CSC111 Project 2 Report: Course Master

Vincent Wang, Yuchen Wang, Jiahe Xiang, Yizhou Ai,

April 4, 2024

Project Introduction

At the University of Toronto, students are given the freedom to choose the courses they want to take from a vast range of options. However, each course will usually has some specific requirement, which means students need to fulfill the entire requirement before they are able to take the course. For example, a prerequisite indicates that students must complete all the courses listed before enrolling in this one; And an exclusion indicates that the courses listed is similar in content and should not be taken any more.

Such being the case, students sometimes get lost in arranging the orders of the courses they want to take and end up needing to spend a lot of time on it. In addition, when students accidentally overlook some course requirements, they may even end up being rejected to take the course or end up receiving no credits because the course was listed as an exclusion.

The problem described above deeply annoyed every student in the University of Toronto and motivated us to develop a program with a GUI using the concepts we learnt in CSC111 to help figure out the orders of the courses. Once the user inputs all the courses he/she wants to take in any order, our program tells them step by step what to take. What's more, we also provided a function for the users to visualize the relationship of exclusion between different classes. Under the help of our program, users not only save time in course selection but also prevent potential issues in those courses.

Datasets

We use a python library called scrapy to obtain course information from the Academic Calendar of Uoft among all the three campus (around 10000 classes) , and the data is stored in a file called courses.json , the format of the dataset is as follows:

”name”: ” CSC111H1 - Foundations of Computer Science II”, ”br”: [”The Physical and Mathematical Universes (5)”], ”pre”: [”CSC110Y1”], ”distribution”: [”Science”], ”exclusion”: [”CSC108H1”, ”CSC148H1”, ”CSC165H1”], ”introduction”: [”A continuation of ”, ” to extend principles of programming and mathematical analysis to further topics in computer science.]

name: the course name br: breath requirement category pre: prerequisite courses distribution: distribution category exclusion: exclusion courses introduction: introduction of the courses

Computational Plan

For our project, “Course Master”, we will be mainly using graph and directed graph. For our implementation, we will use a directed graph to represent the prerequisite of the courses, all the courses which are of prerequisite of on course will be the neighbors of that course. If a course has no prerequisite, it just simply means the degree of that vertex is 0. One thing deserves to mention is that the relationship of prerequisite courses are only in one direction (a is prerequisite of b does not imply b is prerequisite of a), so that's why we use directed graph instead of graph to represent it. In terms of exclusion, we will use a different undirected graph to represent. If the two courses are exclusion to each others, then they are adjacent in the graph since exclusion is a relationship in both directions.

After obtaining data from the 2023-24 Academic Calendar [1] [3][4]of the University of Toronto using scrappy library, we will first filter and tackle the data into more friendly format and then using a function to transform it into both directed graph and undirected graph.

Our program mainly has two ways to report the result: one is generating an image to visualize the relationship of courses, and the other is displaying text information. Additionally, users can further explore the data through interactive graphs and text entry.

In terms of the libraries we use, we first employ Scrapy to obtain data from the internet and store it in a well-formatted JSON file. Next, we utilize Tkinter to create an interactive GUI for users to interact with the data. Finally, we use Networkx and Plotly to generate visually appealing images for users to visualize the relationships between different courses(code adapted form [2]).

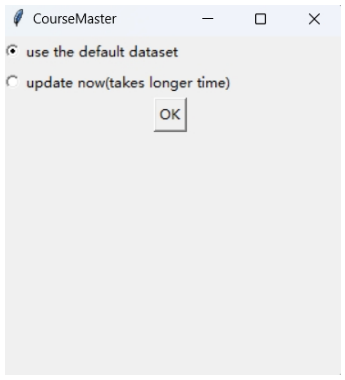
Two key function in our program:

1. Arrange courses: In this function, the input is a list of unordered courses and the output is the courses in order based on prerequisite. To implement this function, we use a while loop to first iterate the copy of the unordered list to check if there are courses whose degree in directed graph is 0, if so, we remove the course from the list and add it to step1; after the first iteration, we again iterates the remaining courses in the list, this time we check if the neighbors of the courses appear in the previous step, if so, we add the course to the step2. Do the following iteration until the sorted list is empty, and return the courses each step contains. If the list can not be empty, then the courses can not be arranged due to lack of prerequisite.

