# Project 2 - CS 5381: Analysis of Algorithms

Minh Chau Pham
chaupham@ttu.edu

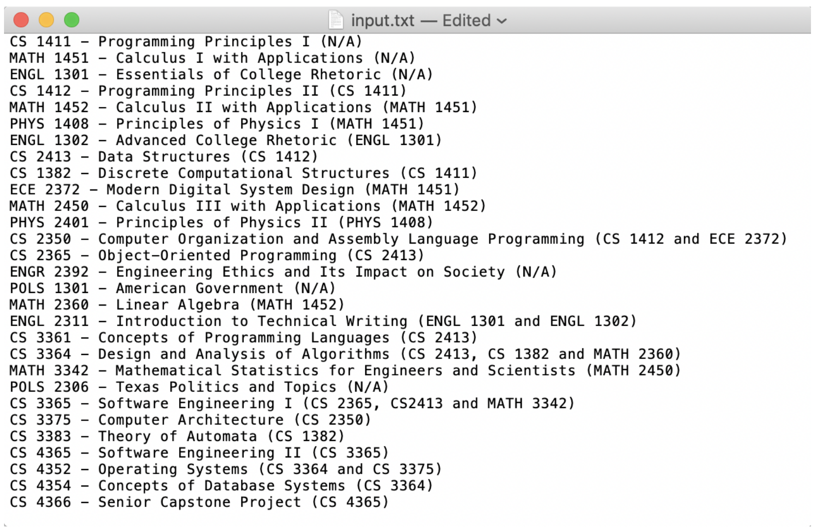
April 6, 2020

## 1 Problem Analysis

Topological orderings have many uses in our lives. For example, our day to day routine has been through a topological sort, or finding the order to take the courses is also considered a topological ordering problem. In this project, we implement a topological ordering algorithm by using Depth First Search (DFS) to determine an order for taking courses in the university.

## 2 Solution Design

First of all, we process the input for the project. The input data is a text file, in which each line indicates a subject following by its prerequisites in the bracket (Figure 1). For example, we can register the course *CS 1411 - Programming Principles I* for any semester due to no required prerequisite, but when it comes to *CS 2350 - Computer Organization and Assembly Language Programming*, we need to complete both *CS 1412* and *ECE 2372* first before taking the course. The data we have after the process step is a list of pairs *(subject, its prerequisite)* shown in Figure 2.



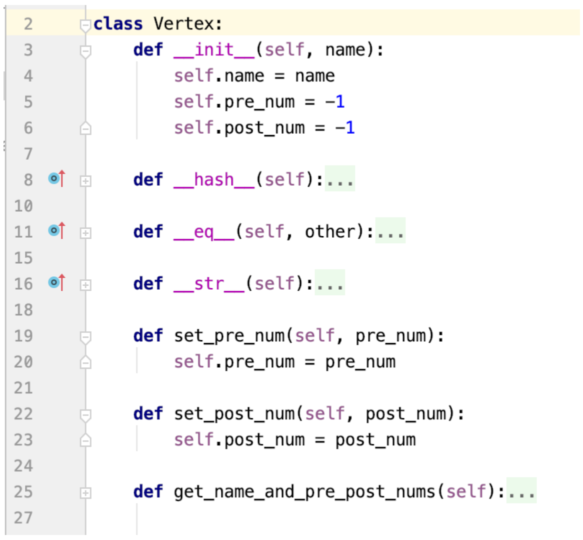
```
CS 1411 - Programming Principles I (N/A)
MATH 1451 - Calculus I with Applications (N/A)
ENGL 1301 - Essentials of College Rhetoric (N/A)
CS 1412 - Programming Principles II (CS 1411)
MATH 1452 - Calculus II with Applications (MATH 1451)
PHYS 1408 - Principles of Physics I (MATH 1451)
ENGL 1302 - Advanced College Rhetoric (ENGL 1301)
CS 2413 - Data Structures (CS 1412)
CS 1382 - Discrete Computational Structures (CS 1411)
ECE 2372 - Modern Digital System Design (MATH 1451)
MATH 2450 - Calculus III with Applications (MATH 1452)
PHYS 2401 - Principles of Physics II (PHYS 1408)
CS 2350 - Computer Organization and Assembly Language Programming (CS 1412 and ECE 2372)
CS 2365 - Object-Oriented Programming (CS 2413)
ENGR 2392 - Engineering Ethics and Its Impact on Society (N/A)
POLS 1301 - American Government (N/A)
MATH 2360 - Linear Algebra (MATH 1452)
ENGL 2311 - Introduction to Technical Writing (ENGL 1301 and ENGL 1302)
CS 3361 - Concepts of Programming Languages (CS 2413)
CS 3364 - Design and Analysis of Algorithms (CS 2413, CS 1382 and MATH 2360)
MATH 3342 - Mathematical Statistics for Engineers and Scientists (MATH 2450)
POLS 2306 - Texas Politics and Topics (N/A)
CS 3365 - Software Engineering I (CS 2365, CS2413 and MATH 3342)
CS 3375 - Computer Architecture (CS 2350)
CS 3383 - Theory of Automata (CS 1382)
CS 4365 - Software Engineering II (CS 3365)
CS 4352 - Operating Systems (CS 3364 and CS 3375)
CS 4354 - Concepts of Database Systems (CS 3364)
CS 4366 - Senior Capstone Project (CS 4365)
```

