Weekly Challenge 07: Playing Registrar

CS/CE 412/471 Algorithms: Design and Analysis

Spring 2025

Purpose

In this WC, you will model exam scheduling as a *graph coloring* problem using a *conflict graph*, transforming exam scheduling into a *combinatorial optimization* problem. This reinforces key algorithmic concepts such as graph representation, graph coloring, and combinatorial optimization. The problem closely mirrors real-world scheduling challenges in university timetabling, operating systems, and resource allocation.

Skills

This WC builds your skills in problem-solving and quantitative reasoning, invites you to apply computational thinking to real-world problems, and reinforces your skills related to algorithmic problem-solving and analysis. These align with the university's learning goals, the CS program's learning outcomes, and the course's learning objectives.

Problems similar to the one described in this WC arise in real-world scheduling problems (e.g., timetabling, task scheduling, and resource allocation) and are widely applicable in job scheduling in operating systems, network optimization, and artificial intelligence.

Applying computer science concepts and computational thinking to real world problems in this manner will make you a more capable and versatile computer scientist.

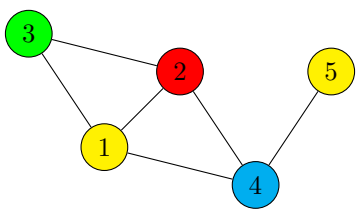
Specifically, this WC develops expertise in:

- Graph Representation and Modeling: building a conflict graph from student enrollment data
- Graph Coloring Algorithms: using graph coloring to minimize exam slots
- Algorithm Design and Justification: developing and implementing an efficient scheduling algorithm
- Data Visualization: using Python libraries to visualize conflict graphs
- Computational Thinking and Randomization: generating realistic random course enrollment data

Background and Requirements

To attempt and submit the WC, you will require:

- An understanding of graphs
- The ability to apply algorithmic thinking to solve a problem
- The ability to program in Python and to read and follow technical tutorials
- Working knowledge of LaTeX



The Setup

A *graph* $G = (V, E)$ consists of V, a nonempty set of *vertices* (or *nodes*), and E, a set of *edges* (or *links*). Each edge is a set of unordered pairs $\{v_1, v_2\}$ of vertices. The vertices v_i and v_2 of an edge $\{v_i, v_2\}$ are called the edge's *endpoints*.

Of course, as a capable student of Algorithms, you already know this! Moreover, Registrar Office (RO) knows that you know this, and is seeking your help to schedule final exams.

You are provided the data of n courses, each of which requires an exam to be scheduled. For each course, i, $1 \leq i \leq n$, you are also provided the data of all the students enrolled in the course. Specifically, you are provided the student IDs of the enrolled students.

RO wants you to schedule exams such that every student can take their exam. That is, every student takes the final exam of every course that they are enrolled in and no student has more than one exam scheduled at any given time. RO also wants you to minimize the total number of time slots in which exams are scheduled.

Employing your algorithmic prowess, you come up with a *conflict graph*. It is a graph containing n nodes, where each node represents an exam to be scheduled. Two nodes have an edge between them iff the corresponding courses *conflict*, i.e., they cannot be scheduled at the same time because there is at least one student who is enrolled in both of them.

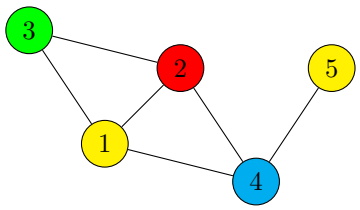
Remembering the saying, "the world isn't just black and white", from your Core courses, you have decided to assign colors to time slots in your conflict graph. All courses whose nodes have the same color in the graph will be scheduled at the same time.

Thus, you have transformed your scheduling task to a *graph coloring* problem! You just have to figure out the minimum number of colors to assign to your nodes such that the endpoints of any given edge have a different color. Feeling smart, happy, and humbled, you make a pilgrimage to Uzbekistan in order to honor Al-Khwarizmi.

Example

Shown below on the left is sample enrollment data of 5 courses, where the i-th line contains the student IDs of the students enrolled in course i. On its right is a resulting conflict graph of the exams. The graph is colored. The graph shows that course 1 conflicts with courses 2, 3, and 4; course 2 conflicts with courses 1, 3, and 4; and so on. A total of 4 time slots (colors) are used. Exams for course 1 and course 5 take place in the same time slot. All of the other exams take place in different, separate time slots.

- S1, S3, S6, S13, S17, S18
- S4, S7, S9, S14, S17, S18
- S1, S2, S6, S7, S12
- S3, S4, S5, S8, S9, S16
- S5, S8, S19, S20



Tasks

1. Given a conflict graph with n nodes, what is the *maximum* number of colors that you can use to color your nodes such that neighboring nodes have different colors? Justify your answer.

Solution: The **maximum number of colors** in a graph depends on it's structure. This occurs when every node has a conflict with every other node (resulting in a complete graph). In a complete graph (denoted as G_c), there are n nodes, and each node is connected to every other node. Each node will require a different colour than the others; hence, we will want exactly n different colours, one for each node. This is the worst-case scenario, in which no tests can be scheduled at the same time because at least one student from each course overlaps with another.