2. Visualize course exclusion: In this function, we want the output to generate an image showing not only the exclusion of one course but also the courses excluded from those exclusions. In other words, we want to get not only the neighbors of one vertex, but also the neighbors of neighbors. To implement this, we need to use recursion, we first create a list to store all the edges between the neighbors and the vertex, and go through every neighbor. The base case is that all the possible edges are added into that list and the function return.

Running our program

1. Download the data.zip file from Markus and unzip it into the **same directory** as the main.py file.
2. Install all Python libraries listed under the requirements.txt file
3. Run a file We submit called main.py. You should be able to see the following window:

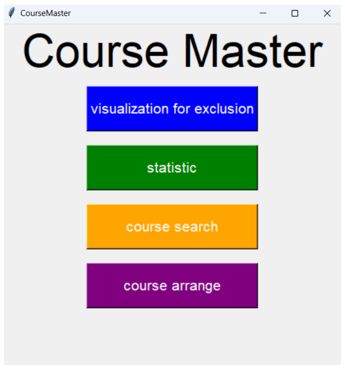

CourseMaster

[✓ use the default dataset]

update now(takes longer time)

OK

Selecting the first option will allow the program to run based on the default dataset. Selecting the second option will trigger the program to run Scrapy to obtain the latest data, which may take around 5 minutes.


CourseMaster

Course Master

visualization for exclusion

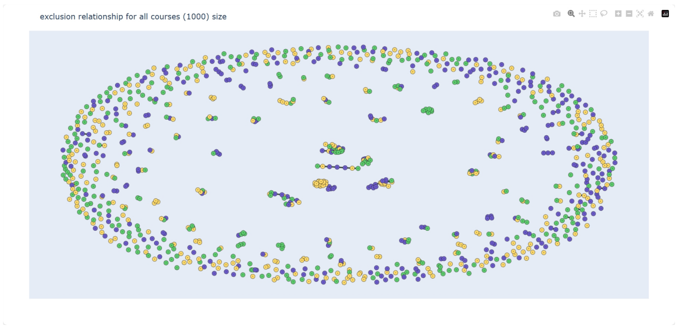
statistic

course search

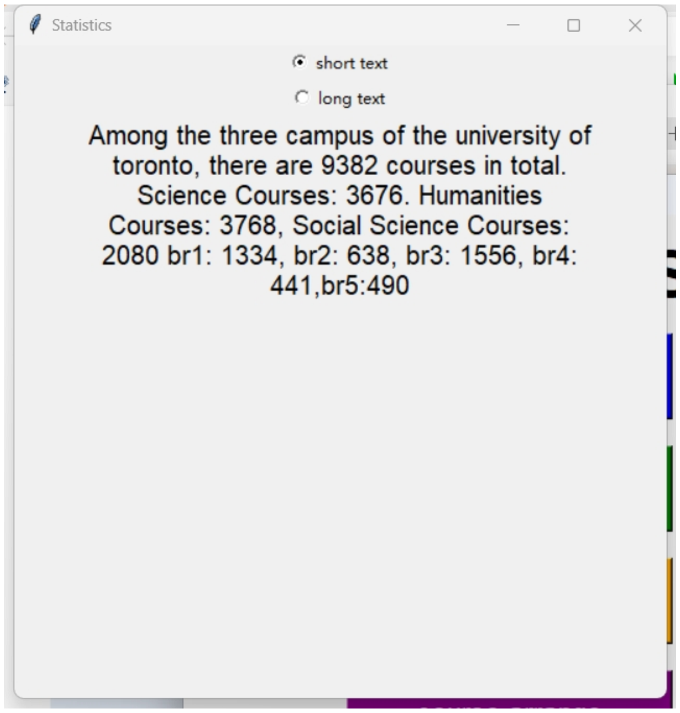
course arrange

The program has 4 main functions:

1. Visualization for exclusion: After building the undirected tree to represent the relationships between courses, we aim to provide users with a better understanding of the overall structure. Therefore, this function creates an image that represents the graph of all the courses.