Figure 1: Input for the project

```
[['CS1412', 'CS1411'],
 ['MATH1452', 'MATH1451'],
 ['PHYS1408', 'MATH1451'],
 ['ENGL1302', 'ENGL1301'],
 ['CS2413', 'CS1412'],
 ['CS1382', 'CS1411'],
 ['ECE2372', 'MATH1451'],
 ['MATH2450', 'MATH1452'],
 ['PHYS2401', 'PHYS1408'],
 ['CS2350', 'CS1412'],
 ['CS2350', 'ECE2372'],
 ['CS2365', 'CS2413'],
 ['MATH2360', 'MATH1452'],
 ['ENGL2311', 'ENGL1301'],
 ['ENGL2311', 'ENGL1302'],
 ['CS3361', 'CS2413'],
 ['CS3364', 'CS2413'],
 ['CS3364', 'CS1382'],
 ['CS3364', 'MATH2360'],
 ['MATH3342', 'MATH2450'],
 ['CS3365', 'CS2365'],
 ['CS3365', 'CS2413'],
 ['CS3365', 'MATH3342'],
 ['CS3375', 'CS2350'],
 ['CS3383', 'CS1382'],
 ['CS4365', 'CS3365'],
 ['CS4352', 'CS3364'],
 ['CS4352', 'CS3375'],
 ['CS4354', 'CS3364'],
 ['CS4366', 'CS4365']]
```

