Algorithms For Dynamic Networks

Saiteja Pendyala
(201538148)

A DISSERTATION

Submitted to
The University of Liverpool

in partial fulfilment of the requirements
for the degree of

MASTER OF SCIENCE

Abstract

Graph models have become more important in nearly every scientific subject, including mathematics, biology, neuroscience, medicine, social networks as well as computer technology. Graphs, unlike other data formats, allow the designing of relationships between entities. Networks, in combination with the vast amount of data available, open up a wide range of design options for theoretical and real-world situations. In graph theory, the study of the structure of graphs has been based on the idea that they are static. However, the graphs that emerge in many applications evolve, making them dynamic.

In recent years there has been an increased interest in analysing the algorithmic foundations and principles involved in various biological processes and discovering the state-of-art algorithmic techniques, as they provide an effective solution to many existing situations. Interesting recent research in this direction was "The complexity of a growing graph" [1], this study provides polynomial-time algorithms that find construction schedules for dynamic graphs that belong to specific graph classes.

$G_t = (V_t, E_t)$ for $t = 1, 2, \ldots, k$ is designed as an undirected growing graph which is highly dynamic where each node generates new nodes in every round(slot) and thus increases the size of the graph. As a result, the difficulty arises in constructing a target network 'G' in an optimal number of time slots from the initial node. To model this, we assume that in every time slot each parent node 'u' can only generate one child node 'v', and that each child node 'v' is connected to its parent node by default, and can activate edges with other nodes only at the time of its birth, as long as these nodes are within a minimal edge-activation distance 'd'.

This dissertation provides insights on experimental evaluation of the theoretical bounds of the graph models contributed in the study "The complexity of a growing graph". Experimental evaluation includes algorithm analysis, algorithm implementation, experimentation and verification of the line and star dynamic graph models.

Student Declaration

I confirm that I have read and understood the University's Academic Integrity Policy.

I confirm that I have acted honestly, ethically and professionally in conduct leading to assessment for the programme of study.

I confirm that I have not copied material from another source nor committed plagiarism or fabricated data when completing the attached piece of work.

I confirm that this dissertation constitutes my work, that where the language of others is set forth, quotation marks so indicate, and that appropriate credit is given where I have used the language, ideas, expressions or writings of another.

I confirm that I have not copied material from another source, nor colluded with any other student in the preparation and production of this work.

I confirm that the dissertation describes original work that has not previously been presented for the award of any other degree of any institution.

SIGNATURE: Saiteja Pendyala

DATE: 27-01-2022

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor, Professor Othon Michail, for his sincerity, excellent instruction, and support, which I will never forget. His guidance from the beginning of the research allowed me to gain a better understanding of the details of my project. I'm grateful for the incredible opportunities he gave me to advance professionally.

I am grateful to my parents for their unwavering love and support, which helped me to stay motivated and confident. They helped me achieve my goals and success because they believed in me. My brother, who has always been supportive of me throughout the thesis process, deserves special mention.

Contents

List of Figures

CHAPTER 1. INTRODUCTION

1.1 Summary of Project Proposal

1.1.1 Problem Statement

This project aims to perform the experimental evaluation of the dynamic graph algorithms.

The key goals of the project are the following.

1. To study and understand the properties of the dynamic graph models.
2. To implement the line and star dynamic algorithms proposed in the research article "The complexity of a growing graph".
3. To test and experiment with the line and star graph algorithms.
4. To verify the theoretical bounds of the graph models stated in the study.

1.1.2 Project Methodology

The experimental evaluation of the line and star graph models has four objectives: algorithm analysis, implementation, testing, experimentation and evaluation.

To analyse the algorithms, I will understand the algorithmic properties of line and star graph models, I will analyse the observations of the dynamic graph templates and understand the process of graph construction. I will deeply study the methods of node generation, edge activation and deletion at each time slot. I will review the concept of algorithm time complexity to understand the theoretical proofs of dynamic algorithms.

I will implement the line graph and star graph algorithms in the python programming language.

To test and experiment with the algorithms, I will manually compute the construction of the graphs with a smaller instance, observe the method of graph construction at each time slot and verify the output of the software with the manual output. I want to experiment the graph models with larger instances and analyse the performance of the software.

To evaluate the dynamic graph algorithms, I will verify the time complexities of the implemented graph algorithms with the theoretical bounds stated in the study.

1.2 <u>Structure of Report</u>

The report is structured as follows: Chapter 1. introduces the problem statement and aims of the project and project methodology. Chapter 2. provides an overview of the background literature required to understand the details of the proposed graph models and to perform an experimental evaluation of the theoretical proofs of the dynamic algorithms. Chapter 3. describes the data required to perform the algorithm. Chapter 4. explains the design methodology involved in the algorithm evaluation. Chapter 5. highlights the implementation of the dynamic algorithms with information regarding the tools and results obtained. Chapter 6. explains the evaluation of the algorithms and also includes the observations. Chapter 7. explains the learning points achieved while working on the project. Chapter 8. describes the project concerning the British Computer Society's (BCS) code of conduct. Chapter 9. Summarises and explains the future work needed in the project.

Chapter 2. Background and review of literature

This chapter aims to explain the background research done in this project to understand the details of the dynamic graph algorithms.

2.1 Motivation

In the past century, and quite naturally, computing research has primarily concerned itself with understanding the fundamental principles of numerical computation. This has led to the development of a beautiful theory of computation and its various subtheories including complexity theory, distributed computing, algorithmic theories of graphs, learning, and games. Driven by the realisation of the fact that many natural processes are essentially algorithmic and by the advent of some low-cost and accessible technologies for programming and controlling various forms of matter, there is a growing recent interest in abstracting biological processes and formalising their algorithmic principles and inventing new algorithmic techniques suitable for the existing technologies [1].

Significant examples of algorithmic methodologies that are inspired by biochemical events are ant colony optimisation, brain computation, dynamically growing networks. Also, there is continuous research in designing the algorithmic foundations of dynamic graphs such as dynamic distributed computing models, structural and algorithmic properties of temporal graphs and the investigation of dynamic networks which has been developing recently [1].

Dynamic networks are graphs that change with time. Graph models introduced in the research article ”The complexity of a growing graph” are highly dynamic and are motivated by a biological process known as embryogenesis. Embryogenesis is a highly complicated process. It involves cellular division and differentiation, controlled tissue growth, and morphogenesis[1].

The work contributed in the study “The complexity of a growing graph” provides distributed and centralised polynomial-time algorithms that find construction schedules for graphs that belong to specific graph classes. Experimental evaluation and verification of the theoretical bounds of the line and star dynamic graph algorithms is the motivation for my project.

2.2 Background Research

The goal of my project is to perform an experimental evaluation and to verify the theoretical bounds of the line and star dynamic graph algorithms proposed in the study “The complexity of a growing graph”.

2.2.1 Dynamic Graphs

Dynamic graphs are an essential tool for representing a wide variety of concepts that evolve with time. Examples include modelling the evolution of relationships and communities in a social network or tracking the activity of users within an enterprise computer network. In the case of static graph representations, random graph models are often useful for analysing and predicting the characteristics of a given network. Even though random dynamic graph models are a trending research topic, the field is still relatively unexplored [7].

A dynamic graph G = (V_t, E_t) for $t = 1, 2, . . . , k$ is designed as an undirected growing graph, the graph initially consists of only a single node ‘u_1’ which is known as the initiator. During the execution, the graph gradually changes by the addition of nodes and/or deletion of the edges. Let ‘u’ be a node that generates a node ‘w’ at time slot ‘t’, now we consider ‘u’ is the parent of child ‘w’, initially each newly generated child is connected to its parent and also can additionally activate some edges at the time of its birth ‘t’, with any node ‘v’ such that the distance between ‘w’ and ‘v’ is at most ‘d’, where ‘d’ is the edge activation distance between nodes[1].

2.2.2 Operations of Growing Graphs

The dynamic graph models allow three operations. Node generation, edge activation and edge deletion.

1) **Node generation:** In each slot 't', each node in the previous slot 't-1' can make a decision either to generate new nodes or not.
2) **Edge-activation:** The set of nodes that were generated at slot 't', in addition to the edge with its parent, can activate edges with nodes that are within its edge-activation distance 'd' at the time of its birth, including nodes that are generated in that round.
3) **Edge deletion:** In each slot 't', any node can decide to remove some of its edges.

2.2.3 Line Graph Algorithm

Line graph algorithm generates a construction schedule for a line graph G with [log n] rounds. Let G = (V, E) be the line graph of size 'n' that needs to construct, where $V = \{u_1, u_2, . . . , u_n\}$, and each $u_i \in V$, $2 \leq i \leq n - 1$, is connected with u_{i-1} and u_{i+1}. The strategy here the algorithm follows similarly construct a line in exactly [log n] slots is that in each time slot 't', all nodes of target graph 'G_t' must generate new nodes, except the last node. Therefore, the problem here is to assign to each node a set of nodes that it needs to generate throughout all time slots while guaranteeing that the final graph will have a spanning line as a sub-graph [1].

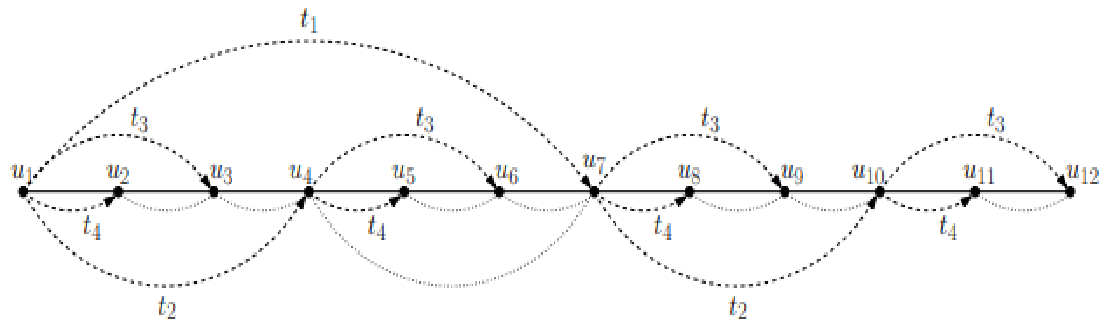


Figure 2.1: Line graph construction

A line graph is constructed in a way where every node lies on the same column or the same row. In particular, at time slot 't', each node 'u_i' that belongs to the target graph 'G_t' generates a new node $u_j = u_{i+[n/2^t]}$ if $u_j \in V$, then the edge 'u_iu_j' is activated. Also $u_j = u_{i+[n/2^{(t-1)}]} \in V_t$, then the edge $u_ju_{i+[n/2^{(t-1)}]}$ is activated. This means that the initial node u_1 generates the node that will be in the middle of the line. This procedure is repeated until it generates its neighbour on the line graph G. Similarly, in each slot t, all nodes construct the middle node between themselves and the next node of the line that was generated in the previous slot $t - 1$. In case such a node does not exist, they generate the middle node between themselves and the end of the line. Finally, each node u_j which is generated by u_i activates an additional edge with the next node of the line that u_i has an edge with (if any) [1].

An excess edge in the line graph is activated during the execution of the algorithm. Edges that are activated in previous rounds and considered as excess edges.

2.2.4 <u>Star Graph Algorithm</u>

Star graph algorithm computes the construction schedule of a graph of size 'n' in polynomial time with [log n] slots and O(n) excess edges. Graph 'G_t' has a star subgraph with node 'u' as its centre. Now, each node that is generated in the ‘t+1’ time slot is in distance two(allowed edge-activation distance(d) between the nodes is 2 i.e$'d' = 2$) from the centre ‘u’ and thus it can activate an edge with ‘u’. This means every child node that is generated by leaf nodes (other than the centre node ‘u’) activates an edge with its parent node(leaf node) and then it disconnects the edge with its parent node and connects the centre node ‘u’. This process is repeated until the slot [log n], where n is the size of the graph [1].

An excess edge in a star graph is activated during the execution of graph construction and is eventually deleted at the end of the round(time slot) ‘t’. Any edge that is activated between two leaf nodes is considered an excess edge. For every node generation, there are at most two edge activations.

2.2.5 <u>Computational Complexity of an Algorithm</u>

In theoretical analysis of algorithms, it is common to estimate their complexity in the asymptotic sense, i.e., to estimate the complexity function for arbitrarily large input. Algorithm analysis is an important part of computational complexity theory, which provides theoretical estimation for the required resources of an algorithm to solve a specific computational problem. Most algorithms are designed to work with inputs of arbitrary length. Analysis of algorithms is the determination of the amount of time and space resources required to execute it. The efficiency or running time of an algorithm is stated as a function relating the input length to the number of steps, known as time complexity, or volume of memory, known as space complexity [2].

Asymptotic notations are mathematical notations that are used to express the runtime of an algorithm when the input tends towards a specific value or a limiting value. Asymptotic notations are classified into three types: Theta notation, the Omega notation, and the Big-O notation. These are used to calculate the algorithm's runtime complexity[4].

2.3 Project requirements gathering

2.3.1 Programming language and libraries

1. **Python:** I have performed the experimental evaluation of algorithms in python to enhance my programming skills in python.
2. **Math library:** The math library in python provides access to some common math functions and constants for complex mathematical computations. I have used the math library for the process of calculations concerning the node generation method [5].
3. **Matplotlib library:** The matplotlib library in python helps to understand the huge amount of data through different visualisations. I have used the matplotlib library to analyse the time complexity of the dynamic graph models for the larger instances[6].

2.3.2 Requirements Management

At the beginning of my project, I got the overall project requirements from the e-project system[3] of COMP702 where it explains the details of the project goals mentioned in section 1.1.1. Later I have discussed with my primary supervisor and gathered the expected outcomes from the project.

2.3.3 Project Plan

The Project plan is outlined in the Gnatt chart in Figure 2.1

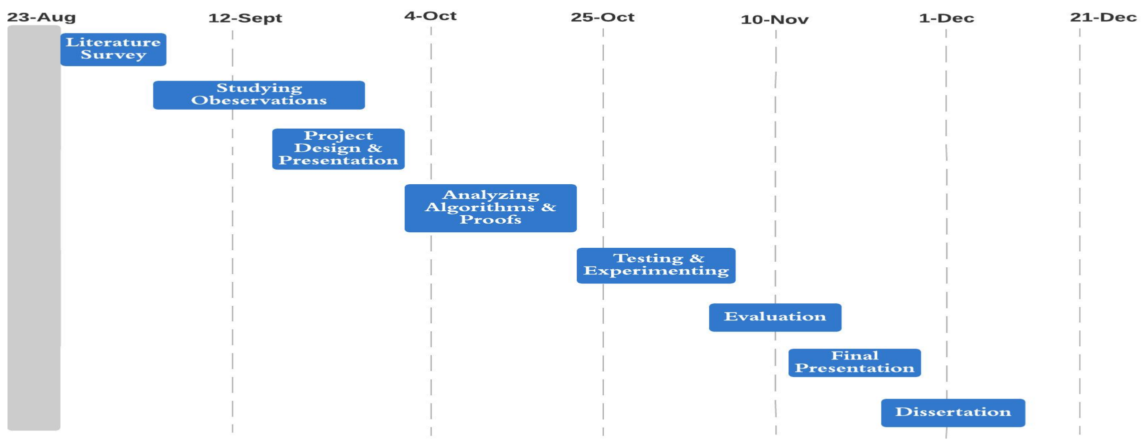


Figure 2.2: Gantt chart

This project was divided into the following stages.

Literature survey

This stage of the project involved background reading and research to gather the resources needed for the project. The above included the decision of programming language and required libraries.

Studying observations

At this stage, I will study the observations of the dynamic graph models proposed in the study “The complexity of a growing graph”.

Design

At this stage, I will have the design and specifications outlined. I have prepared my design for the plan of execution of the project with the details of algorithm implementation, experimentation and evaluation.

Analysing algorithms and proofs

In this stage, I will study, analyse the proof of line and star graph algorithms and understand the method of generation of the construction schedules of the dynamic graph models.

Testing and experimenting

At this stage, I will implement the line and star dynamic graph models. I will perform algorithm experimentation with large instances.

Evaluation

This stage of the project includes verifying the runtime analysis of the line and star graph models with the theoretical bounds started in the study “The complexity of a growing graph”.

Dissertation write-up

The dissertation write-up includes a record of what has transpired during the project. It contains detailed information on the purpose, outcomes, achievements and challenges of my project.

Dissertation review and Submission

At this last stage of the project, I plan to review my final draft, make any modifications needed and submit my dissertation.

Chapter 3. Ethical Use Of Data

The data I have used to test and experiment with the dynamic graph models are artificially created to analyse the performance of the software and to perform the verification of the dynamic graph algorithms contributed in the study "The complexity of a growing graph".

There is no personally identifiable information presented within this dissertation.

.

Chapter 4. Design

4.1 Literature review

The first stage of the design involves the reading and understanding of the approach of dynamic networks proposed in the study “The complexity of a growing graph”. I have started by understanding the growing mechanisms of nature, which have been long inspiring researchers to discover and build artificial systems resembling the study of abstraction algorithmic knowledge of nature.

The work presented in the study “The complexity of a growing graph” is about the networks which are highly dynamic and this research was motivated by a biological process known as embryogenesis in which organisms have a remarkable ability to develop from a single entity and can grow into well-defined global networks and structures.

The algorithmic study of dynamic networks is a relatively new area of research, which is concerned with studying the algorithmic and structural properties of networked systems whose structure changes with time. The model of growing graphs is a type of temporally changing graph in which the set of vertices is non-decreasing, and in contrast, they strictly increase, the edges in the graph models follow some local dynamic constraints. Specifically, the edge can only be activated when the birth of its latest-born endpoint, then remains constantly active for one or more time slots ‘t’, and can be deactivated at most once.

The goal of the project is to perform an experimental evaluation of the line and star dynamic graph algorithms.

4.2 Experimental Evaluation Design

Algorithm experimentation and verification are the two main objectives of my project as this project is concerned with the experimental evaluation of the dynamic graph models.

I have divided the experimental evaluation into four stages as shown in figure 4.1

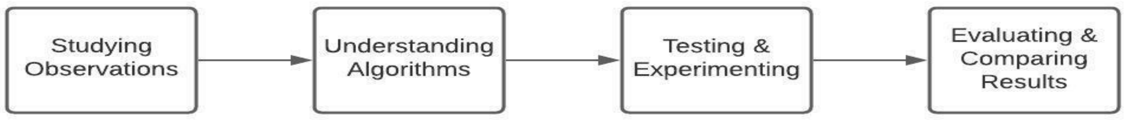


Figure 4.1: Flow diagram for the experimental evaluation of the algorithms

The first stage of the experimental evaluation involves studying the observations of the dynamic algorithms. The second stage is about understanding the line and star graph algorithms. The third stage is algorithm testing and experimenting, which is the most crucial requirement of the project. In the testing and experimenting stage, I will test the performance of implemented dynamic graph models with larger instances. In the last stage, I will evaluate the output of my software with the theoretical bounds stated in the research "The complexity of a growing graph".

4.2.1 Implementation Design

I have selected the Python programming language to implement and verify the theoretical bounds of the algorithms proposed in the study "The complexity of a growing graph". I will be using the "math" library in python to calculate the process of the node generation. I will use the "matplotlib" library in python to analyse the time complexities of the line and star graph models.

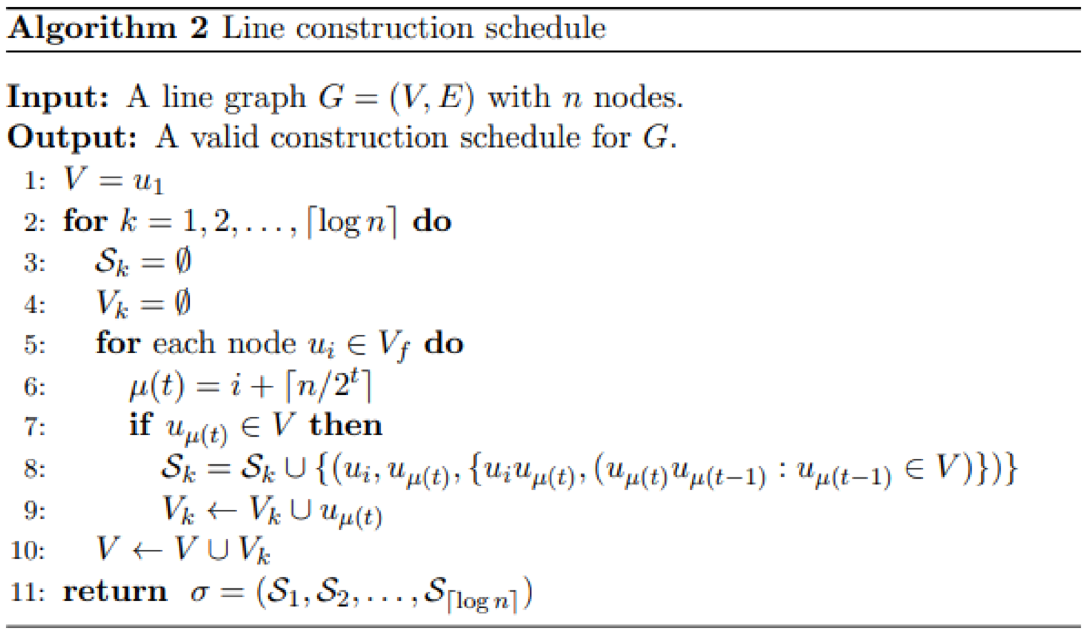
Line Graph Implementation design

The fundamental approach for constructing a line graph of size ‘n’ in exactly [log n] slots is that all nodes of a target graph ‘G_t’ must generate new nodes in each time slot ‘t’, except for the last node.

I want to use a dictionary to store the line graph constructed in every slot ‘t’. To store the excess edges that are generated in each slot I will use a separate dictionary.

The graph, at time slot ‘t = 0’, consists only of a single node ‘u_1’ and here the graph does not have any excess edges. An excess edge of a graph is activated during the execution of line graph construction and is eventually deleted at the end of the round(time slot) ‘t’.

I will use a list to count the total excess edges formed in every round. To store the total number of rounds required to construct the graph I want to use a separate list.



Algorithm 2 Line construction schedule

```
Input: A line graph $G = (V, E)$ with $n$ nodes.
Output: A valid construction schedule for $G$.
 1: $V = u_1$
 2: for $k = 1, 2, \ldots, \lceil \log n \rceil$ do
 3:     $\mathcal{S}_k = \emptyset$
 4:     $V_k = \emptyset$
 5:     for each node $u_i \in V_f$ do
 6:         $\mu(t) = i + \lceil n/2^t \rceil$
 7:         if $u_{\mu(t)} \in V$ then
 8:             $\mathcal{S}_k = \mathcal{S}_k \cup \{(u_i, u_{\mu(t)}, \{u_i u_{\mu(t)}, (u_{\mu(t)} u_{\mu(t-1)} : u_{\mu(t-1)} \in V)\})\}$
 9:             $V_k \leftarrow V_k \cup u_{\mu(t)}$
10:     $V \leftarrow V \cup V_k$
11: return $\sigma = (\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_{\lceil \log n \rceil})$
```

Figure 4.2: Pseudocode for Line graph construction

I will output each line graph that has been constructed in every round with excess edges activated in every time slot and the total number of excess edges formed in each time slot.

More details on the implementation of the line graph will be explained in section 5.1.1.

Star Graph Implementation design

The star graph algorithm computes the construction schedule of a graph of size ‘n’ in polynomial time with [log n] slots and $\Theta(n)$ excess edges. Here each node generated at ‘t+1’ time slot is in distance two from the centre ‘u’ and thus it can activate an edge with ‘u’. In every round ‘t’ each node ‘u_i’ that belongs to ‘t-1’ time slot will generate a child node ‘u_j’ and every new edge $u_i --> u_j : u_i \neq u_1$ is disconnected with the leaf node ‘u_i’ and connects with the centre of the star graph ‘u_1’.

I want to use a dictionary to store the star graph constructed in every slot ‘t’. I will use a list to store the total nodes that are generated. To store the total number of excess edges formed in every round I will use a list.

I will output each star graph that has been constructed in every time slot ’t’, excess edges that are generated, new nodes along with the edges activated in every slot. At the time slot, the ‘t = 0’ graph consists only of a single node which is the centre of the graph ‘u_1’ and here the graph does not have any excess edges.

More details on the implementation of the star graph will be explained in section 5.1.2.

4.2.2 Experimentation Design

I will experiment with software with larger instances and analyse the performance of the software. To analyse the time complexity of the line and star graphs of size ‘n’, by using the “matplotlib” library from python, I will plot a graph that relates the total number of nodes generated and total number slots.

To analyse the complexity of the excess edges that are generated during the execution of the line and star graphs I will plot a graph that relates the total number of excess edges generated and the total number of vertices generated.

4.2.3 **Testing and Evaluation Design**

To test the graph construction in every round, I will analyse the method of the node generation, edge activation and edge deletion in every slot respectively for the graph of a smaller instance according to the algorithms proposed in the research article ”The complexity of a growing graph”.

I will manually compute the process of construction of the graphs in every time slot for the smaller instance of the line and star graphs and I will verify the manually calculated construction schedule with the output of the software.

I will evaluate the efficiency of the algorithms by analysing the runtime complexity of the line and star graphs. Finally, I will verify the complexity of the graphs with the theoretical bounds stated in the research article “The complexity of a growing graph”.

Chapter 5. Realisation

5.1 Implementation

I have implemented the design explained in Chapter 4. A brief explanation of the key elements involved in algorithm implementation is as follows and please refer to Appendix B for the screenshots of implementation of the line and star graph models.

5.1.1 Line Graph Implementation

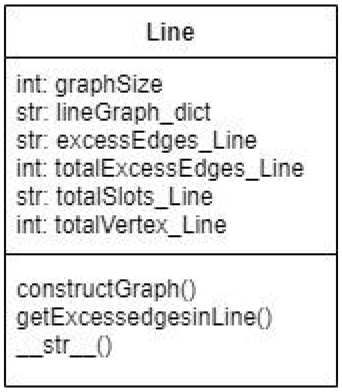


Figure 5.1: Class diagram for line graph algorithm

- **Attributes of the Line**
    - **graphSize**

      The variable 'graphSize' allows integer values as the size of the graph.

    - **lineGraph_dict**

      Variable 'lineGraph_dict' is a dictionary. I have used this variable to store the line graph constructed at each time slot 't'. It stores the slot and the list of nodes generated in each slot as the key-value pair.

- **excessEdges_Line**

  Variable ‘excessEdges_Line’ is a dictionary. I have used this variable to store the excess edges that are formed during the execution of the graph at each slot ‘t’. It stores the slot and the list of excess edges formed in each slot as the key-value pair.

- **totalExcessEdges_Line**

  This variable is a list and I have used ‘totalExcessEdges_Line’ to store the total number of excess edges formed in each slot ‘t’.

- **totalSlots_Line**

  Variable ‘totalSlots_Line’ is a list. I have used this variable to store the total number of slots required to construct the line graph

- **totalVertex_Line**

  This variable is a list and it has all the nodes that are generated in the line graph construction

● **Methods of the Line**

- **constructGraph( )**

  I have taken a variable ‘lineNodes’ as a list and initialised it to 1 to represent the initiator ‘u_1’. The list ‘lineNodes’ stores the list of nodes that will be generated during the execution. I have used the ‘math.log( )’ method from the ‘math’ library from python to return the natural logarithm to base 2 of the graph size. I have used the ‘math.ceil( )’ method to return the integral value greater than the natural logarithm of the size of the graph. For each time slot within range ‘log n’ where n is the size, each parent node in the ,list ‘lineNodes’ generates a child node. I have taken the variable ‘nextNode’ to calculate the child node. I used a dictionary ‘lineGraph_dict’ to store the graph constructed at each slot ‘t’. This method constructs the line graph of size ‘n’ in ‘log n’ slots.

- **getExcessedgesinLine( )**

  This method identifies the excess edges that are formed at each slot during the execution of the graph.

- **__str__( )**

  This method prints the line graph constructed at each slot ‘t’, shows the excess edges formed at each slot ‘t’ and shows the complexity of constructing a line graph and the complexity of the excess edges constructed during the execution of the algorithm.

