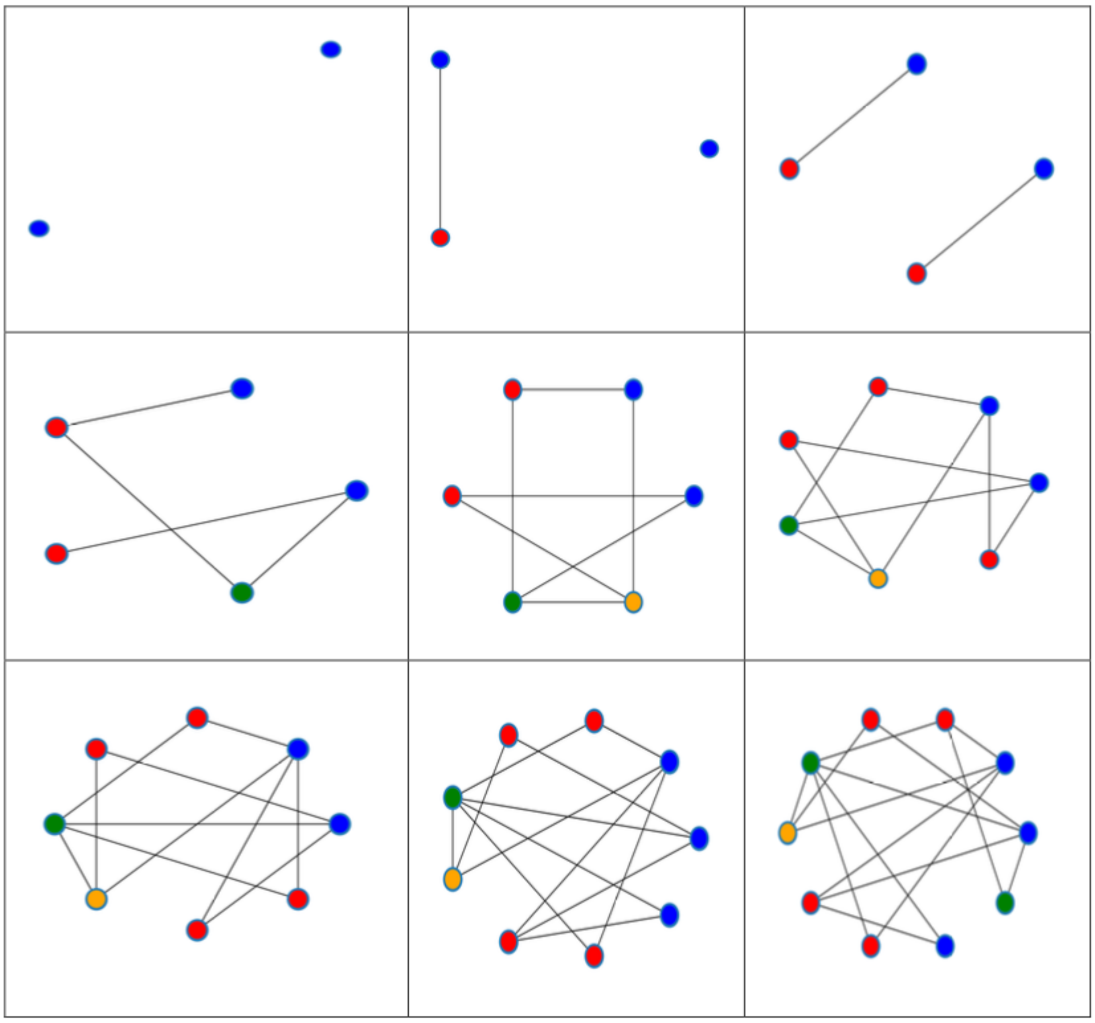
4.1 Step by step implementation of 10 vertices graph



