Description

It is the final year for a cohort of students, and they all want to choose an interesting project to study. We want to know if it's possible to allocate the students projects and supervisors such that no project is or supervisor is overloaded.

Notation

- There is a set P of l projects (labeled 0, 1, 2, etc.)
- There is a set T of m teachers (labeled 0, 1, 2, etc.)
- There is a set S of n students (labeled 0, 1, 2, etc.)
- Each project i can be studied by at most $p_i \in \mathbb{Z}^+$ students.
- Each teacher i is willing to supervise at most $s_i \in \mathbb{Z}^+$ students.
- Each teacher i is willing to supervise students studying a subset $t_i \subseteq P$ of the projects.
- Each student i is willing to studying any project from a subset $w_i \subseteq P$ of the projects.

Example

Suppose there were:

- $l = 3$ projects $P = \{0, 1, 2\}$
  - Project 0 can have up to $p_0 = 2$ students studying it.
  - Project 1 can have up to $p_1 = 3$ students studying it.
  - Project 2 can have up to $p_2 = 1$ student studying it.
- $m = 2$ teachers $T = \{0, 1\}$
  - Teacher 0 is willing to teach up to $s_0 = 3$ students, studying projects $t_0 = \{0, 1\}$
  - Teacher 1 is willing to teach up to $s_1 = 2$ students, studying projects $t_1 = \{1, 2\}$
- 5 students
  - Student 0 is interested in projects $w_0 = \{0, 1, 2\}$
  - Student 1 is interested in project $w_1 = \{0\}$
  - Student 2 is interested in projects $w_2 = \{1, 2\}$
  - Student 3 is interested in projects $w_3 = \{0, 2\}$
  - Student 4 is interested in projects $w_4 = \{0, 1\}$

The answer in this case would be YES, because it is possible to match everything up. For example:

- Allocate student 0 to project 0 and teacher 0
- Allocate student 1 to project 0 and teacher 0
- Allocate student 2 to project 1 and teacher 0
- Allocate student 3 to project 2 and teacher 1
- Allocate student 4 to project 1 and teacher 1

Consider the following decision problems[1]:

1. Student Projects:

   "Given the constraints described above, is it possible to allocate projects such that every student has a project to study on?"

2. Min-cut decision:

   "Given a flow network G and an integer d, is the capacity of the min-cut of G at least d?"

Task 1: [30 marks]

Design an algorithm to solve problem (1) by reducing it to problem (2). Your algorithm should have the following structure:

- "Transform the input"

  Transform the input of the Student Projects problem into input for the min-cut decision problem. i.e. a flow network G and integer d.

- "Solve the transformed problem"

  Efficiently solve the min-cut decision problem with input (G, d)

- "Transform the output"

  Use the output of the min-cut decision problem to determine the answer to the Student Projects problem

Task 2: [30 marks]

Prove that this reduction works. Specifically you should prove the following two theorems:

$\Rightarrow$ "If the answer to the min-cut decision problem is YES, then the answer to the Student Projects decision problem must also be YES."

$\Leftarrow$ "If the answer to the Student Projects problem is YES, then the answer to the min-cut decision problem must also be YES."

Task 3: [20 marks]

Reason about the time complexity of your algorithm. Don't forget to consider:

- How long does it take to transform the input?
- How long does it take to solve the transformed problem?
- How long does it take to transform the output?

Make sure your answer is given in terms of the input data to the Student Projects problem.

[1] decision problems are problems which always have a YES or NO answer

Task 4: [20 marks]

Implement the algorithm (on Ed), in the programming language of your choice.

Your program will read the input as a series of lines sent as standard input (i.e. as if the user had typed them at the console)

- The first line is l, the number of projects
- The second line is l integers representing p_k (max students per project).
- The third line is m, the number of teachers
- The fourth line is m integers representing t_j (max students per teacher).
- The next m lines each contain up to l integers, representing the set of projects the corresponding teacher is interested in
- The next line is n, the number of students
- The remaining n lines each contain up to l integers, representing the set of projects the corresponding student is interested in

The original example would look like this:

```
3
2 3 1
2
3 2
0 1
1 2
5
0 1 2
0
1 2
0 2
0 1
```

The output should simply be the word YES or NO. For the example, it would be

```
YES
```

Some skeleton code is provided in Python which reads the input for you. You can use this code, or start from scratch in this or another language if you prefer.

For your convenience the skeleton code will also include access to an implementation of Ford-Fulkerson. You are also welcome to use an implementation of Ford-Fulkerson you have found elsewhere (but you must appropriately reference your source, and ideally should keep it in a separate source file to your own code.)

Please note that the test cases will not cover any weird edge cases around the input format itself (i.e. the input will always be valid, have the expected number of lines, all the numbers will be positive integers, with the specified whitespace etc.)

There will be limits on both *time* and *space* used by your algorithm. So try to keep your implementation reasonably efficient in both. If you get the "killed" message in Ed, it probably means your submission used too much memory – check with a tutor if you're in doubt of what caused this particular error message. If your submission used too much time, or gave the wrong answer, you should get a clearer feedback message from Ed.

Submission details

- Submission deadline is Sunday 27th May, at 23:59pm. Late submissions without special consideration will be subject to the penalties specified in the first lecture (25% per day or part thereof.)
- Submit your answers to Tasks 1 – 3 as a single document (preferably a pdf or doc file) to Canvas. Your work must be typed text (no images of text, although you can use diagrams if you think it helps.) Please try to be reasonably concise.
- Submit your code for Task 4 to `edstem.org`
- Both your code and report will be subject to automatic and manual plagiarism detection systems. Remember, it's acceptable to discuss high level ideas with your peers, but you should not share the detail of your work, such as (but not limited to) parts of the actual algorithms, (counter)examples, proofs, writing, or code.

Additional question for COMP3927 students only: [40 marks]

This question is for COMP3927 students only. The answer to this question should be submitted separately to the rest of your report (there will be a separate submission point for it on canvas.)

Assume that we are given a machine M and n jobs $1, \ldots, n$ that are specified by their length ℓ_j and their weight w_j, $1 \leq j \leq n$. Precedence constraints are given by a partial order of jobs; if job i precedes job j in the partial order, denoted by $i \to j$, then i must be processed before j. The goal is to schedule the jobs on M such that the total completion time $\sum_{j=1}^{n} w_j \cdot C_j$ is minimized; here C_j is the time at which the job j is completed in the given schedule. A brute force algorithm would try all $O(n!)$ permutations of the jobs. Your task is to design a more efficient algorithm.