5.1.2 Star Graph Implementation

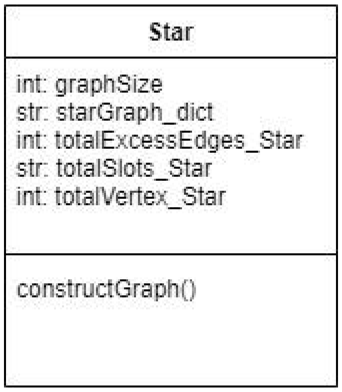


Figure 5.2: Class diagram for star graph algorithm

- **Attributes of the Star**
    - **graphSize**

      The variable ‘graphSize’ allows integer values as the size of the star graph.

    - **starGraph_dict**

      Variable ‘starGraph_dict’ is a dictionary. I have used this variable to store the star graph constructed at each time slot ‘t’. It stores the slot and the list of nodes generated at each slot as the key-value pair.

    - **totalExcessEdges_Star**

      This variable is a list and I have used ‘totalExcessEdges_Star’ to store the total number of excess edges formed at each slot ‘t’.

    - **totalSlots_Star**

      Variable ‘totalSlots_Star’ is a list. I have used this variable to store the total number of slots required to construct the star graph

    - **totalVertex_Star**

      This variable is a list and it has all the nodes that are generated in the star graph construction

- **Methods of the Star**
    - **constructGraph( )**

      I have taken a variable ‘starNodes’ as a list and initialised it with 1 to represent the centre of the star graph ‘u_1’. The list ‘starNodes’ stores the list of nodes that will be generated during the execution. I have used the 'math.log( )' method from the 'math' library from python to return the natural logarithm to base 2 of the star graph size of 'n'. I have used the 'math.ceil( )' method to return the integral value greater than the natural logarithm of the size 'n' of the star graph. For each time slot within range 'log n' where n is the size, each parent node in the list 'starNodes' generates a child node. I have taken the variable 'nextNode' to calculate the child node. I used a dictionary 'starGraph_dict' to store the graph constructed at each slot 't'.

This method identifies and prints the excess edges that are constructed during the execution of the star graph at each slot 't'. It prints the new nodes that are generated at each slot. It shows edges that are activated during the construction of the graph in every round. It shows the star construction schedule for graph size of 'n'. Finally shows the complexity of construing a star graph and the complexity of the excess edges in the star graph.

5.2 Experimentation

I have experimented with the software with larger instances to test the performance of the algorithms.

5.2.1 Line Graph Experimentation

I have experimented with the line graph algorithm with larger instances and analysed the performance of the software, please refer to Appendix A for the code screenshots of experimentation of the line graph algorithm.

5.2.2 Star Graph Experimentation

I have experimented with the star graph algorithm with larger instances and analysed the performance of the software, please refer to Appendix A for the code screenshots of experimentation of the line graph algorithm.

5.3 Testing and Evaluation

To test the line and star algorithms, I have manually implemented the construction schedules for the smaller instance of graph size ‘n = 16’. I have tested the output of the software of the graph size ‘n = 16’ compared it with the manually computed graph construction data and verified the construction schedules of the graphs.

To evaluate the runtime analysis of construction schedules of the graph models, I have used the “matplotlib” library to plot the complexity graph that relates the number of nodes generated and the number of time slots required to construct the graph. To verify the complexity of excess edges of the graphs I have plotted a complexity graph that relates the total number of the excess edges formed during the execution of the graphs and the total number of nodes that are generated.

5.3.1 Line Graph Testing

The construction schedule of the line graph of size ‘n =16’ is computed in [log 16] base two which is in four slots by the design of the line construction algorithm.

- **Manual testing and verification**

For the slot ‘t1’:

parent node = u_1; child node = i + [$n/2^t$] where ‘i’ is the index of the node, ‘n’ is the size of the graph and ‘t’ is the number of time slots.

i = 1; n = 16; t = 1

child node = $1 + [16/2^1]$; $1 + [8] = 9$; child node = u_9

Line graph at slot t1: $u_1 - u_9$

For the slot ’t2’:

Every node in the previous slot generates a child node respectively.

parent nodes = u_1, u_9

i = 1; n = 16; t = 2

child node = $1 + [16/2^2] = 5$; child node = u_5; line graph = $u_1 - u_5$

i = 9; n = 16; t = 2

child node = $9 + [16/2^2] = 13$; child node = u_{13}; line graph = $u_9 - u_{13}$

Excess edges at t2: u_1u_9

Line graph at slot t2: $u_1 - u_5 - u_9 - u_{13}$

For the slot ‘t3’:

parent nodes = u_1, u_5, u_9, u_{13}

i = 1; n = 16; t = 3

child node = $1 + [16/2^3] = 3$; child node = u_3; line graph = $u_1 - u_3$

$i = 5$; $n = 16$; $t = 3$

child node = $5 + [16/2^3] = 7$; child node = u_7; line graph = $u_5 – u_7$

$i = 9$; $n = 16$; $t = 3$

child node = $9 + [16/2^3] = 11$; child node = u_{11}; line graph = $u_9 – u_{11}$

$i = 13$; $n = 16$; $t = 3$

child node = $13 + [16/2^3] = 15$; child node = u_{15}; line graph = $u_{13} – u_{15}$

Excess edges at t3: u_1u_5, u_9u_{13}, u_5u_9, u_1u_9

Line graph at slot t3: $u_1 – u_3 – u_5 – u_7 – u_9 – u_{11} – u_{13} – u_{15}$

For the slot 't4':

parent nodes = $u_1,u_3,u_5,u_{7,}u_9,u_{11},u_{13},u_{15}$

$i = 1$; $n = 16$; $t = 4$

child node = $1 + [16/2^4] = 2$; child node = u_2; line graph = $u_1 – u_2$

$i = 3$; $n = 16$; $t = 4$

child node = $3 + [16/2^4] = 4$; child node = u_4; line graph = $u_3 – u_4$

$i = 5$; $n = 16$; $t = 4$

child node = $5 + [16/2^4] = 6$; child node = u_6; line graph = $u_5 – u_6$

$i = 7$; $n = 16$; $t = 4$

child node = $7 + [16/2^4] = 8$; child node = u_8; line graph = $u_7 – u_8$

i = 9; n = 16; t = 4

child node = 9 + [16/2^4] = 10; child node = u_{10}; line graph = $u_9 – u_{10}$

i = 11; n = 16; t = 4

child node = 11 + [16/2^4] = 12; child node = u_{12}; line graph = $u_{11} – u_{12}$

i = 13; n = 16; t = 4

child node = 13 + [16/2^4] = 14; child node = u_{14}; line graph = $u_{13} – u_{14}$

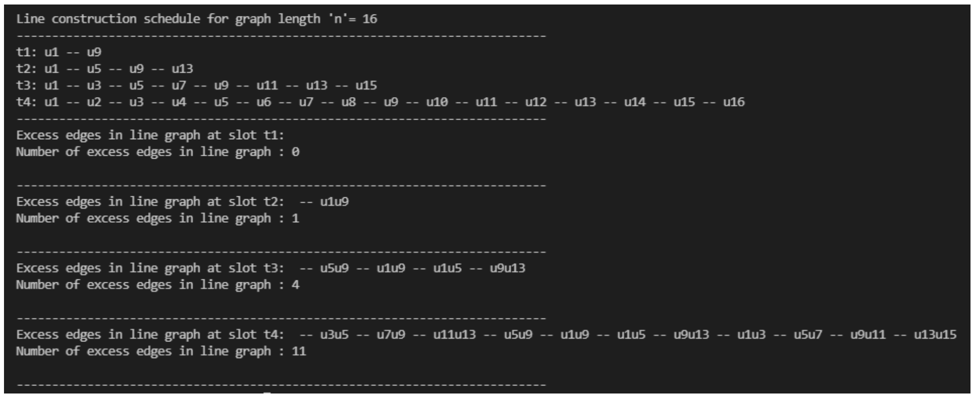
i = 15; n = 16; t = 4

child node = 15 + [16/2^4] = 16; child node = u_{16}; line graph = $u_{15} – u_{16}$

Excess edges at t4: u_1u_5, u_9u_{13}, u_5u_9, u_1u_9, u_1u_3, u_3u_5, u_5u_7, u_7u_9, u_9u_{11}, $u_{11}u_{13}$, $u_{13}u_{15}$

Line graph at slot t4: $u_1 – u_2 – u_3 – u_4 – u_5 – u_6 – u_7 – u_8 – u_9 – u_{10} – u_{11} – u_{12} – u_{13} – u_{14} – u_{15} – u_{16}$

Line Graph output for size ‘n = 16’:



```
Line construction schedule for graph length 'n'= 16
------------------------------------------------------------------------
t1: u1 -- u9
t2: u1 -- u5 -- u9 -- u13
t3: u1 -- u3 -- u5 -- u7 -- u9 -- u11 -- u13 -- u15
t4: u1 -- u2 -- u3 -- u4 -- u5 -- u6 -- u7 -- u8 -- u9 -- u10 -- u11 -- u12 -- u13 -- u14 -- u15 -- u16
------------------------------------------------------------------------
Excess edges in line graph at slot t1:
Number of excess edges in line graph : 0

------------------------------------------------------------------------
Excess edges in line graph at slot t2:  -- u1u9
Number of excess edges in line graph : 1

------------------------------------------------------------------------
Excess edges in line graph at slot t3:  -- u5u9 -- u1u9 -- u1u5 -- u9u13
Number of excess edges in line graph : 4

------------------------------------------------------------------------
Excess edges in line graph at slot t4:  -- u3u5 -- u7u9 -- u11u13 -- u5u9 -- u1u9 -- u1u5 -- u9u13 -- u1u3 -- u5u7 -- u9u11 -- u13u15
Number of excess edges in line graph : 11

------------------------------------------------------------------------
```

Figure 5.3: Line graph construction for graph size ‘n = 16’

5.3.2 Star Graph Testing

The construction schedule of the star graph of size ‘n =8’ is computed in [log 8] base two which is in three slots by the design of the star construction algorithm.

- **Manual testing and verification**

Graph initially consists of a centre node ‘u_1’.

For slot ‘t1’:

Parent node ‘u_1’ generates a child node ‘u2’

Star graph at ‘t1’: $u_1 \rightarrow u_2$

Leaf nodes after ‘t1’: u_2

For slot ‘t2’:

Parent nodes: u_1, u_2

Edges activated: $u_1 \rightarrow u_3$; $u_2 \rightarrow u_4$; Edges activated by leaf nodes are connected with the centre of the star i.e ‘u_1’.

Star graph at ‘t2’: $u_1 \rightarrow u_3$; $u_1 \rightarrow u_4$;

New nodes at ‘t2’: u_3, u_4

Edges activated by the leaf nodes are considered as the excess edges.

Excess edges at ‘t2’: $u_2 \rightarrow u_4$

Leaf nodes after ‘t2’: u_2, u_3, u_4

For slot ‘t3’:

Parent nodes: u_1, u_2, u_3, u_4

Edges activated: $u_1 \rightarrow u_5$; $u_2 \rightarrow u_6$; $u_3 \rightarrow u_7$; $u_4 \rightarrow u_8$; Edges activated by leaf nodes are connected with the centre of the star i.e ‘u_1’.

Star graph at ‘t3’: $u_1 \rightarrow u_5$; $u_1 \rightarrow u_6$; $u_1 \rightarrow u_7$; $u_1 \rightarrow u_8$;

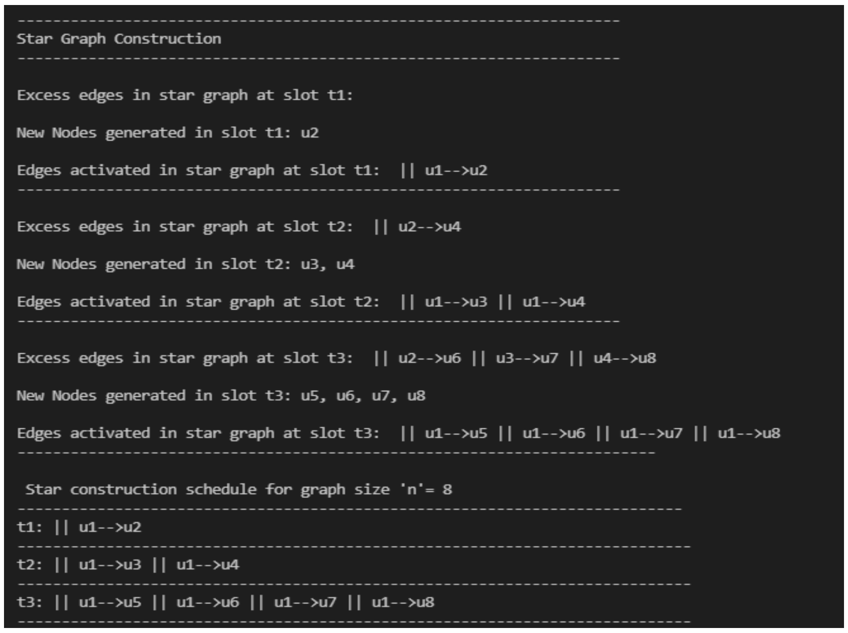
New nodes at ‘t3’: u_5, u_6, u_7, u_8

Edges activated by the leaf nodes are considered as the excess edges.

Excess edges at ‘t3’: $u_2 \rightarrow u_6$; $u_3 \rightarrow u_7$; $u_4 \rightarrow u_8$;

Leaf nodes after ‘t3’: u_2, u_3, u_4, u_5, u_6, u_7, u_8

Star graph output for size ‘n = 8’:



```
--------------------------------------------------------------------
Star Graph Construction
--------------------------------------------------------------------

Excess edges in star graph at slot t1:

New Nodes generated in slot t1: u2

Edges activated in star graph at slot t1:  || u1-->u2
--------------------------------------------------------------------

Excess edges in star graph at slot t2:  || u2-->u4

New Nodes generated in slot t2: u3, u4

Edges activated in star graph at slot t2:  || u1-->u3 || u1-->u4
--------------------------------------------------------------------

Excess edges in star graph at slot t3:  || u2-->u6 || u3-->u7 || u4-->u8

New Nodes generated in slot t3: u5, u6, u7, u8

Edges activated in star graph at slot t3:  || u1-->u5 || u1-->u6 || u1-->u7 || u1-->u8
------------------------------------------------------------------------

 Star construction schedule for graph size 'n'= 8
---------------------------------------------------------------------------
t1: || u1-->u2
----------------------------------------------------------------------------
t2: || u1-->u3 || u1-->u4
----------------------------------------------------------------------------
t3: || u1-->u5 || u1-->u6 || u1-->u7 || u1-->u8
----------------------------------------------------------------------------
```

Figure 5.4: Star graph construction for graph size ‘n = 8’

5.3.3 Line Graph Evaluation

After the successful algorithm analysis, implementation and experimentation of the line graph, I have verified the runtime of the line graph construction with the theoretical bounds of the line graph algorithm stated in the article ”The complexity of a Growing graph”.

To construct the graph of size ‘n’, the line graph model takes [log n] time complexity. Figure 5.5 shows the complexity of constructing a line graph of size ‘n’ in [log n] time slots. The complexity of excess edges in the line graph model is O[n]. Figure 5.6 shows the complexity of the excess edges that are formed during the execution of the line graph .

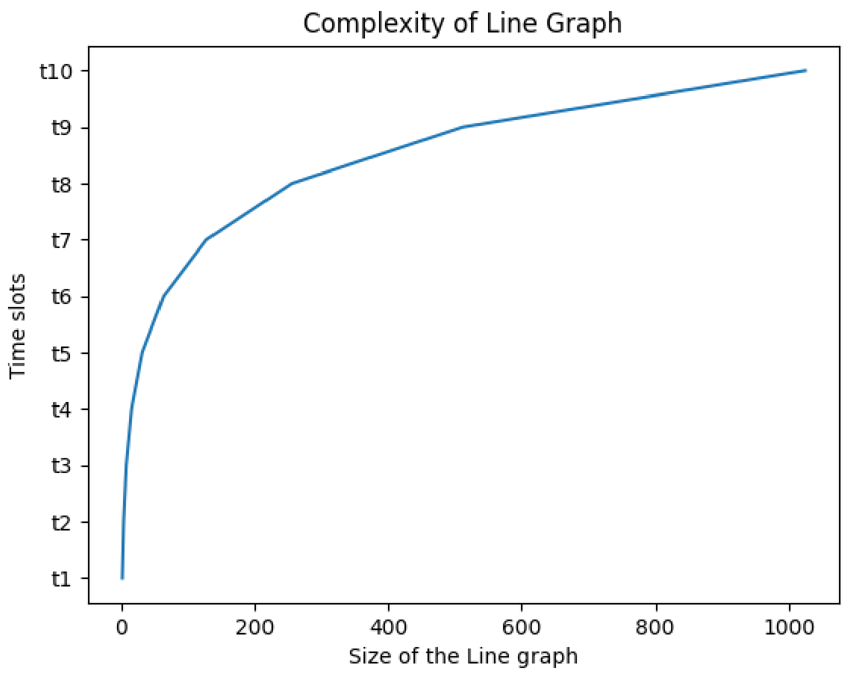


Figure 5.5: Complexity of Line graph construction for graph size ‘n = 1000’

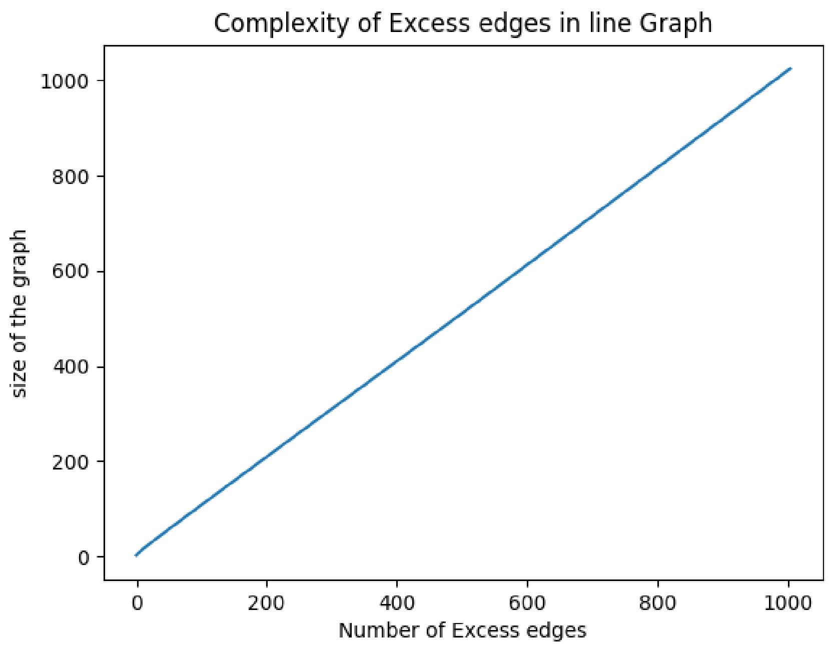


Figure 5.6: Complexity of the excess edges for graph size ‘n = 1000’

5.3.4 <u>Star Graph Evaluation</u>

After the successful algorithm analysis, implementation and experimentation of the star graph, I have verified the runtime of the star graph construction with the theoretical bounds of the star graph algorithm stated in the study ”The complexity of a Growing graph”.

To construct the graph of size ‘n’, the star graph model takes [log n] time complexity. Figure 5.7 shows the complexity of constructing a star graph of size ‘n’ in [log n] time slots. The complexity of excess edges in the star graph model is O[n]. Figure 5.8 shows the complexity of the excess edges that are formed during the execution of the star graph.

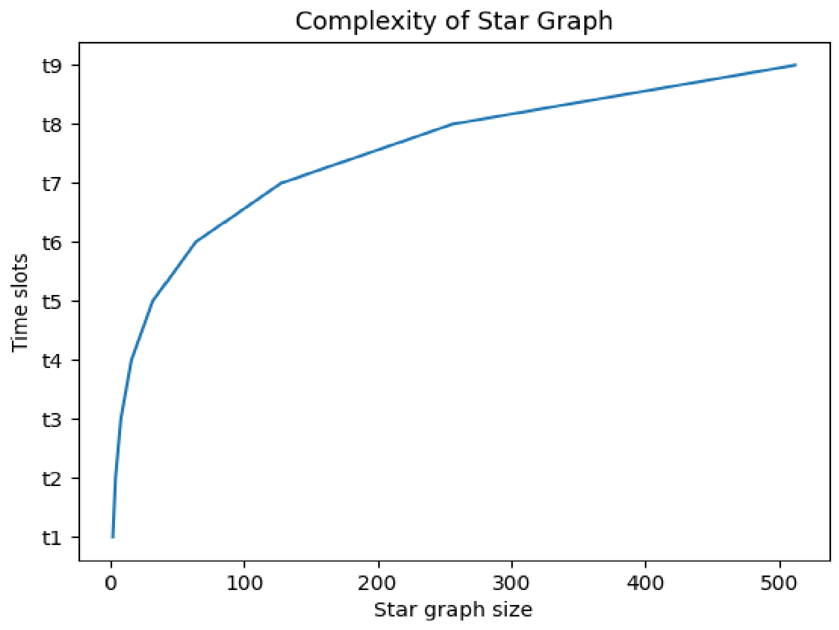


Figure 5.7: Complexity of Star graph construction for graph size ‘n = 500’

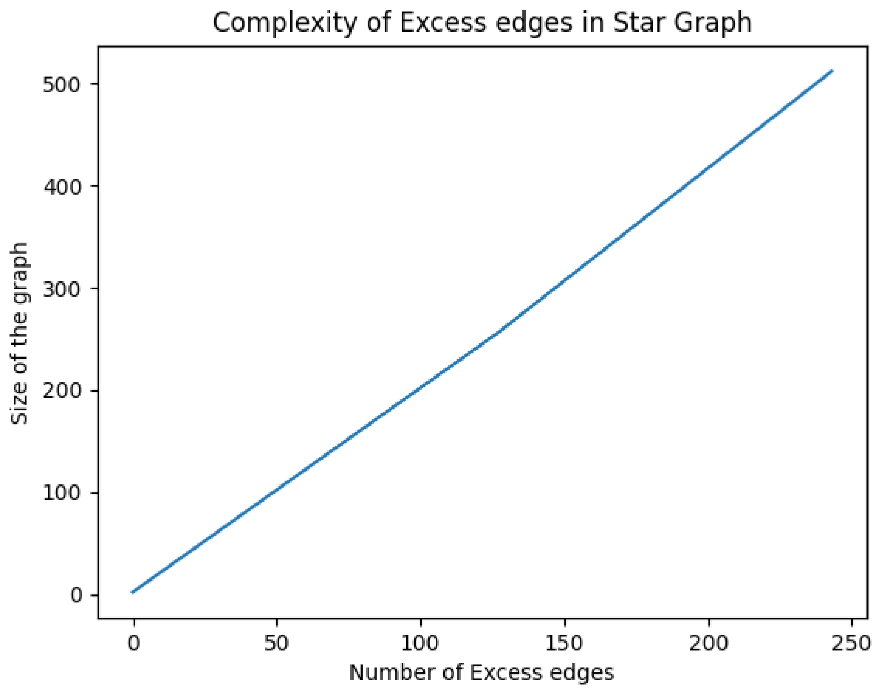


Figure 5.8: Complexity of excess edges in Star graph for graph size ‘n = 500’

Chapter 6. Evaluation

6.1 Successes of the Project

I believe that my project was successful as I have accomplished all the project goals. I have implemented, experimented, tested and verified the theoretical bounds of the line and star graph models as per the project plan.

I assessed the complexity of the dynamic graph algorithms by the test results mentioned in section 5.3.

I believe that the proper planning of the analysis of dynamic graph algorithms at the initial stages would have reduced the challenges mentioned in the section below.

6.2 Challenges of the Project

I have faced some challenges during the course of the project. There are multiple scenarios in understanding the graph models. The main challenges were as follows.

- **Construction of Line Graph:**

  At the initial stages I did not understand the process of graph construction and excess edge identification of the line graph algorithm. I have solved the above issue by manually implementing the algorithm with a smaller instance and clearly understanding the process of node generation and excess edge identification at each time slot.

- **Choosing the Data Structure:**

  At the beginning of algorithm implementation, I had observed many obstacles in storing the graph data. I needed a data structure to store the graph construction, to store the excess edges data. To handle the above challenge, I have updated the data structure from list to dictionary. So that I can use each slot as a key of the dictionary and graph data as the value.

- **Algorithm Complexity:**

  I have reviewed the concepts of runtime analysis of the algorithms to properly understand the complexity of the algorithms.

Chapter 7. Learning Points

Throughout the project, I learned several skills. In this chapter, I will detail the skills I learned to accomplish my project.

7.1 Research

I have gathered the resources that are required for my project. By analysing and evaluating the information I have improved my critical analysis. I have improved my research skills.

7.2 Problem Solving

By identifying various problems in understanding the graph models, analysing the causes for the problems while implementing the algorithms, prioritising the solutions, generating new ideas to solve the issues. I can confidently say that I have become more resilient in problem-solving.

7.3 Object-Oriented Programming using Python

The object-oriented approach of the project reduced the complexity of the implementation of dynamic graph algorithms. I implemented OOPs concepts such as polymorphism and encapsulation in this project which made me understand better the use of object-oriented programming. Overall I would say that using the Python programming language was beneficial to the project and my personal development.

7.4 Project Management Skills

- **Project Planning:**

I designed my complete project by fixing the deadlines, choosing the project methodologies and setting milestones. I implemented my project as per the plan, which helped me understand the importance of project planning at the initial phase.

- **Time Management:**

  The project had five weeks for analysing the algorithms, implementation, testing and verification in a total of fifteen weeks.

- **Decision making:**

  During the project, I had to make decisions on design changes in algorithm implementation, I allocated time for unexpected challenges and finished the project as per the plan. Prioritising the objectives and keeping on track was challenging and making the right decisions was crucial.

CHAPTER 8. PROFESSIONAL ISSUES

In this chapter, I want to explain how the four main principles of the BCS code of conduct were assessed and fulfilled throughout the project.

8.1 Making IT for everyone:

The project focused on the experimental evaluation of the dynamic algorithms that are contributed in the research article ”The Complexity of a growing graph”. This promotes its usage in future research in self-growing algorithmic processes.

8.2 Show what you know, Learn what you don’t:

I have decided to do my project by assessing the practical possibilities of completing it based on my skills. I have implemented an experimental evaluation of the dynamic algorithms on my own using the available resources by continuous learning.

I have learned several skills and technologies as detailed in Chapter 7 during the course of this project. From above, I can say that my project addresses the BCS principle - ”Show what you know, learn what you don’t”

8.3 Respect the organisation or individual you work for

I have maintained professionalism for all the meetings and interactions with everyone involved in my project. I take personal and collective responsibility for my actions while maintaining discretion and ethical standards for my project.

8.4 Keep IT real, Keep IT professional. Pass IT on

The project's work does not include any actions that disrespect the profession. To increase professional standards, I have implemented my project and tested it from the ground up.

Chapter 9. Conclusions

9.1 Personal Conclusion

This project to implement experimental evaluation of the dynamic graph models has been a successful endeavour. I have performed verification of the algorithms proposed in the study "The complexity of growing graphs". I have learned new technologies, principles and improved my problem-solving skills.

If I get an opportunity to complete a similar project in the future, I will concentrate on test automation instead of manual testing, and allocate more time for analysing the self-growing algorithmic processes.

9.2 Future plans

I plan the following tasks as some of the potential expansions for my project.

- **Tree graphs:** Implementing the tree graph algorithm and experimenting with various instances to analyse the performance of the algorithm and verify the theoretical bounds of the algorithm stated.
- **Planar graphs:** Analysing the description and proofs of the planar graph and experimenting with the planar graph model that computes a construction schedule for any given planar graph starting from a single node.
- **Zero-waste construction schedules:** Understanding the concepts of the zero waste construction schedule, graphs that can be constructed with zero excess edges.
- **Visualisations of the algorithms:** Creating a web page to implement the visualisation of the dynamic graph algorithms.

