Project Documentation

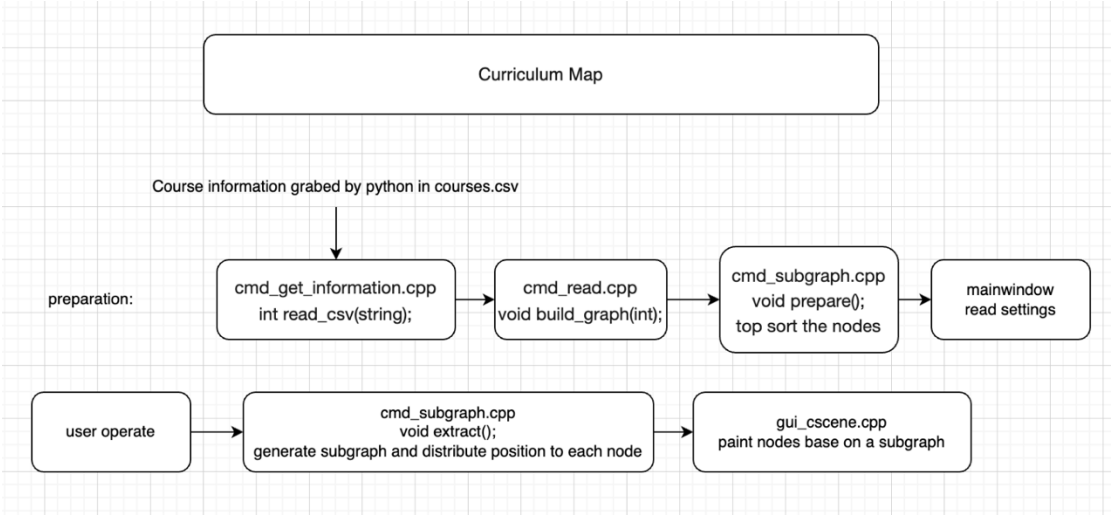
Project Title: Curriculum Map
Project Code: H26
Group Information: XU, Han WANG, Yicheng WANG, Yucheng ZHANG, Yujun

0. Overview

We are committed to providing clear visualization solutions for the intricate curriculum and course relationships of HKUST. As many of the upper-level courses require pre-requisites and even co-requisites, it is very complicated to draft a clear pathway or study plan from the information provided by current text-based websites. Our project aims to solve this difficulty for UST students by offering a highly customized virtualization course map based on users' needs.
The following picture shows the full structure of our project:

```
├── CurriculumMap.pro
├── cmd
│   ├── cmd_Course.h
│   ├── cmd_LogicP.h
│   ├── cmd_Point.h
│   ├── cmd_get_information.cpp
│   ├── cmd_read.cpp
│   ├── cmd_subgraph.cpp
│   └── color.cpp
├── global.cpp
├── global.h
├── gui
│   ├── GraphicsViewer.cpp
│   ├── GraphicsViewer.h
│   ├── gui_arrow.cpp
│   ├── gui_arrow.h
│   ├── gui_cnode.cpp
│   ├── gui_cnode.h
│   ├── gui_cscene.cpp
│   ├── gui_cscene.h
│   ├── gui_node.cpp
│   ├── gui_node.h
│   ├── gui_rnode.cpp
│   ├── gui_rnode.h
│   ├── mainwindow.cpp
│   ├── mainwindow.h
│   └── mainwindow.ui
├── main.cpp
├── res
│   └── style.css
└── resource.qrc
```

A general idea of code realization is shown in the illustration below.



