Project 2

Texas Tech University
CS 3364: Design and Analysis of Algorithms
Professor: Dr. Victor Sheng
November 1, 2023

Dhruv Maniar R11713343
Atharva Dalvi R11765481
Apoorv Rana R1172307
Alex Murangira R11648884

Introduction

The primary aim of this project is to implement a topological ordering algorithm to determine the optimal sequence for undertaking courses within a Computer Science undergraduate program. This objective aligns with real-life scenarios where courses possess prerequisites, requiring a well-defined order for completion.

Problem

The objective is to identify the correct sequence of courses for a Computer Science major based on their prerequisites. A sample set of courses, along with their prerequisites, is provided. The challenge is to implement the topological ordering algorithm, primarily based on Depth-First Search (DFS), to determine the optimal sequence for course completion.

Solution

The project requires the design, implementation, and testing of an abstract data type (ADT) graph prototype, along with the topological ordering algorithm using DFS. The primary focus is on outputting the course sequence, sorted by their post-visit numbers derived from DFS traversal.

We have found a practical way to solve this issue by using different sorting algorithms: Merge Sort, Quick Sort and Insertion Sort to compare the 5 search engines and find the optimum one. We would be calculating the combined rank for each of the web page by all 5 sources (adding all 10000 ranks elements wise). Next step, would be sorting the combined data (in ascending order).

With the help of combined data, we would be rearranging the data in each of the 5 source files according to same index positions. After that, we would be applying different sorting algorithms on all of the 5 rearranged file arrays to calculate the inversions in each of the files. The file having least inversions would be considered as the best search engine.

To address the problem, the team adopted a systematic approach:

1. Parsing Course Data: Extraction of course details and their prerequisites from the file 'Data.txt'.
2. Graph Construction: Creation of an Abstract Data Type (ADT) graph to establish relationships among courses and their prerequisites.
3. DFS-Based Topological Sorting: Implementation of a DFS algorithm to perform topological sorting, capturing pre-visit and post-visit numbers.
4. Mapping Post-Visit Numbers: Assigning post-visit numbers to course codes, facilitating the sorted display of courses in the correct sequence.

Design and Implementation

1. Data Parsing: The team meticulously extracted and organized course information along with prerequisites from the provided file, structured into Course Info objects.

2. Graph Creation: Development of an ADT graph to effectively illustrate interconnections between courses and their respective prerequisites.
3. DFS Algorithm Implementation: The team designed and implemented the topological sorting algorithm using DFS, ensuring the meticulous tracking of pre-visit and post-visit numbers.
4. Post-Visit Numbers Assignment: Mapping post-visit numbers to course codes, enabling the precise sorting of courses in the optimal sequence.

Experimental Test Case

Rigorous testing was conducted to verify the accuracy and efficacy of the DFS algorithm. Various data inputs were employed to ensure the correctness of the optimal course sequencing. We found out that in the question file, the input data has a typing error. For CS 3365 the pre req CS 2413 does not have a space, hence we made the necessary correction in our Data.txt to ensure the program runs smoothly.

Conclusion and Future Work

Based on the post-visit numbers derived from DFS traversal, the courses were sorted in the order they should be taken, considering their prerequisites. The final output accurately demonstrates the sequence in which courses should be completed.

For future enhancements, the project could be expanded to include additional functionalities, such as visualizing the course sequence graphically. Furthermore, the code could be optimized for improved efficiency.

Team Work

The team functioned cohesively, ensuring equal contributions from all members. Individual team members were assigned distinct roles; some focused-on coding, others on developing the methodology and testing the code, while another member commented on the code and prepared the project report. This collaborative effort ensured the timely and successful completion of the project, providing each member with the opportunity to contribute meaningfully and gain valuable insights for their future projects.