CS 170 Efficient Algorithms and Intractable Problems
Fall 2020 U. Vazirani and A. Tal Version 1

Problem Statement

2020 has been a stressful year, and CS 170 course staff is trying to reduce student stress and increase student happiness as much as possible. Since school has been reduced to a series of awkward Zoom breakout rooms, we figured this is a good place to start. We noticed that student stress and happiness fluctuate greatly depending on how they are split into breakout rooms, so we are looking to find a way to divide up stressed 170 students to make them a little happier. However, this sounds like a really difficult task, so we decided to outsource this to you.

Your job is to place n students into Zoom breakout rooms. For each pair of students i and j, there is one value h_{ij} quantifying how much happiness these two students give each other and one value s_{ij} quantifying how much stress they give each other. The total happiness value of a room H_{room} is the sum of the happiness values h_{ij} of every student pair in that room, and the total stress value of a room S_{room} is the sum of the stress values s_{ij} of every student pair in that room. Knowing that trying to eliminate student stress is impossible, we have settled with keeping total student stress low enough so that it does not surpass S_{max}/k in each room, where k is the number of breakout rooms you choose to open. Your goal is to maximize total happiness H_{total} across all rooms, while keeping the total stress below the threshold S_{max}/k in each room.

With notation:

Input parameters:

- $n =$ Number of students in the class
- $h_{ij} =$ Happiness student i and j give each other
- $s_{ij} =$ Stress student i and j induce on each other
- $S_{max} =$ Maximum total stress across all breakout rooms

Output values:

- $H_r =$ Happiness of breakout room r
- $S_r =$ Stress of breakout room r
- $H_{total} =$ Total happiness across all breakout rooms
- $k =$ Total number of breakout rooms opened

If students $0, 1, 2, ..., m$ were placed in room r, then:

- $H_r = \sum_{i,j \in \{0,1,...,m\}} h_{ij}$
- $S_r = \sum_{i,j \in \{0,1,...,m\}} s_{ij}$

$H_{total} = \sum_{r \in [k]} H_r$

1 Your Tasks

1.1 Phase 0: Team Selection (Due 11/17, 11:59pm)

Please fill out this form by November 17th, 11:59pm.

- Only **one** member from each group should fill out this form. If multiple members fill it out, there may be issues with registering your group.
- You must submit this form even if you are working alone.

1.2 Phase 1: Input Phase (Due 11/24, 11:59pm)

Once your team has been formed, work with your teammates to create three hard inputs to the problem, and a solution to each. You are not required to guarantee that your solution is optimal, but the performance of other teams' solvers will be compared against your solution to test hardness of your input, so for more points you should ensure that a higher-quality solution is unlikely to exist. It is best if finding your solution to the problem, or a similar-quality solution, is very difficult, as a small part of your grade will be the hardness of your inputs (and since the solver phase is competitive, it is in your best interest to create inputs which other teams will find difficult to solve). You'll submit your three inputs and their solutions to gradescope in the format described below.

In order to receive credit for this portion of the project, your input and output files should be formatted correctly (as described below), and the solutions described by your output files should be feasible for the corresponding input. **Only one member of your team should submit, and that member must add the other members on Gradescope.**

In order to get out a complete list of inputs to students in a timely fashion, there will be **NO late submissions**. As with the homeworks, **our official policy is that if you can submit your inputs to Gradescope, then your submission is considered to be submitted on time. Anything that is submitted past the deadline will not be accepted.**

1.3 Phase 2: Solver Phase (Due 12/7, 11:59pm)

Overview

You will be provided the pool of inputs generated by the class categorized by their size. Design and implement an algorithm and run it on the entire pool of input files. You will **submit your output for every input provided** in the format described below to the solutions assignment on gradescope. The autograder will check whether your solution is feasible and what the objective value is, and you'll be given a score based on how well you did compared to the solution submitted with each input, on average. Choose a team name; it'll be used for an anonymous leaderboard showing the scores given to each team. **Please keep team names civil and PG-13 and no more than 30 characters long**. Your grade for this phase will be determined in part by how your solver performs compared to those of other students. In addition to your outputs, you will **write a reflection** on the approaches you took and how they performed.

We have released starter code (see Piazza) to parse inputs and check outputs (you do not have to use the starter code). You may use any programming language you wish, as long as your input and output files follow our specified format. However, we cannot guarantee that staff will be able to help you with usage of languages and libraries outside of Python and NetworkX.

Services, Libraries, Tools

If you use non-standard libraries or computing resources beyond your personal computer, **you may only use free services and tools**. Services and tools which are normally paid are okay if used under free, easily available academic licenses. This includes things like free AWS credits for students. If you use AWS or any other cloud computing

service, you must cite how you used it, and how much, in your project report. If you use any non-standard libraries, you need to explicitly cite which you used, and describe in your reflection document why you chose to use them. If you choose to use the instructional machines[1], **you may only use one at a time per team**. We will be strict about enforcing this; there will be a Google form for students to self-report anyone that they see using multiple instructional machines. **Anybody caught using multiple instructional machines will receive a zero for this part of the project**.

