Course Prerequisites: A Demonstration of Directed Graphs

Dylan Sullins

16 Nov 2024

Table of Contents

Introduction

This project is a study of directed graphs through a simplified representation of courses from the Computer Science department at the University of Missouri Kansas City (UMKC) and their prerequisites. The purpose of this project is to establish a deeper understanding of the implementation of directed graphs. The scope is limited to computer science courses at UMKC, plus a few external courses from other departments as required. Implementation is rudimentary, with courses displayed in a grid of nodes connected by edges.

Objectives

The goal of this documentation is to serve as a reflection on the planning, framing, and implementation of the directed graph of courses. The main objective is to model course dependencies using a directed graph.

Methodology

The software implementation of this project was created in C++ using the Qt graphical library to display the graph. All data was obtained from the UMKC Course Catalog.

Concepts and Implementation

What is a directed graph? To answer this question, we first must understand the concept of a graph. A graph G is a collected of points known as vertices (singular: vertex), represented by V, connected by edges, represented by E. A directed graph is a graph where the edges are directional. In other words, each edges points from one vertex to another. In a directed graph, if edge e_1 points from vertex v_1 to vertex v_2, this only shows a relationship from $v_1 \rightarrow v_2$ and does not imply $v_2 \rightarrow v_1$.

How can a directed graph of courses and their prerequisites be modeled? Physically, this graph can be modeled by plotting a series of nodes that represent each course, connected by a series of lines representing prerequisite relationships. On paper, this is a simple process that quickly becomes tedious and time consuming with a large amount of courses. A more efficient method of modeling the directed graph is to represent it through a graphical user interface (GUI).

For this project, C++ was chosen as the language, and the Qt library was selected for GUI design and development. C++ is especially relevant in this project, as earlier UMKC computer science courses mainly use C++ in their instruction. Qt is a cross platform library for GUI design and development using C++, and was selected for its relative ease of use.

Initially, the directed graph was created as a series of user-defined objects known as Courses, with each Course containing a vector of Course objects representing their prerequisites. A data file was created using information gathered from UMKC's course catalog, and formatted to ensure ease of text-parsing. Fake courses were created to represent testing requirements, departmental consent, and junior status requirements, as it makes sense to count these as prerequisites of sorts. The information from the catalog file was then read into the program which created a directed graph of courses and their corresponding prerequisites.

Once this was complete, a quick GUI was developed to create a graphical node for each Course object and an arrow for each edge connecting prerequisite nodes to their corresponding courses. The course nodes are then displayed on a grid with black arrows between them. Upon hovering over a course node, an informational popup appears with the course information and the edges connecting its immediate prerequisites are highlighted.

The courses with the minimum and maximum prerequisites are displayed in the command line interface console. For the Catalog.dat that is included in the source code, the course with the minimum prerequisites was found to be COMP-SCI 424 Software Methods and Tools with no prerequisites listed, and the course with the maximum prerequisites was found to be COMP-SCI 421 Foundations of Data Networks with 12 prerequisites. In order to count all of the prerequisites, a recursive function was used that counts all of a course's immediate prerequisites, and all of those courses' prerequisites, ignoring duplicates.

One notable restriction, the directed graph does not model logical OR relationships. For example, in cases where MATH 110 OR MATH 120 are required, the software counts them as both being required for the sake of simplicity. This restriction implies that the theoretical prerequisite count generated may be slightly higher than the actual prerequisite count. However, for the purposes of this representation, this does not largely affect the results.

Conclusion

While the directed graph may be busy, it is a clear and simple representation of courses and their prerequisites. While one may assume that lower courses would be most likely to have the minimum amount of prerequisites, the actual minimum was a higher level course with zero prerequisites. It is possible that there are additional prerequisites not listed on the catalog, but for the purposes and scope of this project, those are not considered. As expected, there were no circular dependencies in the section of the course catalog analyzed.