Bibliography

[1] George B. Mertzios, Othon Michail, George Skretas, Paul G. Spirakis, And Michail Theofilatos. *The Complexity of Growing a Graph*.
arXiv: 2107.14126 [cs.DS] (2021) Available at https://arxiv.org/abs/2107.14126.

[2] Mohammad Mohtashim. *Tutorials Point*, "Design and Analysis of Algorithm"[online]. Available at: https://www.tutorialspoint.com/design_and_analysis_of_algorithms/analysis_of_algorithms.htm [Accessed 13 September 2021].

[3] sam.csc.liv.ac.uk. (n.d.). *CSJS702 - E-Project System*. [online] Available at: https://sam.csc.liv.ac.uk/CSJS702/2021/project/1498 [Accessed 12 Dec. 2021].

[4] Intersog: how to calculate the complexity of an algorithm -Intersog [online] Available at:
https://intersog.com/blog/algorithm-complexity-estimation-a-bit-of-theory-and-why-it-is-necessary-to-know/ [Accessed 17 September 2021].

[5] "math — Mathematical functions — Python 3.10.0 documentation." *Python Docs*, Available at https://docs.python.org/3/library/math.html[Accessed 16 November 2021]

[6] Matplotlib: pyplot tutorial - *matplotlib 3.4.3 documentation* [online] Available at
https://matplotlib.org/stable/tutorials/introductory/pyplot.html[Accessed 17 September 2021].

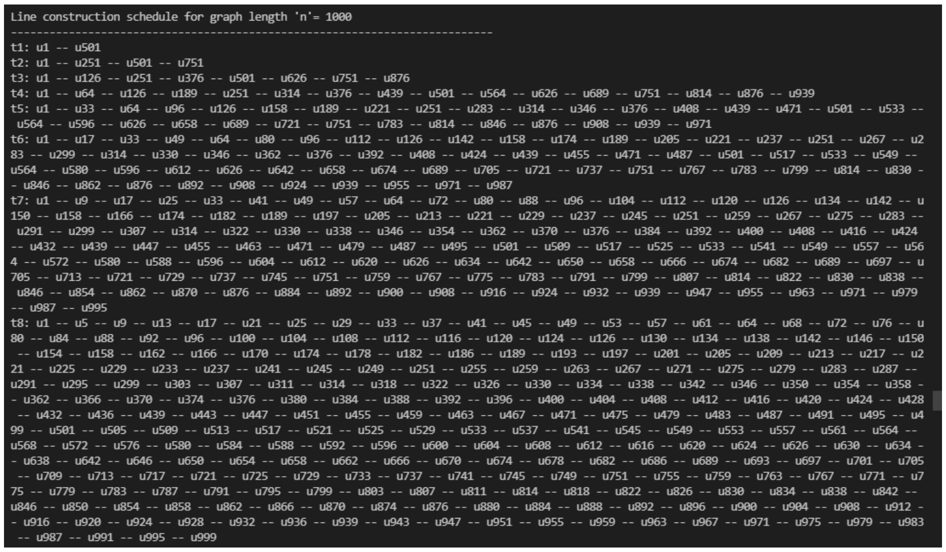
[7] ACM. Association for Computing Machinery: Aaron Scott Pope, Daniel R. Tauritz, Chris Rawlings. *Automated design of random dynamic graph models*. (retrieved from https://dl.acm.org/doi/10.1145/3319619.3326859)

Appendix A

Experimentation Results

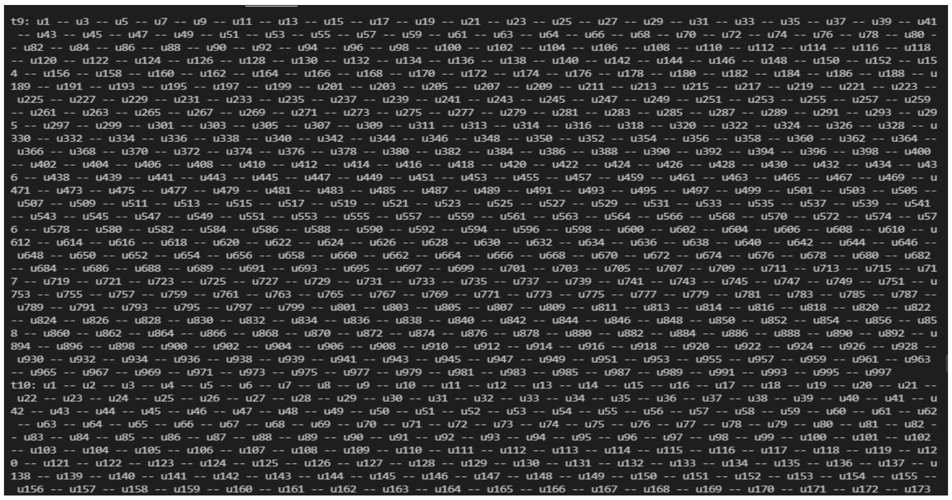
A.1 Line Graph

Line graph experimentation with size of the graph 'n = 1000'



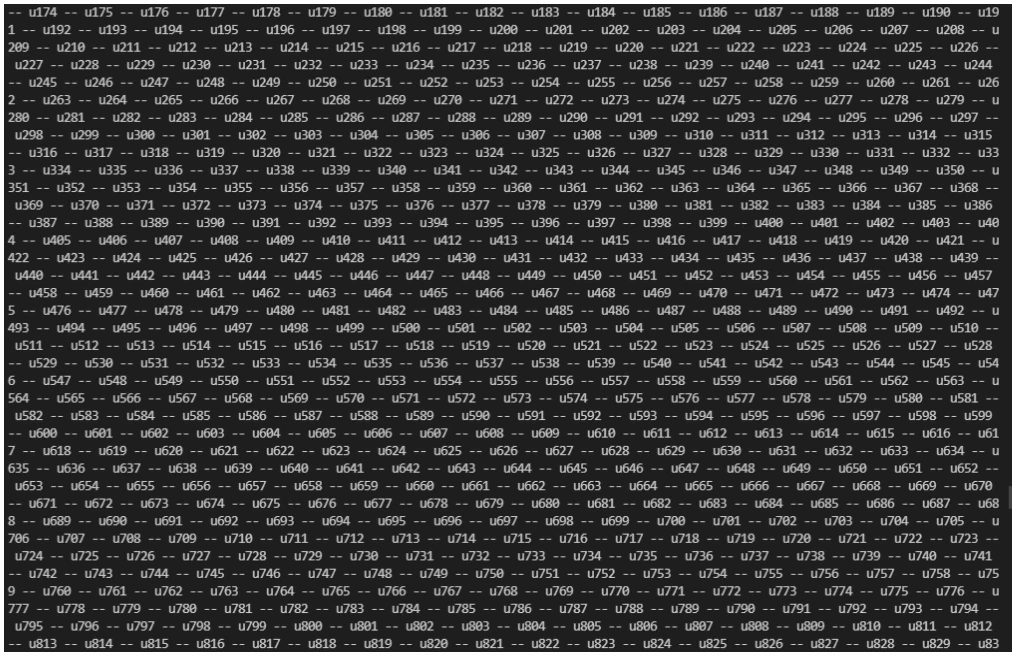
```
Line construction schedule for graph length 'n'= 1000
---------------------------------------------------------------------------
t1: u1 -- u501
t2: u1 -- u251 -- u501 -- u751
t3: u1 -- u126 -- u251 -- u376 -- u501 -- u626 -- u751 -- u876
t4: u1 -- u64 -- u126 -- u189 -- u251 -- u314 -- u376 -- u439 -- u501 -- u564 -- u626 -- u689 -- u751 -- u814 -- u876 -- u939
t5: u1 -- u33 -- u64 -- u96 -- u126 -- u158 -- u189 -- u221 -- u251 -- u283 -- u314 -- u346 -- u376 -- u408 -- u439 -- u471 -- u501 -- u533 --
 u564 -- u596 -- u626 -- u658 -- u689 -- u721 -- u751 -- u783 -- u814 -- u846 -- u876 -- u908 -- u939 -- u971
t6: u1 -- u17 -- u33 -- u49 -- u64 -- u80 -- u96 -- u112 -- u126 -- u142 -- u158 -- u174 -- u189 -- u205 -- u221 -- u237 -- u251 -- u267 -- u2
83 -- u299 -- u314 -- u330 -- u346 -- u362 -- u376 -- u392 -- u408 -- u424 -- u439 -- u455 -- u471 -- u487 -- u501 -- u517 -- u533 -- u549 --
u564 -- u580 -- u596 -- u612 -- u626 -- u642 -- u658 -- u674 -- u689 -- u705 -- u721 -- u737 -- u751 -- u767 -- u783 -- u799 -- u814 -- u830 -
- u846 -- u862 -- u876 -- u892 -- u908 -- u924 -- u939 -- u955 -- u971 -- u987
t7: u1 -- u9 -- u17 -- u25 -- u33 -- u41 -- u49 -- u57 -- u64 -- u72 -- u80 -- u88 -- u96 -- u104 -- u112 -- u120 -- u126 -- u134 -- u142 -- u
150 -- u158 -- u166 -- u174 -- u182 -- u189 -- u197 -- u205 -- u213 -- u221 -- u229 -- u237 -- u245 -- u251 -- u259 -- u267 -- u275 -- u283 --
 u291 -- u299 -- u307 -- u314 -- u322 -- u330 -- u338 -- u346 -- u354 -- u362 -- u370 -- u376 -- u384 -- u392 -- u400 -- u408 -- u416 -- u424
-- u432 -- u439 -- u447 -- u455 -- u463 -- u471 -- u479 -- u487 -- u495 -- u501 -- u509 -- u517 -- u525 -- u533 -- u541 -- u549 -- u557 -- u56
4 -- u572 -- u580 -- u588 -- u596 -- u604 -- u612 -- u620 -- u626 -- u634 -- u642 -- u650 -- u658 -- u666 -- u674 -- u682 -- u689 -- u697 -- u
705 -- u713 -- u721 -- u729 -- u737 -- u745 -- u751 -- u759 -- u767 -- u775 -- u783 -- u791 -- u799 -- u807 -- u814 -- u822 -- u830 -- u838 --
 u846 -- u854 -- u862 -- u870 -- u876 -- u884 -- u892 -- u900 -- u908 -- u916 -- u924 -- u932 -- u939 -- u947 -- u955 -- u963 -- u971 -- u979
-- u987 -- u995
t8: u1 -- u5 -- u9 -- u13 -- u17 -- u21 -- u25 -- u29 -- u33 -- u37 -- u41 -- u45 -- u49 -- u53 -- u57 -- u61 -- u64 -- u68 -- u72 -- u76 -- u
80 -- u84 -- u88 -- u92 -- u96 -- u100 -- u104 -- u108 -- u112 -- u116 -- u120 -- u124 -- u126 -- u130 -- u134 -- u138 -- u142 -- u146 -- u150
 -- u154 -- u158 -- u162 -- u166 -- u170 -- u174 -- u178 -- u182 -- u186 -- u189 -- u193 -- u197 -- u201 -- u205 -- u209 -- u213 -- u217 -- u2
21 -- u225 -- u229 -- u233 -- u237 -- u241 -- u245 -- u249 -- u251 -- u255 -- u259 -- u263 -- u267 -- u271 -- u275 -- u279 -- u283 -- u287 --
u291 -- u295 -- u299 -- u303 -- u307 -- u311 -- u314 -- u318 -- u322 -- u326 -- u330 -- u334 -- u338 -- u342 -- u346 -- u350 -- u354 -- u358 -
- u362 -- u366 -- u370 -- u374 -- u376 -- u380 -- u384 -- u388 -- u392 -- u396 -- u400 -- u404 -- u408 -- u412 -- u416 -- u420 -- u424 -- u428
 -- u432 -- u436 -- u439 -- u443 -- u447 -- u451 -- u455 -- u459 -- u463 -- u467 -- u471 -- u475 -- u479 -- u483 -- u487 -- u491 -- u495 -- u4
99 -- u501 -- u505 -- u509 -- u513 -- u517 -- u521 -- u525 -- u529 -- u533 -- u537 -- u541 -- u545 -- u549 -- u553 -- u557 -- u561 -- u564 --
u568 -- u572 -- u576 -- u580 -- u584 -- u588 -- u592 -- u596 -- u600 -- u604 -- u608 -- u612 -- u616 -- u620 -- u624 -- u626 -- u630 -- u634 -
- u638 -- u642 -- u646 -- u650 -- u654 -- u658 -- u662 -- u666 -- u670 -- u674 -- u678 -- u682 -- u686 -- u689 -- u693 -- u697 -- u701 -- u705
 -- u709 -- u713 -- u717 -- u721 -- u725 -- u729 -- u733 -- u737 -- u741 -- u745 -- u749 -- u751 -- u755 -- u759 -- u763 -- u767 -- u771 -- u7
75 -- u779 -- u783 -- u787 -- u791 -- u795 -- u799 -- u803 -- u807 -- u811 -- u814 -- u818 -- u822 -- u826 -- u830 -- u834 -- u838 -- u842 --
u846 -- u850 -- u854 -- u858 -- u862 -- u866 -- u870 -- u874 -- u876 -- u880 -- u884 -- u888 -- u892 -- u896 -- u900 -- u904 -- u908 -- u912 -
- u916 -- u920 -- u924 -- u928 -- u932 -- u936 -- u939 -- u943 -- u947 -- u951 -- u955 -- u959 -- u963 -- u967 -- u971 -- u975 -- u979 -- u983
 -- u987 -- u991 -- u995 -- u999
```

Figure A.1 : Line graph construction from 't1' to 't8' slots



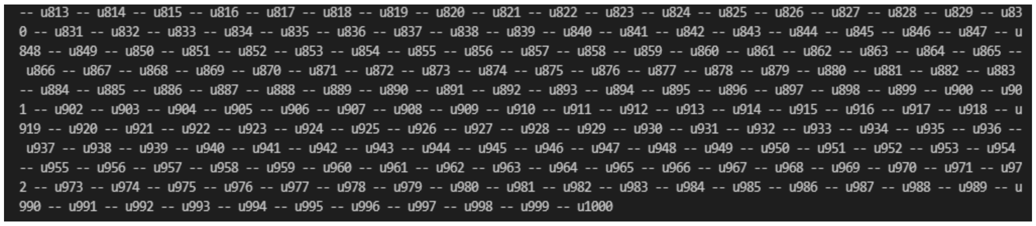
```
t9: u1 -- u3 -- u5 -- u7 -- u9 -- u11 -- u13 -- u15 -- u17 -- u19 -- u21 -- u23 -- u25 -- u27 -- u29 -- u31 -- u33 -- u35 -- u37 -- u39 -- u41
 -- u43 -- u45 -- u47 -- u49 -- u51 -- u53 -- u55 -- u57 -- u59 -- u61 -- u63 -- u64 -- u66 -- u68 -- u70 -- u72 -- u74 -- u76 -- u78 -- u80 -
- u82 -- u84 -- u86 -- u88 -- u90 -- u92 -- u94 -- u96 -- u98 -- u100 -- u102 -- u104 -- u106 -- u108 -- u110 -- u112 -- u114 -- u116 -- u118
-- u120 -- u122 -- u124 -- u126 -- u128 -- u130 -- u132 -- u134 -- u136 -- u138 -- u140 -- u142 -- u144 -- u146 -- u148 -- u150 -- u152 -- u15
4 -- u156 -- u158 -- u160 -- u162 -- u164 -- u166 -- u168 -- u170 -- u172 -- u174 -- u176 -- u178 -- u180 -- u182 -- u184 -- u186 -- u188 -- u
189 -- u191 -- u193 -- u195 -- u197 -- u199 -- u201 -- u203 -- u205 -- u207 -- u209 -- u211 -- u213 -- u215 -- u217 -- u219 -- u221 -- u223 --
 u225 -- u227 -- u229 -- u231 -- u233 -- u235 -- u237 -- u239 -- u241 -- u243 -- u245 -- u247 -- u249 -- u251 -- u253 -- u255 -- u257 -- u259
-- u261 -- u263 -- u265 -- u267 -- u269 -- u271 -- u273 -- u275 -- u277 -- u279 -- u281 -- u283 -- u285 -- u287 -- u289 -- u291 -- u293 -- u29
5 -- u297 -- u299 -- u301 -- u303 -- u305 -- u307 -- u309 -- u311 -- u313 -- u314 -- u316 -- u318 -- u320 -- u322 -- u324 -- u326 -- u328 -- u
330 -- u332 -- u334 -- u336 -- u338 -- u340 -- u342 -- u344 -- u346 -- u348 -- u350 -- u352 -- u354 -- u356 -- u358 -- u360 -- u362 -- u364 --
 u366 -- u368 -- u370 -- u372 -- u374 -- u376 -- u378 -- u380 -- u382 -- u384 -- u386 -- u388 -- u390 -- u392 -- u394 -- u396 -- u398 -- u400
-- u402 -- u404 -- u406 -- u408 -- u410 -- u412 -- u414 -- u416 -- u418 -- u420 -- u422 -- u424 -- u426 -- u428 -- u430 -- u432 -- u434 -- u43
6 -- u438 -- u439 -- u441 -- u443 -- u445 -- u447 -- u449 -- u451 -- u453 -- u455 -- u457 -- u459 -- u461 -- u463 -- u465 -- u467 -- u469 -- u
471 -- u473 -- u475 -- u477 -- u479 -- u481 -- u483 -- u485 -- u487 -- u489 -- u491 -- u493 -- u495 -- u497 -- u499 -- u501 -- u503 -- u505 --
 u507 -- u509 -- u511 -- u513 -- u515 -- u517 -- u519 -- u521 -- u523 -- u525 -- u527 -- u529 -- u531 -- u533 -- u535 -- u537 -- u539 -- u541
-- u543 -- u545 -- u547 -- u549 -- u551 -- u553 -- u555 -- u557 -- u559 -- u561 -- u563 -- u564 -- u566 -- u568 -- u570 -- u572 -- u574 -- u57
6 -- u578 -- u580 -- u582 -- u584 -- u586 -- u588 -- u590 -- u592 -- u594 -- u596 -- u598 -- u600 -- u602 -- u604 -- u606 -- u608 -- u610 -- u
612 -- u614 -- u616 -- u618 -- u620 -- u622 -- u624 -- u626 -- u628 -- u630 -- u632 -- u634 -- u636 -- u638 -- u640 -- u642 -- u644 -- u646 --
 u648 -- u650 -- u652 -- u654 -- u656 -- u658 -- u660 -- u662 -- u664 -- u666 -- u668 -- u670 -- u672 -- u674 -- u676 -- u678 -- u680 -- u682
-- u684 -- u686 -- u688 -- u689 -- u691 -- u693 -- u695 -- u697 -- u699 -- u701 -- u703 -- u705 -- u707 -- u709 -- u711 -- u713 -- u715 -- u71
7 -- u719 -- u721 -- u723 -- u725 -- u727 -- u729 -- u731 -- u733 -- u735 -- u737 -- u739 -- u741 -- u743 -- u745 -- u747 -- u749 -- u751 -- u
753 -- u755 -- u757 -- u759 -- u761 -- u763 -- u765 -- u767 -- u769 -- u771 -- u773 -- u775 -- u777 -- u779 -- u781 -- u783 -- u785 -- u787 --
 u789 -- u791 -- u793 -- u795 -- u797 -- u799 -- u801 -- u803 -- u805 -- u807 -- u809 -- u811 -- u813 -- u814 -- u816 -- u818 -- u820 -- u822
-- u824 -- u826 -- u828 -- u830 -- u832 -- u834 -- u836 -- u838 -- u840 -- u842 -- u844 -- u846 -- u848 -- u850 -- u852 -- u854 -- u856 -- u85
8 -- u860 -- u862 -- u864 -- u866 -- u868 -- u870 -- u872 -- u874 -- u876 -- u878 -- u880 -- u882 -- u884 -- u886 -- u888 -- u890 -- u892 -- u
894 -- u896 -- u898 -- u900 -- u902 -- u904 -- u906 -- u908 -- u910 -- u912 -- u914 -- u916 -- u918 -- u920 -- u922 -- u924 -- u926 -- u928 --
 u930 -- u932 -- u934 -- u936 -- u938 -- u939 -- u941 -- u943 -- u945 -- u947 -- u949 -- u951 -- u953 -- u955 -- u957 -- u959 -- u961 -- u963
-- u965 -- u967 -- u969 -- u971 -- u973 -- u975 -- u977 -- u979 -- u981 -- u983 -- u985 -- u987 -- u989 -- u991 -- u993 -- u995 -- u997
t10: u1 -- u2 -- u3 -- u4 -- u5 -- u6 -- u7 -- u8 -- u9 -- u10 -- u11 -- u12 -- u13 -- u14 -- u15 -- u16 -- u17 -- u18 -- u19 -- u20 -- u21 --
 u22 -- u23 -- u24 -- u25 -- u26 -- u27 -- u28 -- u29 -- u30 -- u31 -- u32 -- u33 -- u34 -- u35 -- u36 -- u37 -- u38 -- u39 -- u40 -- u41 -- u
42 -- u43 -- u44 -- u45 -- u46 -- u47 -- u48 -- u49 -- u50 -- u51 -- u52 -- u53 -- u54 -- u55 -- u56 -- u57 -- u58 -- u59 -- u60 -- u61 -- u62
 -- u63 -- u64 -- u65 -- u66 -- u67 -- u68 -- u69 -- u70 -- u71 -- u72 -- u73 -- u74 -- u75 -- u76 -- u77 -- u78 -- u79 -- u80 -- u81 -- u82 -
- u83 -- u84 -- u85 -- u86 -- u87 -- u88 -- u89 -- u90 -- u91 -- u92 -- u93 -- u94 -- u95 -- u96 -- u97 -- u98 -- u99 -- u100 -- u101 -- u102
-- u103 -- u104 -- u105 -- u106 -- u107 -- u108 -- u109 -- u110 -- u111 -- u112 -- u113 -- u114 -- u115 -- u116 -- u117 -- u118 -- u119 -- u12
0 -- u121 -- u122 -- u123 -- u124 -- u125 -- u126 -- u127 -- u128 -- u129 -- u130 -- u131 -- u132 -- u133 -- u134 -- u135 -- u136 -- u137 -- u
138 -- u139 -- u140 -- u141 -- u142 -- u143 -- u144 -- u145 -- u146 -- u147 -- u148 -- u149 -- u150 -- u151 -- u152 -- u153 -- u154 -- u155 --
 u156 -- u157 -- u158 -- u159 -- u160 -- u161 -- u162 -- u163 -- u164 -- u165 -- u166 -- u167 -- u168 -- u169 -- u170 -- u171 -- u172 -- u173
```

Figure A.2 : Line graph construction at slot 't9' and 't10'



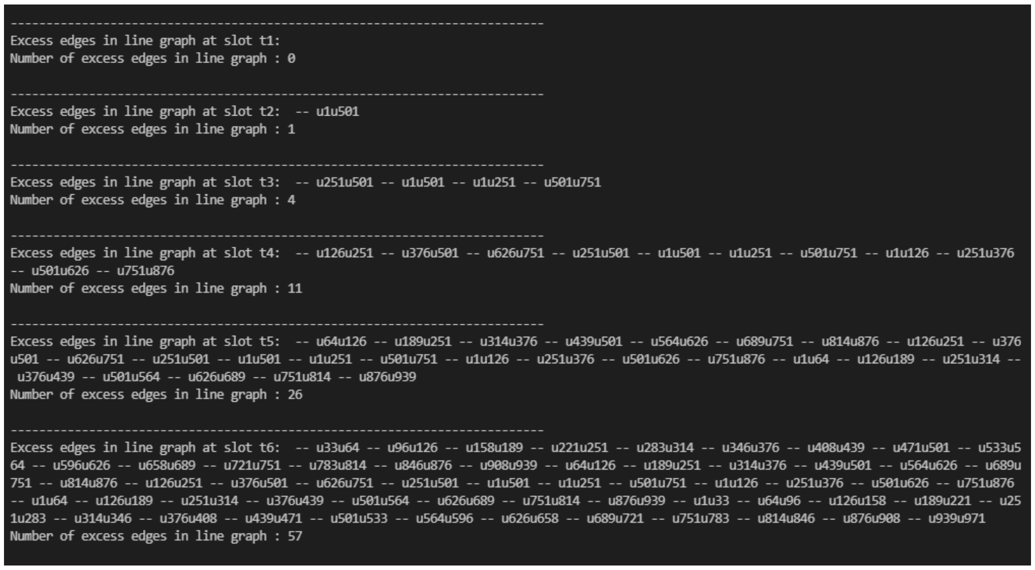
```
-- u174 -- u175 -- u176 -- u177 -- u178 -- u179 -- u180 -- u181 -- u182 -- u183 -- u184 -- u185 -- u186 -- u187 -- u188 -- u189 -- u190 -- u19
1 -- u192 -- u193 -- u194 -- u195 -- u196 -- u197 -- u198 -- u199 -- u200 -- u201 -- u202 -- u203 -- u204 -- u205 -- u206 -- u207 -- u208 -- u
209 -- u210 -- u211 -- u212 -- u213 -- u214 -- u215 -- u216 -- u217 -- u218 -- u219 -- u220 -- u221 -- u222 -- u223 -- u224 -- u225 -- u226 --
 u227 -- u228 -- u229 -- u230 -- u231 -- u232 -- u233 -- u234 -- u235 -- u236 -- u237 -- u238 -- u239 -- u240 -- u241 -- u242 -- u243 -- u244
-- u245 -- u246 -- u247 -- u248 -- u249 -- u250 -- u251 -- u252 -- u253 -- u254 -- u255 -- u256 -- u257 -- u258 -- u259 -- u260 -- u261 -- u26
2 -- u263 -- u264 -- u265 -- u266 -- u267 -- u268 -- u269 -- u270 -- u271 -- u272 -- u273 -- u274 -- u275 -- u276 -- u277 -- u278 -- u279 -- u
280 -- u281 -- u282 -- u283 -- u284 -- u285 -- u286 -- u287 -- u288 -- u289 -- u290 -- u291 -- u292 -- u293 -- u294 -- u295 -- u296 -- u297 --
 u298 -- u299 -- u300 -- u301 -- u302 -- u303 -- u304 -- u305 -- u306 -- u307 -- u308 -- u309 -- u310 -- u311 -- u312 -- u313 -- u314 -- u315
-- u316 -- u317 -- u318 -- u319 -- u320 -- u321 -- u322 -- u323 -- u324 -- u325 -- u326 -- u327 -- u328 -- u329 -- u330 -- u331 -- u332 -- u33
3 -- u334 -- u335 -- u336 -- u337 -- u338 -- u339 -- u340 -- u341 -- u342 -- u343 -- u344 -- u345 -- u346 -- u347 -- u348 -- u349 -- u350 -- u
351 -- u352 -- u353 -- u354 -- u355 -- u356 -- u357 -- u358 -- u359 -- u360 -- u361 -- u362 -- u363 -- u364 -- u365 -- u366 -- u367 -- u368 --
 u369 -- u370 -- u371 -- u372 -- u373 -- u374 -- u375 -- u376 -- u377 -- u378 -- u379 -- u380 -- u381 -- u382 -- u383 -- u384 -- u385 -- u386
-- u387 -- u388 -- u389 -- u390 -- u391 -- u392 -- u393 -- u394 -- u395 -- u396 -- u397 -- u398 -- u399 -- u400 -- u401 -- u402 -- u403 -- u40
4 -- u405 -- u406 -- u407 -- u408 -- u409 -- u410 -- u411 -- u412 -- u413 -- u414 -- u415 -- u416 -- u417 -- u418 -- u419 -- u420 -- u421 -- u
422 -- u423 -- u424 -- u425 -- u426 -- u427 -- u428 -- u429 -- u430 -- u431 -- u432 -- u433 -- u434 -- u435 -- u436 -- u437 -- u438 -- u439 --
 u440 -- u441 -- u442 -- u443 -- u444 -- u445 -- u446 -- u447 -- u448 -- u449 -- u450 -- u451 -- u452 -- u453 -- u454 -- u455 -- u456 -- u457
-- u458 -- u459 -- u460 -- u461 -- u462 -- u463 -- u464 -- u465 -- u466 -- u467 -- u468 -- u469 -- u470 -- u471 -- u472 -- u473 -- u474 -- u47
5 -- u476 -- u477 -- u478 -- u479 -- u480 -- u481 -- u482 -- u483 -- u484 -- u485 -- u486 -- u487 -- u488 -- u489 -- u490 -- u491 -- u492 -- u
493 -- u494 -- u495 -- u496 -- u497 -- u498 -- u499 -- u500 -- u501 -- u502 -- u503 -- u504 -- u505 -- u506 -- u507 -- u508 -- u509 -- u510 --
 u511 -- u512 -- u513 -- u514 -- u515 -- u516 -- u517 -- u518 -- u519 -- u520 -- u521 -- u522 -- u523 -- u524 -- u525 -- u526 -- u527 -- u528
-- u529 -- u530 -- u531 -- u532 -- u533 -- u534 -- u535 -- u536 -- u537 -- u538 -- u539 -- u540 -- u541 -- u542 -- u543 -- u544 -- u545 -- u54
6 -- u547 -- u548 -- u549 -- u550 -- u551 -- u552 -- u553 -- u554 -- u555 -- u556 -- u557 -- u558 -- u559 -- u560 -- u561 -- u562 -- u563 -- u
564 -- u565 -- u566 -- u567 -- u568 -- u569 -- u570 -- u571 -- u572 -- u573 -- u574 -- u575 -- u576 -- u577 -- u578 -- u579 -- u580 -- u581 --
 u582 -- u583 -- u584 -- u585 -- u586 -- u587 -- u588 -- u589 -- u590 -- u591 -- u592 -- u593 -- u594 -- u595 -- u596 -- u597 -- u598 -- u599
-- u600 -- u601 -- u602 -- u603 -- u604 -- u605 -- u606 -- u607 -- u608 -- u609 -- u610 -- u611 -- u612 -- u613 -- u614 -- u615 -- u616 -- u61
7 -- u618 -- u619 -- u620 -- u621 -- u622 -- u623 -- u624 -- u625 -- u626 -- u627 -- u628 -- u629 -- u630 -- u631 -- u632 -- u633 -- u634 -- u
635 -- u636 -- u637 -- u638 -- u639 -- u640 -- u641 -- u642 -- u643 -- u644 -- u645 -- u646 -- u647 -- u648 -- u649 -- u650 -- u651 -- u652 --
 u653 -- u654 -- u655 -- u656 -- u657 -- u658 -- u659 -- u660 -- u661 -- u662 -- u663 -- u664 -- u665 -- u666 -- u667 -- u668 -- u669 -- u670
-- u671 -- u672 -- u673 -- u674 -- u675 -- u676 -- u677 -- u678 -- u679 -- u680 -- u681 -- u682 -- u683 -- u684 -- u685 -- u686 -- u687 -- u68
8 -- u689 -- u690 -- u691 -- u692 -- u693 -- u694 -- u695 -- u696 -- u697 -- u698 -- u699 -- u700 -- u701 -- u702 -- u703 -- u704 -- u705 -- u
706 -- u707 -- u708 -- u709 -- u710 -- u711 -- u712 -- u713 -- u714 -- u715 -- u716 -- u717 -- u718 -- u719 -- u720 -- u721 -- u722 -- u723 --
 u724 -- u725 -- u726 -- u727 -- u728 -- u729 -- u730 -- u731 -- u732 -- u733 -- u734 -- u735 -- u736 -- u737 -- u738 -- u739 -- u740 -- u741
-- u742 -- u743 -- u744 -- u745 -- u746 -- u747 -- u748 -- u749 -- u750 -- u751 -- u752 -- u753 -- u754 -- u755 -- u756 -- u757 -- u758 -- u75
9 -- u760 -- u761 -- u762 -- u763 -- u764 -- u765 -- u766 -- u767 -- u768 -- u769 -- u770 -- u771 -- u772 -- u773 -- u774 -- u775 -- u776 -- u
777 -- u778 -- u779 -- u780 -- u781 -- u782 -- u783 -- u784 -- u785 -- u786 -- u787 -- u788 -- u789 -- u790 -- u791 -- u792 -- u793 -- u794 --
 u795 -- u796 -- u797 -- u798 -- u799 -- u800 -- u801 -- u802 -- u803 -- u804 -- u805 -- u806 -- u807 -- u808 -- u809 -- u810 -- u811 -- u812
-- u813 -- u814 -- u815 -- u816 -- u817 -- u818 -- u819 -- u820 -- u821 -- u822 -- u823 -- u824 -- u825 -- u826 -- u827 -- u828 -- u829 -- u83
```