The reason for this rule is that CS 170 students in the past have overloaded the instructional machines the week before the project is due, and this makes them unavailable to other students. We want to make sure that you are not inconveniencing other students with your project work.

Reflection

Your reflection should address the following questions:

- Describe (briefly and informally) the approach you used to generate your inputs, and what other approaches (if any) you considered or would like to consider if given more time.
- Describe the algorithm you used for your solver. Why do you think it is a good approach?
- What other approaches did you try? How did they perform?
- What computational resources did you use? (e.g. AWS, instructional machines, etc.)
- If given more time, what else would you try to improve your approach?

Your reflection should be **no more than 1000 words** long, although it may be shorter as long as you respond to the questions above.

We strongly suggest you start working on Phase 2 early. In fact we recommend that students to begin working on working on solvers *before* the Phase 1 deadline. Among other benefits, this will allow you to test whether your inputs are actually hard before submitting them.

2 Inputs Format

You will submit three different inputs: one with 10, one with 20, and one with 50 students. The name of the file is the input size, and should have the extension ".in". For example, for an input size of 50 students, the input file should be "50.in".

Each input should be formatted as follows:

- Each input should start with a single integer equal to n, the number of students.
- Second line should be a single number equal to S_{max}, the total stress budget. S_{max} must be greater than 0, less than 100, and have at most 3 decimal places.
- Each following line should be a space-separated list of 4 numbers: i j h_{ij} s_{ij}. Students i and j are represented as integers ranging from 0 to $n-1$. The happiness value h_{ij} and stress value s_{ij} must be at least 0, less than 100, and have at most 3 decimal places.
- All students give themselves zero happiness and zero stress, and pairs are symmetric, so include all and only those lines where $i < j$.

[1] *Note on accessing instructional machines:* to use the instructional machines, first access EECS Acropolis and get your account credentials (e.g. `cs170-abc`). Once you have your account, SSH into one of the instructional machines listed on Hivemind (e.g. `ssh cs170-abc@asbhy.cs.berkeley.edu`). You should now have terminal access to your instructional account.

Sample Input:

```
4
42.314
0 1 9.2 3
0 2 5.4 40.8
0 3 2.123 98
1 2 75.4 8
1 3 18 57.904
2 3 87 9.4
```

3 Output Format

The output file corresponding to an input file must have the same name, except with the extension ".in" replaced by ".out". For example, the output file for "50.in" must be named "50.out".

The output should be formatted as follows:

- There is one line per student. If there are 50 students, there should be 50 lines in the output file.
- Each line should have two space-separated integers: *i r*. This denotes that student *i* is to be placed in room *r*.

Sample Output:

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

4 Grading

Overall, this project is worth 5% of your final grade. You will earn these points as follows:

- 0.5% will come from validity of your inputs.
- 1% will come from quality of your inputs.
- 1% will come from your reflection.
- 2.5% will come from the validity and quality of your outputs.

We will release more details about how outputs are scored closer to the release of **Phase 2**.

We encourage students to create the best solver they possibly can, and as staff we love to see a healthy competitive spirit in project groups. However, we'd like to emphasize that **the point of this project is not to be purely an evaluation of your abilities against those of your peers**. We really want to encourage students to view this as an opportunity to apply what they've learned over the semester to an open-ended project in an exploratory way. Do your best, try approaches that sound interesting to you, and have fun!

We will not accept late submissions. Submissions made after the deadline will not be considered.

5 Academic Honesty

Here are some rules and guidelines to keep in mind while doing the project:

1. No sharing of any files (input files or output files), in any form.
2. No sharing of code, in any form.
3. Informing others about available libraries is encouraged in the spirit of the project. You are encouraged to do this openly, on Piazza.
4. Informing others about available research papers that are potentially relevant is encouraged in the spirit of the project. You are encouraged to do this openly, on Piazza.
5. Informing others about possible reductions is fine, but treat it like a homework question – you are not to give any substantial guidance on how to actually think about formulating the reduction to anyone not in your team.
6. As a general rule of thumb: don't discuss things in such detail that after the discussion, the other teams write code that produces effectively the same outputs as you. This should be a general guideline about how deep your discussions should be.
7. If in doubt, ask us. Ignorance is not an excuse for over-collaborating.

If you observe a team cheating, or have any reason to suspect someone is not playing by the rules, please report it here.

As a final note from the staff, we generally trust that students will make the right decisions when it comes to academic honesty, and can distinguish collaboration from cheating. However, we'd like to point out that what has made this project so interesting in the past is the diversity of student approaches to a single problem. We could have elected to give you yet another problem set, but we believe that the open-ended nature of this project is a valuable experience to have. Do not deprive yourselves (or us) of this by over-collaborating or simply taking the same approaches you find your peers using. Again, try your best and have fun!