Graph Theory and GitHub

Angelica Tellez and Prof. Johann Thiel

New York City College of Technology, Honors Scholars Program

Abstract

This project explores the basics of graph theory and its application in GitHub, a widely-used version control management system and collaboration platform. Graph theory finds extensive applications in diverse fields such as computer networks, communication systems, social networks, hardware design, and software. GitHub utilizes directed graphs to manage and track the various versions of a project, making a foundational understanding of graph theory essential for optimizing its use. By understanding these concepts, individuals can seamlessly collaborate on complex projects using GitHub's tools, enhancing their adaptability and value in an increasingly technology-driven workforce.

Introduction

In the workforce, collaboration with peers is crucial for solving complex problems. GitHub is a widely known collaborative management system that fosters individual creativity until it is ready to be shared. Understanding where you stand on the "work tree" is essential for knowing when to commit your changes or merge them into the main branch as the final copy. An understanding of graph theory provides the tools needed to navigate the work tree and work seamlessly with others in your field. In this project, we used basic GitHub commands to build a work tree that we then analyzed using graph theory principles.

GitHub

We began by creating a new repository in GitHub and adding a single text file. GitHub supports various other files, we simply used text file for this example. This file initially had the phrase: *"This is the original version."* This version became the root of our work tree. After deciding on an improved version of the text file, we updated the phrase to: *"This is the better version."* This change was reflected to other collaborators when the edit was named and committed to the work tree. When a change is committed to the shared copy, a new node is added to the tree, extending it (as shown in Figure 1).

Another branch of the tree was created to test a new feature in the text file. In our case, this branch was named Version-1 and stemmed from the "first commit" node. At this stage, two branches existed with copies of the same text file. In the origin branch, changes continued to be committed to progress the work. However, the final version was created and decided upon in the **Version-1** branch. The **Version-1** branch was then merged into the origin branch, overriding the previous copy. If a change in a new branch is not approved, the origin branch remains unaffected (as illustrated in Figure 2, the **feature** branch).

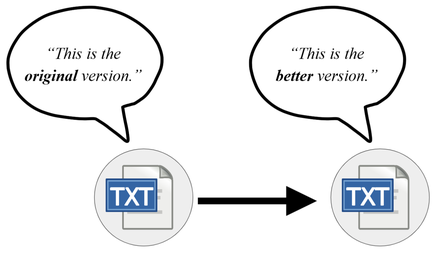

Figure 1: First Commit Example

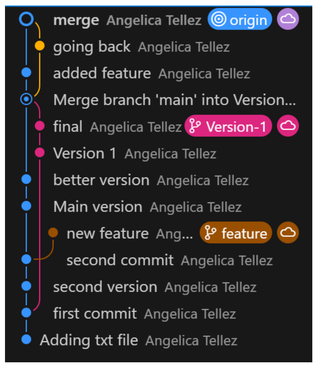

Figure 2: GitHub Work Tree

Graph Theory

In graph theory, GitHub's work tree can be visualized as a directed graph, where each commit represents a vertex, and each edge indicates the progression from one commit to the next. These commits capture significant changes in the project, creating a traceable history within the work tree. The edges reflect a one-way relationship between vertices, always pointing toward the most recent version of the collaborative work. This structure ensures that collaborators can easily track the evolution of the project. Additionally, if a team member wants to review their peers' progress, they can follow the directed path from their current version to the desired commit, simplifying navigation and fostering collaboration on complex problems.

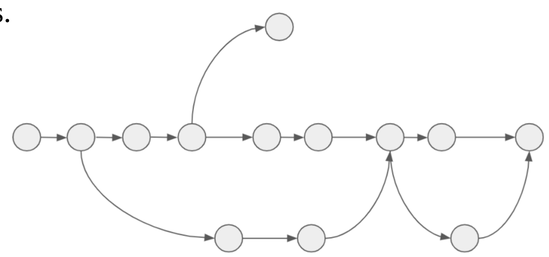

Figure 3: Directed Graph Example

Importance

GitHub's implementation of graph theory allows developers to backtrack through commits, identify when and where specific changes occurred, and recover previous versions if necessary. The branching system helps individuals or teams isolate their work without disrupting the main project, and the ability to merge changes ensures that contributions are systematically integrated. By utilizing graph theory, GitHub enables one to understand their position within the work tree, facilitating seamless collaboration, conflict resolution, and the ability to track the development process across multiple contributors.

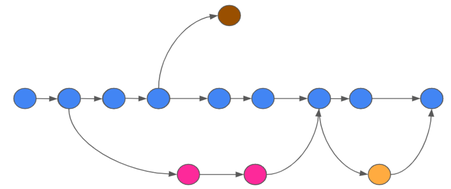

Figure 4: GitHub Directed Graph

Conclusion

In this project, we explored how GitHub's work tree applies graph theory to enable effective collaboration on complex problems. By visualizing commits as vertices and their relationships as directed edges, we demonstrated how GitHub systematically tracks project development. This structure, combined with an understanding of graph theory, equips professionals with the tools to manage projects efficiently, resolve conflicts, and maintain transparency in team-based environments. Future steps could include advanced GitHub features like pull requests, which provide a structured way to propose and review changes, and forking repositories, which allows for experimentation or contributions to external projects. These skills are essential in industries where version control and collaborative development are critical to thrive.

Sources

"Hello World." GitHub Docs, docs.github.com/en/get-started/start-your-journey/hello-world.

Rosen, Kenneth H. *Discrete Mathematics and Its Applications*. 7th ed., McGraw-Hill, 2013.

Wilson, Robin J. *Introduction to Graph Theory*. 4th ed., Prentice Hall.