CSC111 Project: Currere

William Wu, David Duan, Hecate Li, Declan Pang

Saturday, April 1st, 2022

1 Problem Description:

How can we improve the course selection process at UofT to make it more convenient and straightforward for students?

Course selection is an essential process for students at UofT as it determines their academic journey and sets the foundation for their future career. However, the current course selection system can be tedious and inaccessible for students, leading to frustration and confusion during the registration period. The problem is that the course selection process at UofT requires students to navigate a complex system that involves multiple steps and criteria. For instance, students have to select courses based on their majors, track their prerequisites, and consider course schedules and timings. Furthermore, the course selection system often experiences technical glitches and errors, causing delays and uncertainty. For newly admitted students at UofT, the course selection process can be particularly overwhelming and confusing. These students are not familiar with the university's systems and interfaces, which makes navigating the course selection process even more challenging. A graphical representation of data would be helpful for understanding. If students are not careful during their course selection in their first year at UofT, they may miss prerequisites for upper-level courses that they need to take later on. Prerequisites are essential as they ensure that students have the foundational knowledge and skills needed to succeed in more advanced courses. Missing a prerequisite can result in having to delay progress towards graduation or even having to take additional courses to make up for the missing prerequisite. **The goal of this project is to address the challenges that students face during the course selection process and propose solutions that enhance accessibility and efficiency. To achieve this, the study will draw on existing course selection systems topped with our computer science knowledge and represent the project with graphs and other data structures.**

2 Data Set Description:

We've gathered all relevant course data from the website https://artsci.calendar.utoronto.ca and scraped it using the Python library BeautifulSoup, then storing it in a csv file named '*combined_math_cs_sta.csv*'. This file contains basic information on undergraduate courses from all three faculties with their perspective prerequisites, corequisites and minimum cut-offs.

3 Computational Overview:

3.1 Graph representation of the project:

CourseGraph and Course CourseGraph and Course are the two most important data class in our project. We;ve chose to represent our data class with the graph data structure. To simplify, CourseGraph resembles a graph, and Course is similar to a vertex in the graph. The edge within the courses are represented as the prerequisites, which are the essential pathway(s) one must take in order to study in the desired course.
In a CourseGraph, there is only one attribute, which is courses. It's a dictionary that contains course codes as keys and maps to the specific Course objects. For example, 'MAT137Y1': Course('MAT137Y1', 'Calculus with Proofs'). In the CourseGraph, there are various methods, which include some basic graph methods including add_course and add_edge, which have similar functions as the ones in the graph class. There are also some special methods for our CourseGraph, for example, a method called find_all_prereq that uses recursion to return all of the prerequisites of a specific course, as well as all of the pre-prerequisites, until the base prerequisite is met. In a Course, there are four attributes.

- name, which stores the course code of the course as a string. For example, 'MAT157Y1'.
- prereq, which stores the prerequisites of this course as a list with dictionaries or tuples in it. For example, if the prereq of course is '(grade $\geq$ 70 for course MAT137 and pass for MAT223), or grade $\geq$ 60 for MAT157', then self.prereq will look like:[('MAT137': 70, 'MAT223':50), 'MAT157':60]. In this case, the prerequisite is satisfied if and only if any of the requirements for (MAT137 and MAT223) is satisfied, or the score for MAT157 is not lower than 60. The 'and' condition is represented by the tuple data class, and the 'or' condition is represented by the list data class. If the prereq of the course is MAT137 or MAT157, then self.prereq will be like ['MAT137':50, 'MAT157':50], which minimum grade requirement is set to 50 by default indicating the student passed the course and earned the prerequisite credit.
- higher_courses, which is a set that stores the courses that use this course as a prerequisite. For example, the higher_courses attribute for CSC110Y1 is 'STA442H1', 'STA414H1', 'STA313H1', 'STA314H1', 'STA302H1', 'CSC111H1', 'STA410H1'.
- key_words, which store some basic description of the course as a string. This is later used in the interactive model in our project.

