PA9: Tofu Factory

This programming assignment is to be completed in either Java or Python (your choice) and submitted on Gradescope.

Collaboration is restricted to just 2 people (other than yourself), and it must be "Whiteboard Only". In summary, this means that you may discuss the assignment with 2 others, but you cannot preserve any artifacts (digital or physical) from these discussions. In particular, this means you may not take and keep photos, notes, video, audio recordings, whiteboard contents, etc. from any discussions.

All collaborators and other resources consulted should be listed as a comment at the top of your code submission. Course staff and official course materials do not need to be listed.

Motivation

We are trying to help a tofu factory schedule shifts among its workers. The factory has divided its manufacturing process into four steps:

1. **Blend**: Take soaked soy beans, wash them, add water, then blend until smooth.
2. **Cook**: Simmer the mixture, stirring constantly.
3. **Strain**: Strain out the soy bean solids using a cloth.
4. **Finish**: Add nigari (a coagulant derived from seawater) to the remaining liquid. Add the formed curds to a mold.

Each of these steps requires a specific number of workers per shift, and these workers must be trained in that particular task. For this assignment you will be given the workers' details and the shift schedule and you must determine whether it is possible for the factory to staff all of its shifts.

You solution should operate by using the input to create a flow network, then using the provided maxflow code in order to find the maxflow of the graph, and finally use the maxflow response to derive the final answer. In other words, your code should perform a reduction from this problem to maxflow.

Problem Description

Your algorithm will be given input in two components. The first is what we will refer to as the "worker details". This input consists of a list of employees, where for each employee you have:

– The stations that this employee is trained on (a list consisting of a subset of "blend", "cook", "strain", and "finish")

– The times the employee is available for working an hour-long shift (a list of times between 0 and 23)

The second part of the input is going to be the "shift schedule". The shift schedule has a list stations ("blend", "cook", "strain", and "finish"). For each station it provides:

– The number of people needed to staff that station per shift (in integer)

– A list of shifts that must be covered (a list of times between 0 and 23)

Your algorithm should determine whether it is possible to fully staff all of the stations for each shift. That is, it must determine whether it is posssible to:

1. assign employees to shifts so that
2. no employee is assigned to two shifts at the same time,
3. each employee is only assigned to stations that they have been trained on,
4. every shift for every station is fully staffed (that is the required number of employees are assigned to each of the shifts)

Your algorithm should return a boolean indicating whether or not it is possible to satisfy all of these requirements. Note that this list of requirements is exhaustive. Do not assume any other requirements (e.g. note that we do not require every employee have at least 1 shift, that is intentional, your algorithm should not depend on that or anything else not mentioned).

You can make a completely correct submission that involves just adding nodes/edges to a graph (using the graph class provided), invoking maxflow (also using the graph class provided), then choosing to answer True/False depending on the amount of flow found.

Input Format

Input will be given in the console (same as with the previous PAs). To provide an input with n employees the input will have $2n + 10$ total lines as follows:

– The first 8 lines will give the shift schedule. in the following way:

  – Line 1 has the word "blend" to indicate that this is the shift schedule for the blend station, then a space, then an integer representing the number of employees who must be assigned to each shift of the blend station

  - Line 2 contains integers representing the start times of the hour-long shifts for the blend station separated by spaces
  - Lines 3 and 4 are the same but for the cook station
  - Lines 5 and 6 give informtion for the strain station
  - Lines 7 and 8 give information for the finish station
- Line 9 has the word "employees" and then gives the integer n, i.e. the total number of employees
- The next $2n$ lines give the information for each of the n employees. Each employee's information is provided on two lines:
  - The first line contains space-separated strings representing the stations on which that employee is trained.
  - the second line contains space-separated integers representing the times that this employee is available to work shifts.
- The last line is just the word "done", indicating the end of the input.

Here is an example input for 4 employees. The answer for this input should be True.

```
blend 1
0 1 2 3
cook 1
1 3
strain 2
4 5
finish 1
5 6
employees 4
blend cook
0 1 2 3
strain
2 4 5
strain finish
4 5 6
blend finish
1 3 5
done
```

For this input:

- Blending requires 1 employee per shift and shifts occur at 00:00, 01:00, 02:00, and 03:00
- Cooking requires 1 employee per shift and shifts occur at 01:00, and 03:00
- Straining requires 2 employees per shift and shifts occur at 04:00, and 05:00
- Finishing requires 1 employee per shift and shifts occur at 01:00, and 03:00
- Employee 0 is trained in blending and cooking and can shifts at 00:00, 01:00, 02:00, and 03:00

– Employee 1 is trained in straining and can do shifts at 02:00, 04:00, and 05:00

– Employee 2 is trained in finishing and straining and can do shifts at 04:00, 05:00, and 06:00

– Employee 3 is trained in blending and finishing and can do shifts at 01:00, 03:00, and 05:00

Output Format

Your program should return a single boolean indicating whether or not the given employees can fully-staff the given shift schedule.

Test Cases Summary

We provide you only with the test case provided above. The autograder will test your code against an additional 11 test cases that you will not have access to (but you will see the outcomes for all of them). Since the answer is a boolean, and exactly half of the tests have a correct answer of each of True and False, your grade will be determined by the number of tests over 6 that you get correct. (i.e. submitting code that always answers false would get 6 tests correct, so you have to get more than 6 tests correct before you will earn a positive score).

Starter Code Summary

The starter code provided consists of two files:

– `Flownetwork.java/flownetwork.py`: You will not submit this program, instead the autograder will be using its own version to interact with your code. This program implements a flownetwork class. Below we describe the most important methods for you to be aware of in the the class. You should not need to use any of the methods other than these.

  – `Constructor`: To construct a FlowNetwork object you should provide just the node that will be the source and the node that will be the sink. In Python nodes may have any immutable type as the label (int, string, float, tuple, etc.). In Java the nodes are labelled with Strings.

  – `add_node`: This allows you to add a node to the flownetwork. simply give the name of the node you would like to add

  – `add_edge`: This allows you to add an edge to the flownetwork. The parameters are the start node of the edge, the destination node of the edge, and the capacity of the edge. If either endpoint of the node is not in the graph then it will be added to the graph automatically.

  – `maxflow`: This implements the Edmunds-Karp maxflow algorithm. Its return value in an integer representing the total flow that was assigned through the graph.

  – `str/toString()`: A string representation of the flow network has been provided (that is, you can cast a flownetwork to a string in python and use the toString method in java). The string produced represents one edge per line and shows the start node for that edge, the destination node, the currently-assigned flow across that edge, and the capacity of that edge.

– `Scheduler.java/scheduler.py`: This is the file you will edit and submit. There is one function defined in this file called `schedule`. The input parameters to this function collectively contain all information in the shift schedule and employee information. There is a comment in the function describing the input parameter format.

– `InputReader.java/input_reader.py`: This is the program you will use to run your code. The purpose of this is to read the input provided in the described format, parse that input, and then invoke your scheduler code. You should not have to change anything here.