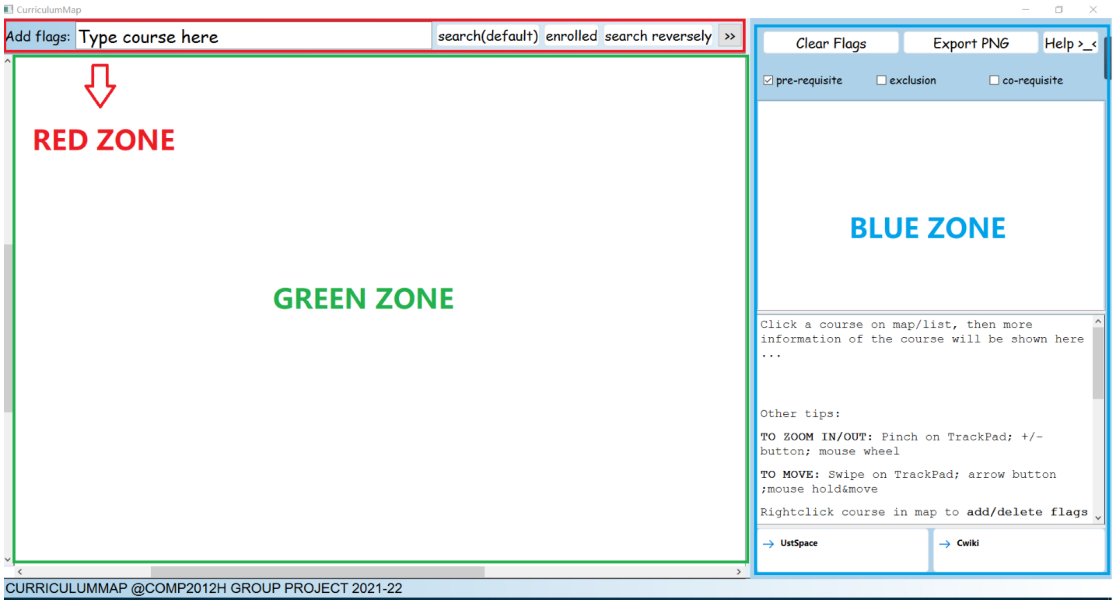
1. Compile and run the program

Notice: Qt edition 6.1.1 and above will be required to compile the code.

We haven't used any third-party library except for Qt library, so just compiling it would be fine.
If you think the courses.csv in get-information folder is old, you could run get-information.py to get the newest one.

2. How to use Curriculum Map

The design of the Curriculum Map aims to provide a user-friendly tool, so the use of the program is quite straightforward. The users can type and search for the courses. The courses will then show on the map with their relative courses, such as pre-requisite courses. Users can then mark this course as enrolled status by clicking the "enrolled" button, check the detailed information of the course by clicking it, or marked the box of "exclusion", "pre-requisite" or "co-requisite" to see the special arrow on the map to enhance these two relations.



The main window is divided into three zones as shown on the above graph.

RED ZONE - Query Area

"Textbox": allows users to type the course name, which is case insensitive and spaces tolerable, e.g., "COMP 2012H" "coMp2012h"

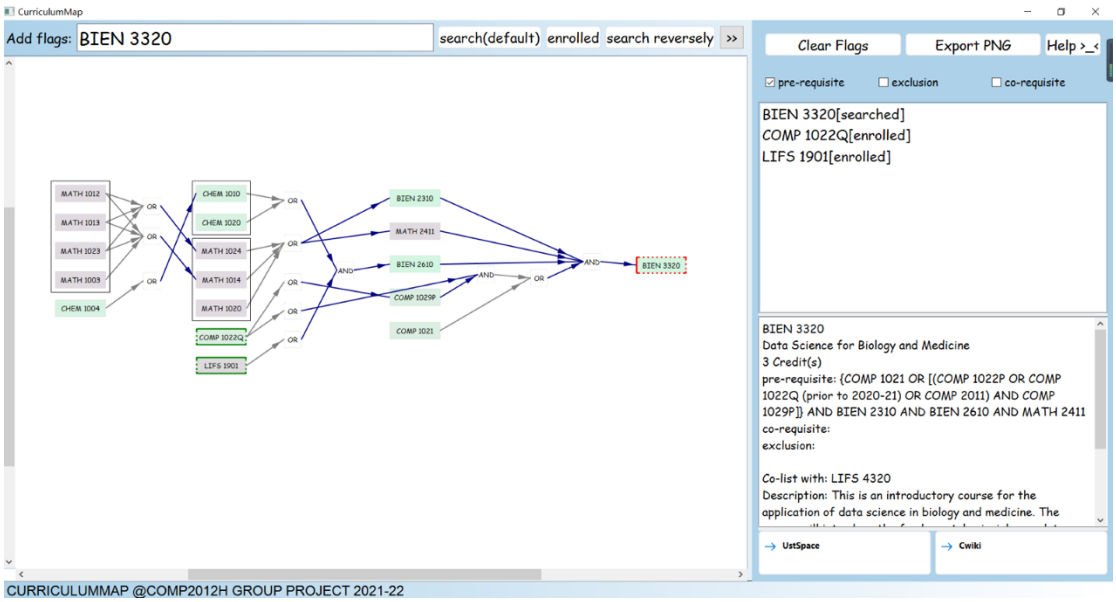
"search" button: enquiry the course and show all the pre-requisites of this course. This is the default operation, which can be also activated by the "Enter" key.