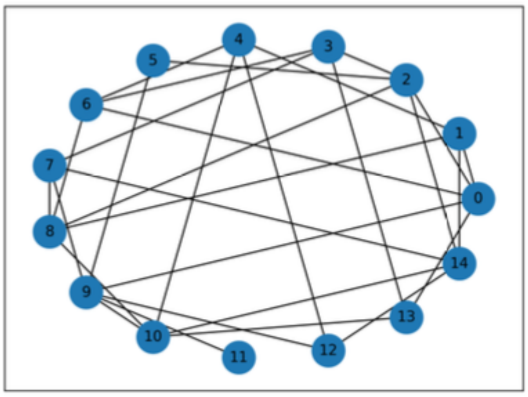


Graph

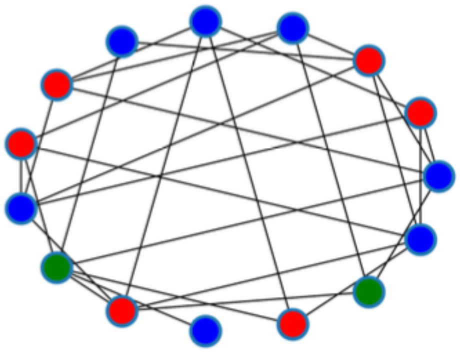
Output



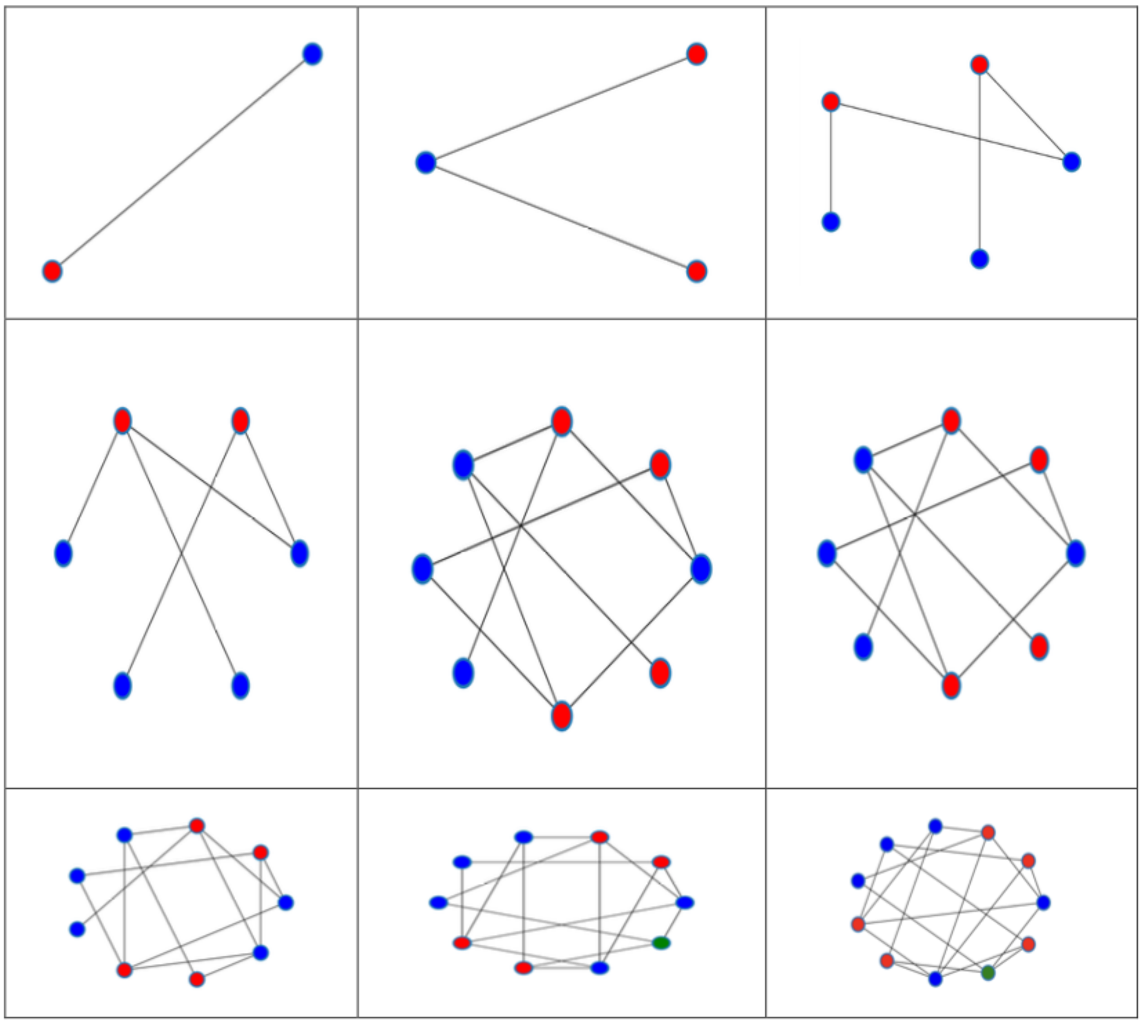
4.2 Step by step implementation of 15 vertices graph

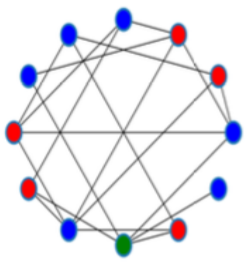
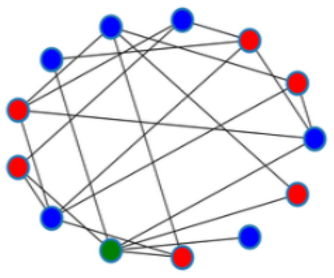
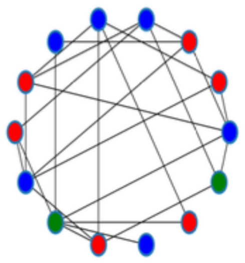
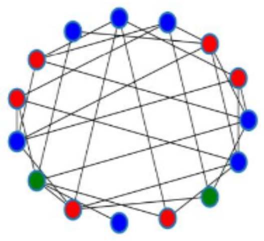