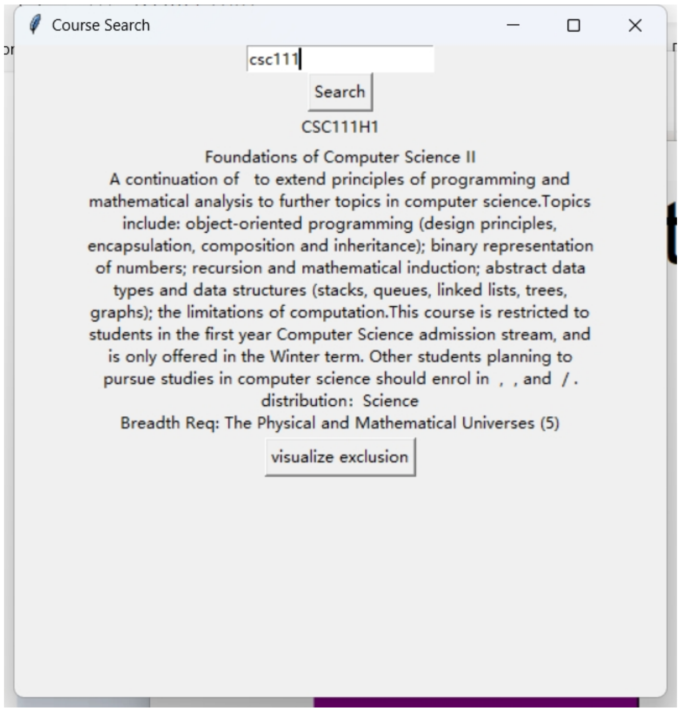


2. Statistic: The data we collected also includes other information, such as breadth requirements and distributions. We provide users with an overall insight by calculating the ratio of courses in different categories and some statistical numbers.



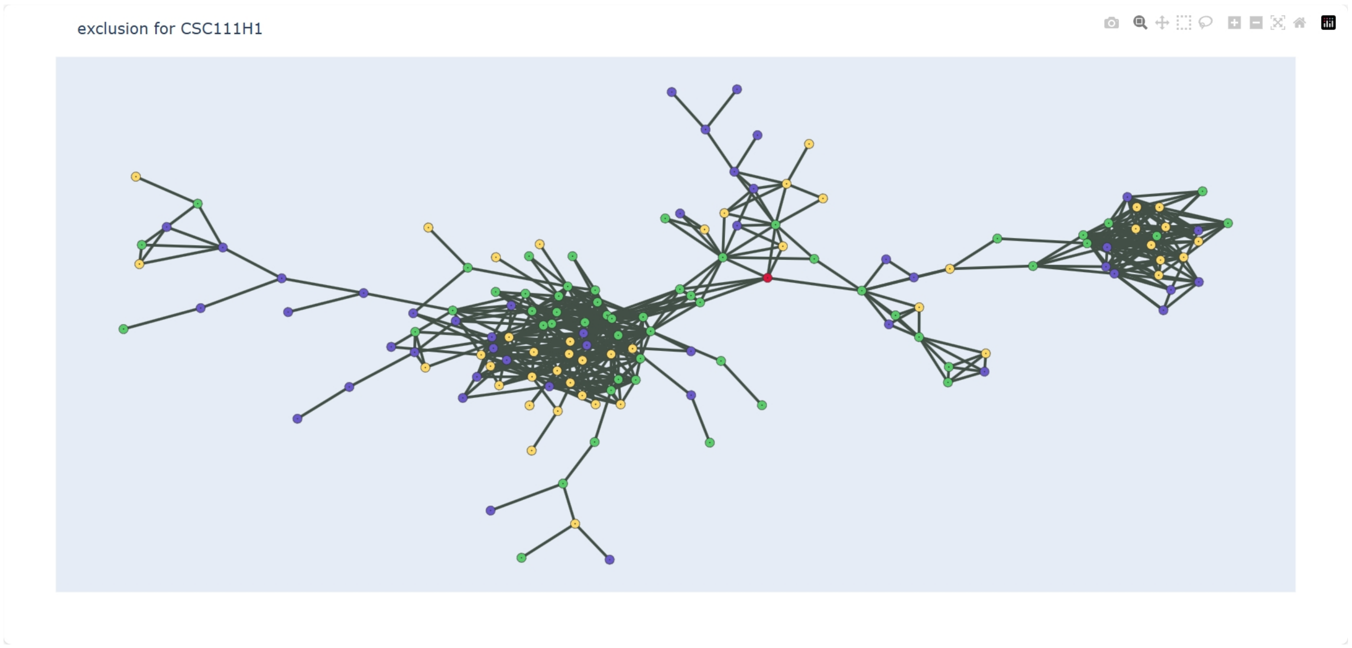
Among the three campus of the university of toronto, there are 9382 courses in total. Science Courses: 3676. Humanities Courses: 3768, Social Science Courses: 2080 br1: 1334, br2: 638, br3: 1556, br4: 441,br5:490