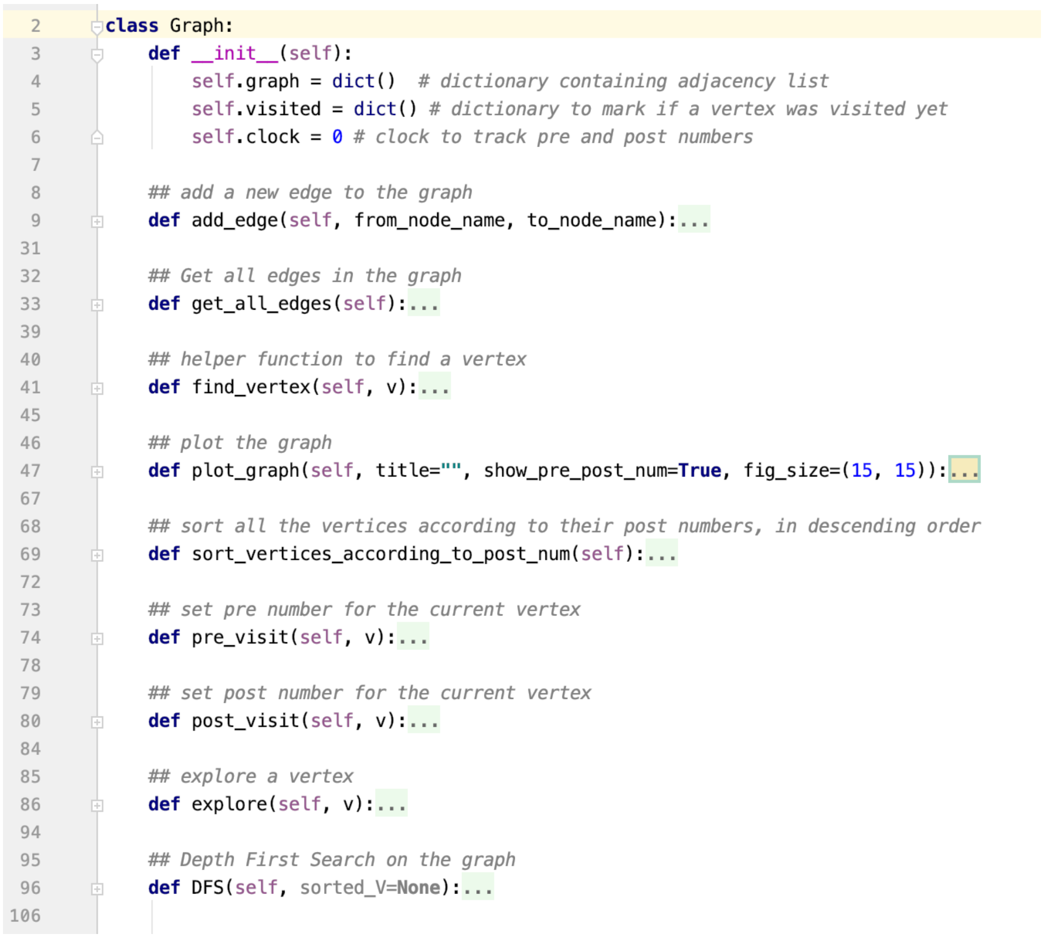
Figure 2: The input data after process step

Next, we implement two classes namely *Vertex* and *Graph*. The first class is to represent vertex in the graph, with some basic methods and attributes as in Figure 3. Basically, each vertex has a *name*, *pre_num* and *post_num* as the attributes (line 4-6, Figure 3). The *pre_num* and *post_num* can be updated when performing DFS algorithm.
Figure 4 shows the other class named *Graph*. In this class, we use a *HashMap(key, value)*, with each *key* is a vertex, and the *value* is the list of vertices that can go from the *key* (line 4, Figure 4). Another *HashMap* is adopted to mark if a vertex has been visited yet (line 5, Figure 4). The class also has a *clock* to keep track of its *pre numbers* and *post numbers* (line 6, Figure 4). In terms of methods, we have a function called *add_edge()* to build the graph by adding a new edge at a time (line 9, Figure 4). Besides, *DFS()* and *explore()* are the other main methods (Figure 5) which allow finding the topological ordering.



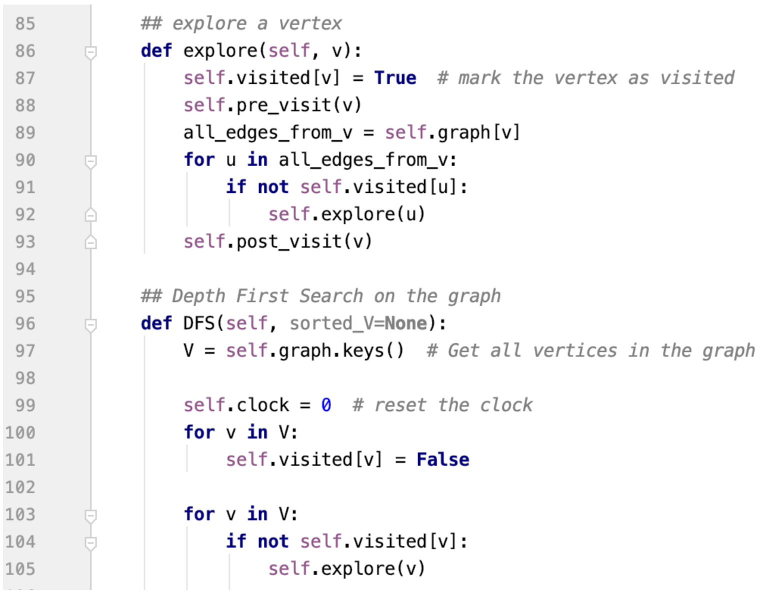
```python
class Vertex:
    def __init__(self, name):
        self.name = name
        self.pre_num = -1
        self.post_num = -1

    def __hash__(self):...

    def __eq__(self, other):...

    def __str__(self):...

    def set_pre_num(self, pre_num):
        self.pre_num = pre_num

    def set_post_num(self, post_num):
        self.post_num = post_num

    def get_name_and_pre_post_nums(self):...
```