Graph



Resultant Colored Graph

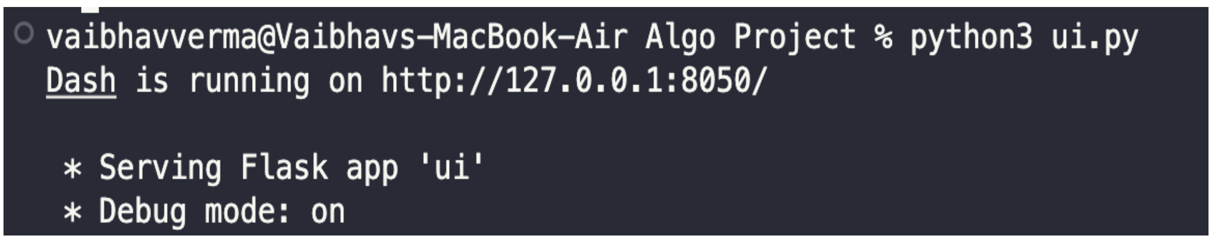


4.3 User Interface

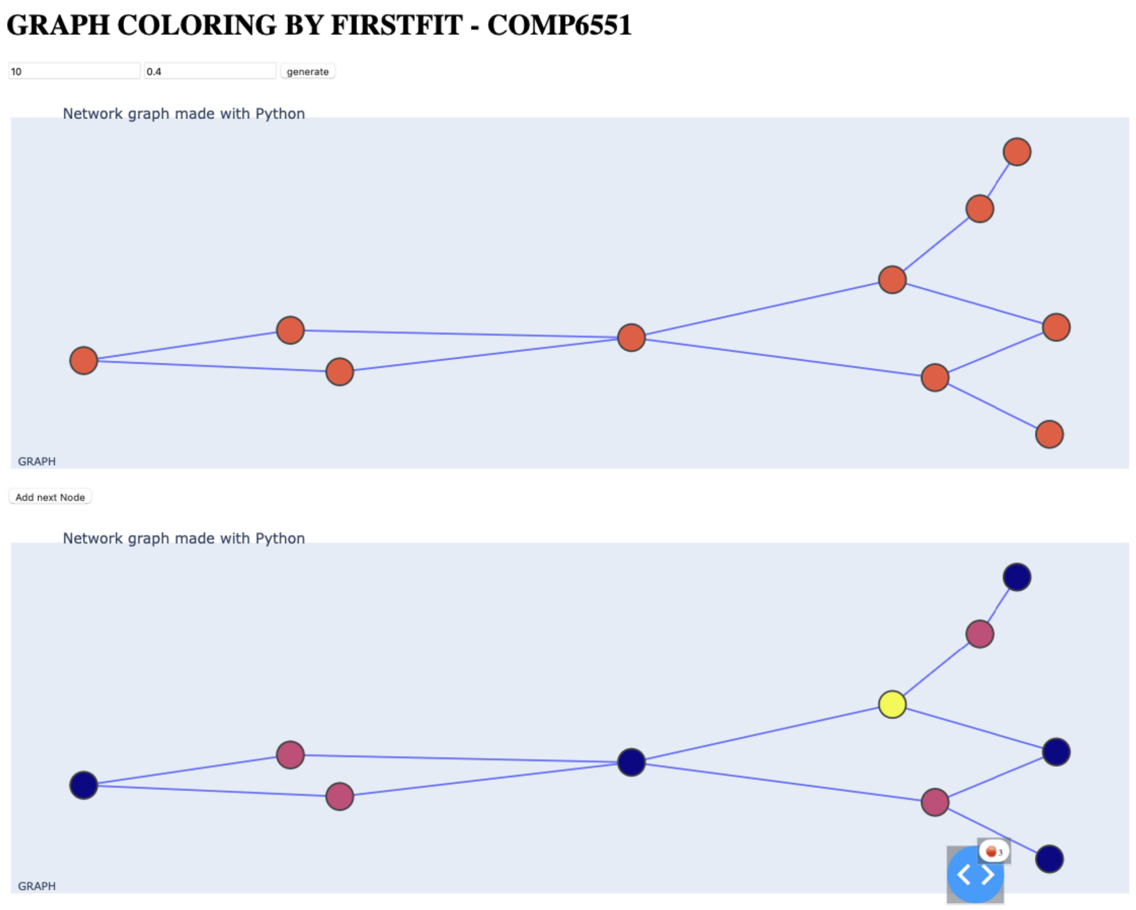
Upon running the user interface via the command *python ui.py*, a user can generate a graph by entering the number of vertices of their choice and the probability of edges. After clicking on the **generate** button, a graph is generated in a coloured form. By clicking the **add next node** button below the user can see a new node come in an online fashion and are given the colour according to our first fit algorithm.

The user can follow the link provided in the terminal output to use the UI.



```
○ vaibhavverma@Vaibhavs-MacBook-Air Algo Project % python3 ui.py
  Dash is running on http://127.0.0.1:8050/

   * Serving Flask app 'ui'
   * Debug mode: on
```