Figure A.3: Line graph construction at slot ‘t10’



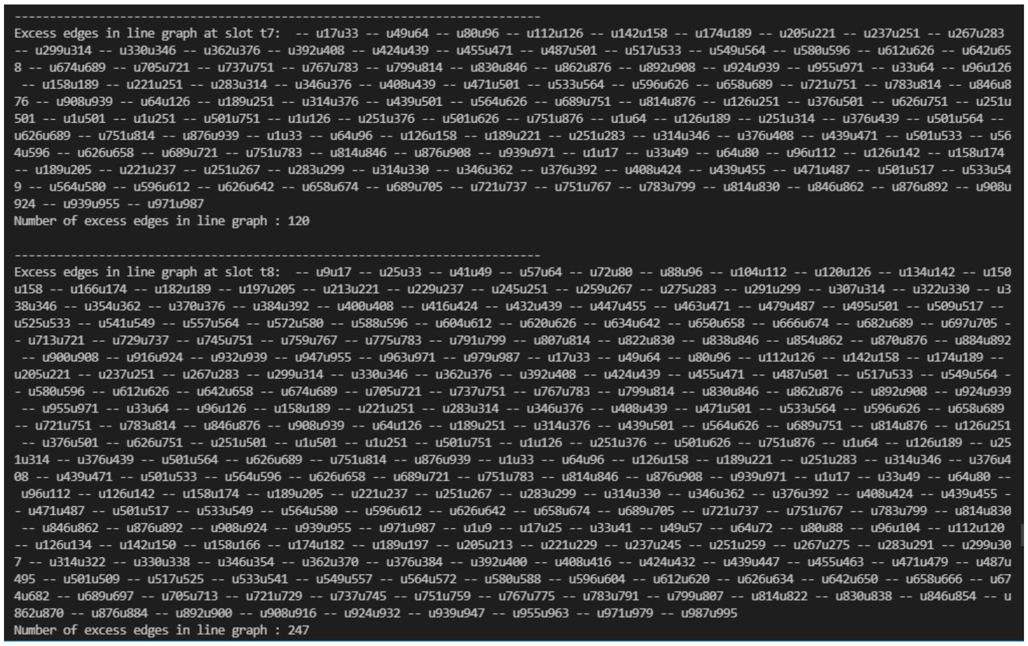
```
-- u813 -- u814 -- u815 -- u816 -- u817 -- u818 -- u819 -- u820 -- u821 -- u822 -- u823 -- u824 -- u825 -- u826 -- u827 -- u828 -- u829 -- u83
0 -- u831 -- u832 -- u833 -- u834 -- u835 -- u836 -- u837 -- u838 -- u839 -- u840 -- u841 -- u842 -- u843 -- u844 -- u845 -- u846 -- u847 -- u
848 -- u849 -- u850 -- u851 -- u852 -- u853 -- u854 -- u855 -- u856 -- u857 -- u858 -- u859 -- u860 -- u861 -- u862 -- u863 -- u864 -- u865 --
 u866 -- u867 -- u868 -- u869 -- u870 -- u871 -- u872 -- u873 -- u874 -- u875 -- u876 -- u877 -- u878 -- u879 -- u880 -- u881 -- u882 -- u883
-- u884 -- u885 -- u886 -- u887 -- u888 -- u889 -- u890 -- u891 -- u892 -- u893 -- u894 -- u895 -- u896 -- u897 -- u898 -- u899 -- u900 -- u90
1 -- u902 -- u903 -- u904 -- u905 -- u906 -- u907 -- u908 -- u909 -- u910 -- u911 -- u912 -- u913 -- u914 -- u915 -- u916 -- u917 -- u918 -- u
919 -- u920 -- u921 -- u922 -- u923 -- u924 -- u925 -- u926 -- u927 -- u928 -- u929 -- u930 -- u931 -- u932 -- u933 -- u934 -- u935 -- u936 --
 u937 -- u938 -- u939 -- u940 -- u941 -- u942 -- u943 -- u944 -- u945 -- u946 -- u947 -- u948 -- u949 -- u950 -- u951 -- u952 -- u953 -- u954
-- u955 -- u956 -- u957 -- u958 -- u959 -- u960 -- u961 -- u962 -- u963 -- u964 -- u965 -- u966 -- u967 -- u968 -- u969 -- u970 -- u971 -- u97
2 -- u973 -- u974 -- u975 -- u976 -- u977 -- u978 -- u979 -- u980 -- u981 -- u982 -- u983 -- u984 -- u985 -- u986 -- u987 -- u988 -- u989 -- u
990 -- u991 -- u992 -- u993 -- u994 -- u995 -- u996 -- u997 -- u998 -- u999 -- u1000
```

Figure A.4: Line graph construction at slot ‘t10’



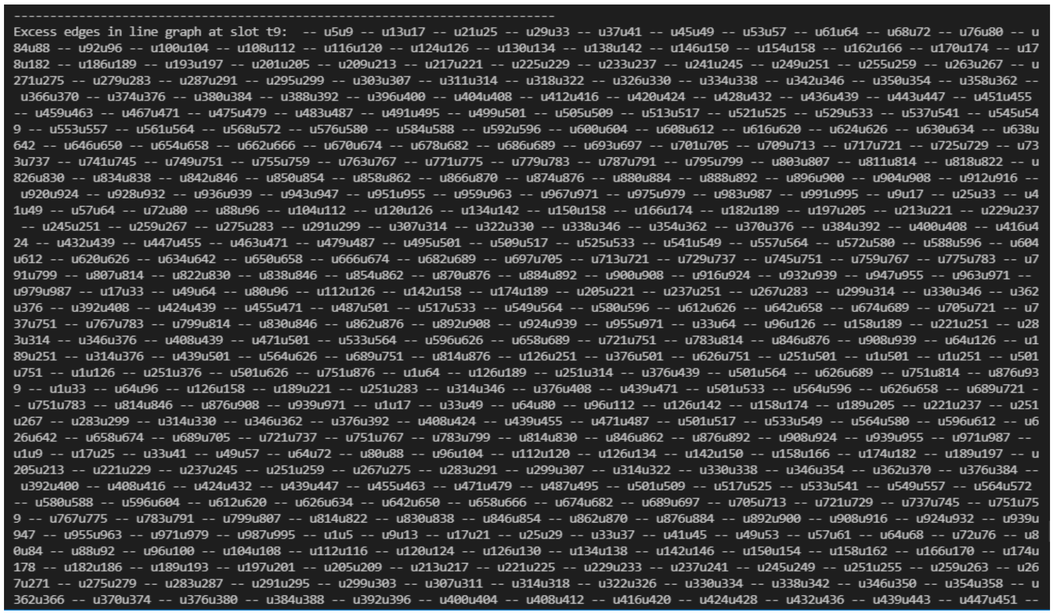
```
-------------------------------------------------------------------------
Excess edges in line graph at slot t1:
Number of excess edges in line graph : 0

-------------------------------------------------------------------------
Excess edges in line graph at slot t2:  -- u1u501
Number of excess edges in line graph : 1

-------------------------------------------------------------------------
Excess edges in line graph at slot t3:  -- u251u501 -- u1u501 -- u1u251 -- u501u751
Number of excess edges in line graph : 4

-------------------------------------------------------------------------
Excess edges in line graph at slot t4:  -- u126u251 -- u376u501 -- u626u751 -- u251u501 -- u1u501 -- u1u251 -- u501u751 -- u1u126 -- u251u376
-- u501u626 -- u751u876
Number of excess edges in line graph : 11

-------------------------------------------------------------------------
Excess edges in line graph at slot t5:  -- u64u126 -- u189u251 -- u314u376 -- u439u501 -- u564u626 -- u689u751 -- u814u876 -- u126u251 -- u376
u501 -- u626u751 -- u251u501 -- u1u501 -- u1u251 -- u501u751 -- u1u126 -- u251u376 -- u501u626 -- u751u876 -- u1u64 -- u126u189 -- u251u314 --
 u376u439 -- u501u564 -- u626u689 -- u751u814 -- u876u939
Number of excess edges in line graph : 26

-------------------------------------------------------------------------
Excess edges in line graph at slot t6:  -- u33u64 -- u96u126 -- u158u189 -- u221u251 -- u283u314 -- u346u376 -- u408u439 -- u471u501 -- u533u5
64 -- u596u626 -- u658u689 -- u721u751 -- u783u814 -- u846u876 -- u908u939 -- u64u126 -- u189u251 -- u314u376 -- u439u501 -- u564u626 -- u689u
751 -- u814u876 -- u126u251 -- u376u501 -- u626u751 -- u251u501 -- u1u501 -- u1u251 -- u501u751 -- u1u126 -- u251u376 -- u501u626 -- u751u876
-- u1u64 -- u126u189 -- u251u314 -- u376u439 -- u501u564 -- u626u689 -- u751u814 -- u876u939 -- u1u33 -- u64u96 -- u126u158 -- u189u221 -- u25
1u283 -- u314u346 -- u376u408 -- u439u471 -- u501u533 -- u564u596 -- u626u658 -- u689u721 -- u751u783 -- u814u846 -- u876u908 -- u939u971
Number of excess edges in line graph : 57
```

Figure A.5: Excess edges in line graph construction at slots ‘t1’ to ‘t6’



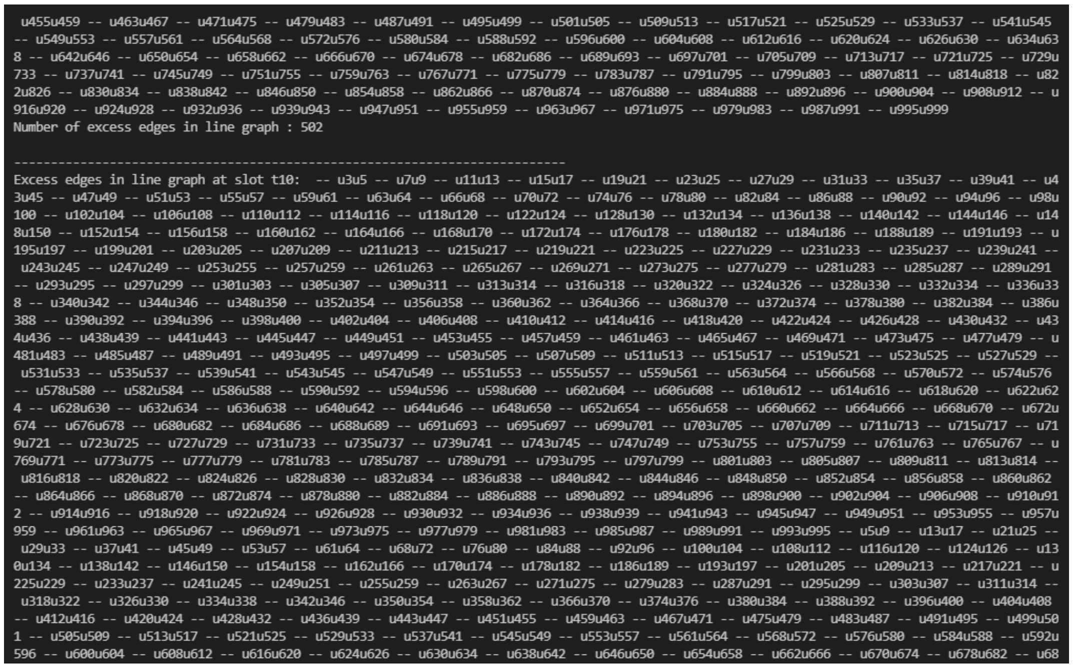
```
-------------------------------------------------------------------------
Excess edges in line graph at slot t7:  -- u17u33 -- u49u64 -- u80u96 -- u112u126 -- u142u158 -- u174u189 -- u205u221 -- u237u251 -- u267u283
-- u299u314 -- u330u346 -- u362u376 -- u392u408 -- u424u439 -- u455u471 -- u487u501 -- u517u533 -- u549u564 -- u580u596 -- u612u626 -- u642u65
8 -- u674u689 -- u705u721 -- u737u751 -- u767u783 -- u799u814 -- u830u846 -- u862u876 -- u892u908 -- u924u939 -- u955u971 -- u33u64 -- u96u126
 -- u158u189 -- u221u251 -- u283u314 -- u346u376 -- u408u439 -- u471u501 -- u533u564 -- u596u626 -- u658u689 -- u721u751 -- u783u814 -- u846u8
76 -- u908u939 -- u64u126 -- u189u251 -- u314u376 -- u439u501 -- u564u626 -- u689u751 -- u814u876 -- u126u251 -- u376u501 -- u626u751 -- u251u
501 -- u1u501 -- u1u251 -- u501u751 -- u1u126 -- u251u376 -- u501u626 -- u751u876 -- u1u64 -- u126u189 -- u251u314 -- u376u439 -- u501u564 --
u626u689 -- u751u814 -- u876u939 -- u1u33 -- u64u96 -- u126u158 -- u189u221 -- u251u283 -- u314u346 -- u376u408 -- u439u471 -- u501u533 -- u56
4u596 -- u626u658 -- u689u721 -- u751u783 -- u814u846 -- u876u908 -- u939u971 -- u1u17 -- u33u49 -- u64u80 -- u96u112 -- u126u142 -- u158u174
-- u189u205 -- u221u237 -- u251u267 -- u283u299 -- u314u330 -- u346u362 -- u376u392 -- u408u424 -- u439u455 -- u471u487 -- u501u517 -- u533u54
9 -- u564u580 -- u596u612 -- u626u642 -- u658u674 -- u689u705 -- u721u737 -- u751u767 -- u783u799 -- u814u830 -- u846u862 -- u876u892 -- u908u
924 -- u939u955 -- u971u987
Number of excess edges in line graph : 120

-------------------------------------------------------------------------
Excess edges in line graph at slot t8:  -- u9u17 -- u25u33 -- u41u49 -- u57u64 -- u72u80 -- u88u96 -- u104u112 -- u120u126 -- u134u142 -- u150
u158 -- u166u174 -- u182u189 -- u197u205 -- u213u221 -- u229u237 -- u245u251 -- u259u267 -- u275u283 -- u291u299 -- u307u314 -- u322u330 -- u3
38u346 -- u354u362 -- u370u376 -- u384u392 -- u400u408 -- u416u424 -- u432u439 -- u447u455 -- u463u471 -- u479u487 -- u495u501 -- u509u517 --
u525u533 -- u541u549 -- u557u564 -- u572u580 -- u588u596 -- u604u612 -- u620u626 -- u634u642 -- u650u658 -- u666u674 -- u682u689 -- u697u705 -
- u713u721 -- u729u737 -- u745u751 -- u759u767 -- u775u783 -- u791u799 -- u807u814 -- u822u830 -- u838u846 -- u854u862 -- u870u876 -- u884u892
 -- u900u908 -- u916u924 -- u932u939 -- u947u955 -- u963u971 -- u979u987 -- u17u33 -- u49u64 -- u80u96 -- u112u126 -- u142u158 -- u174u189 --
u205u221 -- u237u251 -- u267u283 -- u299u314 -- u330u346 -- u362u376 -- u392u408 -- u424u439 -- u455u471 -- u487u501 -- u517u533 -- u549u564 -
- u580u596 -- u612u626 -- u642u658 -- u674u689 -- u705u721 -- u737u751 -- u767u783 -- u799u814 -- u830u846 -- u862u876 -- u892u908 -- u924u939
 -- u955u971 -- u33u64 -- u96u126 -- u158u189 -- u221u251 -- u283u314 -- u346u376 -- u408u439 -- u471u501 -- u533u564 -- u596u626 -- u658u689
-- u721u751 -- u783u814 -- u846u876 -- u908u939 -- u64u126 -- u189u251 -- u314u376 -- u439u501 -- u564u626 -- u689u751 -- u814u876 -- u126u251
 -- u376u501 -- u626u751 -- u251u501 -- u1u501 -- u1u251 -- u501u751 -- u1u126 -- u251u376 -- u501u626 -- u751u876 -- u1u64 -- u126u189 -- u25
1u314 -- u376u439 -- u501u564 -- u626u689 -- u751u814 -- u876u939 -- u1u33 -- u64u96 -- u126u158 -- u189u221 -- u251u283 -- u314u346 -- u376u4
08 -- u439u471 -- u501u533 -- u564u596 -- u626u658 -- u689u721 -- u751u783 -- u814u846 -- u876u908 -- u939u971 -- u1u17 -- u33u49 -- u64u80 --
 u96u112 -- u126u142 -- u158u174 -- u189u205 -- u221u237 -- u251u267 -- u283u299 -- u314u330 -- u346u362 -- u376u392 -- u408u424 -- u439u455 -
- u471u487 -- u501u517 -- u533u549 -- u564u580 -- u596u612 -- u626u642 -- u658u674 -- u689u705 -- u721u737 -- u751u767 -- u783u799 -- u814u830
 -- u846u862 -- u876u892 -- u908u924 -- u939u955 -- u971u987 -- u1u9 -- u17u25 -- u33u41 -- u49u57 -- u64u72 -- u80u88 -- u96u104 -- u112u120
-- u126u134 -- u142u150 -- u158u166 -- u174u182 -- u189u197 -- u205u213 -- u221u229 -- u237u245 -- u251u259 -- u267u275 -- u283u291 -- u299u30
7 -- u314u322 -- u330u338 -- u346u354 -- u362u370 -- u376u384 -- u392u400 -- u408u416 -- u424u432 -- u439u447 -- u455u463 -- u471u479 -- u487u
495 -- u501u509 -- u517u525 -- u533u541 -- u549u557 -- u564u572 -- u580u588 -- u596u604 -- u612u620 -- u626u634 -- u642u650 -- u658u666 -- u67
4u682 -- u689u697 -- u705u713 -- u721u729 -- u737u745 -- u751u759 -- u767u775 -- u783u791 -- u799u807 -- u814u822 -- u830u838 -- u846u854 -- u
862u870 -- u876u884 -- u892u900 -- u908u916 -- u924u932 -- u939u947 -- u955u963 -- u971u979 -- u987u995
Number of excess edges in line graph : 247
```

Figure A.6: Excess edges in line graph construction at slots ‘t7’ and ‘t8’



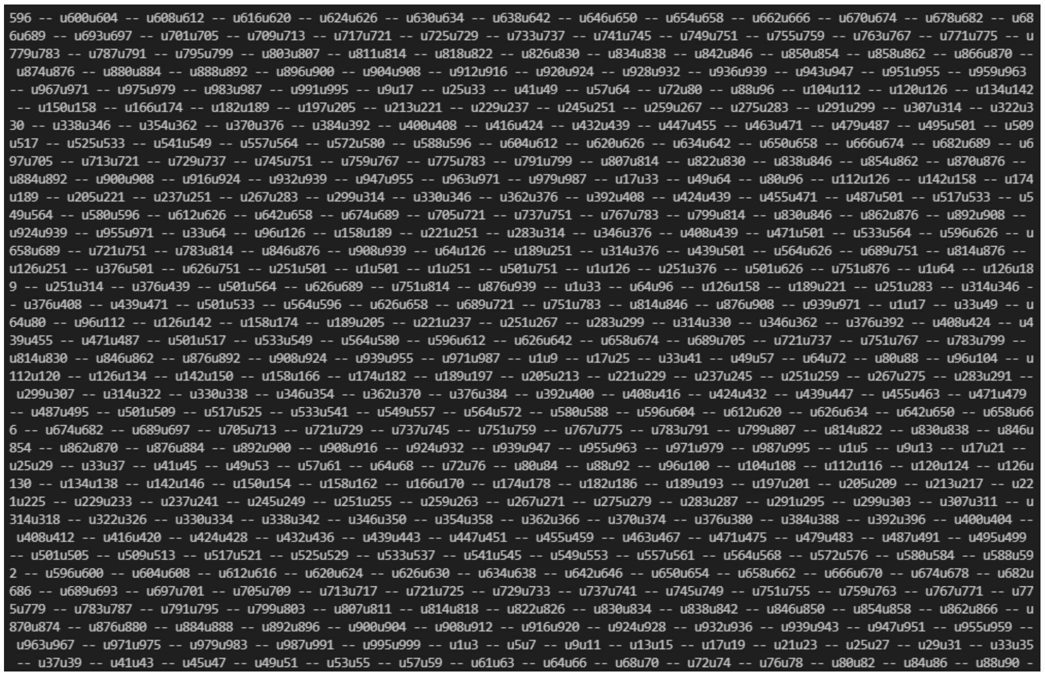
```
-------------------------------------------------------------------------
Excess edges in line graph at slot t9:  -- u5u9 -- u13u17 -- u21u25 -- u29u33 -- u37u41 -- u45u49 -- u53u57 -- u61u64 -- u68u72 -- u76u80 -- u
84u88 -- u92u96 -- u100u104 -- u108u112 -- u116u120 -- u124u126 -- u130u134 -- u138u142 -- u146u150 -- u154u158 -- u162u166 -- u170u174 -- u17
8u182 -- u186u189 -- u193u197 -- u201u205 -- u209u213 -- u217u221 -- u225u229 -- u233u237 -- u241u245 -- u249u251 -- u255u259 -- u263u267 -- u
271u275 -- u279u283 -- u287u291 -- u295u299 -- u303u307 -- u311u314 -- u318u322 -- u326u330 -- u334u338 -- u342u346 -- u350u354 -- u358u362 --
 u366u370 -- u374u376 -- u380u384 -- u388u392 -- u396u400 -- u404u408 -- u412u416 -- u420u424 -- u428u432 -- u436u439 -- u443u447 -- u451u455
-- u459u463 -- u467u471 -- u475u479 -- u483u487 -- u491u495 -- u499u501 -- u505u509 -- u513u517 -- u521u525 -- u529u533 -- u537u541 -- u545u54
9 -- u553u557 -- u561u564 -- u568u572 -- u576u580 -- u584u588 -- u592u596 -- u600u604 -- u608u612 -- u616u620 -- u624u626 -- u630u634 -- u638u
642 -- u646u650 -- u654u658 -- u662u666 -- u670u674 -- u678u682 -- u686u689 -- u693u697 -- u701u705 -- u709u713 -- u717u721 -- u725u729 -- u73
3u737 -- u741u745 -- u749u751 -- u755u759 -- u763u767 -- u771u775 -- u779u783 -- u787u791 -- u795u799 -- u803u807 -- u811u814 -- u818u822 -- u
826u830 -- u834u838 -- u842u846 -- u850u854 -- u858u862 -- u866u870 -- u874u876 -- u880u884 -- u888u892 -- u896u900 -- u904u908 -- u912u916 --
 u920u924 -- u928u932 -- u936u939 -- u943u947 -- u951u955 -- u959u963 -- u967u971 -- u975u979 -- u983u987 -- u991u995 -- u9u17 -- u25u33 -- u4
1u49 -- u57u64 -- u72u80 -- u88u96 -- u104u112 -- u120u126 -- u134u142 -- u150u158 -- u166u174 -- u182u189 -- u197u205 -- u213u221 -- u229u237
 -- u245u251 -- u259u267 -- u275u283 -- u291u299 -- u307u314 -- u322u330 -- u338u346 -- u354u362 -- u370u376 -- u384u392 -- u400u408 -- u416u4
24 -- u432u439 -- u447u455 -- u463u471 -- u479u487 -- u495u501 -- u509u517 -- u525u533 -- u541u549 -- u557u564 -- u572u580 -- u588u596 -- u604
u612 -- u620u626 -- u634u642 -- u650u658 -- u666u674 -- u682u689 -- u697u705 -- u713u721 -- u729u737 -- u745u751 -- u759u767 -- u775u783 -- u7
91u799 -- u807u814 -- u822u830 -- u838u846 -- u854u862 -- u870u876 -- u884u892 -- u900u908 -- u916u924 -- u932u939 -- u947u955 -- u963u971 --
u979u987 -- u17u33 -- u49u64 -- u80u96 -- u112u126 -- u142u158 -- u174u189 -- u205u221 -- u237u251 -- u267u283 -- u299u314 -- u330u346 -- u362
u376 -- u392u408 -- u424u439 -- u455u471 -- u487u501 -- u517u533 -- u549u564 -- u580u596 -- u612u626 -- u642u658 -- u674u689 -- u705u721 -- u7
37u751 -- u767u783 -- u799u814 -- u830u846 -- u862u876 -- u892u908 -- u924u939 -- u955u971 -- u33u64 -- u96u126 -- u158u189 -- u221u251 -- u28
3u314 -- u346u376 -- u408u439 -- u471u501 -- u533u564 -- u596u626 -- u658u689 -- u721u751 -- u783u814 -- u846u876 -- u908u939 -- u64u126 -- u1
89u251 -- u314u376 -- u439u501 -- u564u626 -- u689u751 -- u814u876 -- u126u251 -- u376u501 -- u626u751 -- u251u501 -- u1u501 -- u1u251 -- u501
u751 -- u1u126 -- u251u376 -- u501u626 -- u751u876 -- u1u64 -- u126u189 -- u251u314 -- u376u439 -- u501u564 -- u626u689 -- u751u814 -- u876u93
9 -- u1u33 -- u64u96 -- u126u158 -- u189u221 -- u251u283 -- u314u346 -- u376u408 -- u439u471 -- u501u533 -- u564u596 -- u626u658 -- u689u721 -
- u751u783 -- u814u846 -- u876u908 -- u939u971 -- u1u17 -- u33u49 -- u64u80 -- u96u112 -- u126u142 -- u158u174 -- u189u205 -- u221u237 -- u251
u267 -- u283u299 -- u314u330 -- u346u362 -- u376u392 -- u408u424 -- u439u455 -- u471u487 -- u501u517 -- u533u549 -- u564u580 -- u596u612 -- u6
26u642 -- u658u674 -- u689u705 -- u721u737 -- u751u767 -- u783u799 -- u814u830 -- u846u862 -- u876u892 -- u908u924 -- u939u955 -- u971u987 --
u1u9 -- u17u25 -- u33u41 -- u49u57 -- u64u72 -- u80u88 -- u96u104 -- u112u120 -- u126u134 -- u142u150 -- u158u166 -- u174u182 -- u189u197 -- u
205u213 -- u221u229 -- u237u245 -- u251u259 -- u267u275 -- u283u291 -- u299u307 -- u314u322 -- u330u338 -- u346u354 -- u362u370 -- u376u384 --
 u392u400 -- u408u416 -- u424u432 -- u439u447 -- u455u463 -- u471u479 -- u487u495 -- u501u509 -- u517u525 -- u533u541 -- u549u557 -- u564u572
-- u580u588 -- u596u604 -- u612u620 -- u626u634 -- u642u650 -- u658u666 -- u674u682 -- u689u697 -- u705u713 -- u721u729 -- u737u745 -- u751u75
9 -- u767u775 -- u783u791 -- u799u807 -- u814u822 -- u830u838 -- u846u854 -- u862u870 -- u876u884 -- u892u900 -- u908u916 -- u924u932 -- u939u
947 -- u955u963 -- u971u979 -- u987u995 -- u1u5 -- u9u13 -- u17u21 -- u25u29 -- u33u37 -- u41u45 -- u49u53 -- u57u61 -- u64u68 -- u72u76 -- u8
0u84 -- u88u92 -- u96u100 -- u104u108 -- u112u116 -- u120u124 -- u126u130 -- u134u138 -- u142u146 -- u150u154 -- u158u162 -- u166u170 -- u174u
178 -- u182u186 -- u189u193 -- u197u201 -- u205u209 -- u213u217 -- u221u225 -- u229u233 -- u237u241 -- u245u249 -- u251u255 -- u259u263 -- u26
7u271 -- u275u279 -- u283u287 -- u291u295 -- u299u303 -- u307u311 -- u314u318 -- u322u326 -- u330u334 -- u338u342 -- u346u350 -- u354u358 -- u
362u366 -- u370u374 -- u376u380 -- u384u388 -- u392u396 -- u400u404 -- u408u412 -- u416u420 -- u424u428 -- u432u436 -- u439u443 -- u447u451 --
```