3.2 Web scraper: Beautiful Soup

The web scraping portion is done in the file 'csc111_proj_data.py', which we imported Beautiful Soup and regular expression(re). With experimentation, we've discovered that the format .xls, then convert into .csv format.
Scraping was a new and challenging process. We've idtendified the html div with class name "view-content" containing the courses on the website. However, there are 3 such div of class "view-content", and we only took the last one as our target. Within this big div, we looped through courses to record each course individually, and then storing them. This process required regular expression replacement and identification since the Faculty of Arts and Science(FAS), Utsc, Utm and engineering courses all had different representation on the academic calender website. We have to treat it on a case basis. For example, if a course is from FAS, it's html element would be

<a href="/course/MAT235Y1">MAT235Y1</a>

and we would keep the 'MAT235' portion with re.sub() function. Likewise, courses from UTSC, UTM, and engineering would have a hyprelink that start with utsc, utm, and eng, respectively.
There also existed some inconsistency within the web page. Some engineering course started with ' href="/course/', and other times they start with 'href="https://engineering.calendar.utoronto.ca/. This added some additional work towards course identification. So we added more layers of re.sub(). It is also a problem that some courses are no longer offered, so we manually deleted those instances. Our automated portion consists of identifying cases that doesn't lead to a course page.

3.3 Data cleaning and processing

After using the web scraper to save the courses in math, statistics, and computer science department as CSV files, we did some data cleaning, which modified the space between some characters and added a small amount of information that the web scraper failed to catch. For each line of the modified CSV file, the first column contains the course code as well as a short description of the course, and the second column contains the prerequisites of this course. For each prerequisite, we use the function compute_prereq, which inputs the string, removes unnecessary information, and uses a combination of str.split() and str. replace() methods to return a list of prerequisites that can be used as the prereq attribute in our Course class. After that, the graph can be generated using the read_csv function, which basically loops through each row of the modified CSV file, call the helper function compute_prereq, and uses add_course and add_edge to add courses and the relationship between courses to the CourseGraph. As mentioned in the previous section 3.

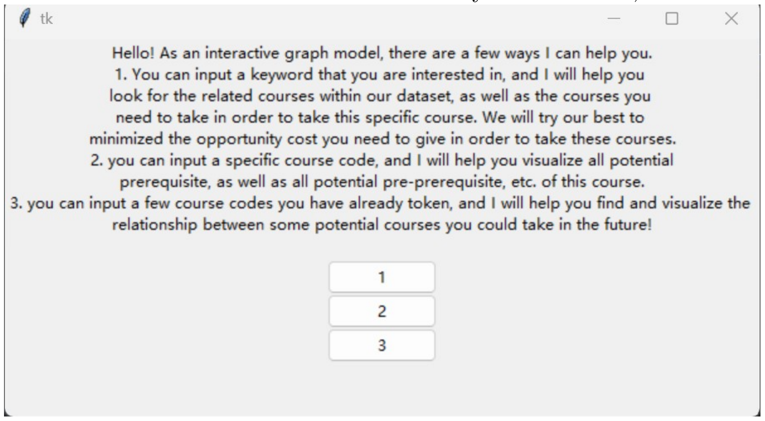
3.4 Visualization: Networkx and matplotlib

In the visualization part of the project, we use two new libraries to visualize the graph. They are NetworkX and Matplotlib.
By using NetworkX, we create a digraph by calling the DiGraph() class, and we use its add_node and add_edge methods to add the courses and the relationship between courses into the DiGraph. In other words, we convert the CourseGraph into a DiGraph. Then, we use the draw() function from NetworkX to create a basic plot of the graph

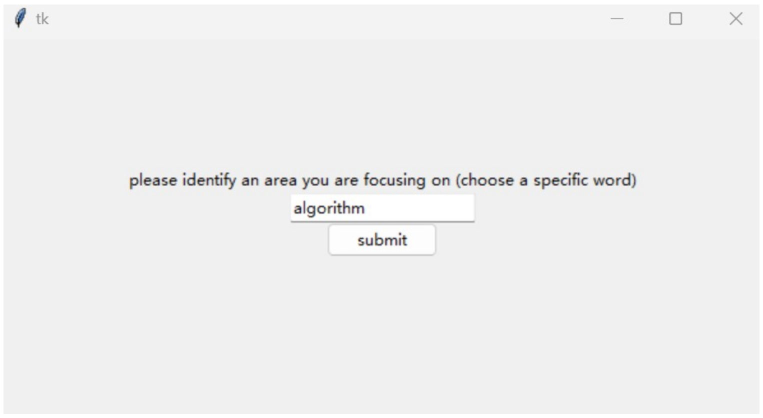
and use the show() function from Matplotlib to display the plot.
In general, we used NetworkX to create ad manipulate the graph data, and use matplotlib to visualize the graph. There are two visualize functions, visualize_course_graph, and visualize_course_graph_node. The first function returns the visualization of the CourseGraph as a whole, and the second function returns some specific part of the graph(basically it asks the user to input a list of courses, and returns a visualization of the relationship between these courses within the whole CourseGraph.