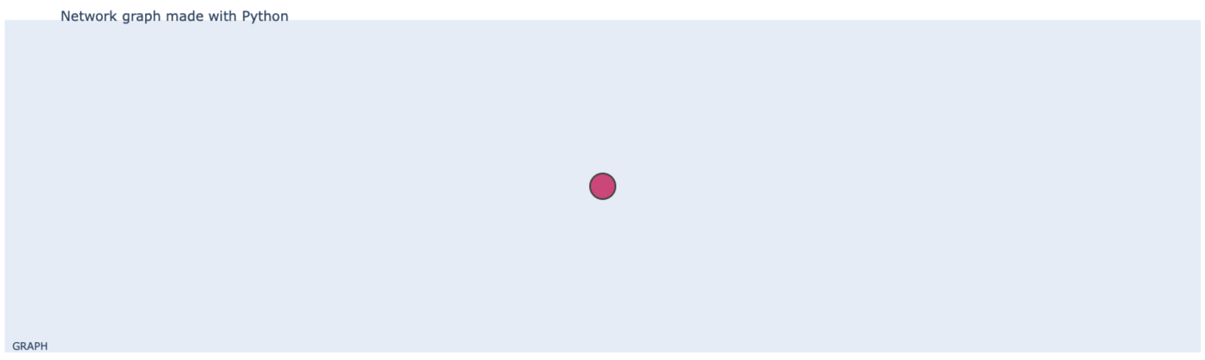


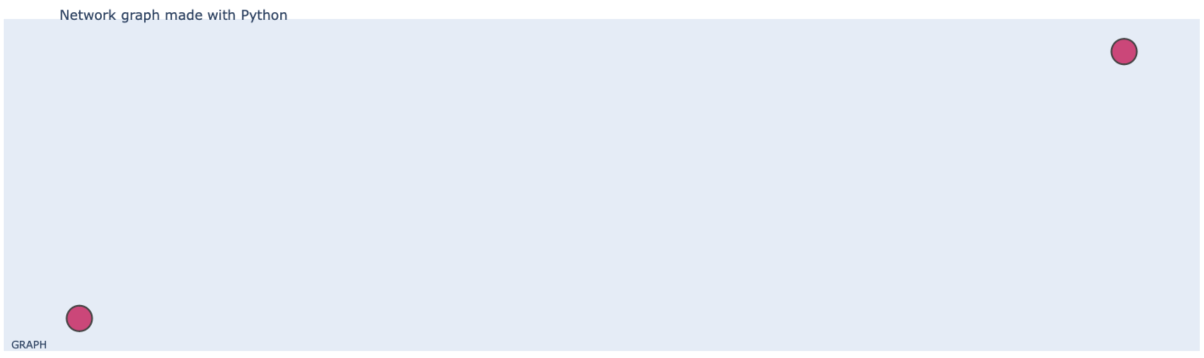
GRAPH COLORING BY FIRSTFIT - COMP6551

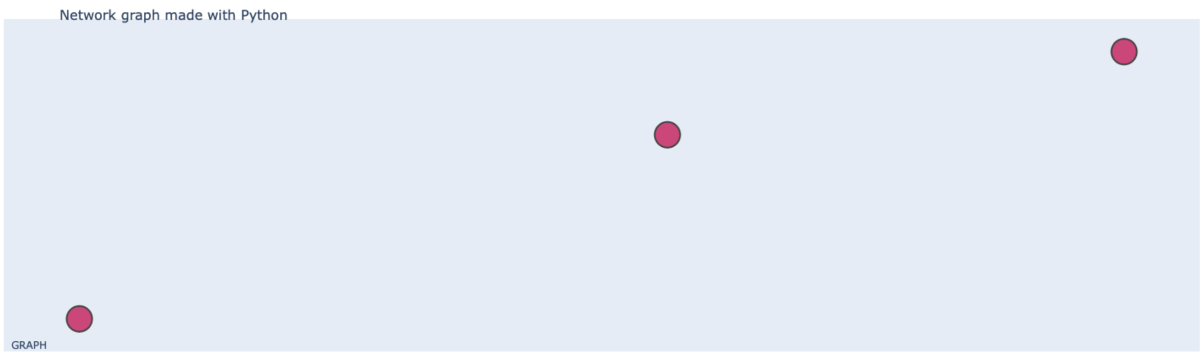
10
0.4
generate
Network graph made with Python
GRAPH
Add next Node
Network graph made with Python
GRAPH

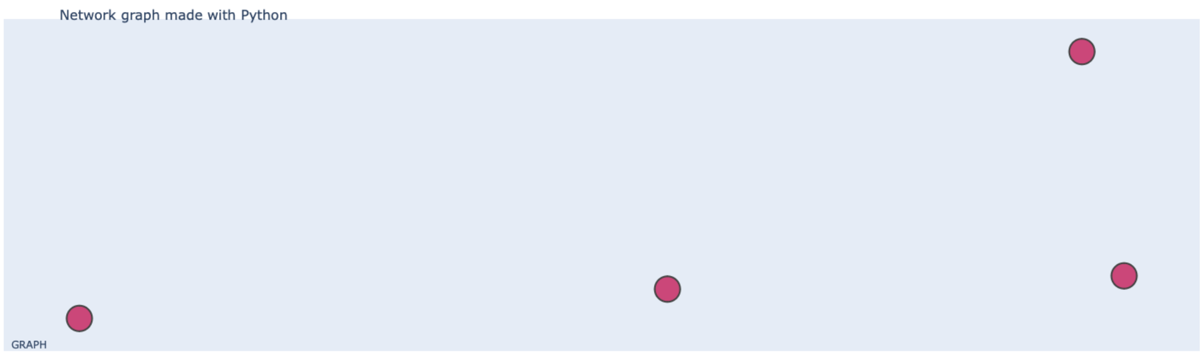
As we are using networkX to generate the graph above, the colour of the nodes might be different as the colour codes in that module are different from our first-fit coloured graph.