Figure A.7: Excess edges in line graph construction at slot ‘t9’



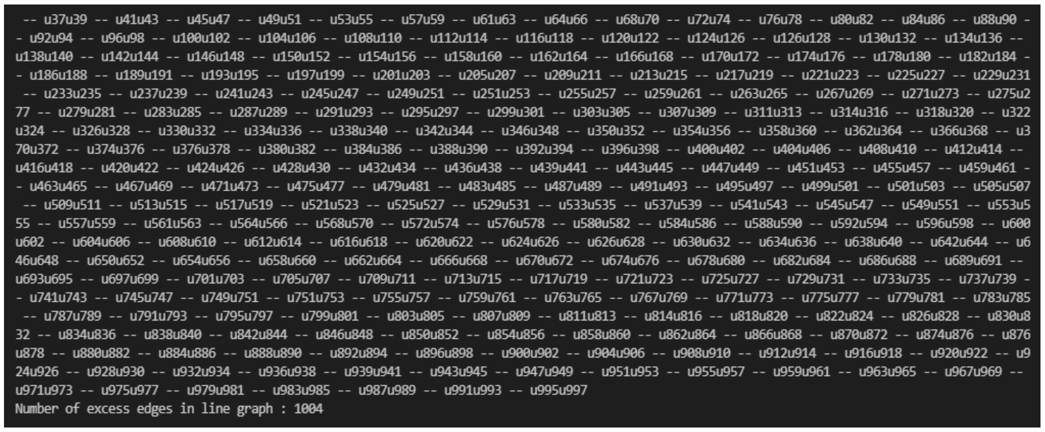
```
 u455u459 -- u463u467 -- u471u475 -- u479u483 -- u487u491 -- u495u499 -- u501u505 -- u509u513 -- u517u521 -- u525u529 -- u533u537 -- u541u545
-- u549u553 -- u557u561 -- u564u568 -- u572u576 -- u580u584 -- u588u592 -- u596u600 -- u604u608 -- u612u616 -- u620u624 -- u626u630 -- u634u63
8 -- u642u646 -- u650u654 -- u658u662 -- u666u670 -- u674u678 -- u682u686 -- u689u693 -- u697u701 -- u705u709 -- u713u717 -- u721u725 -- u729u
733 -- u737u741 -- u745u749 -- u751u755 -- u759u763 -- u767u771 -- u775u779 -- u783u787 -- u791u795 -- u799u803 -- u807u811 -- u814u818 -- u82
2u826 -- u830u834 -- u838u842 -- u846u850 -- u854u858 -- u862u866 -- u870u874 -- u876u880 -- u884u888 -- u892u896 -- u900u904 -- u908u912 -- u
916u920 -- u924u928 -- u932u936 -- u939u943 -- u947u951 -- u955u959 -- u963u967 -- u971u975 -- u979u983 -- u987u991 -- u995u999
Number of excess edges in line graph : 502

-------------------------------------------------------------------------
Excess edges in line graph at slot t10:  -- u3u5 -- u7u9 -- u11u13 -- u15u17 -- u19u21 -- u23u25 -- u27u29 -- u31u33 -- u35u37 -- u39u41 -- u4
3u45 -- u47u49 -- u51u53 -- u55u57 -- u59u61 -- u63u64 -- u66u68 -- u70u72 -- u74u76 -- u78u80 -- u82u84 -- u86u88 -- u90u92 -- u94u96 -- u98u
100 -- u102u104 -- u106u108 -- u110u112 -- u114u116 -- u118u120 -- u122u124 -- u128u130 -- u132u134 -- u136u138 -- u140u142 -- u144u146 -- u14
8u150 -- u152u154 -- u156u158 -- u160u162 -- u164u166 -- u168u170 -- u172u174 -- u176u178 -- u180u182 -- u184u186 -- u188u189 -- u191u193 -- u
195u197 -- u199u201 -- u203u205 -- u207u209 -- u211u213 -- u215u217 -- u219u221 -- u223u225 -- u227u229 -- u231u233 -- u235u237 -- u239u241 --
 u243u245 -- u247u249 -- u253u255 -- u257u259 -- u261u263 -- u265u267 -- u269u271 -- u273u275 -- u277u279 -- u281u283 -- u285u287 -- u289u291
-- u293u295 -- u297u299 -- u301u303 -- u305u307 -- u309u311 -- u313u314 -- u316u318 -- u320u322 -- u324u326 -- u328u330 -- u332u334 -- u336u33
8 -- u340u342 -- u344u346 -- u348u350 -- u352u354 -- u356u358 -- u360u362 -- u364u366 -- u368u370 -- u372u374 -- u378u380 -- u382u384 -- u386u
388 -- u390u392 -- u394u396 -- u398u400 -- u402u404 -- u406u408 -- u410u412 -- u414u416 -- u418u420 -- u422u424 -- u426u428 -- u430u432 -- u43
4u436 -- u438u439 -- u441u443 -- u445u447 -- u449u451 -- u453u455 -- u457u459 -- u461u463 -- u465u467 -- u469u471 -- u473u475 -- u477u479 -- u
481u483 -- u485u487 -- u489u491 -- u493u495 -- u497u499 -- u503u505 -- u507u509 -- u511u513 -- u515u517 -- u519u521 -- u523u525 -- u527u529 --
 u531u533 -- u535u537 -- u539u541 -- u543u545 -- u547u549 -- u551u553 -- u555u557 -- u559u561 -- u563u564 -- u566u568 -- u570u572 -- u574u576
-- u578u580 -- u582u584 -- u586u588 -- u590u592 -- u594u596 -- u598u600 -- u602u604 -- u606u608 -- u610u612 -- u614u616 -- u618u620 -- u622u62
4 -- u628u630 -- u632u634 -- u636u638 -- u640u642 -- u644u646 -- u648u650 -- u652u654 -- u656u658 -- u660u662 -- u664u666 -- u668u670 -- u672u
674 -- u676u678 -- u680u682 -- u684u686 -- u688u689 -- u691u693 -- u695u697 -- u699u701 -- u703u705 -- u707u709 -- u711u713 -- u715u717 -- u71
9u721 -- u723u725 -- u727u729 -- u731u733 -- u735u737 -- u739u741 -- u743u745 -- u747u749 -- u753u755 -- u757u759 -- u761u763 -- u765u767 -- u
769u771 -- u773u775 -- u777u779 -- u781u783 -- u785u787 -- u789u791 -- u793u795 -- u797u799 -- u801u803 -- u805u807 -- u809u811 -- u813u814 --
 u816u818 -- u820u822 -- u824u826 -- u828u830 -- u832u834 -- u836u838 -- u840u842 -- u844u846 -- u848u850 -- u852u854 -- u856u858 -- u860u862
-- u864u866 -- u868u870 -- u872u874 -- u878u880 -- u882u884 -- u886u888 -- u890u892 -- u894u896 -- u898u900 -- u902u904 -- u906u908 -- u910u91
2 -- u914u916 -- u918u920 -- u922u924 -- u926u928 -- u930u932 -- u934u936 -- u938u939 -- u941u943 -- u945u947 -- u949u951 -- u953u955 -- u957u
959 -- u961u963 -- u965u967 -- u969u971 -- u973u975 -- u977u979 -- u981u983 -- u985u987 -- u989u991 -- u993u995 -- u5u9 -- u13u17 -- u21u25 --
 u29u33 -- u37u41 -- u45u49 -- u53u57 -- u61u64 -- u68u72 -- u76u80 -- u84u88 -- u92u96 -- u100u104 -- u108u112 -- u116u120 -- u124u126 -- u13
0u134 -- u138u142 -- u146u150 -- u154u158 -- u162u166 -- u170u174 -- u178u182 -- u186u189 -- u193u197 -- u201u205 -- u209u213 -- u217u221 -- u
225u229 -- u233u237 -- u241u245 -- u249u251 -- u255u259 -- u263u267 -- u271u275 -- u279u283 -- u287u291 -- u295u299 -- u303u307 -- u311u314 --
 u318u322 -- u326u330 -- u334u338 -- u342u346 -- u350u354 -- u358u362 -- u366u370 -- u374u376 -- u380u384 -- u388u392 -- u396u400 -- u404u408
-- u412u416 -- u420u424 -- u428u432 -- u436u439 -- u443u447 -- u451u455 -- u459u463 -- u467u471 -- u475u479 -- u483u487 -- u491u495 -- u499u50
1 -- u505u509 -- u513u517 -- u521u525 -- u529u533 -- u537u541 -- u545u549 -- u553u557 -- u561u564 -- u568u572 -- u576u580 -- u584u588 -- u592u
596 -- u600u604 -- u608u612 -- u616u620 -- u624u626 -- u630u634 -- u638u642 -- u646u650 -- u654u658 -- u662u666 -- u670u674 -- u678u682 -- u68
```

Figure A.8: Excess edges in line graph construction at slot ‘t9’ and ‘t10’



```
596 -- u600u604 -- u608u612 -- u616u620 -- u624u626 -- u630u634 -- u638u642 -- u646u650 -- u654u658 -- u662u666 -- u670u674 -- u678u682 -- u68
6u689 -- u693u697 -- u701u705 -- u709u713 -- u717u721 -- u725u729 -- u733u737 -- u741u745 -- u749u751 -- u755u759 -- u763u767 -- u771u775 -- u
779u783 -- u787u791 -- u795u799 -- u803u807 -- u811u814 -- u818u822 -- u826u830 -- u834u838 -- u842u846 -- u850u854 -- u858u862 -- u866u870 --
 u874u876 -- u880u884 -- u888u892 -- u896u900 -- u904u908 -- u912u916 -- u920u924 -- u928u932 -- u936u939 -- u943u947 -- u951u955 -- u959u963
-- u967u971 -- u975u979 -- u983u987 -- u991u995 -- u9u17 -- u25u33 -- u41u49 -- u57u64 -- u72u80 -- u88u96 -- u104u112 -- u120u126 -- u134u142
 -- u150u158 -- u166u174 -- u182u189 -- u197u205 -- u213u221 -- u229u237 -- u245u251 -- u259u267 -- u275u283 -- u291u299 -- u307u314 -- u322u3
30 -- u338u346 -- u354u362 -- u370u376 -- u384u392 -- u400u408 -- u416u424 -- u432u439 -- u447u455 -- u463u471 -- u479u487 -- u495u501 -- u509
u517 -- u525u533 -- u541u549 -- u557u564 -- u572u580 -- u588u596 -- u604u612 -- u620u626 -- u634u642 -- u650u658 -- u666u674 -- u682u689 -- u6
97u705 -- u713u721 -- u729u737 -- u745u751 -- u759u767 -- u775u783 -- u791u799 -- u807u814 -- u822u830 -- u838u846 -- u854u862 -- u870u876 --
u884u892 -- u900u908 -- u916u924 -- u932u939 -- u947u955 -- u963u971 -- u979u987 -- u17u33 -- u49u64 -- u80u96 -- u112u126 -- u142u158 -- u174
u189 -- u205u221 -- u237u251 -- u267u283 -- u299u314 -- u330u346 -- u362u376 -- u392u408 -- u424u439 -- u455u471 -- u487u501 -- u517u533 -- u5
49u564 -- u580u596 -- u612u626 -- u642u658 -- u674u689 -- u705u721 -- u737u751 -- u767u783 -- u799u814 -- u830u846 -- u862u876 -- u892u908 --
u924u939 -- u955u971 -- u33u64 -- u96u126 -- u158u189 -- u221u251 -- u283u314 -- u346u376 -- u408u439 -- u471u501 -- u533u564 -- u596u626 -- u
658u689 -- u721u751 -- u783u814 -- u846u876 -- u908u939 -- u64u126 -- u189u251 -- u314u376 -- u439u501 -- u564u626 -- u689u751 -- u814u876 --
u126u251 -- u376u501 -- u626u751 -- u251u501 -- u1u501 -- u1u251 -- u501u751 -- u1u126 -- u251u376 -- u501u626 -- u751u876 -- u1u64 -- u126u18
9 -- u251u314 -- u376u439 -- u501u564 -- u626u689 -- u751u814 -- u876u939 -- u1u33 -- u64u96 -- u126u158 -- u189u221 -- u251u283 -- u314u346 -
- u376u408 -- u439u471 -- u501u533 -- u564u596 -- u626u658 -- u689u721 -- u751u783 -- u814u846 -- u876u908 -- u939u971 -- u1u17 -- u33u49 -- u
64u80 -- u96u112 -- u126u142 -- u158u174 -- u189u205 -- u221u237 -- u251u267 -- u283u299 -- u314u330 -- u346u362 -- u376u392 -- u408u424 -- u4
39u455 -- u471u487 -- u501u517 -- u533u549 -- u564u580 -- u596u612 -- u626u642 -- u658u674 -- u689u705 -- u721u737 -- u751u767 -- u783u799 --
u814u830 -- u846u862 -- u876u892 -- u908u924 -- u939u955 -- u971u987 -- u1u9 -- u17u25 -- u33u41 -- u49u57 -- u64u72 -- u80u88 -- u96u104 -- u
112u120 -- u126u134 -- u142u150 -- u158u166 -- u174u182 -- u189u197 -- u205u213 -- u221u229 -- u237u245 -- u251u259 -- u267u275 -- u283u291 --
 u299u307 -- u314u322 -- u330u338 -- u346u354 -- u362u370 -- u376u384 -- u392u400 -- u408u416 -- u424u432 -- u439u447 -- u455u463 -- u471u479
-- u487u495 -- u501u509 -- u517u525 -- u533u541 -- u549u557 -- u564u572 -- u580u588 -- u596u604 -- u612u620 -- u626u634 -- u642u650 -- u658u66
6 -- u674u682 -- u689u697 -- u705u713 -- u721u729 -- u737u745 -- u751u759 -- u767u775 -- u783u791 -- u799u807 -- u814u822 -- u830u838 -- u846u
854 -- u862u870 -- u876u884 -- u892u900 -- u908u916 -- u924u932 -- u939u947 -- u955u963 -- u971u979 -- u987u995 -- u1u5 -- u9u13 -- u17u21 --
u25u29 -- u33u37 -- u41u45 -- u49u53 -- u57u61 -- u64u68 -- u72u76 -- u80u84 -- u88u92 -- u96u100 -- u104u108 -- u112u116 -- u120u124 -- u126u
130 -- u134u138 -- u142u146 -- u150u154 -- u158u162 -- u166u170 -- u174u178 -- u182u186 -- u189u193 -- u197u201 -- u205u209 -- u213u217 -- u22
1u225 -- u229u233 -- u237u241 -- u245u249 -- u251u255 -- u259u263 -- u267u271 -- u275u279 -- u283u287 -- u291u295 -- u299u303 -- u307u311 -- u
314u318 -- u322u326 -- u330u334 -- u338u342 -- u346u350 -- u354u358 -- u362u366 -- u370u374 -- u376u380 -- u384u388 -- u392u396 -- u400u404 --
 u408u412 -- u416u420 -- u424u428 -- u432u436 -- u439u443 -- u447u451 -- u455u459 -- u463u467 -- u471u475 -- u479u483 -- u487u491 -- u495u499
-- u501u505 -- u509u513 -- u517u521 -- u525u529 -- u533u537 -- u541u545 -- u549u553 -- u557u561 -- u564u568 -- u572u576 -- u580u584 -- u588u59
2 -- u596u600 -- u604u608 -- u612u616 -- u620u624 -- u626u630 -- u634u638 -- u642u646 -- u650u654 -- u658u662 -- u666u670 -- u674u678 -- u682u
686 -- u689u693 -- u697u701 -- u705u709 -- u713u717 -- u721u725 -- u729u733 -- u737u741 -- u745u749 -- u751u755 -- u759u763 -- u767u771 -- u77
5u779 -- u783u787 -- u791u795 -- u799u803 -- u807u811 -- u814u818 -- u822u826 -- u830u834 -- u838u842 -- u846u850 -- u854u858 -- u862u866 -- u
870u874 -- u876u880 -- u884u888 -- u892u896 -- u900u904 -- u908u912 -- u916u920 -- u924u928 -- u932u936 -- u939u943 -- u947u951 -- u955u959 --
 u963u967 -- u971u975 -- u979u983 -- u987u991 -- u995u999 -- u1u3 -- u5u7 -- u9u11 -- u13u15 -- u17u19 -- u21u23 -- u25u27 -- u29u31 -- u33u35
 -- u37u39 -- u41u43 -- u45u47 -- u49u51 -- u53u55 -- u57u59 -- u61u63 -- u64u66 -- u68u70 -- u72u74 -- u76u78 -- u80u82 -- u84u86 -- u88u90 -
```

Figure A.9: Excess edges in line graph construction at slot ‘t10’



```
 -- u37u39 -- u41u43 -- u45u47 -- u49u51 -- u53u55 -- u57u59 -- u61u63 -- u64u66 -- u68u70 -- u72u74 -- u76u78 -- u80u82 -- u84u86 -- u88u90 -
- u92u94 -- u96u98 -- u100u102 -- u104u106 -- u108u110 -- u112u114 -- u116u118 -- u120u122 -- u124u126 -- u126u128 -- u130u132 -- u134u136 --
u138u140 -- u142u144 -- u146u148 -- u150u152 -- u154u156 -- u158u160 -- u162u164 -- u166u168 -- u170u172 -- u174u176 -- u178u180 -- u182u184 -
- u186u188 -- u189u191 -- u193u195 -- u197u199 -- u201u203 -- u205u207 -- u209u211 -- u213u215 -- u217u219 -- u221u223 -- u225u227 -- u229u231
 -- u233u235 -- u237u239 -- u241u243 -- u245u247 -- u249u251 -- u251u253 -- u255u257 -- u259u261 -- u263u265 -- u267u269 -- u271u273 -- u275u2
77 -- u279u281 -- u283u285 -- u287u289 -- u291u293 -- u295u297 -- u299u301 -- u303u305 -- u307u309 -- u311u313 -- u314u316 -- u318u320 -- u322
u324 -- u326u328 -- u330u332 -- u334u336 -- u338u340 -- u342u344 -- u346u348 -- u350u352 -- u354u356 -- u358u360 -- u362u364 -- u366u368 -- u3
70u372 -- u374u376 -- u376u378 -- u380u382 -- u384u386 -- u388u390 -- u392u394 -- u396u398 -- u400u402 -- u404u406 -- u408u410 -- u412u414 --
u416u418 -- u420u422 -- u424u426 -- u428u430 -- u432u434 -- u436u438 -- u439u441 -- u443u445 -- u447u449 -- u451u453 -- u455u457 -- u459u461 -
- u463u465 -- u467u469 -- u471u473 -- u475u477 -- u479u481 -- u483u485 -- u487u489 -- u491u493 -- u495u497 -- u499u501 -- u501u503 -- u505u507
 -- u509u511 -- u513u515 -- u517u519 -- u521u523 -- u525u527 -- u529u531 -- u533u535 -- u537u539 -- u541u543 -- u545u547 -- u549u551 -- u553u5
55 -- u557u559 -- u561u563 -- u564u566 -- u568u570 -- u572u574 -- u576u578 -- u580u582 -- u584u586 -- u588u590 -- u592u594 -- u596u598 -- u600
u602 -- u604u606 -- u608u610 -- u612u614 -- u616u618 -- u620u622 -- u624u626 -- u626u628 -- u630u632 -- u634u636 -- u638u640 -- u642u644 -- u6
46u648 -- u650u652 -- u654u656 -- u658u660 -- u662u664 -- u666u668 -- u670u672 -- u674u676 -- u678u680 -- u682u684 -- u686u688 -- u689u691 --
u693u695 -- u697u699 -- u701u703 -- u705u707 -- u709u711 -- u713u715 -- u717u719 -- u721u723 -- u725u727 -- u729u731 -- u733u735 -- u737u739 -
- u741u743 -- u745u747 -- u749u751 -- u751u753 -- u755u757 -- u759u761 -- u763u765 -- u767u769 -- u771u773 -- u775u777 -- u779u781 -- u783u785
 -- u787u789 -- u791u793 -- u795u797 -- u799u801 -- u803u805 -- u807u809 -- u811u813 -- u814u816 -- u818u820 -- u822u824 -- u826u828 -- u830u8
32 -- u834u836 -- u838u840 -- u842u844 -- u846u848 -- u850u852 -- u854u856 -- u858u860 -- u862u864 -- u866u868 -- u870u872 -- u874u876 -- u876
u878 -- u880u882 -- u884u886 -- u888u890 -- u892u894 -- u896u898 -- u900u902 -- u904u906 -- u908u910 -- u912u914 -- u916u918 -- u920u922 -- u9
24u926 -- u928u930 -- u932u934 -- u936u938 -- u939u941 -- u943u945 -- u947u949 -- u951u953 -- u955u957 -- u959u961 -- u963u965 -- u967u969 --
u971u973 -- u975u977 -- u979u981 -- u983u985 -- u987u989 -- u991u993 -- u995u997
Number of excess edges in line graph : 1004
```

Figure A.10: Excess edges in line graph construction at slot ‘t10’

