Daily Coding Problem #92

Problem

This problem was asked by Airbnb.

We're given a hashmap associating each `courseId` key with a list of `courseIds` values, which represents that the prerequisites of `courseId` are `courseIds`. Return a sorted ordering of courses such that we can finish all courses.

Return null if there is no such ordering.

For example, given {'CSC300': ['CSC100', 'CSC200'], 'CSC200': ['CSC100'], 'CSC100': []}, should return ['CSC100', 'CSC200', 'CSCS300'].

Solution

This is a classic topological sorting question. One way to think about this problem is to think about how would you go

about solving it manually? We can divide it into these two steps:

1. Put all courses with no pre-requisites into our todo list.
2. For each course C in the todo list, find each course D which have C as a prerequisite and remove C from its list. Add D to the todo list.

If in the end we couldn't take some courses, this means that were was a circular dependency.

```
def courses_to_take(course_to_prereqs):
    # Copy list values into a set for faster removal.
    course_to_prereqs = {c: set(p) for c, p in course_to_prereqs.items()}

    todo = [c for c, p in course_to_prereqs.items() if not p]

    # Used to find courses D which have C as a prerequiste
    prereq_to_coures = {}
    for course in course_to_prereqs:
        for prereq in course_to_prereqs[course]:
            if prereq not in prereq_to_coures:
                prereq_to_coures[prereq] = []

            prereq_to_coures[prereq].append(course)

    result = [] # courses we need to take in order

    while todo:
        prereq = todo.pop()
        result.append(prereq)

        # Find which courses are now free to take
```

```
    for c in prereq_to_coures.get(prereq, []):
        course_to_prereqs[c].remove(prereq)
        if not course_to_prereqs[c]:
            todo.append(c)

# Cicrcular dependency
if len(result) < len(course_to_prereqs):
    return None
return result
```