Figure 3: Vertex class



```python
class Graph:
    def __init__(self):
        self.graph = dict()  # dictionary containing adjacency list
        self.visited = dict() # dictionary to mark if a vertex was visited yet
        self.clock = 0 # clock to track pre and post numbers

    ## add a new edge to the graph
    def add_edge(self, from_node_name, to_node_name):...

    ## Get all edges in the graph
    def get_all_edges(self):...

    ## helper function to find a vertex
    def find_vertex(self, v):...

    ## plot the graph
    def plot_graph(self, title="", show_pre_post_num=True, fig_size=(15, 15)):...

    ## sort all the vertices according to their post numbers, in descending order
    def sort_vertices_according_to_post_num(self):...

    ## set pre number for the current vertex
    def pre_visit(self, v):...

    ## set post number for the current vertex
    def post_visit(self, v):...

    ## explore a vertex
    def explore(self, v):...

    ## Depth First Search on the graph
    def DFS(self, sorted_V=None):...
```

Figure 4: Graph class



```python
    ## explore a vertex
    def explore(self, v):
        self.visited[v] = True  # mark the vertex as visited
        self.pre_visit(v)
        all_edges_from_v = self.graph[v]
        for u in all_edges_from_v:
            if not self.visited[u]:
                self.explore(u)
        self.post_visit(v)

    ## Depth First Search on the graph
    def DFS(self, sorted_V=None):
        V = self.graph.keys()  # Get all vertices in the graph

        self.clock = 0  # reset the clock
        for v in V:
            self.visited[v] = False

        for v in V:
            if not self.visited[v]:
                self.explore(v)
```

Figure 5: The two main functions of Graph class

From the input, we produce a graph that is drawn as in Figure 6 (using *plot_graph()* method (line 47, Figure 4)). The graph has 26 vertices (subjects) and 30 edges. We then perform *DFS()* on the graph by first resetting its *clock*, and mark all the vertices as unvisited. Then, we employ *explore()* on each vertex to determine its *pre number* and *post number*. By doing this, we get all *pre nums* and *post nums* of the graph (Figure 7). We sort all the vertices according to their *post numbers* in descending order (using *sort_vertices_according_to_post_num()* method (line 69, Figure 4)) to get the topological ordering of the graph, which is also the order of taking courses.