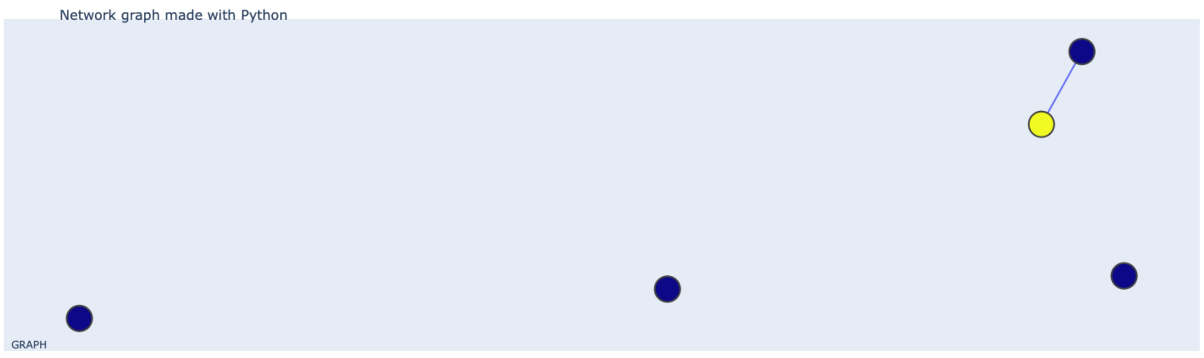
Step by step implementation

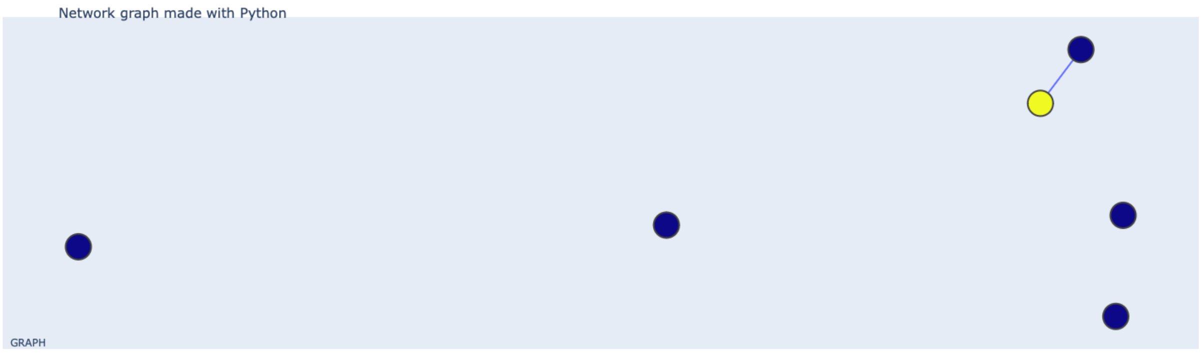


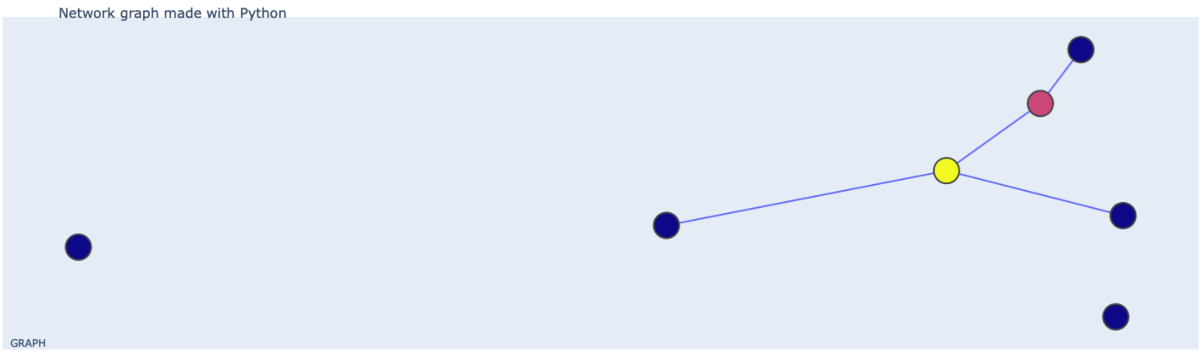

Network graph made with Python
GRAPH

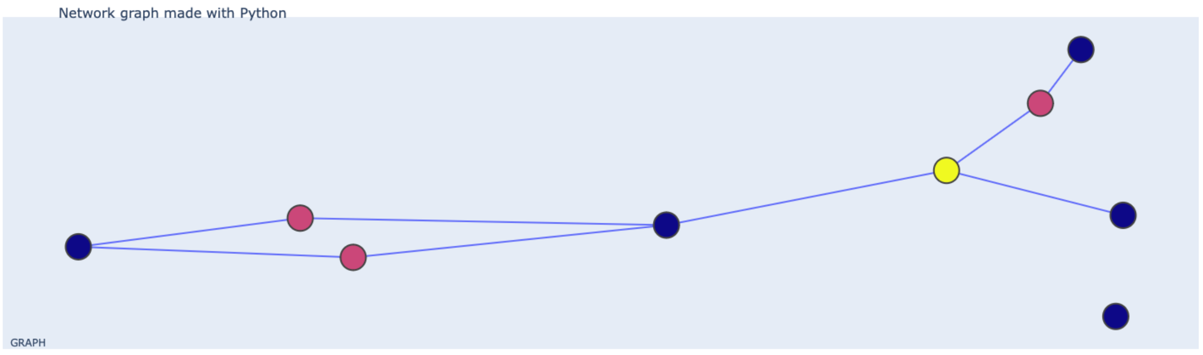

Network graph made with Python
GRAPH

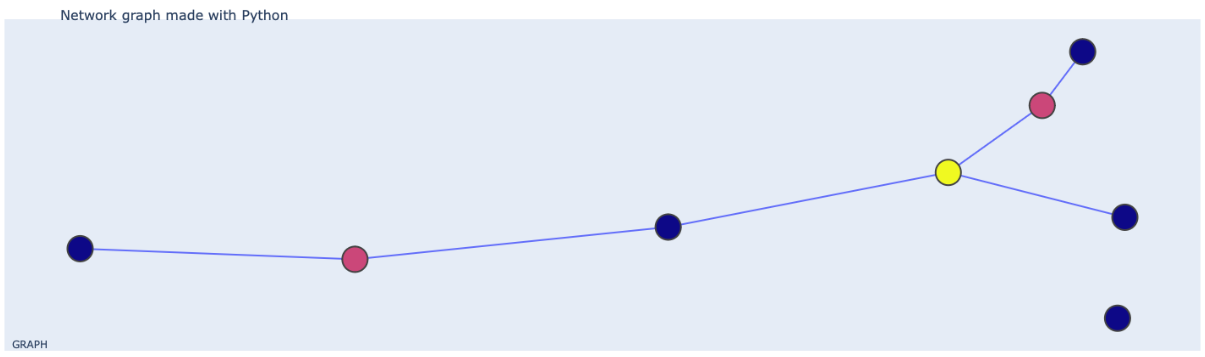

Network graph made with Python
GRAPH

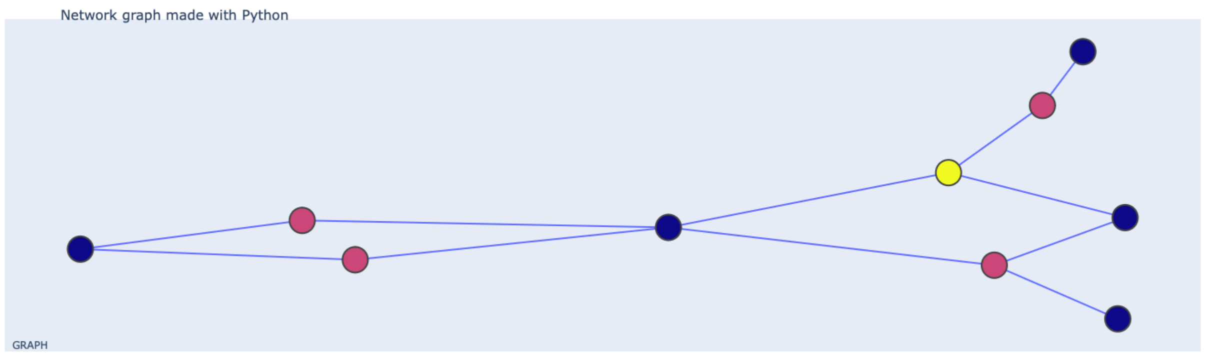

Network graph made with Python
GRAPH


Network graph made with Python
GRAPH


Network graph made with Python
GRAPH


Network graph made with Python
GRAPH





5. RESULTS