"enrolled" button: state the enrolled course, all the pre-requisites of this course will not be shown in the Curriculum Map to give a better demonstration effect.

"search reversely" button: reversely enquiry the course and show all the courses that require pre-requisites of this course, typically the senior level of courses.

"arrow" button: fold and open the sidebar to give a better demonstration effect on the visualization area

GREEN ZONE - Visualization Area



(a simple example of usage with only function pre-requisite)

The green zone is a Graphicsviewer, self-modified class inherited from QGraphicsView. It is the space for the visualization Curriculum Map. When the course is searched in the textbox, which means that the "search" button is clicked, the course nodes will then be shown on this area as well as its related courses. For example, simply searching BIEN 3320 with COMP1022Q and LIFS 1901 enrolled will be the case shown above.

The zone supports more operations than the normal classes. For example, **zoom in and zoom out** (+/- button, mouse wheel, pinch on the trackpad), **move** (Swipe on the trackpad, arrow button, mouse hold and move) and **scale centred at the mouse cursor.** This will make the project more user-friendly and fit for multiple user situations.

BLUE ZONE - Side widget control button and Information Box

"Clear Flags" button: when it is clicked, the Visualization Area will become blank, and all searching history will be clear.

"Export" button: allow users to export the Curriculum Map to a Png picture file and save it locally on users' computers.

"Help" button: a popup window will appear to kindly remind the user of some basic information and use guide of our project. It is designed for first-time users to get a better interactive experience.

"***pre-requisite" "exclusion"*** and ***"co-requisite"*** button: after clicking, the corresponding arrow will then be shown between courses on the Visualization Area.

"Information Box upper": show the modified history of the courses map.

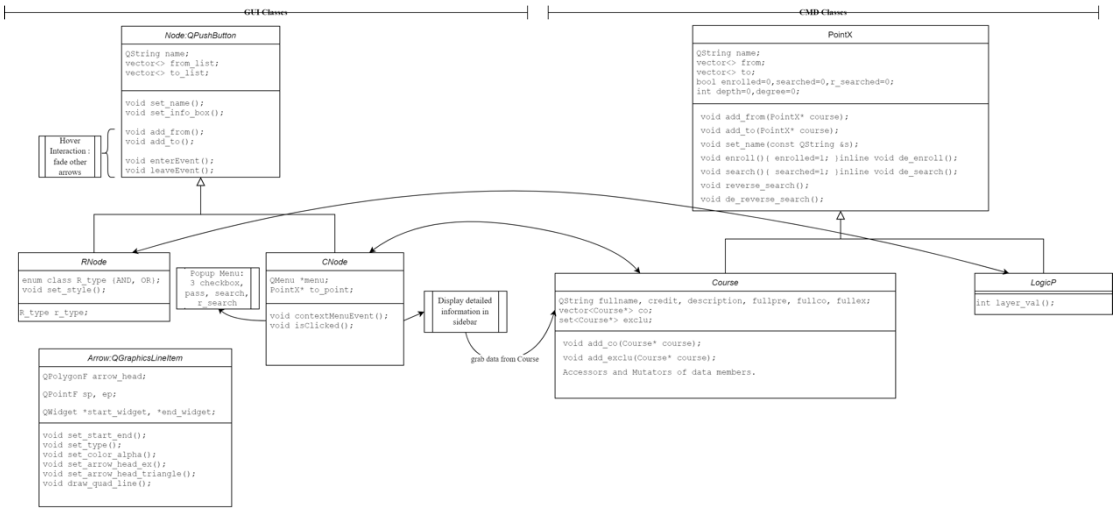
"Information Box lower": give the detailed information of the course once users click the course button (CNode) in the visualization zone.

"external sources" button: "***ustspace"*** and ***"cwiki"***, click to directly link to the outside website for users to further research the courses-related information.

3. OOP constructs and techniques:

We have two sets of classes with similar structures in GUI and CMD, respectively.

The relationships of the classes are virtualized in this map:



I. CMD

- **PointX and its derived classes:**

  Introduction: To save memory and improve readability, all attributes of courses will be stored here.

  Base Class: PointX
  Store graph information, including edges, start from and end at the point, degree, depth, layer. It provides support for interactions that are related to graphs, i.e., three kinds of marks in the context menu. It also stores the common data of its derived classes, like name, visibility.

