VISVESVARAYA TECHNOLOGICAL UNIVERSITY

JNANA SANGAMA,BELAGAVI – 590018 KARNATAKA

Mini Project Report

On

"COURSE PREREQUISITE ON TOPOLOGICAL SORTING"

SUBMITTED IN PARTIAL FULFILLMENT OF THE ASSIGNMENT

FOR THE Analysis & Design of Algorithms(BCS401)
COURSE OF IV SEMESTER

Submitted by

KEERTHANA L D

[1CG22CS053]

Guide:
Mr.Asif Ulla KhanM. Tech.
Asst. Prof., Dept. of CSE
CIT, Gubbi.

HOD:
Dr. Shantala C P PhD.,
Head, Dept. of CSE
CIT, Gubbi.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Channabasaveshwara Institute of Technology
(Affiliated to VTU, Belgaum & Approved by AICTE, New Delhi)
(NAAC Accredited & ISO 9001:2015 Certified Institution)
NH 206 (B.H. Road), Gubbi, Tumkur – 572 216. Karnataka.

2023-24

Rubric – B.E. Mini-Project [BCS401]

Course outcome	Rubric/Level	Excellent (91-100%)	Good (81-90%)	Average (61-80%)	Moderate (40-60%)	Score
CO1	Identification of project proposal (05 Marks)					
CO2	Design and Implementation (10 Marks)					
CO3	Presentation skill (05 Marks)					
CO4	Individual or in a team development					
CO5	Report (05 Marks)					
Total						

Course outcome:

CO 1: Identification of project proposal which is relevant to subject of engineering.

CO 2: Design and implement proposed project methodology.

CO 3: Effective communication skill to assimilate their project work.

CO 4: Work as an individual or in a team in development of technical projects.

CO 5: Understanding overall project progress and performance.

Student Signature **Faculty signature**

INDEX

ABSTRACT

Course prerequisites refer to the specific courses, qualifications, or skills that students must have completed or attained before they can enroll in a particular course. These prerequisites ensure that students possess the necessary knowledge, understanding, and foundational skills required to succeed in the more advanced coursework. Prerequisites vary across different fields and institutions, reflecting the sequential nature of learning where earlier concepts and competencies serve as building blocks for more complex and specialized topics. Understanding course prerequisites is crucial for students to plan their academic paths effectively and to ensure they are adequately prepared for the challenges of higher-level courses.

CHAPTER 1:

INTRODUCTION

Course prerequisites serve as essential foundations for academic progression, ensuring that students have the fundamental knowledge and skills necessary to engage effectively with advanced coursework. These prerequisites are designed to create a structured learning pathway where each course builds upon earlier concepts and competencies. By establishing prerequisite requirements, educational institutions ensure that students enter higher-level courses prepared to delve deeper into specialized subjects without encountering gaps in their understanding.

Understanding the role of course prerequisites is crucial for both students and academic advisors. For students, knowing and fulfilling prerequisite requirements enables them to plan their academic trajectories strategically, ensuring they meet all prerequisites before enrolling in more advanced courses. This preparation not only enhances their learning experience but also increases their likelihood of academic success.

Moreover, course prerequisites vary across disciplines and institutions, reflecting the unique demands and progression paths of different fields of study. They may include specific prerequisite courses, minimum grade requirements, or proficiency in certain skills or knowledge areas. This variability underscores the importance of clear communication between students, advisors, and academic departments to ensure accurate understanding and fulfillment of prerequisite requirements.

In summary, course prerequisites play a vital role in structuring academic programs, guiding students through a coherent sequence of learning, and ensuring they possess the necessary foundation to excel in their chosen fields of study

CHAPTER 2:

PROBLEM STATEMENT

You are tasked with developing a system to manage course prerequisites at a university. Given a set of courses and their prerequisites, the system should allow students to determine whether they can take a specific course.

ALGORITHM STEPS

Topological Sorting using Kahn's Algorithm (BFS Approach):

1. **Create a Graph:** Build an adjacency list from the prerequisite list.
2. **Compute In-degrees:** Count the number of incoming edges (prerequisites) for each course.
3. **Initialize Queue:** Start with courses having zero in-degrees (i.e., no prerequisites).
4. **Process the Queue:**
    - Remove a course from the queue and add it to the sorted order.
    - For each course dependent on it (i.e., each course it points to), reduce the in-degree by 1.
    - If any dependent course's in-degree becomes zero, add it to the queue.
5. **Check for Cycles:** If the sorted order contains all the courses, then a valid order exists. If not, the graph has a cycle, making it impossible to sort topologically.

CHAPTER 3:

IMPLEMENTATION

This program provides the solution for the above problem statement:

```
# Course class to represent each course
class Course:
    def __init__(self, id, name, prerequisites=None):
        self.id = id
        self.name = name
        self.prerequisites = prerequisites if prerequisites else []

# List to store courses
courses = []

# Function to add a new course with prerequisites
def add_course(id, name, prerequisites=None):
    new_course = Course(id, name, prerequisites)
    courses.append(new_course)

# Function to check if a course can be taken based on its prerequisites
def can_take_course(course_id, taken_courses):
    # Find the course in the courses list
    course = next((c for c in courses if c.id == course_id), None)
    if not course:
        print(f"Course {course_id} not found.")
```

```
        return False

    # Check if all prerequisites of the course are in taken_courses
    for prerequisite in course.prerequisites:
        if prerequisite not in taken_courses:
            return False  # Missing prerequisite
    return True  # All prerequisites are met
# Example usage
# Adding courses with their prerequisites
add_course(101, "Introduction to Computer Science")
add_course(201, "Data Structures", [101])
add_course(301, "Algorithms", [201])

# Example: Check if a course can be taken based on provided course ID
taken_courses = [101]  # Assuming course 101 (Intro to CS) is taken

course_to_take =104 # Course ID to check
if can_take_course(course_to_take, taken_courses):
    print(f"You can take course {course_to_take}: {next((c.name for c in courses if c.id == course_to_take), 'Course not found')}")
else:
    print(f"Cannot take course {course_to_take}: {next((c.name for c in courses if c.id == course_to_take), 'Course not found')}")
```