We generated a large enough data set of random triangle free graphs from number of vertices n range(25, 800) with several variances. We found the relationship of the average number of colours used for graph colouring compared to the corresponding number of vertices in the triangle free graph to be non-perfect linear.

N = number of graphs

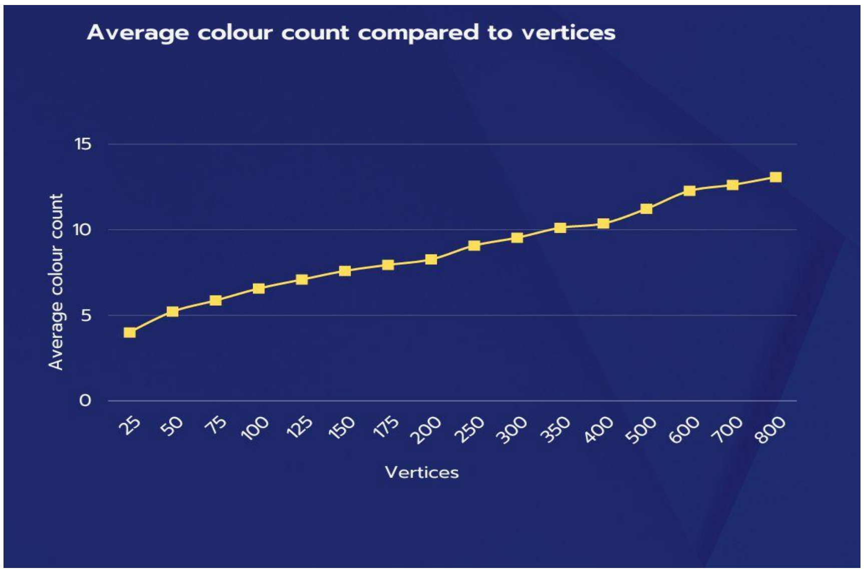
n = number of vertices

k = number of colours used

Dataset and Results:

N=100 n=?	25	50	75	100	125	150	175	200
k=?	3.99	5.21	5.86	6.56	7.08	7.59	7.94	8.26

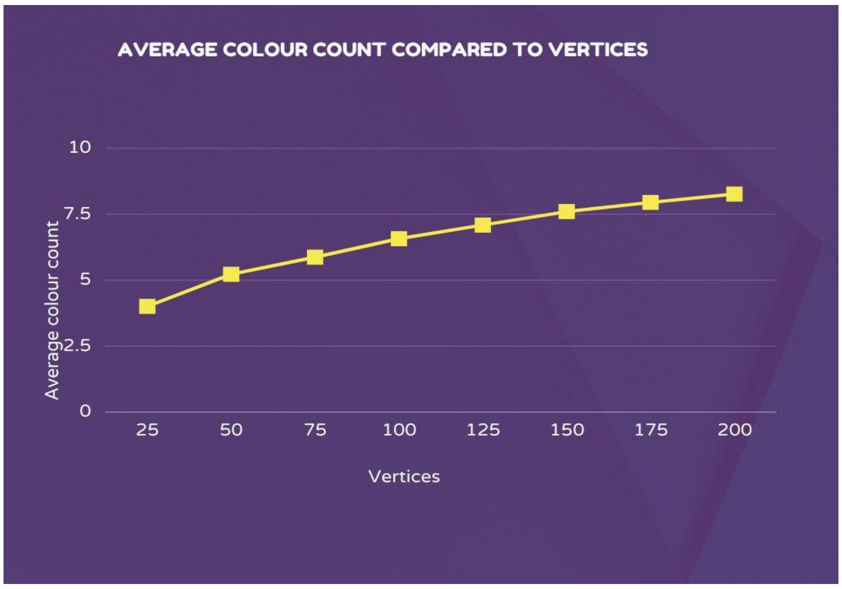
N=100 n=?	250	300	350	400	500	600	700	800
k=?	9.07	9.53	10.11	10.36	11.22	12.27	12.62	13.08



