Question 2. Course Prerequisites

Please see these slides for a more graphical explanation of this question.

In most post-secondary programs, courses have prerequisites. For example, you cannot take EECS 3101 until you have passed EECS 2011. How can we represent such a system of dependencies?

A natural choice is a directed graph.

- Each vertex represents a course.
- Each directed edge represents a prerequisite
- A directed edge from Course U to Course V means that Course U must be taken before Course V.

We also want to be able to find the information for a particular course quickly. The course number provides a convenient key that can be used to organize course records in a sorted map, implemented as a binary search tree (cf. A3Q1).

Thus it makes sense to represent courses using both a sorted map (for efficient access) and a directed graph (to represent dependencies). By storing a reference to the directed graph vertex for a course in the sorted map, we can efficiently access course dependencies.

It is important that the course prerequisite graph be a directed acyclic graph (DAG), otherwise the dependencies are circular and no one could satisfy them!!

In this question, you are provided with a basic implementation of a system to represent courses and dependencies. This system relies upon the net.datastructures library.

Methods for adding courses and getting prerequisites are provided. You need only write the method for adding a prerequisite. This method will use a depth-first-search algorithm (also provided) that can be used to prevent the addition of prerequisites that introduce cycles.

We will use the TreeMap class to represent the sorted map (cf. A3Q1). We will use the AdjacencyMapGraph class to represent the directed graph. This implementation uses ProbeHashMap, a linear probe hash table, to represent the incoming and outgoing edges for each vertex.

Here is the code you need, extracted from the net.datastructures library. You do not need the whole net.datastructures library, but if you wish to download it you can find it here.

You will modify only the class **Courses** in order to implement the method **putRequisite**, which adds a dependency between two courses.

New: an example of how to use the DFS method provided by the Courses class can be found in the DFS_Complete method of the GraphAlgorithms class.

I have provided a test function **testCourses** which provides simple test cases. However, remember to test your implementation on a broader range of test cases, paying particular attention to boundary conditions.