And if we try to colour the graph with fewer than n colours, some adjacent nodes will certainly share the same colour, which violates the requirement that neighbouring nodes be different colours. Hence, the chromatic number of a complete graph is n, in other words the **maximum number of colors required for a conflict graph with n nodes is** n.

2. Of course, RO is interested in the *minimum* number of colors. What is the minimum number of colors that can be used in a conflict graph with n nodes? Justify your answer.

Solution: If no courses have any conflict, the conflict graph is unconnected with no edges, and all tests can be completed in one time slot. As a result, **one colour is sufficient to colour the conflict graph**, with one being the Registrar's minimal number of colours required.

3. Devise an algorithm to assign the minimum number of colors to the nodes in a conflict graph. Provide the algorithm here in CLRS format.

Solution:
Graph-Coloring(G):

```
1:  for each node v in V do
2:      color[v] = −1
3:  end for
4:  Sort vertices in V by degree in descending order.
5:  for each node v in V do
6:      available[] = false
7:      for each neighbor u in Neighbors(v) do
8:          if color[u] ≠ −1 then
9:              available[color[u]] = true
10:         end if
11:     end for
12:     c = 0
13:     while available[c] == true do
14:         c = c + 1
15:     end while
16:     color[v] = c
17: end for
18: return color[]
```

Neighbors(v):

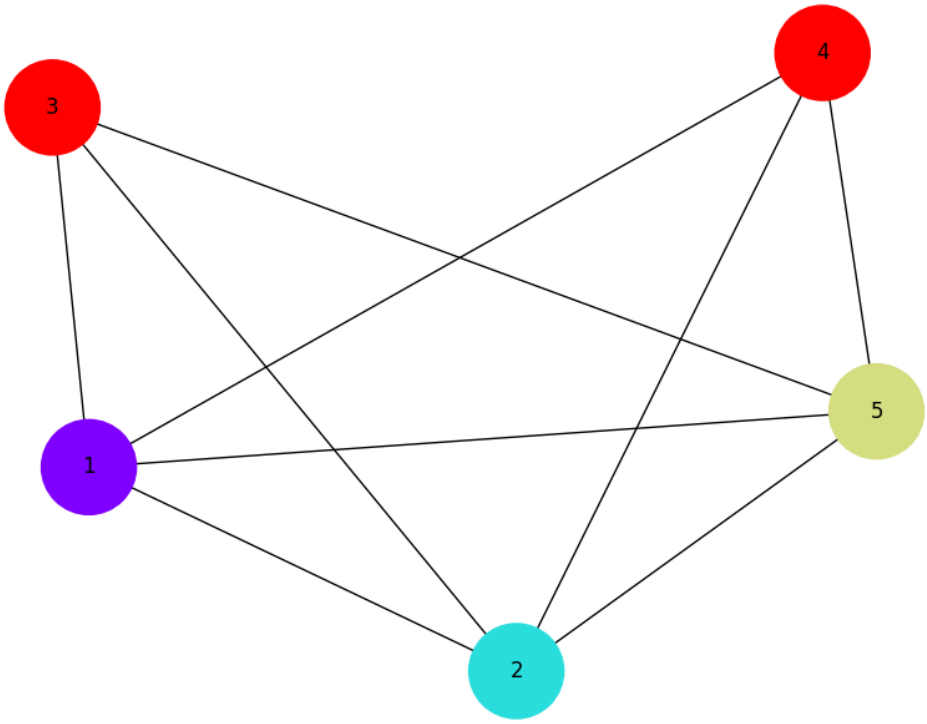
```
1:  neighbors = []
2:  for each edge (v, u) in E
3:      if v == edge[0]
4:          [len(neighbours) :] = edge[1]
5:      else if v == edge[1]
6:          [len(neighbours) :] = edge[0]
7:  end for
8:  return neighbors
```

4. Provide below, sample enrollment data for a desired number, n, of courses. The i-th line, $1 \leq i \leq n$, contains the student IDs of the students enrolled in course i. Ensure a good amount of cross-enrollment, i.e., students enrolled in multiple and overlapping courses.

Solution:
Course 1: S1, S3, S5, S7, S9
Course 2: S2, S3, S4, S6, S7
Course 3: S1, S2, S5, S8
Course 4: S4, S6, S7, S9
Course 5: S2, S3, S5, S6, S8

5. Include below a visualization of your conflict graph which is colored according to your algorithm above. Make sure that each node is labeled with its course ID.

Solution:



6. (Optional, no points) Include below a picture or drawing of you honoring Al-Khwarizmi. The picture may be hand drawn. Al-Khwarizmi would probably feel more honored if it were algorithmically generated.

Solution:

Hints

- Use the `networkx` package in Python to model a graph and the `matplotlib` package to visualize it. You can find related instructions here.
- Instructions on adding color to the visualization are available here.

Submission

Fill in the solution boxes above and submit the generated PDF file.