By changing the variation of the number of nodes to increase in intervals, as we increase the difference number of nodes from 25 to 50 to 100 we still see the same non-perfect linear relationship.

Variation of number of vertices: 25

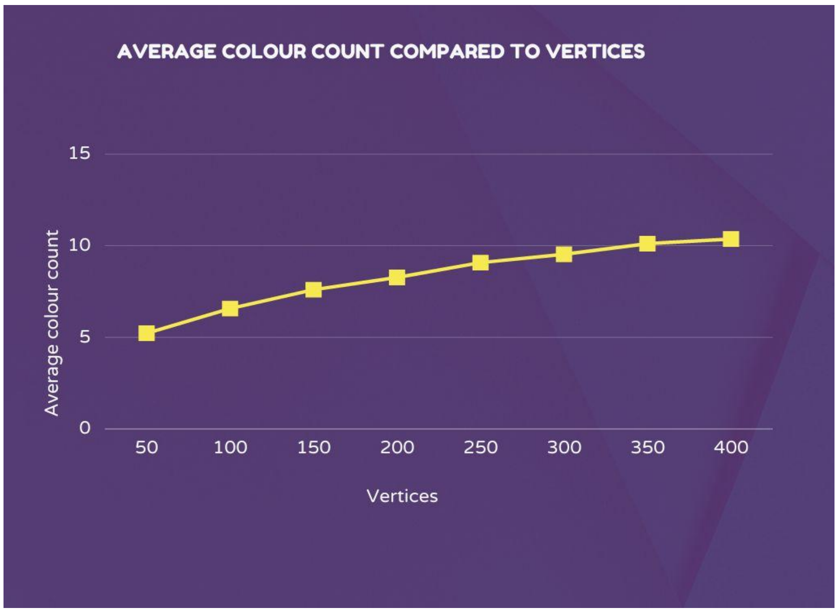
N=100 n=?	25	50	75	100	125	150	175	200
k=?	3.99	5.21	5.86	6.56	7.08	7.59	7.94	8.26



In the given table, we have the values of k for various values of n, where N(Number of Generated Graphs) is fixed at 100. The values of k in the table represent the average number of colours required to colour a graph of n=25 to 200 nodes with increment of 25 for n and with edges such that no two adjacent vertices connected by an edge have the same colour. From the given table, we can observe that as the value of n increases, the value of k also increases. This can be seen by the fact that the values of k in the table increase as we move from left to right, i.e., from n=25 to n=200.

Variation of number of vertices: 50

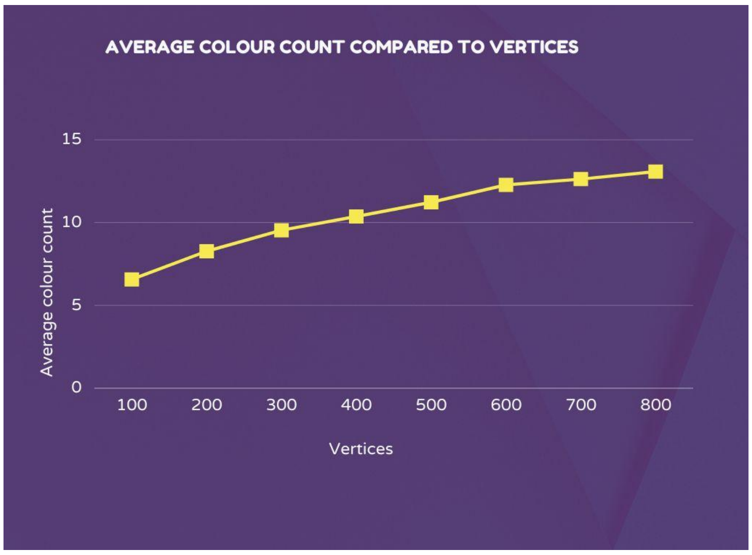
N=100 n=?	50	100	150	200	250	300	350	400
k=?	5.21	6.56	7.59	8.26	9.07	9.53	10.11	10.36



In the given table, we have the values of k for various values of n, where N(Number of Generated Graphs) is fixed at 100. The values of k in the table represent the average number of colours required to colour a graph of n=50 to 400 nodes with increment of 50 for n and with edges such that no two adjacent vertices connected by an edge have the same colour. From the given table, we can observe that as the value of n increases, the value of k also increases. This can be seen by the fact that the values of k in the table increase as we move from left to right, i.e., from n=50 to n=400.