3. Course search: User can enter courses in the search box to get information of a certain course

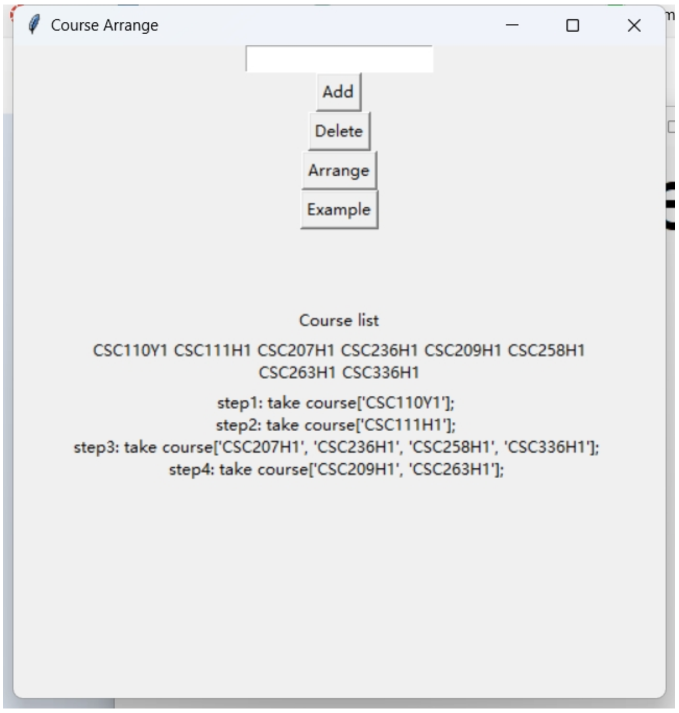


CSC111H1

Foundations of Computer Science II
A continuation of to extend principles of programming and mathematical analysis to further topics in computer science.Topics include: object-oriented programming (design principles, encapsulation, composition and inheritance); binary representation of numbers; recursion and mathematical induction; abstract data types and data structures (stacks, queues, linked lists, trees, graphs); the limitations of computation.This course is restricted to students in the first year Computer Science admission stream, and is only offered in the Winter term. Other students planning to pursue studies in computer science should enrol in , , and / .
distribution: Science
Breadth Req: The Physical and Mathematical Universes (5)



4. Course arrange: User can enter a list of courses and the program will be able to arrange the courses in orders based on prerequisite.



```
CSC110Y1 CSC111H1 CSC207H1 CSC236H1 CSC209H1 CSC258H1
CSC263H1 CSC336H1
step1: take course['CSC110Y1'];
step2: take course['CSC111H1'];
step3: take course['CSC207H1', 'CSC236H1', 'CSC258H1', 'CSC336H1'];
step4: take course['CSC209H1', 'CSC263H1'];
```

Difference from Project Proposal

1. Instead of just focusing on CS, MATH, and Statistic courses, we focus on a broader range of all the courses that the university offer. Such being the case, more students can benefit from our program.