3.5 Interactive prereq explorer

There are three main interactive functions in our project. The first one asks the user to input a specific keyword, for example, "algorithm", and print out a recommended courses for this user, as well as its potential prerequisite that minimizes the opportunity cost(a year course has the opportunity cost of 1 and half year course have 0.5) for taking this course. The list of returned courses will be organized into the first year, second year, third year, and fourth year, which may become a potential future course selection for the user. Then, the user can choose whether or not to Visualize the relationship between the recommended courses and their potential prerequisite.
In this interactive function, it uses a combination of if-else statements, while loop and input() to ask the user to input a keyword, uses the CourseGraph method course_with_keywords to loop through the courses attribute of the CourseGraph and randomly select a course whose description contains the keyword, and uses a recursive CourseGraph method compute_cost to get the list of prerequisites as well as all the potential pre-prerequisite, until the base prerequisite is met, that minimize the opportunity cost for each course above separately. Then, it uses the above visualizing function visualize_course_graph_node to visualize the relationship between the courses.
The second interactive function Asks the user to input a specific course code, for example, MAT137Y1, and shows all of the prerequisites the user can take in order to take this course. including the prerequisite of prerequisite, until the base prerequisite is met. This method utilizes a CourseGraph recursive method called find_all_prereq and uses a combination of while loops, and if-else statements to let the user choose whether or not to continue, and use the same visualizing function as above to return a visualization of the relationship between the return courses for the user.
The third interactive function asks the user to input some course he/she already took, and return 5 potential possible courses the user could take in the future. It utilizes a CourseGraph method called find_higher_courses, which inputs a list of courses the user took, and uses recursion in the Course attribute higher_courses to return the possible courses he/she can take in the future. It also uses a combination of while loops and if-else statements to let the user choose whether or not to continue, and whether or not to visualize the result. For the visualization of this method, it use the same visualizing function as above.
At last, we combined these three interactive functions into one last interactive function, which we decided to wrap up using Tkinter, making our research much more like an app, making it more user-friendly. Users would first click on buttons to choose which function they want to call,

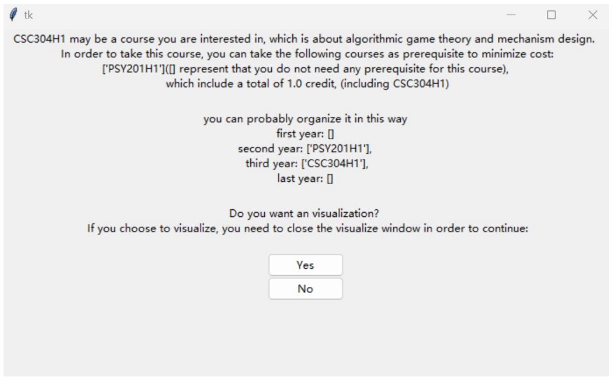

Hello! As an interactive graph model, there are a few ways I can help you.
1. You can input a keyword that you are interested in, and I will help you look for the related courses within our dataset, as well as the courses you need to take in order to take this specific course. We will try our best to minimized the opportunity cost you need to give in order to take these courses.
2. you can input a specific course code, and I will help you visualize all potential prerequisite, as well as all potential pre-prerequisite, etc. of this course.
3. you can input a few course codes you have already token, and I will help you find and visualize the relationship between some potential courses you could take in the future!

and then for all three functions, a new window would pop up where the user would be able to enter relevant information in a textbox


please identify an area you are focusing on (choose a specific word)

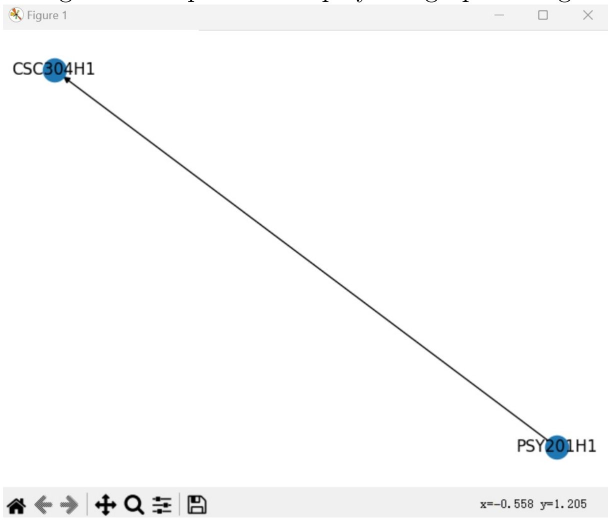
algorithm

Lastly, the desired info will be given and the user will be given the option to display the graphs using networkx