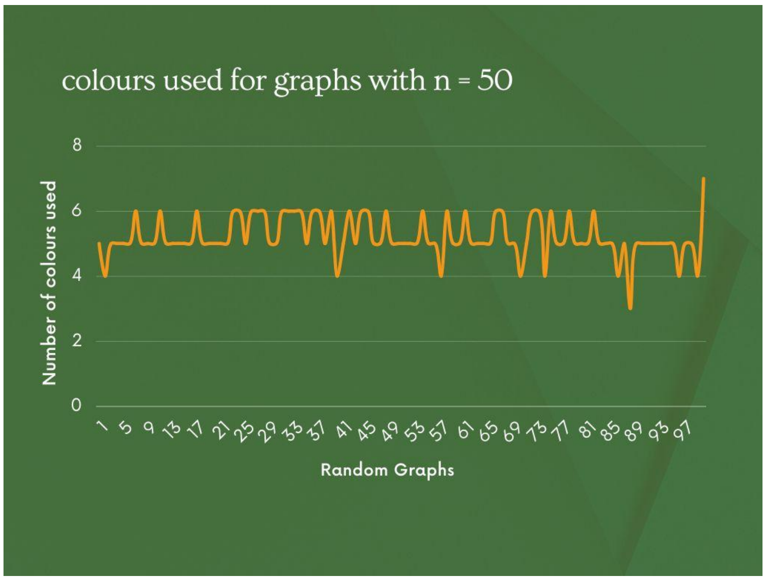
Variation of number of vertices: 100

N=100 n=?	100	200	300	400	500	600	700	800
k=?	6.56	8.26	9.53	10.36	11.22	12.27	12.62	13.08



In the given table, we have the values of k for various values of n, where N(Number of Generated Graphs) is fixed at 100. The values of k in the table represent the average number of colours required to colour a graph of n=100 to 800 nodes with increment of 100 for n and with edges such that no two adjacent vertices connected by an edge have the same colour. From the given table, we can observe that as the value of n increases, the value of k also increases. This can be seen by the fact that the values of k in the table increase as we move from left to right, i.e., from n=100 to n=800.

100 Graphs with number of vertices = 50



This data also shows a variation in the number of colours used for an equal number of vertices i.e n = 50. This relationship is heavily influenced by various peaks and valleys caused by the variance of types of graphs generated by the algorithm, which affects the number of colours used by our algorithm to colour the graphs.

6. CONCLUSION

This dependency of k on n can be explained by the fact that as the number of edges in a graph increases, the number of adjacent vertices also increases. This, in turn, increases the number of vertices that need to be coloured differently from their adjacent vertices, which requires more colours. Furthermore, the dependency of k on n can also be influenced by the nature of the graph being coloured. For example, certain types of graphs such as highly connected graphs(Triangle Free) may require more colours than other types of graphs with the same number of edges. In summary, the dependency of k on n can be influenced by various factors such as the number of edges in the graph, the nature of the graph, and the first-fit colouring algorithm used. However, in general, we can observe that as the number of edges in a graph increases, the number of colours required to colour the graph also increases.

Thus, in conclusion our analysis of the results from our algorithm that generated random triangle-free graphs showed a non-perfect linear(directly proportional) relationship between the average number of colours used for graph colouring and the corresponding number of vertices. This is because the generated graphs varied in connectivity, with some being highly connected and others sparsely connected, resulting in a non-linear relationship between the two variables.

7. TEAMWORK DISTRIBUTION

Jimi Mukeshchandra Mehta (40225526)	Triangle-Free Graph Generation Algorithm, First-Fit Algorithm, Implementation, Result Generation, Generating Result Test Data, Result Data Sheets, Result Analysis, Conclusion, Report
Rishab Kumar (40199196)	Code Generation, Triangle-Free Graph Generation Algorithm, First-Fit Algorithm, Implementation, Result Generation, Report, UI Coding and Implementation
Vaibhav Verma (40195571)	Test Data Analysis, Result Graph Generation, Result Analysis, Conclusion, Report, UI development
Vignesh Pugazhendhi (40230262)	Research on first-fit and triangle free graph generation, First-fit algorithm implementation, coding UI, Result analysis and report

Saketh Oppula (40221013)	Triangle free graph generation, UI implementation, Conclusion in report
Amrita Awasthi (40203422)	Research on Problem Statement (Online graph algorithm, Triangle-free algorithm, Fist-fit online graph algorithm), Introduction, Background information, Implementation, Analysis on result and providing insights, Report.

8. References

[1] Yaqiao Li, Vishnu V. Narayan, Denis Pankratov, "Online Colouring and a New Type of Adversary for Online Graph Problems" May 25, 2020

[2] Susanne Albers and Sebastian Schraink. Tight bounds for online coloring of basic graph classes. In 25th Annual European Symposium on Algorithms (ESA 2017). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2017.

[3] Maria Paola Bianchi, Hans-Joachim B¨ockenhauer, Juraj Hromkoviˇc, and Lucia Keller. Online coloring of bipartite graphs with and without advice. Algorithmica, 70(1):92–111, 2014.

[4] Andr´as Gy´arf´as and Jen¨o Lehel. On-line and first fit colorings of graphs. Journal of Graph theory, 12(2):217–227, 1988.

[5] Hal A Kierstead. On-line coloring k-colorable graphs. Israel Journal of Mathematics, 105(1):93–104, 1998.

[6] https://en.wikipedia.org/wiki/Triangle-free_graph

[7] https://arxiv.org/pdf/2005.10852.pdf

[8] Aric A. Hagberg, Pieter J. Swart, and Daniel S. Chult. NetworkX. Available online at https://networkx.org, 2008-2021.