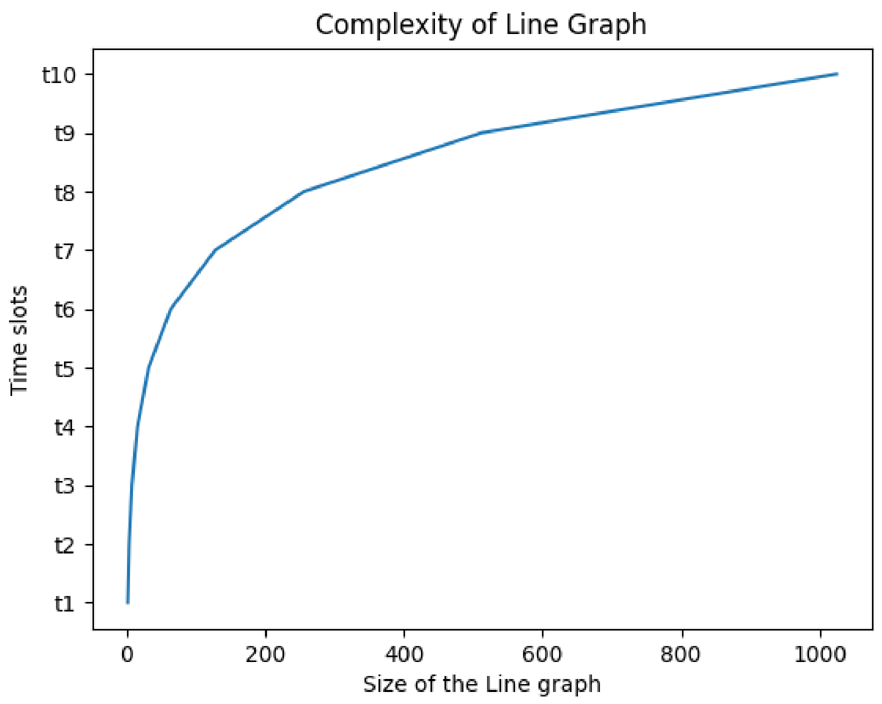


Figure A.11: Complexity of the line graph construction

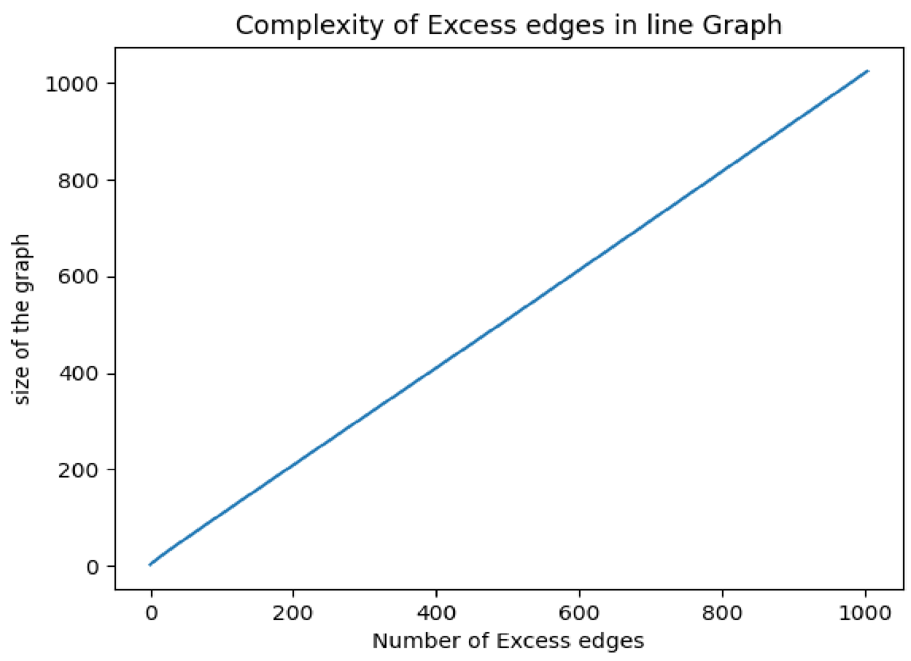
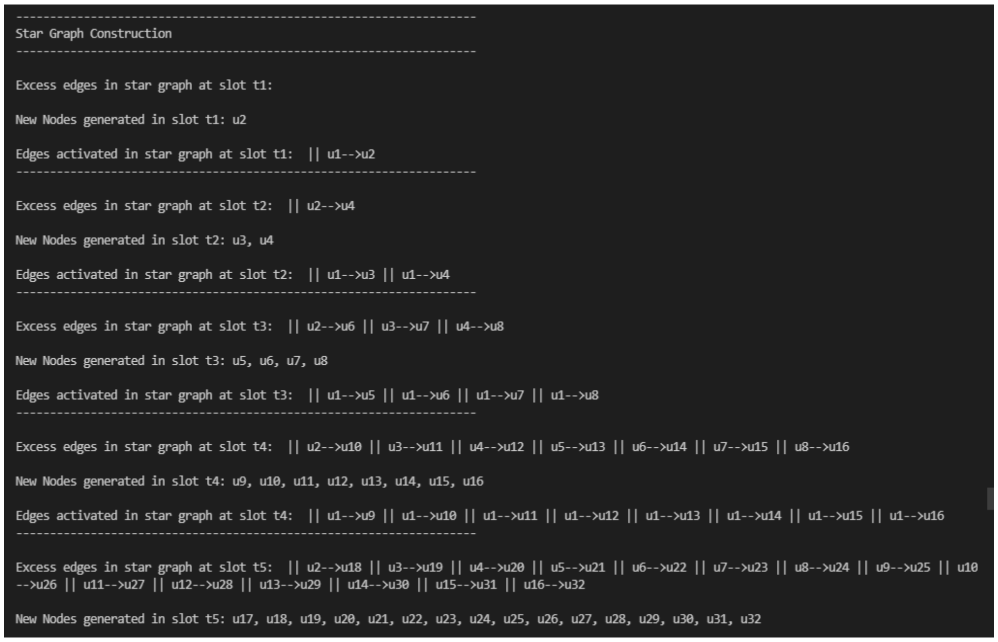


Figure A.12: Complexity of the excess edges in line graph construction

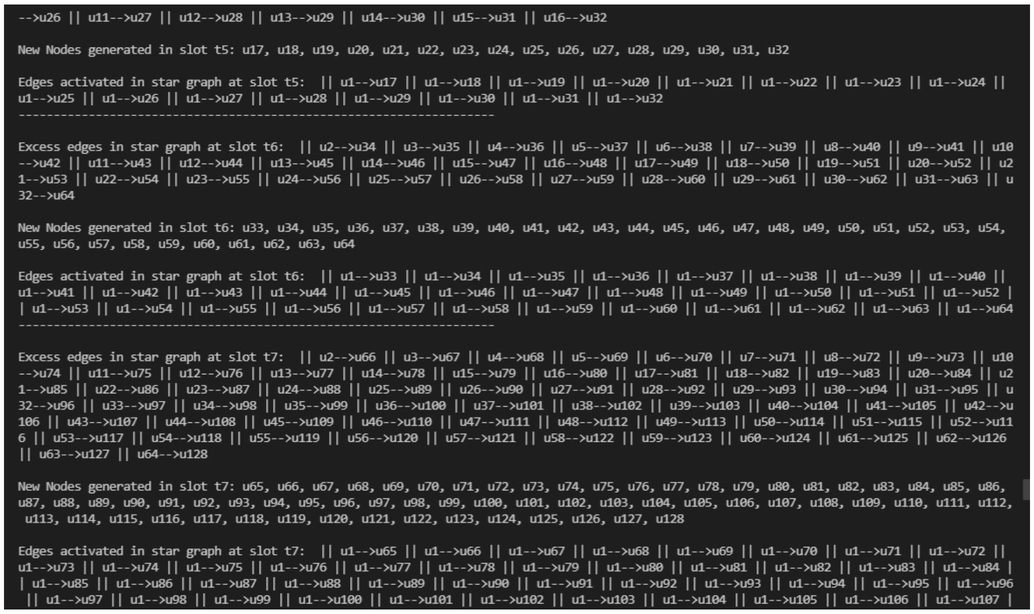
A.2 Star Graph

Star graph algorithm experimentation for the graph size of ‘n = 500’.



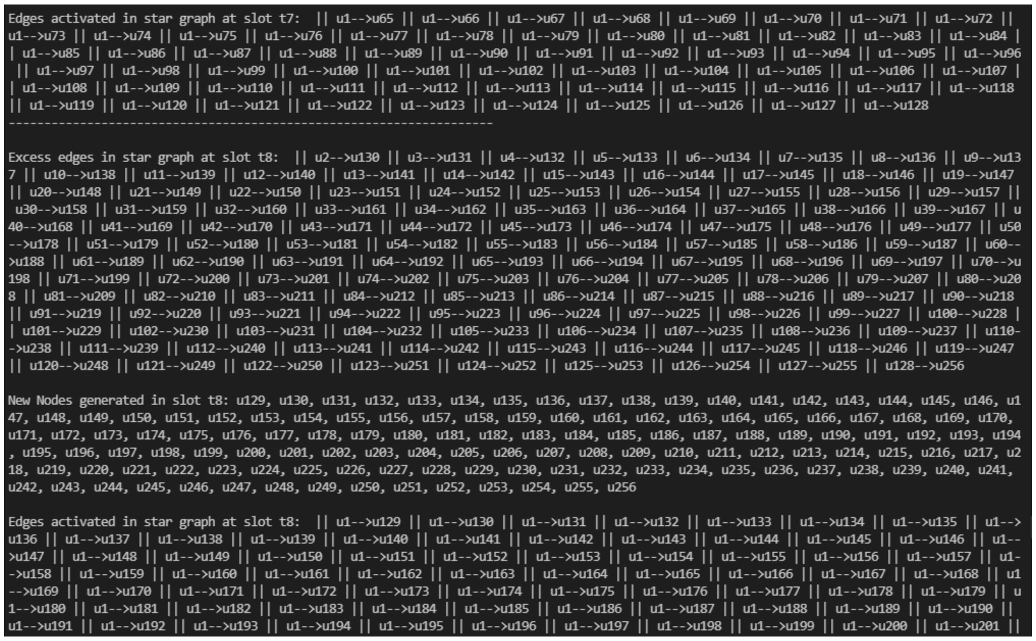
```
-----------------------------------------------------------------
Star Graph Construction
-----------------------------------------------------------------

Excess edges in star graph at slot t1:

New Nodes generated in slot t1: u2

Edges activated in star graph at slot t1:  || u1-->u2
-----------------------------------------------------------------

Excess edges in star graph at slot t2:  || u2-->u4

New Nodes generated in slot t2: u3, u4

Edges activated in star graph at slot t2:  || u1-->u3 || u1-->u4
-----------------------------------------------------------------

Excess edges in star graph at slot t3:  || u2-->u6 || u3-->u7 || u4-->u8

New Nodes generated in slot t3: u5, u6, u7, u8

Edges activated in star graph at slot t3:  || u1-->u5 || u1-->u6 || u1-->u7 || u1-->u8
-----------------------------------------------------------------

Excess edges in star graph at slot t4:  || u2-->u10 || u3-->u11 || u4-->u12 || u5-->u13 || u6-->u14 || u7-->u15 || u8-->u16

New Nodes generated in slot t4: u9, u10, u11, u12, u13, u14, u15, u16

Edges activated in star graph at slot t4:  || u1-->u9 || u1-->u10 || u1-->u11 || u1-->u12 || u1-->u13 || u1-->u14 || u1-->u15 || u1-->u16
-----------------------------------------------------------------

Excess edges in star graph at slot t5:  || u2-->u18 || u3-->u19 || u4-->u20 || u5-->u21 || u6-->u22 || u7-->u23 || u8-->u24 || u9-->u25 || u10
-->u26 || u11-->u27 || u12-->u28 || u13-->u29 || u14-->u30 || u15-->u31 || u16-->u32

New Nodes generated in slot t5: u17, u18, u19, u20, u21, u22, u23, u24, u25, u26, u27, u28, u29, u30, u31, u32
```

Figure A.13: Excess edges, new nodes generated and edges activated at slots ‘t1’ to ‘t5’



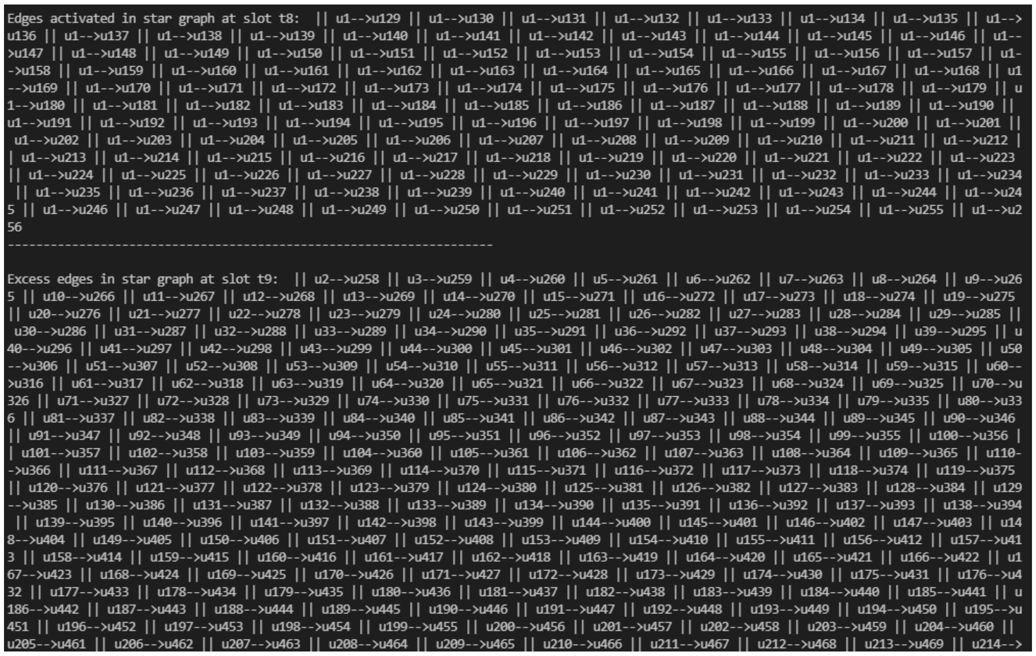
```
-->u26 || u11-->u27 || u12-->u28 || u13-->u29 || u14-->u30 || u15-->u31 || u16-->u32

New Nodes generated in slot t5: u17, u18, u19, u20, u21, u22, u23, u24, u25, u26, u27, u28, u29, u30, u31, u32

Edges activated in star graph at slot t5:  || u1-->u17 || u1-->u18 || u1-->u19 || u1-->u20 || u1-->u21 || u1-->u22 || u1-->u23 || u1-->u24 ||
u1-->u25 || u1-->u26 || u1-->u27 || u1-->u28 || u1-->u29 || u1-->u30 || u1-->u31 || u1-->u32
-----------------------------------------------------------------

Excess edges in star graph at slot t6:  || u2-->u34 || u3-->u35 || u4-->u36 || u5-->u37 || u6-->u38 || u7-->u39 || u8-->u40 || u9-->u41 || u10
-->u42 || u11-->u43 || u12-->u44 || u13-->u45 || u14-->u46 || u15-->u47 || u16-->u48 || u17-->u49 || u18-->u50 || u19-->u51 || u20-->u52 || u2
1-->u53 || u22-->u54 || u23-->u55 || u24-->u56 || u25-->u57 || u26-->u58 || u27-->u59 || u28-->u60 || u29-->u61 || u30-->u62 || u31-->u63 || u
32-->u64

New Nodes generated in slot t6: u33, u34, u35, u36, u37, u38, u39, u40, u41, u42, u43, u44, u45, u46, u47, u48, u49, u50, u51, u52, u53, u54,
u55, u56, u57, u58, u59, u60, u61, u62, u63, u64

Edges activated in star graph at slot t6:  || u1-->u33 || u1-->u34 || u1-->u35 || u1-->u36 || u1-->u37 || u1-->u38 || u1-->u39 || u1-->u40 ||
u1-->u41 || u1-->u42 || u1-->u43 || u1-->u44 || u1-->u45 || u1-->u46 || u1-->u47 || u1-->u48 || u1-->u49 || u1-->u50 || u1-->u51 || u1-->u52 |
| u1-->u53 || u1-->u54 || u1-->u55 || u1-->u56 || u1-->u57 || u1-->u58 || u1-->u59 || u1-->u60 || u1-->u61 || u1-->u62 || u1-->u63 || u1-->u64
-----------------------------------------------------------------

Excess edges in star graph at slot t7:  || u2-->u66 || u3-->u67 || u4-->u68 || u5-->u69 || u6-->u70 || u7-->u71 || u8-->u72 || u9-->u73 || u10
-->u74 || u11-->u75 || u12-->u76 || u13-->u77 || u14-->u78 || u15-->u79 || u16-->u80 || u17-->u81 || u18-->u82 || u19-->u83 || u20-->u84 || u2
1-->u85 || u22-->u86 || u23-->u87 || u24-->u88 || u25-->u89 || u26-->u90 || u27-->u91 || u28-->u92 || u29-->u93 || u30-->u94 || u31-->u95 || u
32-->u96 || u33-->u97 || u34-->u98 || u35-->u99 || u36-->u100 || u37-->u101 || u38-->u102 || u39-->u103 || u40-->u104 || u41-->u105 || u42-->u
106 || u43-->u107 || u44-->u108 || u45-->u109 || u46-->u110 || u47-->u111 || u48-->u112 || u49-->u113 || u50-->u114 || u51-->u115 || u52-->u11
6 || u53-->u117 || u54-->u118 || u55-->u119 || u56-->u120 || u57-->u121 || u58-->u122 || u59-->u123 || u60-->u124 || u61-->u125 || u62-->u126
|| u63-->u127 || u64-->u128

New Nodes generated in slot t7: u65, u66, u67, u68, u69, u70, u71, u72, u73, u74, u75, u76, u77, u78, u79, u80, u81, u82, u83, u84, u85, u86,
u87, u88, u89, u90, u91, u92, u93, u94, u95, u96, u97, u98, u99, u100, u101, u102, u103, u104, u105, u106, u107, u108, u109, u110, u111, u112,
 u113, u114, u115, u116, u117, u118, u119, u120, u121, u122, u123, u124, u125, u126, u127, u128

Edges activated in star graph at slot t7:  || u1-->u65 || u1-->u66 || u1-->u67 || u1-->u68 || u1-->u69 || u1-->u70 || u1-->u71 || u1-->u72 ||
u1-->u73 || u1-->u74 || u1-->u75 || u1-->u76 || u1-->u77 || u1-->u78 || u1-->u79 || u1-->u80 || u1-->u81 || u1-->u82 || u1-->u83 || u1-->u84 |
| u1-->u85 || u1-->u86 || u1-->u87 || u1-->u88 || u1-->u89 || u1-->u90 || u1-->u91 || u1-->u92 || u1-->u93 || u1-->u94 || u1-->u95 || u1-->u96
 || u1-->u97 || u1-->u98 || u1-->u99 || u1-->u100 || u1-->u101 || u1-->u102 || u1-->u103 || u1-->u104 || u1-->u105 || u1-->u106 || u1-->u107 |
```

Figure A.14: Excess edges, new nodes generated and edges activated at slots ‘t5’ to ‘t7’



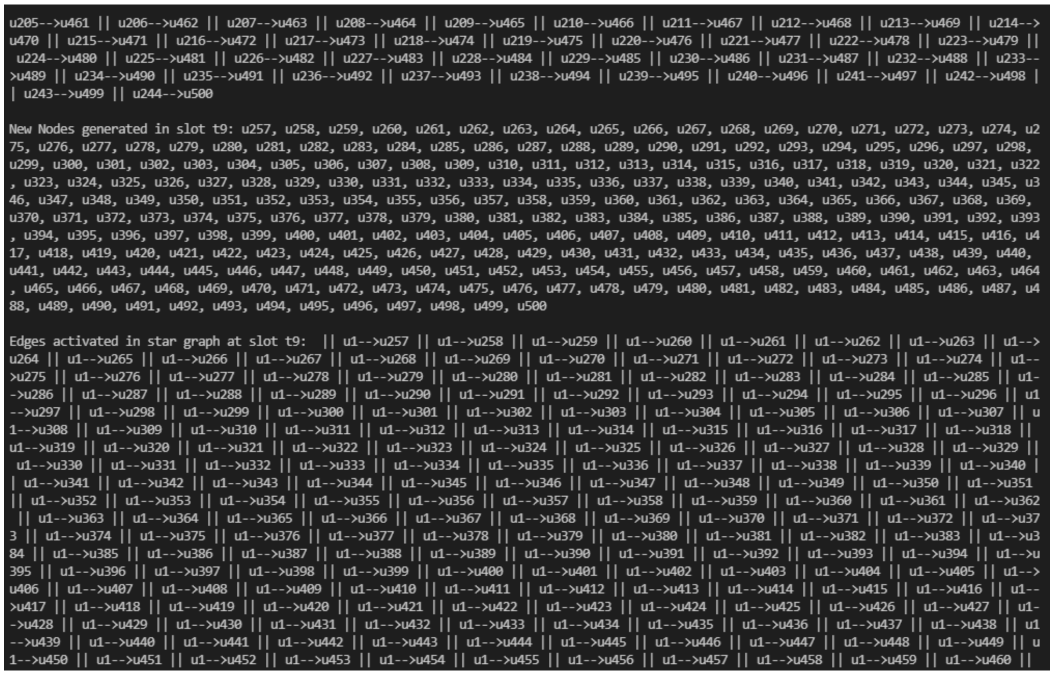
```
Edges activated in star graph at slot t7:  || u1-->u65 || u1-->u66 || u1-->u67 || u1-->u68 || u1-->u69 || u1-->u70 || u1-->u71 || u1-->u72 ||
u1-->u73 || u1-->u74 || u1-->u75 || u1-->u76 || u1-->u77 || u1-->u78 || u1-->u79 || u1-->u80 || u1-->u81 || u1-->u82 || u1-->u83 || u1-->u84 |
| u1-->u85 || u1-->u86 || u1-->u87 || u1-->u88 || u1-->u89 || u1-->u90 || u1-->u91 || u1-->u92 || u1-->u93 || u1-->u94 || u1-->u95 || u1-->u96
 || u1-->u97 || u1-->u98 || u1-->u99 || u1-->u100 || u1-->u101 || u1-->u102 || u1-->u103 || u1-->u104 || u1-->u105 || u1-->u106 || u1-->u107 |
| u1-->u108 || u1-->u109 || u1-->u110 || u1-->u111 || u1-->u112 || u1-->u113 || u1-->u114 || u1-->u115 || u1-->u116 || u1-->u117 || u1-->u118
|| u1-->u119 || u1-->u120 || u1-->u121 || u1-->u122 || u1-->u123 || u1-->u124 || u1-->u125 || u1-->u126 || u1-->u127 || u1-->u128
-------------------------------------------------------------------

Excess edges in star graph at slot t8:  || u2-->u130 || u3-->u131 || u4-->u132 || u5-->u133 || u6-->u134 || u7-->u135 || u8-->u136 || u9-->u13
7 || u10-->u138 || u11-->u139 || u12-->u140 || u13-->u141 || u14-->u142 || u15-->u143 || u16-->u144 || u17-->u145 || u18-->u146 || u19-->u147
|| u20-->u148 || u21-->u149 || u22-->u150 || u23-->u151 || u24-->u152 || u25-->u153 || u26-->u154 || u27-->u155 || u28-->u156 || u29-->u157 ||
 u30-->u158 || u31-->u159 || u32-->u160 || u33-->u161 || u34-->u162 || u35-->u163 || u36-->u164 || u37-->u165 || u38-->u166 || u39-->u167 || u
40-->u168 || u41-->u169 || u42-->u170 || u43-->u171 || u44-->u172 || u45-->u173 || u46-->u174 || u47-->u175 || u48-->u176 || u49-->u177 || u50
-->u178 || u51-->u179 || u52-->u180 || u53-->u181 || u54-->u182 || u55-->u183 || u56-->u184 || u57-->u185 || u58-->u186 || u59-->u187 || u60--
>u188 || u61-->u189 || u62-->u190 || u63-->u191 || u64-->u192 || u65-->u193 || u66-->u194 || u67-->u195 || u68-->u196 || u69-->u197 || u70-->u
198 || u71-->u199 || u72-->u200 || u73-->u201 || u74-->u202 || u75-->u203 || u76-->u204 || u77-->u205 || u78-->u206 || u79-->u207 || u80-->u20
8 || u81-->u209 || u82-->u210 || u83-->u211 || u84-->u212 || u85-->u213 || u86-->u214 || u87-->u215 || u88-->u216 || u89-->u217 || u90-->u218
|| u91-->u219 || u92-->u220 || u93-->u221 || u94-->u222 || u95-->u223 || u96-->u224 || u97-->u225 || u98-->u226 || u99-->u227 || u100-->u228 |
| u101-->u229 || u102-->u230 || u103-->u231 || u104-->u232 || u105-->u233 || u106-->u234 || u107-->u235 || u108-->u236 || u109-->u237 || u110-
->u238 || u111-->u239 || u112-->u240 || u113-->u241 || u114-->u242 || u115-->u243 || u116-->u244 || u117-->u245 || u118-->u246 || u119-->u247
|| u120-->u248 || u121-->u249 || u122-->u250 || u123-->u251 || u124-->u252 || u125-->u253 || u126-->u254 || u127-->u255 || u128-->u256

New Nodes generated in slot t8: u129, u130, u131, u132, u133, u134, u135, u136, u137, u138, u139, u140, u141, u142, u143, u144, u145, u146, u1
47, u148, u149, u150, u151, u152, u153, u154, u155, u156, u157, u158, u159, u160, u161, u162, u163, u164, u165, u166, u167, u168, u169, u170,
u171, u172, u173, u174, u175, u176, u177, u178, u179, u180, u181, u182, u183, u184, u185, u186, u187, u188, u189, u190, u191, u192, u193, u194
, u195, u196, u197, u198, u199, u200, u201, u202, u203, u204, u205, u206, u207, u208, u209, u210, u211, u212, u213, u214, u215, u216, u217, u2
18, u219, u220, u221, u222, u223, u224, u225, u226, u227, u228, u229, u230, u231, u232, u233, u234, u235, u236, u237, u238, u239, u240, u241,
u242, u243, u244, u245, u246, u247, u248, u249, u250, u251, u252, u253, u254, u255, u256

Edges activated in star graph at slot t8:  || u1-->u129 || u1-->u130 || u1-->u131 || u1-->u132 || u1-->u133 || u1-->u134 || u1-->u135 || u1-->
u136 || u1-->u137 || u1-->u138 || u1-->u139 || u1-->u140 || u1-->u141 || u1-->u142 || u1-->u143 || u1-->u144 || u1-->u145 || u1-->u146 || u1--
>u147 || u1-->u148 || u1-->u149 || u1-->u150 || u1-->u151 || u1-->u152 || u1-->u153 || u1-->u154 || u1-->u155 || u1-->u156 || u1-->u157 || u1-
->u158 || u1-->u159 || u1-->u160 || u1-->u161 || u1-->u162 || u1-->u163 || u1-->u164 || u1-->u165 || u1-->u166 || u1-->u167 || u1-->u168 || u1
-->u169 || u1-->u170 || u1-->u171 || u1-->u172 || u1-->u173 || u1-->u174 || u1-->u175 || u1-->u176 || u1-->u177 || u1-->u178 || u1-->u179 || u
1-->u180 || u1-->u181 || u1-->u182 || u1-->u183 || u1-->u184 || u1-->u185 || u1-->u186 || u1-->u187 || u1-->u188 || u1-->u189 || u1-->u190 ||
u1-->u191 || u1-->u192 || u1-->u193 || u1-->u194 || u1-->u195 || u1-->u196 || u1-->u197 || u1-->u198 || u1-->u199 || u1-->u200 || u1-->u201 ||
```

Figure A.15: Excess edges, new nodes generated and edges activated at slots ‘t7’ and ‘t8’