2. Instead of using command lines to control the program, we create a GUI for the users, which will be much more convenient than just typing the commands.

3. We create a new function called arrange courses,which can automatically arrange a set of courses in orders based on prerequisite.

Discussion Section

The data we end up receiving based off the feature we chose to use, whether it was to arrange the order of classes, view statistics or exclusions, or something else we wanted to figure out, can be easily seen and interpreted from the usage of our program. We wanted to make sure that whenever students choose the courses that they want in the future, they are able make sure they take all their prerequisites or whatever classes they want to take in order. They are also able to see the exclusions to make sure they don't accidentally waste their time and take a class they don't need or can't receive credit from.

The exclusions are easily seen using our visualization tool in which we can go through and see directly which classes we shouldn't take if we took a class. The statistics page is nice since we can see information about the current classes available at the University of Toronto and that we can read up about how many of each category classes there are. Course search is a good feature since we can see the specific for a course directly and by narrowing down our source, we can see the specifics about a course as well as being able to visualize the exclusions for that's specific course. Lastly, our arrange feature is nice since we can just add a bunch of classes that we want to take or plan to and then have our program automatically sort through it to display the order in which we should take those classes.

Some obstacles we received while writing the program was that we had some errors with scrappy at first, we were able to get it to run one time and get all the data but for some reason it didn't work again and we spent a lot of time working on it to see that we had a file saving issue as well as a bunch of tests when we weren't connected to the internet. Still, we were able to get it to end up working which was nice and we were able to get the program to display the data we wanted it to. We also had some issues working with graphs and setting up directed/undirected graphs since graphs are tricky in general and sometimes it's hard to interpret the visualization data and to check it making sure that it is correct. Still, with some time and a lot of checking we were able to make our read our file into correct graphs for our course data.

If we were given more time and wanted to add more, I think if we were able to get course ratings from the course evaluations website, we would be able to make a section on sorting courses by rating or a way of viewing the rating of the courses. Maybe design something that will rank classes based off rating and difficulty. Lastly, a breadth course recommender based off the classes you've taken, and the category would be something we would want to look into.

From the beginning, our mission has been to give students at the University of Toronto with the resources necessary to make good decisions regarding the classes that they take. Through the integration of features aimed at simplifying course selection, such as insights into prerequisites, exclusions, and class statistics, our program aims to assist in the difficult task of making a schedule into something a little more simple.

In conclusion, our program provides valuable tools for students navigating course selection at the University of Toronto. Through features like arranging class orders, viewing statistics, and identifying exclusions, students can efficiently plan their academic paths. Despite encountering challenges such as errors with web scraping and graph visualization complexities, we overcame these obstacles to deliver a functional program. Looking ahead, integrating course ratings and developing a breadth course recommender could further enhance the utility of our tool, aiding students in making informed decisions about their academic journeys. With continued refinement and expansion, our program stands poised to serve as a valuable resource for students seeking to optimize their university experience.

References

[1] Faculty of Arts I& Science. *2023-24 Academic Calendar*. 2024. URL: `https://artsci.calendar.utoronto.ca/`.

[2] CSC111 Teaching Team. *CSC111 Exercise 3: Graphs and Recommender Systems*. 2024. URL: `https://www.teach.cs.toronto.edu/~csc111h/winter/assignments/ex3-graphs-and-recommendations/handout/`.

[3] UTM. *2023-24 Academic Calendar*. 2024. URL: `https://utm.calendar.utoronto.ca/`.

[4] UTSC. *2023-24 Academic Calendar*. 2024. URL: `https://utsc.calendar.utoronto.ca/`.