CSC304H1 may be a course you are interested in, which is about algorithmic game theory and mechanism design.
In order to take this course, you can take the following courses as prerequisite to minimize cost:
['PSY201H1']([] represent that you do not need any prerequisite for this course),
which include a total of 1.0 credit, (including CSC304H1)

you can probably organize it in this way
first year: []
second year: ['PSY201H1'],
third year: ['CSC304H1'],
last year: []

Do you want an visualization?
If you choose to visualize, you need to close the visualize window in order to continue:



4 Instructions for obtaining datasets and running our program

Please download and install all Python libraries listed under our ‘requirements.txt’ file that we submit as part of the project.
Then proceed to download the data set zip file called ‘dataset.zip’ from Markus. Please unzip the file, then proceed to find ‘combined_math_cs_sta.csv‘, this is the dataset file we’re going to use. Then run ’main.py’ which will run our project.
Note: please extract the dataset file ”combined_math_cs_sta.csv” to the same folder as the rest of the files.

5 Some changes to our project plan

In our initial project plan, we used sets to denote the ‘or’ relationships between prerequisites. For example, if the prerequisite for a specific course is MAT137Y1/MAT157Y1, then the prerequisite of the Course object will look like: ‘MAT137Y1’:50, ‘MAT157Y1’:50. However, we found that adding a dictionary into a set will result in the error TypeError: unhashable type: ’dict’. Although this can be fixed by converting the dictionary into a tuple, it needs much more computation between types conversions, and this conflicts with our original definition of tuple. As a result, we used lists to denote the ‘or’ relationships between prerequisites. Also, instead of using the FastAPI and plotly libraries for visualizations as described in the propsosal, we used networkX and metaplotlib instead. This is because we learned about the library networkX during our tutorial, which enhance our efficiency during the computation process.

6 Discussion

- The result of our exploration helped achieve our goal of making course-selection less stressful for new students especially. Even though our end result didn't achieve all of our goals at the beginning, in this short span of time, being able to come up with a nice looking application and the ability to showcase how we used graphs to represent the connections of different, we are satisfied with this program which each of us might actually use in the future.
- The biggest limitation is the amount of time we are given, since the project was given right before midterm started and is due before school end. Beside this, one of the big limitation is the fact that python is not so customizable with displaying graphs as well as GUI, this is mostly due to the fact that python, in the end, is not designed be used for this. If we were given the ability to use other languages, Javascript would've made the graphs look much cleaner and more customizable. What't more, when we visualize the our CourseGraph using mataplotlib and networkx, although the visualization and other outputs are shown correctly, every time when we run the python file and call the function for the first time, an error message pops out. This is probably due to some sort of incompatibility between version of libraries, and we don't know how to solve this problem. However, since the output is not affected by this error, we didn't modify our code.
- For the future plan, since We've only covered three faculties: mathematics, statistics, and computer science. Given the opportunity, we would like to do this for all the faculties as well as engineering, and eventually incorporate javascript on the front-end side, which will make this program much more convenient and user friendly. What's more, since there are exclusion between courses, we would like to add this as another attribute of Course class in the future, and add some graph method to consider the exclusion properties in our interactive model. Further more, since every minor, major, specialist in U of T have different requirement for student, in the future if we have enough time, we can design a new program class, which is a class representation of various minor, major and specialist in U of T. In this case, we can combine this class with the usage of our existing classes, and develop some methods to give the users advice on efficient courses selection for completing the major, specialist or minor he/she currently in. Last but not least, due to the limitation of time, we didn't accomplish, but would like to develop a more complex interactive model, that ask the user to input a combination of more information, and return a complete 4-year courses selection for the user. This required a combination that includes but not limited to keyword searching, prerequisite and future courses searching, program searching/matching, credit calculation for every school year.

References

Faculty of Arts and Science. “2022-23 Calendar.” Academic Calendar, 2022, https://artsci.calendar.utoronto.ca/.
Faculty of Arts and Science. “CSC263H1: Data Structures and Analysis.” CSC263H1 — Academic Calendar, 2022, https://artsci.calendar.utoronto.ca/course/csc263h1.
Liu, David, and Ian Stewart-Binks. “Graph.” Courseography, 2013, https://courseography.cdf.toronto.edu/graph.