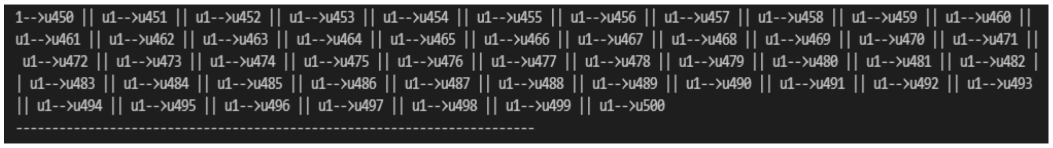
```
Edges activated in star graph at slot t8:  || u1-->u129 || u1-->u130 || u1-->u131 || u1-->u132 || u1-->u133 || u1-->u134 || u1-->u135 || u1-->
u136 || u1-->u137 || u1-->u138 || u1-->u139 || u1-->u140 || u1-->u141 || u1-->u142 || u1-->u143 || u1-->u144 || u1-->u145 || u1-->u146 || u1--
>u147 || u1-->u148 || u1-->u149 || u1-->u150 || u1-->u151 || u1-->u152 || u1-->u153 || u1-->u154 || u1-->u155 || u1-->u156 || u1-->u157 || u1-
->u158 || u1-->u159 || u1-->u160 || u1-->u161 || u1-->u162 || u1-->u163 || u1-->u164 || u1-->u165 || u1-->u166 || u1-->u167 || u1-->u168 || u1
-->u169 || u1-->u170 || u1-->u171 || u1-->u172 || u1-->u173 || u1-->u174 || u1-->u175 || u1-->u176 || u1-->u177 || u1-->u178 || u1-->u179 || u
1-->u180 || u1-->u181 || u1-->u182 || u1-->u183 || u1-->u184 || u1-->u185 || u1-->u186 || u1-->u187 || u1-->u188 || u1-->u189 || u1-->u190 ||
u1-->u191 || u1-->u192 || u1-->u193 || u1-->u194 || u1-->u195 || u1-->u196 || u1-->u197 || u1-->u198 || u1-->u199 || u1-->u200 || u1-->u201 ||
 u1-->u202 || u1-->u203 || u1-->u204 || u1-->u205 || u1-->u206 || u1-->u207 || u1-->u208 || u1-->u209 || u1-->u210 || u1-->u211 || u1-->u212 |
| u1-->u213 || u1-->u214 || u1-->u215 || u1-->u216 || u1-->u217 || u1-->u218 || u1-->u219 || u1-->u220 || u1-->u221 || u1-->u222 || u1-->u223
|| u1-->u224 || u1-->u225 || u1-->u226 || u1-->u227 || u1-->u228 || u1-->u229 || u1-->u230 || u1-->u231 || u1-->u232 || u1-->u233 || u1-->u234
 || u1-->u235 || u1-->u236 || u1-->u237 || u1-->u238 || u1-->u239 || u1-->u240 || u1-->u241 || u1-->u242 || u1-->u243 || u1-->u244 || u1-->u24
5 || u1-->u246 || u1-->u247 || u1-->u248 || u1-->u249 || u1-->u250 || u1-->u251 || u1-->u252 || u1-->u253 || u1-->u254 || u1-->u255 || u1-->u2
56
-------------------------------------------------------------------

Excess edges in star graph at slot t9:  || u2-->u258 || u3-->u259 || u4-->u260 || u5-->u261 || u6-->u262 || u7-->u263 || u8-->u264 || u9-->u26
5 || u10-->u266 || u11-->u267 || u12-->u268 || u13-->u269 || u14-->u270 || u15-->u271 || u16-->u272 || u17-->u273 || u18-->u274 || u19-->u275
|| u20-->u276 || u21-->u277 || u22-->u278 || u23-->u279 || u24-->u280 || u25-->u281 || u26-->u282 || u27-->u283 || u28-->u284 || u29-->u285 ||
 u30-->u286 || u31-->u287 || u32-->u288 || u33-->u289 || u34-->u290 || u35-->u291 || u36-->u292 || u37-->u293 || u38-->u294 || u39-->u295 || u
40-->u296 || u41-->u297 || u42-->u298 || u43-->u299 || u44-->u300 || u45-->u301 || u46-->u302 || u47-->u303 || u48-->u304 || u49-->u305 || u50
-->u306 || u51-->u307 || u52-->u308 || u53-->u309 || u54-->u310 || u55-->u311 || u56-->u312 || u57-->u313 || u58-->u314 || u59-->u315 || u60--
>u316 || u61-->u317 || u62-->u318 || u63-->u319 || u64-->u320 || u65-->u321 || u66-->u322 || u67-->u323 || u68-->u324 || u69-->u325 || u70-->u
326 || u71-->u327 || u72-->u328 || u73-->u329 || u74-->u330 || u75-->u331 || u76-->u332 || u77-->u333 || u78-->u334 || u79-->u335 || u80-->u33
6 || u81-->u337 || u82-->u338 || u83-->u339 || u84-->u340 || u85-->u341 || u86-->u342 || u87-->u343 || u88-->u344 || u89-->u345 || u90-->u346
|| u91-->u347 || u92-->u348 || u93-->u349 || u94-->u350 || u95-->u351 || u96-->u352 || u97-->u353 || u98-->u354 || u99-->u355 || u100-->u356 |
| u101-->u357 || u102-->u358 || u103-->u359 || u104-->u360 || u105-->u361 || u106-->u362 || u107-->u363 || u108-->u364 || u109-->u365 || u110-
->u366 || u111-->u367 || u112-->u368 || u113-->u369 || u114-->u370 || u115-->u371 || u116-->u372 || u117-->u373 || u118-->u374 || u119-->u375
|| u120-->u376 || u121-->u377 || u122-->u378 || u123-->u379 || u124-->u380 || u125-->u381 || u126-->u382 || u127-->u383 || u128-->u384 || u129
-->u385 || u130-->u386 || u131-->u387 || u132-->u388 || u133-->u389 || u134-->u390 || u135-->u391 || u136-->u392 || u137-->u393 || u138-->u394
 || u139-->u395 || u140-->u396 || u141-->u397 || u142-->u398 || u143-->u399 || u144-->u400 || u145-->u401 || u146-->u402 || u147-->u403 || u14
8-->u404 || u149-->u405 || u150-->u406 || u151-->u407 || u152-->u408 || u153-->u409 || u154-->u410 || u155-->u411 || u156-->u412 || u157-->u41
3 || u158-->u414 || u159-->u415 || u160-->u416 || u161-->u417 || u162-->u418 || u163-->u419 || u164-->u420 || u165-->u421 || u166-->u422 || u1
67-->u423 || u168-->u424 || u169-->u425 || u170-->u426 || u171-->u427 || u172-->u428 || u173-->u429 || u174-->u430 || u175-->u431 || u176-->u4
32 || u177-->u433 || u178-->u434 || u179-->u435 || u180-->u436 || u181-->u437 || u182-->u438 || u183-->u439 || u184-->u440 || u185-->u441 || u
186-->u442 || u187-->u443 || u188-->u444 || u189-->u445 || u190-->u446 || u191-->u447 || u192-->u448 || u193-->u449 || u194-->u450 || u195-->u
451 || u196-->u452 || u197-->u453 || u198-->u454 || u199-->u455 || u200-->u456 || u201-->u457 || u202-->u458 || u203-->u459 || u204-->u460 ||
u205-->u461 || u206-->u462 || u207-->u463 || u208-->u464 || u209-->u465 || u210-->u466 || u211-->u467 || u212-->u468 || u213-->u469 || u214-->
```

Figure A.16: Edges activated and excess edges at slots ‘t8’ and ‘t9’ respectively



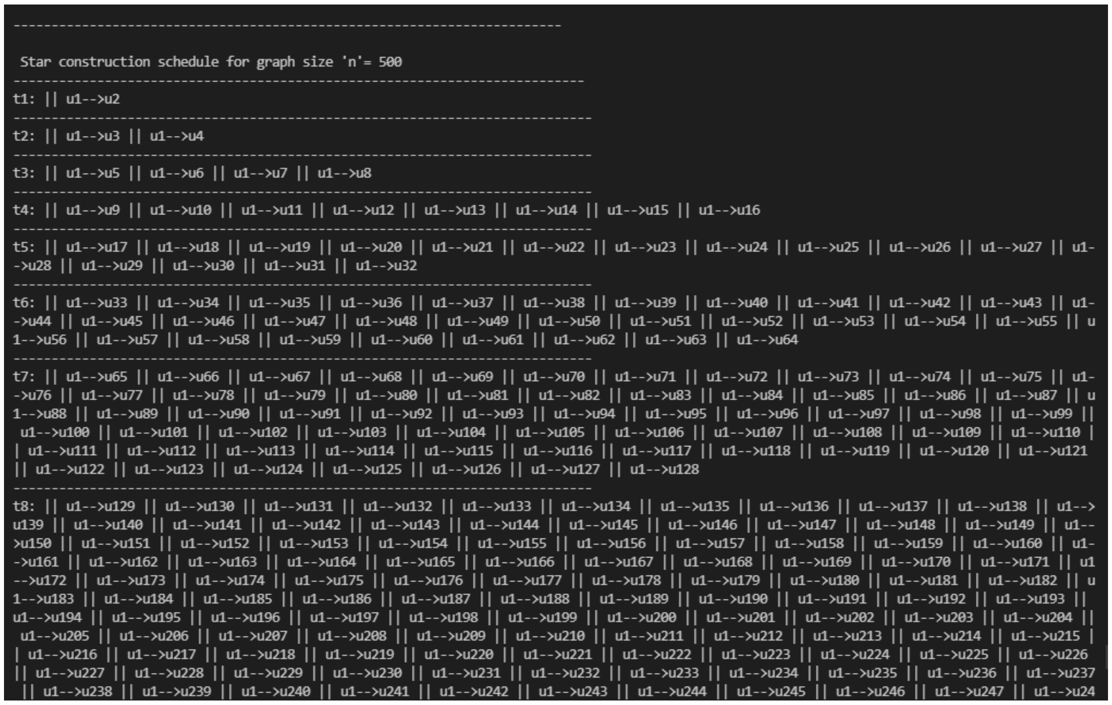
```
u205-->u461 || u206-->u462 || u207-->u463 || u208-->u464 || u209-->u465 || u210-->u466 || u211-->u467 || u212-->u468 || u213-->u469 || u214-->
u470 || u215-->u471 || u216-->u472 || u217-->u473 || u218-->u474 || u219-->u475 || u220-->u476 || u221-->u477 || u222-->u478 || u223-->u479 ||
 u224-->u480 || u225-->u481 || u226-->u482 || u227-->u483 || u228-->u484 || u229-->u485 || u230-->u486 || u231-->u487 || u232-->u488 || u233--
>u489 || u234-->u490 || u235-->u491 || u236-->u492 || u237-->u493 || u238-->u494 || u239-->u495 || u240-->u496 || u241-->u497 || u242-->u498 |
| u243-->u499 || u244-->u500

New Nodes generated in slot t9: u257, u258, u259, u260, u261, u262, u263, u264, u265, u266, u267, u268, u269, u270, u271, u272, u273, u274, u2
75, u276, u277, u278, u279, u280, u281, u282, u283, u284, u285, u286, u287, u288, u289, u290, u291, u292, u293, u294, u295, u296, u297, u298,
u299, u300, u301, u302, u303, u304, u305, u306, u307, u308, u309, u310, u311, u312, u313, u314, u315, u316, u317, u318, u319, u320, u321, u322
, u323, u324, u325, u326, u327, u328, u329, u330, u331, u332, u333, u334, u335, u336, u337, u338, u339, u340, u341, u342, u343, u344, u345, u3
46, u347, u348, u349, u350, u351, u352, u353, u354, u355, u356, u357, u358, u359, u360, u361, u362, u363, u364, u365, u366, u367, u368, u369,
u370, u371, u372, u373, u374, u375, u376, u377, u378, u379, u380, u381, u382, u383, u384, u385, u386, u387, u388, u389, u390, u391, u392, u393
, u394, u395, u396, u397, u398, u399, u400, u401, u402, u403, u404, u405, u406, u407, u408, u409, u410, u411, u412, u413, u414, u415, u416, u4
17, u418, u419, u420, u421, u422, u423, u424, u425, u426, u427, u428, u429, u430, u431, u432, u433, u434, u435, u436, u437, u438, u439, u440,
u441, u442, u443, u444, u445, u446, u447, u448, u449, u450, u451, u452, u453, u454, u455, u456, u457, u458, u459, u460, u461, u462, u463, u464
, u465, u466, u467, u468, u469, u470, u471, u472, u473, u474, u475, u476, u477, u478, u479, u480, u481, u482, u483, u484, u485, u486, u487, u4
88, u489, u490, u491, u492, u493, u494, u495, u496, u497, u498, u499, u500

Edges activated in star graph at slot t9:  || u1-->u257 || u1-->u258 || u1-->u259 || u1-->u260 || u1-->u261 || u1-->u262 || u1-->u263 || u1-->
u264 || u1-->u265 || u1-->u266 || u1-->u267 || u1-->u268 || u1-->u269 || u1-->u270 || u1-->u271 || u1-->u272 || u1-->u273 || u1-->u274 || u1--
>u275 || u1-->u276 || u1-->u277 || u1-->u278 || u1-->u279 || u1-->u280 || u1-->u281 || u1-->u282 || u1-->u283 || u1-->u284 || u1-->u285 || u1-
->u286 || u1-->u287 || u1-->u288 || u1-->u289 || u1-->u290 || u1-->u291 || u1-->u292 || u1-->u293 || u1-->u294 || u1-->u295 || u1-->u296 || u1
-->u297 || u1-->u298 || u1-->u299 || u1-->u300 || u1-->u301 || u1-->u302 || u1-->u303 || u1-->u304 || u1-->u305 || u1-->u306 || u1-->u307 || u
1-->u308 || u1-->u309 || u1-->u310 || u1-->u311 || u1-->u312 || u1-->u313 || u1-->u314 || u1-->u315 || u1-->u316 || u1-->u317 || u1-->u318 ||
u1-->u319 || u1-->u320 || u1-->u321 || u1-->u322 || u1-->u323 || u1-->u324 || u1-->u325 || u1-->u326 || u1-->u327 || u1-->u328 || u1-->u329 ||
 u1-->u330 || u1-->u331 || u1-->u332 || u1-->u333 || u1-->u334 || u1-->u335 || u1-->u336 || u1-->u337 || u1-->u338 || u1-->u339 || u1-->u340 |
| u1-->u341 || u1-->u342 || u1-->u343 || u1-->u344 || u1-->u345 || u1-->u346 || u1-->u347 || u1-->u348 || u1-->u349 || u1-->u350 || u1-->u351
|| u1-->u352 || u1-->u353 || u1-->u354 || u1-->u355 || u1-->u356 || u1-->u357 || u1-->u358 || u1-->u359 || u1-->u360 || u1-->u361 || u1-->u362
 || u1-->u363 || u1-->u364 || u1-->u365 || u1-->u366 || u1-->u367 || u1-->u368 || u1-->u369 || u1-->u370 || u1-->u371 || u1-->u372 || u1-->u37
3 || u1-->u374 || u1-->u375 || u1-->u376 || u1-->u377 || u1-->u378 || u1-->u379 || u1-->u380 || u1-->u381 || u1-->u382 || u1-->u383 || u1-->u3
84 || u1-->u385 || u1-->u386 || u1-->u387 || u1-->u388 || u1-->u389 || u1-->u390 || u1-->u391 || u1-->u392 || u1-->u393 || u1-->u394 || u1-->u
395 || u1-->u396 || u1-->u397 || u1-->u398 || u1-->u399 || u1-->u400 || u1-->u401 || u1-->u402 || u1-->u403 || u1-->u404 || u1-->u405 || u1-->
u406 || u1-->u407 || u1-->u408 || u1-->u409 || u1-->u410 || u1-->u411 || u1-->u412 || u1-->u413 || u1-->u414 || u1-->u415 || u1-->u416 || u1--
>u417 || u1-->u418 || u1-->u419 || u1-->u420 || u1-->u421 || u1-->u422 || u1-->u423 || u1-->u424 || u1-->u425 || u1-->u426 || u1-->u427 || u1-
->u428 || u1-->u429 || u1-->u430 || u1-->u431 || u1-->u432 || u1-->u433 || u1-->u434 || u1-->u435 || u1-->u436 || u1-->u437 || u1-->u438 || u1
-->u439 || u1-->u440 || u1-->u441 || u1-->u442 || u1-->u443 || u1-->u444 || u1-->u445 || u1-->u446 || u1-->u447 || u1-->u448 || u1-->u449 || u
1-->u450 || u1-->u451 || u1-->u452 || u1-->u453 || u1-->u454 || u1-->u455 || u1-->u456 || u1-->u457 || u1-->u458 || u1-->u459 || u1-->u460 ||
```

Figure A.17: Excess edges, new nodes generated and edges activated at slot ‘t9’



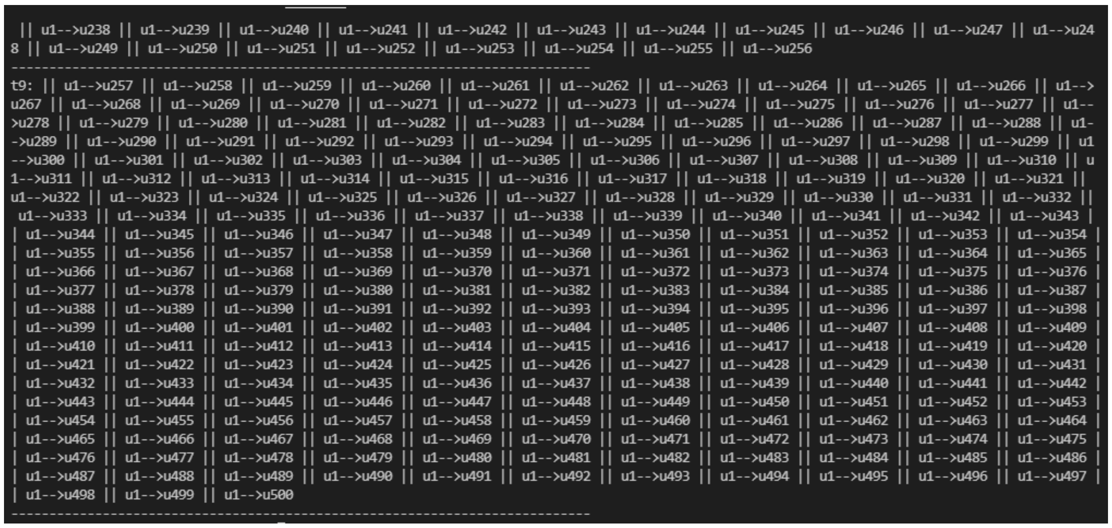
```
1-->u450 || u1-->u451 || u1-->u452 || u1-->u453 || u1-->u454 || u1-->u455 || u1-->u456 || u1-->u457 || u1-->u458 || u1-->u459 || u1-->u460 ||
u1-->u461 || u1-->u462 || u1-->u463 || u1-->u464 || u1-->u465 || u1-->u466 || u1-->u467 || u1-->u468 || u1-->u469 || u1-->u470 || u1-->u471 ||
 u1-->u472 || u1-->u473 || u1-->u474 || u1-->u475 || u1-->u476 || u1-->u477 || u1-->u478 || u1-->u479 || u1-->u480 || u1-->u481 || u1-->u482 |
| u1-->u483 || u1-->u484 || u1-->u485 || u1-->u486 || u1-->u487 || u1-->u488 || u1-->u489 || u1-->u490 || u1-->u491 || u1-->u492 || u1-->u493
|| u1-->u494 || u1-->u495 || u1-->u496 || u1-->u497 || u1-->u498 || u1-->u499 || u1-->u500
-----------------------------------------------------------------------
```

Figure A.18: Edges activated at slot ‘t9’



```
--------------------------------------------------------------------

 Star construction schedule for graph size 'n'= 500
-----------------------------------------------------------------------
t1: || u1-->u2
-----------------------------------------------------------------------
t2: || u1-->u3 || u1-->u4
-----------------------------------------------------------------------
t3: || u1-->u5 || u1-->u6 || u1-->u7 || u1-->u8
-----------------------------------------------------------------------
t4: || u1-->u9 || u1-->u10 || u1-->u11 || u1-->u12 || u1-->u13 || u1-->u14 || u1-->u15 || u1-->u16
-----------------------------------------------------------------------
t5: || u1-->u17 || u1-->u18 || u1-->u19 || u1-->u20 || u1-->u21 || u1-->u22 || u1-->u23 || u1-->u24 || u1-->u25 || u1-->u26 || u1-->u27 || u1-
->u28 || u1-->u29 || u1-->u30 || u1-->u31 || u1-->u32
-----------------------------------------------------------------------
t6: || u1-->u33 || u1-->u34 || u1-->u35 || u1-->u36 || u1-->u37 || u1-->u38 || u1-->u39 || u1-->u40 || u1-->u41 || u1-->u42 || u1-->u43 || u1-
->u44 || u1-->u45 || u1-->u46 || u1-->u47 || u1-->u48 || u1-->u49 || u1-->u50 || u1-->u51 || u1-->u52 || u1-->u53 || u1-->u54 || u1-->u55 || u
1-->u56 || u1-->u57 || u1-->u58 || u1-->u59 || u1-->u60 || u1-->u61 || u1-->u62 || u1-->u63 || u1-->u64
-----------------------------------------------------------------------
t7: || u1-->u65 || u1-->u66 || u1-->u67 || u1-->u68 || u1-->u69 || u1-->u70 || u1-->u71 || u1-->u72 || u1-->u73 || u1-->u74 || u1-->u75 || u1-
->u76 || u1-->u77 || u1-->u78 || u1-->u79 || u1-->u80 || u1-->u81 || u1-->u82 || u1-->u83 || u1-->u84 || u1-->u85 || u1-->u86 || u1-->u87 || u
1-->u88 || u1-->u89 || u1-->u90 || u1-->u91 || u1-->u92 || u1-->u93 || u1-->u94 || u1-->u95 || u1-->u96 || u1-->u97 || u1-->u98 || u1-->u99 ||
 u1-->u100 || u1-->u101 || u1-->u102 || u1-->u103 || u1-->u104 || u1-->u105 || u1-->u106 || u1-->u107 || u1-->u108 || u1-->u109 || u1-->u110 |
| u1-->u111 || u1-->u112 || u1-->u113 || u1-->u114 || u1-->u115 || u1-->u116 || u1-->u117 || u1-->u118 || u1-->u119 || u1-->u120 || u1-->u121
|| u1-->u122 || u1-->u123 || u1-->u124 || u1-->u125 || u1-->u126 || u1-->u127 || u1-->u128
-----------------------------------------------------------------------
t8: || u1-->u129 || u1-->u130 || u1-->u131 || u1-->u132 || u1-->u133 || u1-->u134 || u1-->u135 || u1-->u136 || u1-->u137 || u1-->u138 || u1-->
u139 || u1-->u140 || u1-->u141 || u1-->u142 || u1-->u143 || u1-->u144 || u1-->u145 || u1-->u146 || u1-->u147 || u1-->u148 || u1-->u149 || u1--
>u150 || u1-->u151 || u1-->u152 || u1-->u153 || u1-->u154 || u1-->u155 || u1-->u156 || u1-->u157 || u1-->u158 || u1-->u159 || u1-->u160 || u1-
->u161 || u1-->u162 || u1-->u163 || u1-->u164 || u1-->u165 || u1-->u166 || u1-->u167 || u1-->u168 || u1-->u169 || u1-->u170 || u1-->u171 || u1
-->u172 || u1-->u173 || u1-->u174 || u1-->u175 || u1-->u176 || u1-->u177 || u1-->u178 || u1-->u179 || u1-->u180 || u1-->u181 || u1-->u182 || u
1-->u183 || u1-->u184 || u1-->u185 || u1-->u186 || u1-->u187 || u1-->u188 || u1-->u189 || u1-->u190 || u1-->u191 || u1-->u192 || u1-->u193 ||
u1-->u194 || u1-->u195 || u1-->u196 || u1-->u197 || u1-->u198 || u1-->u199 || u1-->u200 || u1-->u201 || u1-->u202 || u1-->u203 || u1-->u204 ||
 u1-->u205 || u1-->u206 || u1-->u207 || u1-->u208 || u1-->u209 || u1-->u210 || u1-->u211 || u1-->u212 || u1-->u213 || u1-->u214 || u1-->u215 |
| u1-->u216 || u1-->u217 || u1-->u218 || u1-->u219 || u1-->u220 || u1-->u221 || u1-->u222 || u1-->u223 || u1-->u224 || u1-->u225 || u1-->u226
|| u1-->u227 || u1-->u228 || u1-->u229 || u1-->u230 || u1-->u231 || u1-->u232 || u1-->u233 || u1-->u234 || u1-->u235 || u1-->u236 || u1-->u237
 || u1-->u238 || u1-->u239 || u1-->u240 || u1-->u241 || u1-->u242 || u1-->u243 || u1-->u244 || u1-->u245 || u1-->u246 || u1-->u247 || u1-->u24
```

Figure A.19: Star graph construction from slots ‘t1’ to ‘t8’



```
 || u1-->u238 || u1-->u239 || u1-->u240 || u1-->u241 || u1-->u242 || u1-->u243 || u1-->u244 || u1-->u245 || u1-->u246 || u1-->u247 || u1-->u24
8 || u1-->u249 || u1-->u250 || u1-->u251 || u1-->u252 || u1-->u253 || u1-->u254 || u1-->u255 || u1-->u256
-----------------------------------------------------------------------
t9: || u1-->u257 || u1-->u258 || u1-->u259 || u1-->u260 || u1-->u261 || u1-->u262 || u1-->u263 || u1-->u264 || u1-->u265 || u1-->u266 || u1-->
u267 || u1-->u268 || u1-->u269 || u1-->u270 || u1-->u271 || u1-->u272 || u1-->u273 || u1-->u274 || u1-->u275 || u1-->u276 || u1-->u277 || u1--
>u278 || u1-->u279 || u1-->u280 || u1-->u281 || u1-->u282 || u1-->u283 || u1-->u284 || u1-->u285 || u1-->u286 || u1-->u287 || u1-->u288 || u1-
->u289 || u1-->u290 || u1-->u291 || u1-->u292 || u1-->u293 || u1-->u294 || u1-->u295 || u1-->u296 || u1-->u297 || u1-->u298 || u1-->u299 || u1
-->u300 || u1-->u301 || u1-->u302 || u1-->u303 || u1-->u304 || u1-->u305 || u1-->u306 || u1-->u307 || u1-->u308 || u1-->u309 || u1-->u310 || u
1-->u311 || u1-->u312 || u1-->u313 || u1-->u314 || u1-->u315 || u1-->u316 || u1-->u317 || u1-->u318 || u1-->u319 || u1-->u320 || u1-->u321 ||
u1-->u322 || u1-->u323 || u1-->u324 || u1-->u325 || u1-->u326 || u1-->u327 || u1-->u328 || u1-->u329 || u1-->u330 || u1-->u331 || u1-->u332 ||
 u1-->u333 || u1-->u334 || u1-->u335 || u1-->u336 || u1-->u337 || u1-->u338 || u1-->u339 || u1-->u340 || u1-->u341 || u1-->u342 || u1-->u343 |
| u1-->u344 || u1-->u345 || u1-->u346 || u1-->u347 || u1-->u348 || u1-->u349 || u1-->u350 || u1-->u351 || u1-->u352 || u1-->u353 || u1-->u354 |
| u1-->u355 || u1-->u356 || u1-->u357 || u1-->u358 || u1-->u359 || u1-->u360 || u1-->u361 || u1-->u362 || u1-->u363 || u1-->u364 || u1-->u365 |
| u1-->u366 || u1-->u367 || u1-->u368 || u1-->u369 || u1-->u370 || u1-->u371 || u1-->u372 || u1-->u373 || u1-->u374 || u1-->u375 || u1-->u376 |
| u1-->u377 || u1-->u378 || u1-->u379 || u1-->u380 || u1-->u381 || u1-->u382 || u1-->u383 || u1-->u384 || u1-->u385 || u1-->u386 || u1-->u387 |
| u1-->u388 || u1-->u389 || u1-->u390 || u1-->u391 || u1-->u392 || u1-->u393 || u1-->u394 || u1-->u395 || u1-->u396 || u1-->u397 || u1-->u398 |
| u1-->u399 || u1-->u400 || u1-->u401 || u1-->u402 || u1-->u403 || u1-->u404 || u1-->u405 || u1-->u406 || u1-->u407 || u1-->u408 || u1-->u409 |
| u1-->u410 || u1-->u411 || u1-->u412 || u1-->u413 || u1-->u414 || u1-->u415 || u1-->u416 || u1-->u417 || u1-->u418 || u1-->u419 || u1-->u420 |
| u1-->u421 || u1-->u422 || u1-->u423 || u1-->u424 || u1-->u425 || u1-->u426 || u1-->u427 || u1-->u428 || u1-->u429 || u1-->u430 || u1-->u431 |
| u1-->u432 || u1-->u433 || u1-->u434 || u1-->u435 || u1-->u436 || u1-->u437 || u1-->u438 || u1-->u439 || u1-->u440 || u1-->u441 || u1-->u442 |
| u1-->u443 || u1-->u444 || u1-->u445 || u1-->u446 || u1-->u447 || u1-->u448 || u1-->u449 || u1-->u450 || u1-->u451 || u1-->u452 || u1-->u453 |
| u1-->u454 || u1-->u455 || u1-->u456 || u1-->u457 || u1-->u458 || u1-->u459 || u1-->u460 || u1-->u461 || u1-->u462 || u1-->u463 || u1-->u464 |
| u1-->u465 || u1-->u466 || u1-->u467 || u1-->u468 || u1-->u469 || u1-->u470 || u1-->u471 || u1-->u472 || u1-->u473 || u1-->u474 || u1-->u475 |
| u1-->u476 || u1-->u477 || u1-->u478 || u1-->u479 || u1-->u480 || u1-->u481 || u1-->u482 || u1-->u483 || u1-->u484 || u1-->u485 || u1-->u486 |
| u1-->u487 || u1-->u488 || u1-->u489 || u1-->u490 || u1-->u491 || u1-->u492 || u1-->u493 || u1-->u494 || u1-->u495 || u1-->u496 || u1-->u497 |
| u1-->u498 || u1-->u499 || u1-->u500
-----------------------------------------------------------------------
```

Figure A.20: Star graph construction from slots ‘t8’ and ‘t9’

