Patrick Gallagher

October 10, 2025

Project 2 Report

LLM Usage

For Project 2, I did not use any AI assistance. The following prompts are ways that I could have used AI.

Prompts

1. “In this file are three class definitions. Please write the method headers only. Do not write the content of any of the methods. Do not modify the main() function.”
    a. I could have used AI to prepare all of the method headers so that I could spend more time working on the content and function of the methods rather than the busy work of repeating the method definition.
2. “I want to overload the stream operator for TemporalSparseMatrix. Can you give me an example of what that code might look like?”
    a. I struggled to implement a proper overload method for the TemporalSparseMatrix. I decided to have the stream operator trigger the display method as a workaround. A test case generated by AI may have provided some valuable insight into developing that overload function.
3. “This is a sample input file for the TemporalSparseMatrix program and an explanation of the file format. Please develop a series of test inputs to examine special and edge cases.”
    a. Creating test inputs that examine all edge cases can be very time consuming, and possible source of human error. As a large database of programming knowledge, the AI model might be able to rapidly generate a robust series of test cases.

Debug/Testing

Debugging

There were three major disturbances in my debugging process. Firstly, after finishing all of my methods, my code continued to crash on test runs. I used several print statements to find the specific point of the crash and identified that point as the line where I attempt to add a new timeValueNode object to the head object. I realized that I had forgotten to initialize the head object in the sparseRow constructor, which meant that, because I had coded the add method with the assumption of a Null node in the head string, the operation failed. I changed the sparseRow constructor to create a Null timeValueNode as the head object and the issue was resolved. Another issue that I encountered was a misunderstanding of how the find method was intended to work. I had programmed my find method to return the value at that specific timeFrame and, if that timeFrame did not exist, to return a 0. I struggled with failing the input test cases for a period of time before deciding to check the project description to ensure that my implementation was correct. I soon discovered that the find method is supposed to find the most recent value for the input timeFrame and reconfigured my program to accomplish that goal. The last major issue

that I experienced was with the spareRow shifting once the timeValueNode list was empty. After a good amount of investigation using print statements and different test inputs, I discovered that the issue lay with my decision to use the delete function on the empty sparseRow, rather than reassigning the pointers and values. By calling the destructor without initializing a new sparseRow, I had shrunk the effective size of the array, which caused my code to behave in unexpected ways. Once I removed this delete statement, the code functioned as intended and passed all of the autograder tests.