Code explanation and data types used:

1. **Class CoursePrerequisite:**

- This class manages course prerequisites using a dictionary (self.prerequisites), where each key is a course and the corresponding value is a list of prerequisite courses.

2. **Method __init__:**

- Initializes an empty dictionary self.prerequisites when an instance of CoursePrerequisite is created. This dictionary will store all course prerequisites.

3. **Method add_course(course, prerequisite):**

- Adds a prerequisite prerequisite for a given course course.

- If course already exists in self.prerequisites, it appends prerequisite to its list of prerequisites.

- If course does not exist in self.prerequisites yet, it creates a new key-value pair where course maps to a list containing prerequisite.

4. **Method can_take_course(course, taken_courses):**

- Checks if a student can take a course course based on the courses taken_courses they have already completed.

- If course is not in self.prerequisites, it means course has no prerequisites, so it returns True.

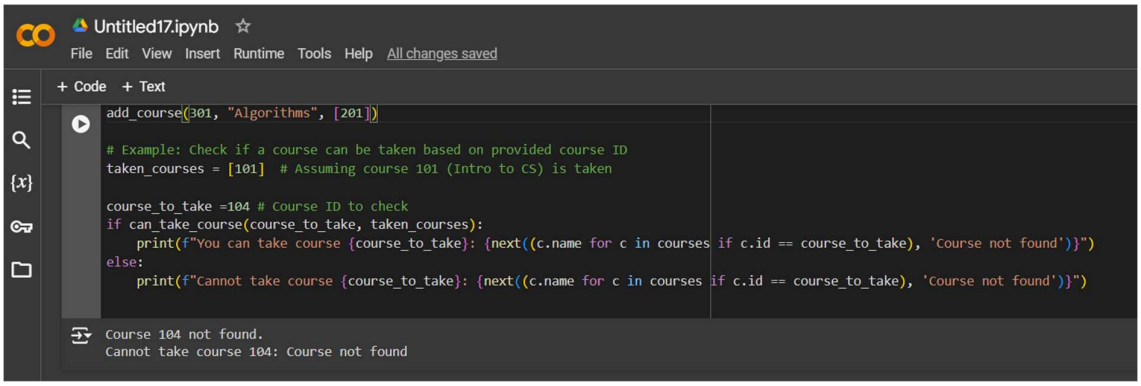
- Retrieves the list of required prerequisites (required_prerequisites) for course from self.prerequisites.

- Iterates through each prerequisite in required_prerequisites and checks if it exists in taken_course

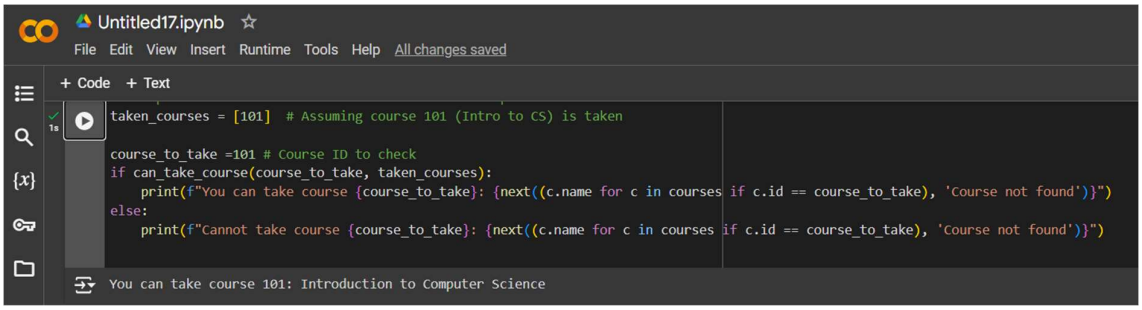
CHAPTER 4:

RESULT/SCREENSHOTS

SCREENSHOT 1:



SCREENSHOT 2:



CHAPTER 5:

CONCLUSION

Using Topological Sorting for Course Prerequisites

In conclusion, topological sorting provides a structured and efficient solution for managing course prerequisites in educational systems. By leveraging its properties, we can ensure that students progress through their academic journey in a systematic manner, adhering to all prerequisite requirements necessary for their chosen courses.

Implementing topological sorting ensures not only the correct sequencing of courses but also enhances the overall management and efficiency of educational programs by automating prerequisite validation processes. This approach not only meets the immediate requirements of course enrollment but also sets a robust foundation for handling future expansions and optimizations within educational institutions.

.

REFERENCE

1. "Introduction to Algorithms" by Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein
2. https://www.geeksforgeeks.org/topological sort/
3. https://www.tutorialspoint.com/data_structures_algorithms/topological_sort_program_in_c. htm
4. "Data Structures and Algorithm Analysis in C" by Mark Allen Weiss