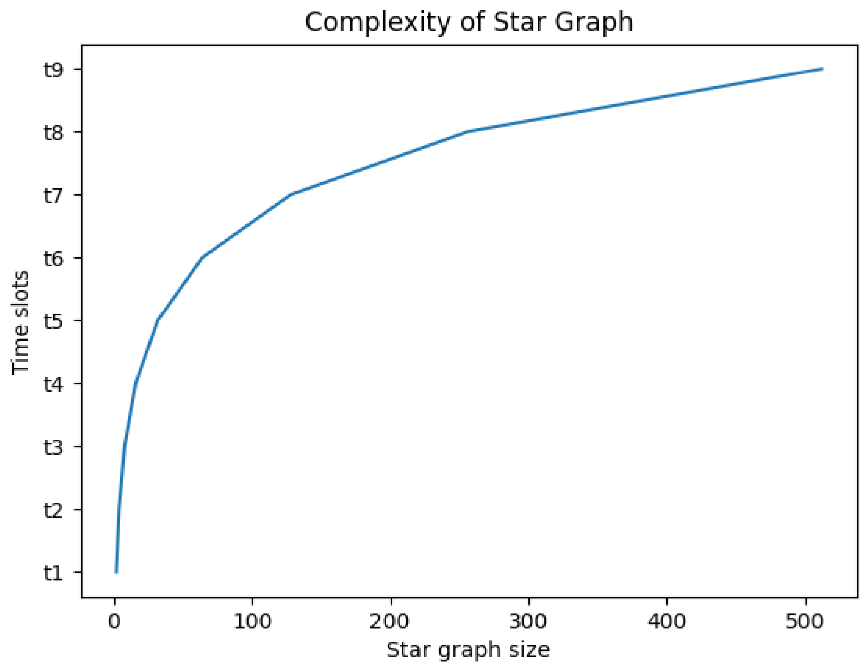


Figure A.21: Complexity of the Star graph construction

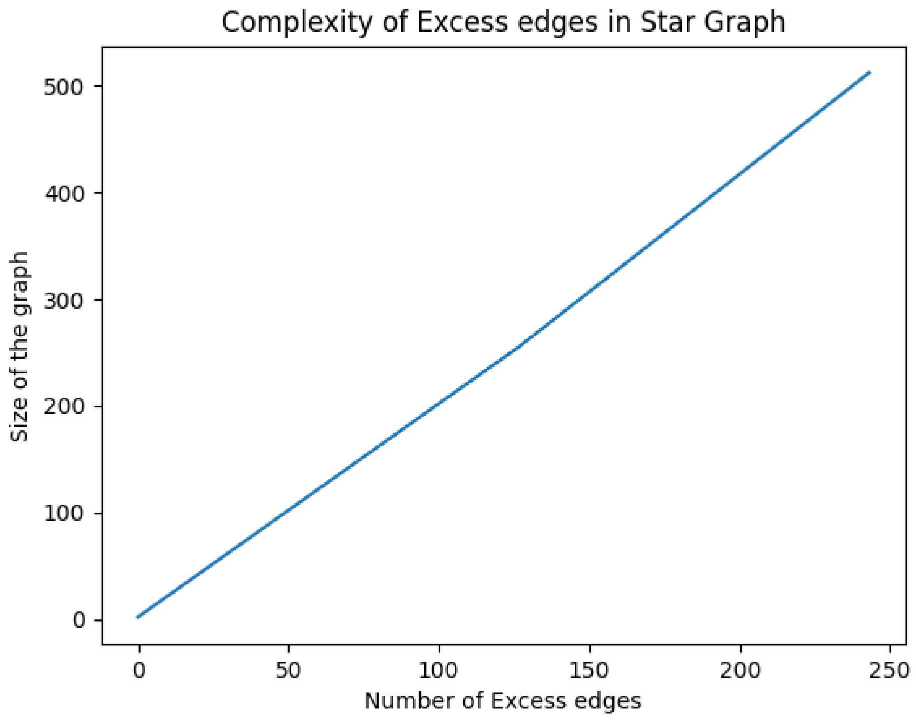
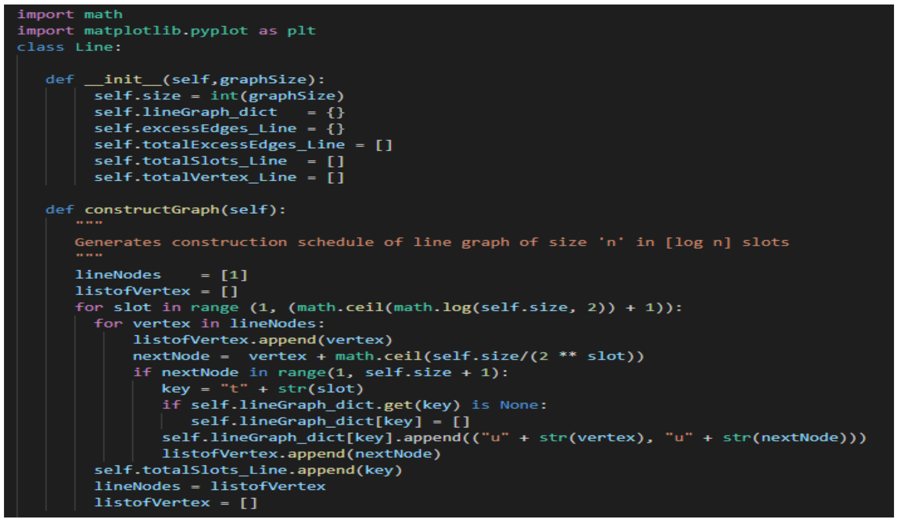


Figure A.22: Complexity of excess edges in the Star graph construction

Appendix B

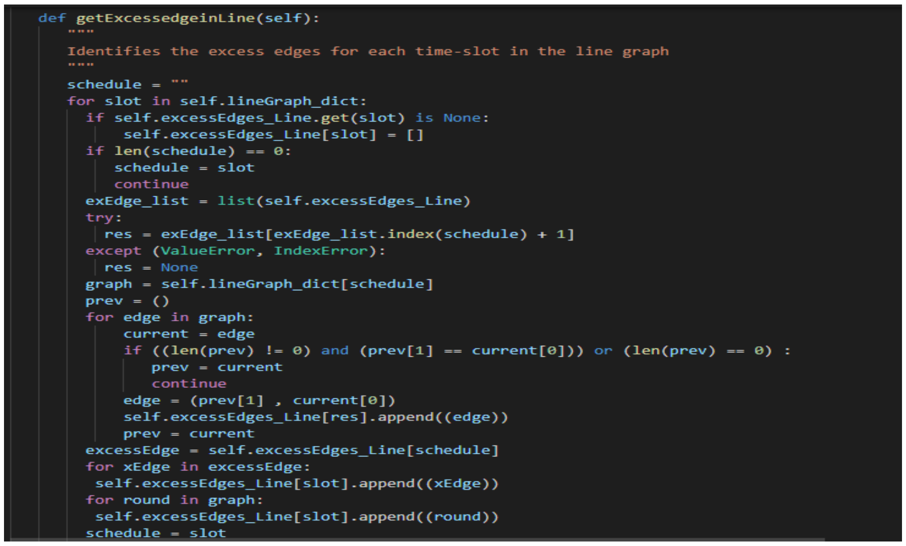
Code for Key Methods

B.1 Line Graph



```python
import math
import matplotlib.pyplot as plt
class Line:

   def __init__(self,graphSize):
        self.size = int(graphSize)
        self.lineGraph_dict    = {}
        self.excessEdges_Line = {}
        self.totalExcessEdges_Line = []
        self.totalSlots_Line  = []
        self.totalVertex_Line = []

   def constructGraph(self):
      """
      Generates construction schedule of line graph of size 'n' in [log n] slots
      """
      lineNodes     = [1]
      listofVertex = []
      for slot in range (1, (math.ceil(math.log(self.size, 2)) + 1)):
        for vertex in lineNodes:
            listofVertex.append(vertex)
            nextNode =  vertex + math.ceil(self.size/(2 ** slot))
            if nextNode in range(1, self.size + 1):
               key = "t" + str(slot)
               if self.lineGraph_dict.get(key) is None:
                  self.lineGraph_dict[key] = []
               self.lineGraph_dict[key].append(("u" + str(vertex), "u" + str(nextNode)))
               listofVertex.append(nextNode)
        self.totalSlots_Line.append(key)
        lineNodes = listofVertex
        listofVertex = []
```

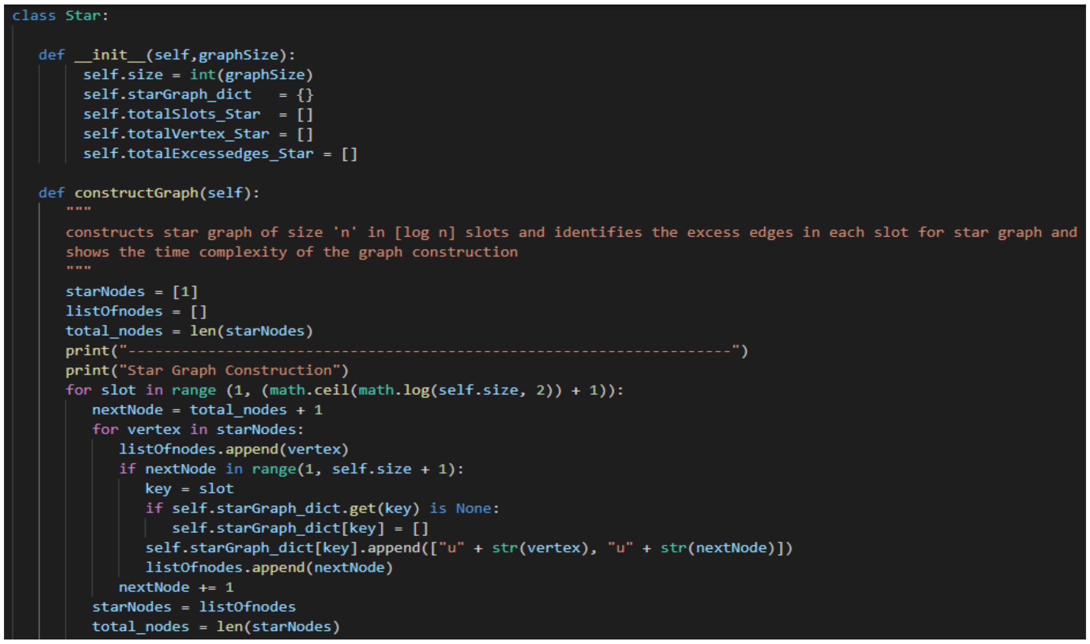
Figure B.1: Code for implementation of the Line graph



```python
    def getExcessedgeinLine(self):
       """
       Identifies the excess edges for each time-slot in the line graph
       """
       schedule = ""
       for slot in self.lineGraph_dict:
         if self.excessEdges_Line.get(slot) is None:
             self.excessEdges_Line[slot] = []
         if len(schedule) == 0:
            schedule = slot
            continue
         exEdge_list = list(self.excessEdges_Line)
         try:
           res = exEdge_list[exEdge_list.index(schedule) + 1]
         except (ValueError, IndexError):
           res = None
         graph = self.lineGraph_dict[schedule]
         prev = ()
         for edge in graph:
             current = edge
             if ((len(prev) != 0) and (prev[1] == current[0])) or (len(prev) == 0) :
                prev = current
                continue
             edge = (prev[1] , current[0])
             self.excessEdges_Line[res].append((edge))
             prev = current
         excessEdge = self.excessEdges_Line[schedule]
         for xEdge in excessEdge:
          self.excessEdges_Line[slot].append((xEdge))
         for round in graph:
          self.excessEdges_Line[slot].append((round))
         schedule = slot
```

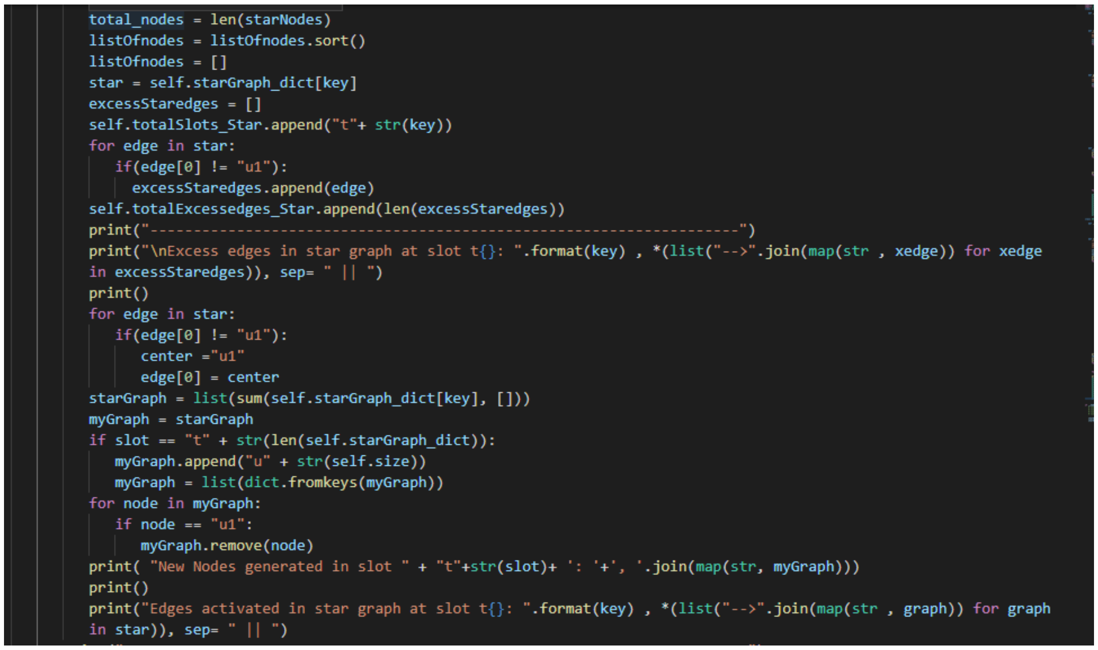
Figure B.2: Code for the excess edges identification in the Line graph

B.2 Star Graph



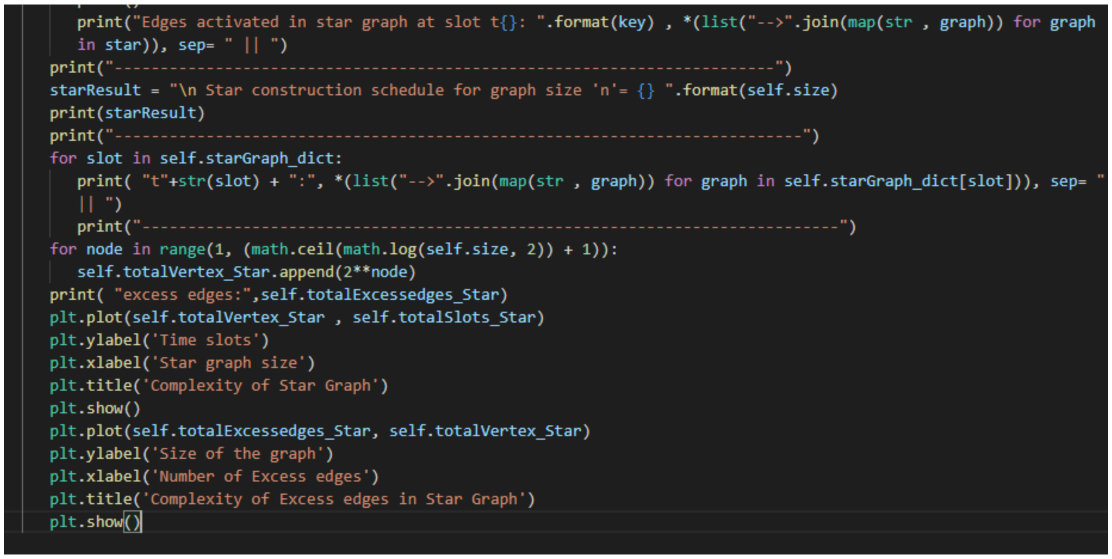
```
class Star:

   def __init__(self,graphSize):
        self.size = int(graphSize)
        self.starGraph_dict   = {}
        self.totalSlots_Star  = []
        self.totalVertex_Star = []
        self.totalExcessedges_Star = []

   def constructGraph(self):
      """
      constructs star graph of size 'n' in [log n] slots and identifies the excess edges in each slot for star graph and
      shows the time complexity of the graph construction
      """
      starNodes = [1]
      listOfnodes = []
      total_nodes = len(starNodes)
      print("------------------------------------------------------------------")
      print("Star Graph Construction")
      for slot in range (1, (math.ceil(math.log(self.size, 2)) + 1)):
         nextNode = total_nodes + 1
         for vertex in starNodes:
            listOfnodes.append(vertex)
            if nextNode in range(1, self.size + 1):
               key = slot
               if self.starGraph_dict.get(key) is None:
                  self.starGraph_dict[key] = []
               self.starGraph_dict[key].append(["u" + str(vertex), "u" + str(nextNode)])
               listOfnodes.append(nextNode)
            nextNode += 1
         starNodes = listOfnodes
         total_nodes = len(starNodes)
```

Figure B.3: Code for the Star Graph construction



```
         total_nodes = len(starNodes)
         listOfnodes = listOfnodes.sort()
         listOfnodes = []
         star = self.starGraph_dict[key]
         excessStaredges = []
         self.totalSlots_Star.append("t"+ str(key))
         for edge in star:
            if(edge[0] != "u1"):
              excessStaredges.append(edge)
         self.totalExcessedges_Star.append(len(excessStaredges))
         print("------------------------------------------------------------------")
         print("\nExcess edges in star graph at slot t{}: ".format(key) , *(list("-->".join(map(str , xedge)) for xedge
         in excessStaredges)), sep= " || ")
         print()
         for edge in star:
            if(edge[0] != "u1"):
               center ="u1"
               edge[0] = center
         starGraph = list(sum(self.starGraph_dict[key], []))
         myGraph = starGraph
         if slot == "t" + str(len(self.starGraph_dict)):
            myGraph.append("u" + str(self.size))
            myGraph = list(dict.fromkeys(myGraph))
         for node in myGraph:
            if node == "u1":
               myGraph.remove(node)
         print( "New Nodes generated in slot " + "t"+str(slot)+ ': '+', '.join(map(str, myGraph)))
         print()
         print("Edges activated in star graph at slot t{}: ".format(key) , *(list("-->".join(map(str , graph)) for graph
         in star)), sep= " || ")
```

Figure B.4: Code for the Star Graph construction (continuation to Figure B.3)



```
        print("Edges activated in star graph at slot t{}: ".format(key) , *(list("-->".join(map(str , graph)) for graph
        in star)), sep= " || ")
    print("------------------------------------------------------------------------")
    starResult = "\n Star construction schedule for graph size 'n'= {} ".format(self.size)
    print(starResult)
    print("---------------------------------------------------------------------------")
    for slot in self.starGraph_dict:
        print( "t"+str(slot) + ":", *(list("-->".join(map(str , graph)) for graph in self.starGraph_dict[slot])), sep= "
        || ")
        print("----------------------------------------------------------------------------")
    for node in range(1, (math.ceil(math.log(self.size, 2)) + 1)):
        self.totalVertex_Star.append(2**node)
    print( "excess edges:",self.totalExcessedges_Star)
    plt.plot(self.totalVertex_Star , self.totalSlots_Star)
    plt.ylabel('Time slots')
    plt.xlabel('Star graph size')
    plt.title('Complexity of Star Graph')
    plt.show()
    plt.plot(self.totalExcessedges_Star, self.totalVertex_Star)
    plt.ylabel('Size of the graph')
    plt.xlabel('Number of Excess edges')
    plt.title('Complexity of Excess edges in Star Graph')
    plt.show()
```

Figure B.5: Code for the Star Graph construction (continuation Figure B.4)

Appendix C

Original design document

I have attached my original design document below for the project to refer to the changes that I made.

University of Liverpool - COMP702 - MSc project

Algorithms for Dynamic Networks

Specification and Proposed Design

Saiteja Pendyala - 201538148

Supervisor - Othon Michail

SUMMARY OF PROPOSAL

The purpose of this project is to investigate an abstraction of networked systems that can expand from a single entity into well-defined global networks and structures. In graph theory, the study of the structure of graphs has been based on the idea that they are static. However, the graphs that emerge in many applications evolve over time, making them dynamic.

Graph representations have become more important in nearly every scientific subject, including mathematics, biology, as well as computer technology. Graphs, unlike other data formats, allow for the modelling of relationships between entities [2]. Graphs, in combination with the ever-increasing amount of data available, open up a wide range of modelling options for theoretical and real-world situations. The modelling capabilities of graphs, on the other hand, have not been adequately used. One reason for this is that there is no easy-to-understand summary of graph kinds or a comparison of their modelling capabilities.

Dynamic graphs are a valuable tool for illustrating a wide range of topics that vary over time. Modelling the evolution of relationships and communities in a social network or tracking user activity in an enterprise computer network are two examples[6].

A growing graph is a distinct network system that is highly dynamic. As nodes, such networks have entities that can self-replicate and so grow the network's size. As a result, the difficulty of constructing a target network G from a single entity arises. To appropriately model this, we assume that each node 'u' can only generate one node 'v' per round and that each formed node 'v' can only activate edges with other nodes at the time of its birth, as long as these nodes are within a minimal distance 'd' of 'v', The most intriguing condition is when the distance is 'd' = 2. Any time slot permits the edge deletions[1]. The objective is to study graph models with particularly efficient construction schedules and to show that a target graph G be produced in the model of growing graphs in at most k time slots and with at most l extra edges given a network G of n nodes. Proposed results include that Unless P=NP, the ideal number of time slots to construct an input target graph with zero waste (i.e., no edge deletions accepted) is difficult to estimate within $n^{1-\varepsilon}$, $for\ any\ \varepsilon > 0$. The purpose of the zero-waste construction schedule problem is to determine whether a target graph G allows construction with k slots and l=0 excess edges. This achieves a natural balance between the speed with which a target network can be formed and the efficiency with which it can be generated.

My contribution involves experimental evaluation of the proposed approach that computes a target graph G = (V, E), where V is a finite set of vertices or nodes and E is a set of Edges, with |V | = 2 having a construction schedule with k = log n slots and

l = 0 extra edges, and then compare my observations to the study's theoretical bounds in [1].

DESIGN

The plan of the experimental evaluation consists of two parts:

1) **Algorithm analysis -** To understand the performance of the algorithms and their computational complexity.
2) **Evaluation -** The process of evaluating an algorithm's property or properties is known as algorithm evaluation.

Algorithm analysis is a crucial aspect of computational complexity theory, as it provides a theoretical estimate of how much time and resources an algorithm will need to solve a given issue. The determination of the amount of time and space resources necessary to execute an algorithm is known as algorithm analysis [3]. Typically, an algorithm's efficiency or running time is expressed as a function linking the input length to the number of steps (time complexity) or the amount of memory (space complexity).

Asymptotic notations are mathematical notations that are used to express the run-time of an algorithm when the input tends toward a specific value or a limiting value. Asymptotic notations are classified into three types: Theta notation, the Omega notation, and the Big-O notation. These are used to calculate the algorithm's runtime complexity.[7]

The first stage of the algorithm experimentation involves studying and analysing the observations, proposed algorithms, and proofs of dynamic graph templates (Line, star, and tree) with their construction schedules, time complexities, and then testing and experimenting the graph models, which is the most significant part of the project.

Firstly, I'll completely understand and examine the proposed centralised and distributed graph models, as well as their theorems, before analysing the proposed algorithms and their theoretical bounds.

Secondly, I'll begin experimenting with whether the proposed algorithm computes a target graph G = (V, E), where V is a finite set of vertices or nodes, and E is a set of Edges, |V | = 2^{δ} has a construction schedule with k = log n slots and l = 0 excess edges. If it does, the algorithm also outputs such a construction schedule, and then compare my findings to the theoretical bounds stated in the study.

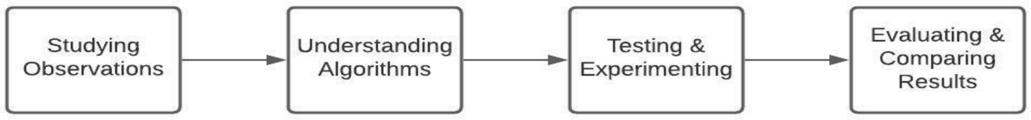


Figure 1: Flow diagram for the design of the project

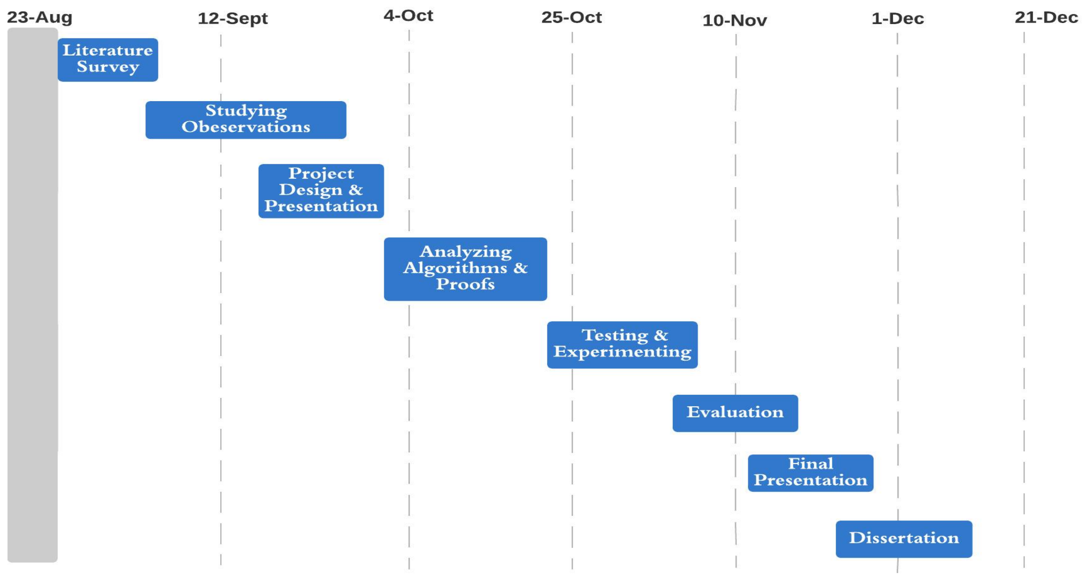
REQUIRED PYTHON LIBRARIES

1) **Numpy** - This library aids in the execution of logical and mathematical operations on multidimensional arrays [4].

2) **Matplotlib -** matplotlib is a Python library that allows you to create a variety of graphs and interactive visualisations [5].

PLAN

Project Plan



My project's Gantt chart looks like this. All of the events are labelled with the appropriate dates. This will help in good time management and progress tracking.

References

[1] George B. Mertzios, Othon Michail, George Skretas, Paul G. Spirakis, And Michail Theofilatos. *The Complexity of Growing a Graph*. arXiv: 2107.14126 [cs.DS] (2021) (available at https://arxiv.org/abs/2107.14126).

[2] Josephine M. Thomas, Silvia Beddar-Wiesing, Alice Moallemy-Oureh, Rüdiger Nather. *Graph type expressivity and transformations*. arXiv: 2109.10708 [cs.DM] (2021) (available at https://arxiv.org/abs/2109.10708).

[3] Mohammad Mohtashim. *Tutorials Point*, "Design and Analysis of Algorithm"[Web]. (available at https://www.tutorialspoint.com/design_and_analysis_of_algorithms/analysis_of_algorithms.htm) [Accessed 13 September 2021].

[4] NumPy: the absolute basics for beginners, *Numpy v1.21 Manual* [Web]. (available at https://numpy.org/doc/stable/user/absolute_beginners.html) [Accessed 17 September 2021].

[5] Matplotlib: pyplot tutorial - *matplotlib 3.4.3 documentation* [Web]. (available at https://matplotlib.org/stable/tutorials/introductory/pyplot.html)[Accessed 17 September 2021].

[6] ACM. Association for Computing Machinery：Aaron Scott Pope, Daniel R. Tauritz, Chris Rawlings. *Automated design of random dynamic graph models*. (retrieved from https://dl.acm.org/doi/10.1145/3319619.3326859)

[7] Intersog: how to calculate complexity of an algorithm -Intersog [Web].(available at https://intersog.com/blog/algorithm-complexity-estimation-a-bit-of-theory-and-why-it-is-necessary-to-know/) [Accessed 17 September 2021].