Examples

Figure 1: Directed Graph

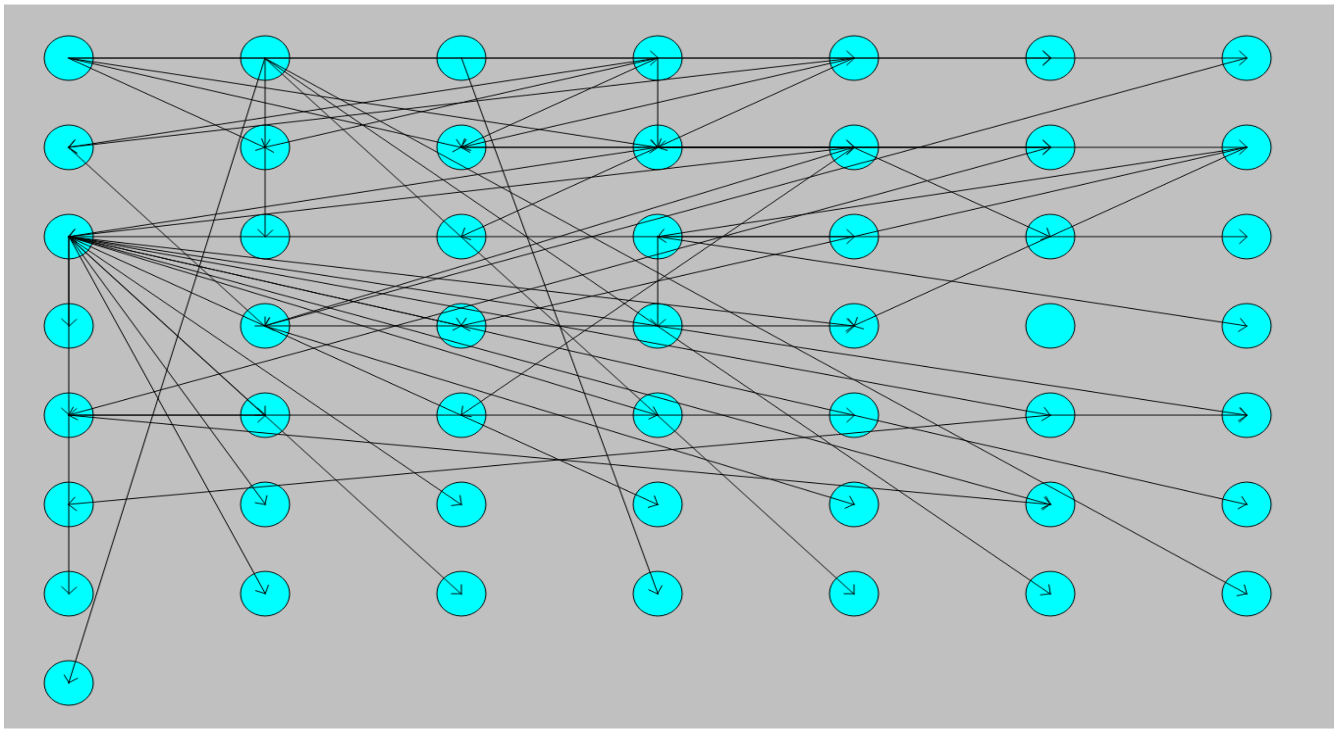


Figure 2: Minimum Prerequisites Hover

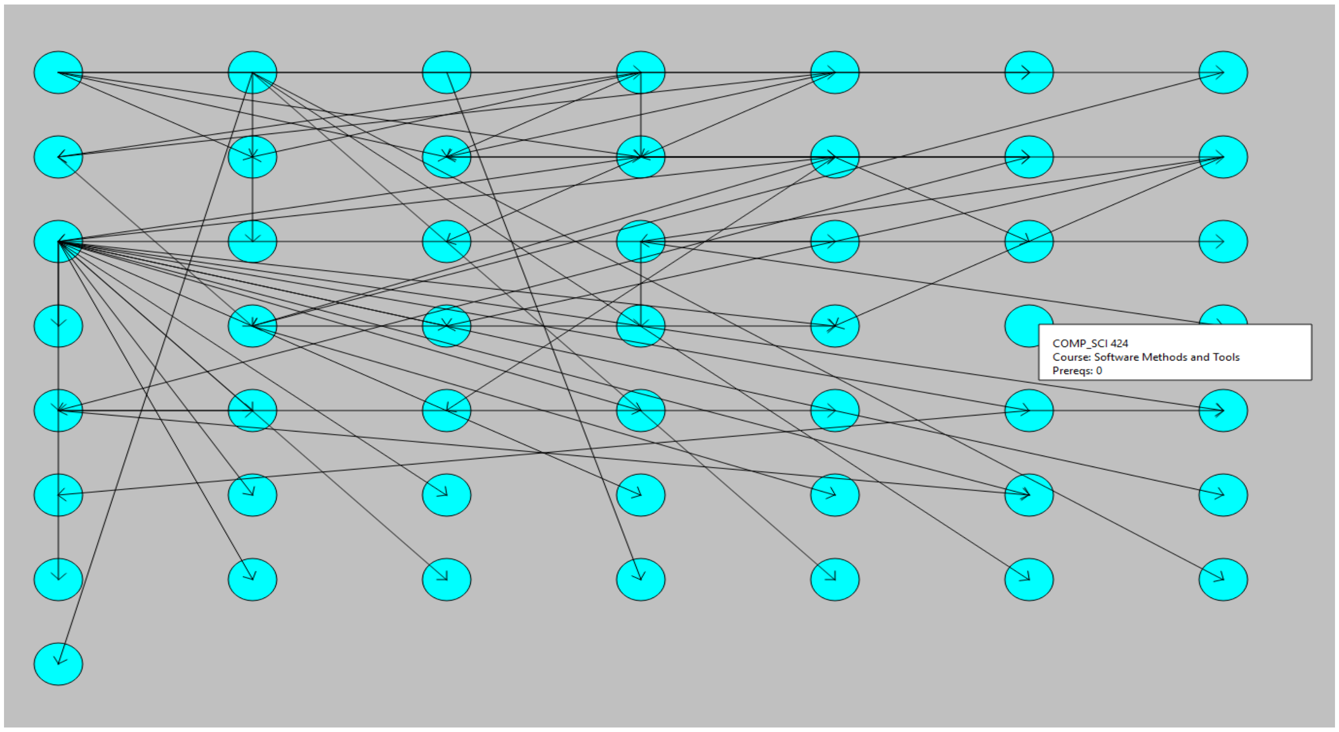
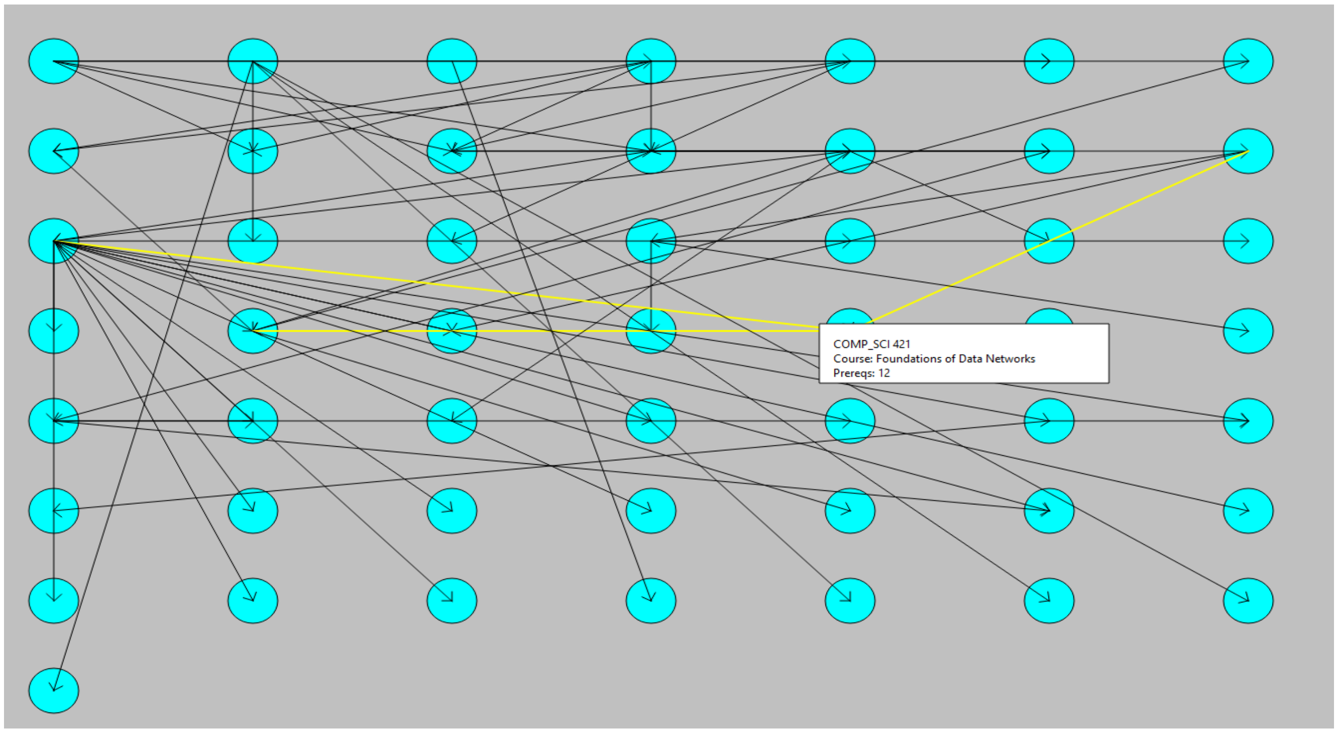


Figure 3: Maximum Prerequisites Hover*


COMP_SCI 421
Course: Foundations of Data Networks
Prereqs: 12

* Note that the edge highlighting only highlights *immediate* prerequisites.

Source Code

The source code for this project is located at: https://github.com/DylanSullins/CS291Project.git

This code was written by Dylan Sullins. Instructions for installation are included in the README.md

References

University of Missouri-Kansas City 2024-2025 Academic Catalog, UMKC, https://catalog.umkc.edu/course-offerings/undergraduate/comp-sci/. Accessed 16 Nov. 2024.