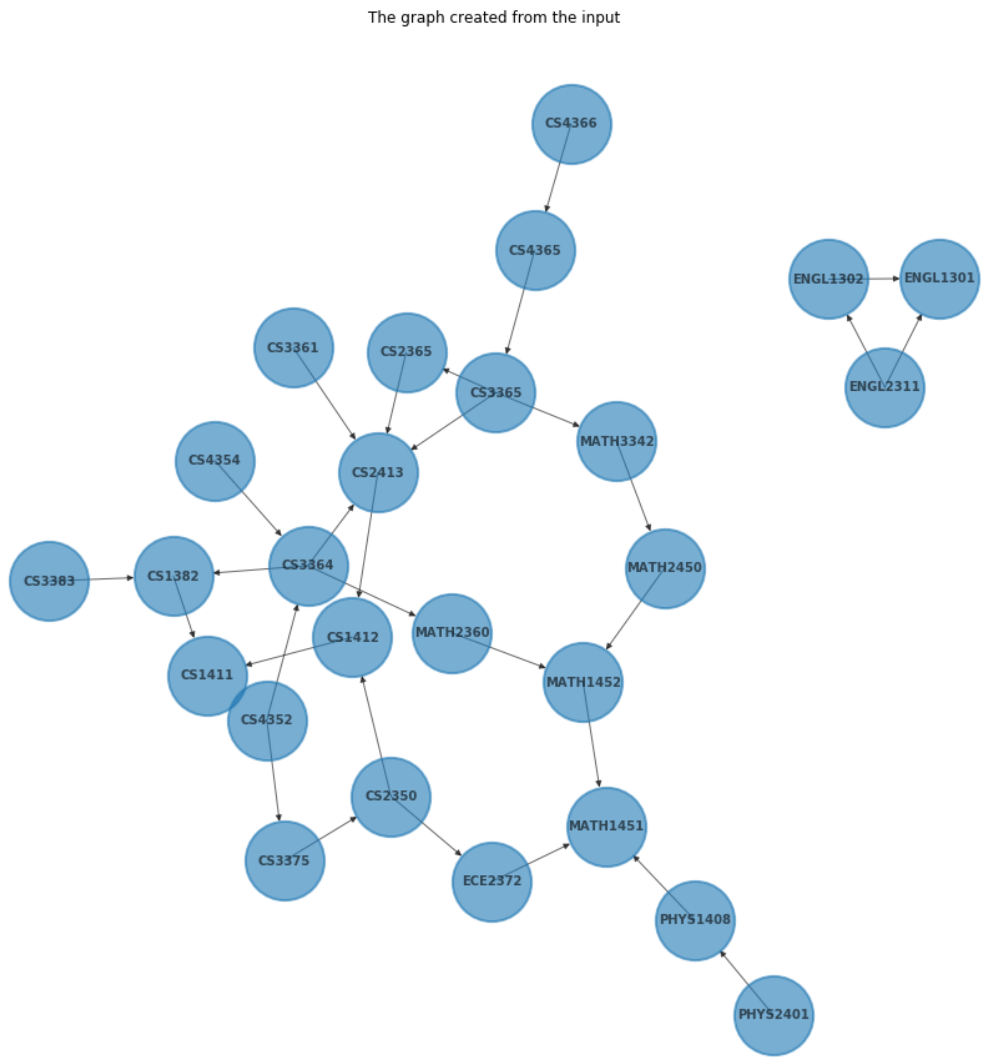


Figure 6: The graph

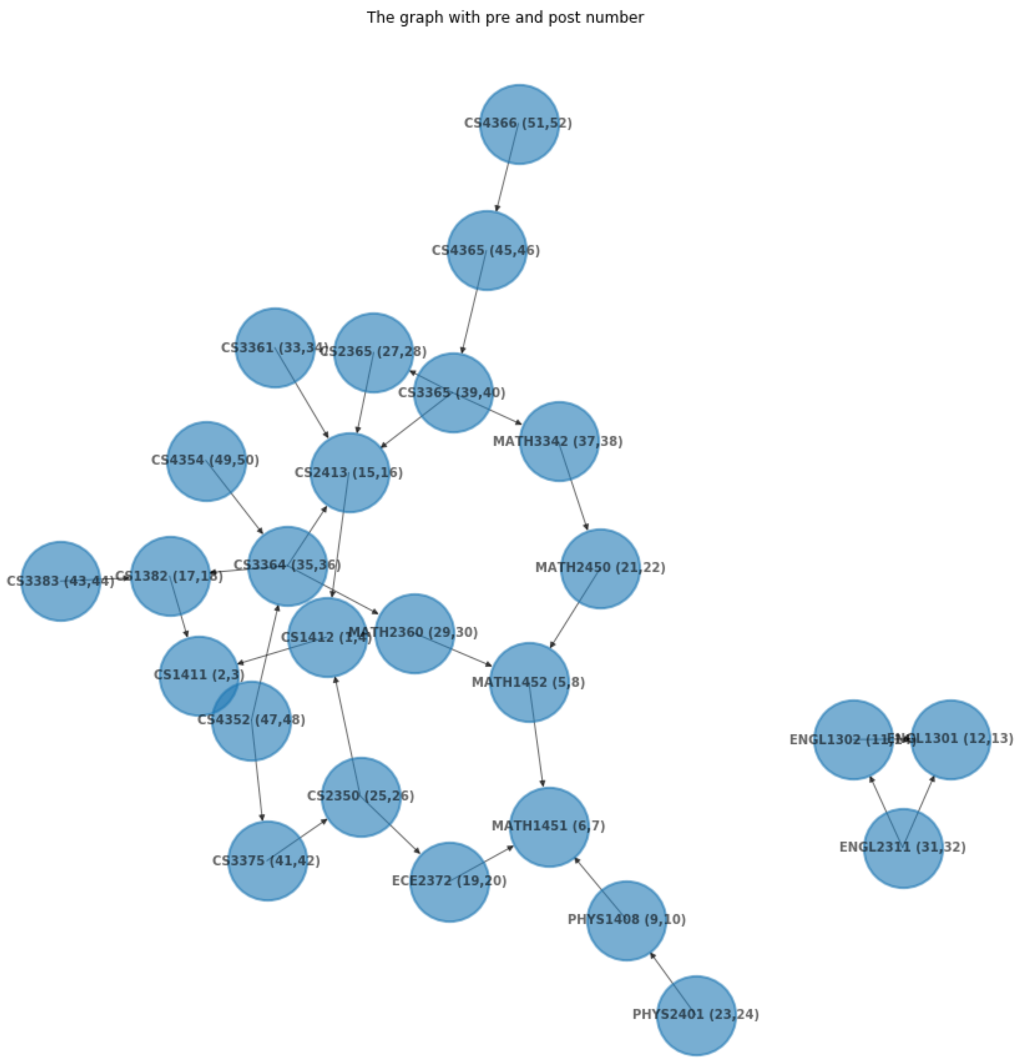


Figure 7: The graph with pre and post numbers for its vertices

## 3 Experimental Results

There may be many different solutions for the topological ordering. In other words, our result is just one of all possible orders. Figure 8 depicts our result, in which the order of taking courses is from the top to the bottom.

```
CS4366 (51,52)
CS4354 (49,50)
CS4352 (47,48)
CS4365 (45,46)
CS3383 (43,44)
CS3375 (41,42)
CS3365 (39,40)
MATH3342 (37,38)
CS3364 (35,36)
CS3361 (33,34)
ENGL2311 (31,32)
MATH2360 (29,30)
CS2365 (27,28)
CS2350 (25,26)
PHYS2401 (23,24)
MATH2450 (21,22)
ECE2372 (19,20)
CS1382 (17,18)
CS2413 (15,16)
ENGL1302 (11,14)
ENGL1301 (12,13)
PHYS1408 (9,10)
MATH1452 (5,8)
MATH1451 (6,7)
CS1412 (1,4)
CS1411 (2,3)
```

Figure 8: The order of taking courses, from top to bottom, with its pre and post numbers respectively in the brackets

## 4 Conclusion and Future Work

In this project, we adopt *DFS* as a topological ordering algorithm to determine the order to take courses. We implement two classes to represent the graph, as well as to obtain the topological ordering.

In future work, we will extend to more algorithms such as Breadth-First Search, Decomposing graph. We will also conduct the experiment on bigger data sets and try to improve the performance in terms of speed and memory usage.

## 5 Submission

The submission folder contains 4 files as below.

- A report (this file): *Report.pdf*
- Input data: *input.txt*
- A *Python* source code, written in *Jupyter Notebook* format: *main.ipynb*

- An HTML file exported from the *Python* source code, which can be easily opened by web browsers: *main.html*