- **Derived Classes**:Course and LogicP

  Course : store basic information for course

II. GUI

- **Node class and its derived classes**

  Introduction: The 3 classes in GUI are only in charge of visualizing the nodes and supporting interactive functions.

  Base Class: Node
  Inherited from QPushButton. It would maintain the position of the node, as well as arrows from and to it. The improve code reusability, the common part of its derived classes will be implemented here, such as name, fading all edges except those connected with the node directly.

  Derived Class: CNode (stands for Course Node), RNode (stands for Relation Node)

  CNode: grab information from the corresponding class (Course) in CMD, and provide user interactions, like displaying detailed information in the information box in sidebar, popping up context menu to set different flags (consider it as enrolled/passed, query the pre-requisites, "reversely" query those courses requiring it as pre-req) of a course. And the different flags have different highlights (more on .css) on the course.

  RNode: display the operator on it

- **Arrow class:** Inherited from QGraphicsLineItem
  To provide different types of arrows (pre-req with AND relation/OR relation, exclusion, co-req). The color and width of an arrow depend on its type. To decrease the crossing and overlap of lines, when two courses are in the same column, it will become a parabola. To clearly show arrows directly connected with one particular node (course or logical operator), when the mouse is hovering on it, other arrows will be set to a lower opacity to underline indirectly connected arrows.

- **CScene Class (stands for curriculum map scene):** Inherited from QGraphicsScene
  A connector that transports information from CMD to GUI objects during their construction and provides support for user interaction. The pointer to all the objects that occurs in the scene will be stored in the class, including pointers to Node objects and Arrow objects. It will read the

graph contents generated by CMD and construct GUI objects accordingly. It also maintains the click event of a course, creates the right-clicked menu, judge when to update and repaint the graph according to the changes of the menu.

4. Other project designs except OOP

I. Data Structures

- Vector, used to store edges of nodes whose sizes are uncertain;
- Stack, used to deal with brackets matching, improve readability;
- Pointer array, to save memory and gain robustness;
- Map, used to map the pointer to GUI classes' objects with CMD classes' objects and used as a hash to identify equivalent logical nodes;
- Set, used to judge bi-directional exclusion.

II. Reusability

This program is well designed to support further development and reuse.

- For program propose, it could be used until the UST courses map course design is changed. Even then, only a few changes in the input part are needed. The same for providing support to other schools.

- For visualization, it could support any data that logically from a Directed acyclic graph as main part with some extra relationship. Logically equivalent requirement could be merged as a communal point, related data could be shown closely … All that features make it helpful especially for -requisite kind data.

- User interactive surface for graphics viewer is strengthened that could support zoom in/out (+/- button, mouse wheel, pinch on trackpad) and move (swipe on trackpad, arrow button, mouse hold & move) with scale center under mouse. The whole class could be used in any other QT project by directly copying GraphicsViewer.cpp&h, and we will do so definitely in the future.

- For exploring, the pre-requisite network could be advanced as a means to visualize the hidden structure in an academic curriculum and get an insight into the academic course designing concept. We could see how

knowledge is built on through previous studies, rather than just a four-digit number.

III. Subgraph design

Given all points and edges built and flags maintained, `extract()`will compute a subgraph that users care about.

First, based on pre-requisite relationship, we get a Directed acyclic graph(DAG), which could be top-sorted for convenience.

After that, we could get all point that is directly or indirectly a pre-requisite for the searched course. We need to distribute them in the different layers by assigning "layer" variable, so that number of point in each layer will not be too biased. We also hope the logic point(OR/AND) relatively close to course point, so line between them will not be too long and occupy too much space. Some interesting algorithmic techniques are applied. See more in code~

We also account for "enrolled" flags. If a "OR" requirement has been satisfied, a non-passed course in this "OR" 's pre-requisite will not initially show up.

This feature makes the map much more clear when the user input some basic class like MATH 1012/MATH 2111 >.<

Some courses like PHYS 3033-3053, MATH 2111-2121-2131 usually appear in the group (pairwise bidirectionally exclusive), so we will merge them as a whole part in these cases to make the map much more clear ;)

Helpful course number to see more features: ELEC 4830, ELEC 4820, ECON 4394, CENG2210, COMP 3721, COMP 4511 ...

Have fun!

References

[1] P. R. Aldrich, "The curriculum prerequisite network: a tool for visualizing and analyzing academic curricula," 22 AUG 2014. https://arxiv.org/abs/1408.5340
[2] Undergraduate Courses 2021-22. https://prog-crs.ust.hk/ugcourse