CS 170 Efficient Algorithms and Intractable Problems
Spring 2020 A. Chiesa and J. Nelson Version 1.1

Problem Statement

You are an engineer at Horizon Wireless, a telecommunications company, and the CEO has tasked you with designing a cell tower network which spans the major cities of the United States. You need to decide which cities to build cell towers in and lay out the fiber network between cities. There are a few factors to consider:

1. You would like each major city in the United States to have access to your network. This means that for each city, there must be a cell tower either in that city or in at least one neighboring city.
2. You need to connect the cell towers with fiber cables. Due to government regulations, each city that your fiber network passes through must have a cell tower. The CEO wants to save money, so he tells you that your fiber network must be a tree.
3. When transmitting a signal between two cell towers, the signal must be sent through the fiber cables. Each cell tower expects to communicate with every other tower efficiently. When designing the network, you want to minimize the average pairwise distance between cell towers along the fiber cables.

Find the placement of the cell towers satisfying conditions (1) and (2) which minimizes the average pairwise distance between towers.

Formally, let $G = (V, E)$ be a weighted, connected, undirected graph. We would like to find a subgraph T of G such that:

1. Every vertex $v \in V$ is either in T or adjacent to a vertex in T.
2. T is a tree.
3. The average pairwise distance between all vertices in T is minimized.

Your cost, which you want to minimize, is

$$\frac{1}{2\binom{|T|}{2}} \sum_{(u,v) \in T \times T} d(u,v)$$

where $d(u,v)$ is the distance between vertices u and v in T. If the tree has only one vertex, then the cost is 0.

Input Format

You will submit three different graphs of sizes up to 25, 50, and 100 vertices as inputs.

Each graph should be formatted as follows:

- The first line should contain a single positive integer equal to $|V|$, the number of vertices of the graph.
- Each following line should be formatted as a space-separated list of three numbers "u v $d(u,v)$" defining an edge of the graph. u and v are integers between 0 and $|V| - 1$ corresponding to vertices and $d(u,v)$ is a floating point number corresponding to the weight of the edge. $d(u,v)$ must be greater than 0, less than 100, and have at most 3 decimal places.

Sample input:

```
5
0 1 0.25
1 2 0.5
2 0 0.75
2 3 1.5
3 4 0.25
```

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".

Your output must be a subgraph of the input graph satisfying conditions (1) and (2) in the problem statement formatted as follows:

- The first line should contain a space-separated list of integers corresponding to the vertices in your chosen subgraph.
- Each following line should be formatted as two space-separated integers "*u v*" corresponding to an edge of the input graph.

Sample Output:

```
2 3 4
2 3
3 4
```

Phase 0 (Due April 12th, 11:59 pm)

Overview

You are allowed to form groups of up to 3 people for the project. This is a hard limit. You must stick with the same group throughout the entire project. You may also work alone or in a group of 2.

Submission Details

Please fill out this form by April 12th, 11:59pm.

- Only **one** member from each group should fill out this form. If multiple members will it out, there may be issues with registering your group.
- After April 12th, **you cannot change your group for any reason**.
- You must submit this form even if you are working alone.

Phase 1 (Due April 19th, 11:59 pm)

Overview

You will submit three input graphs of different sizes.

You must submit exactly 1 input for each of the following size categories:

- Small: up to 25 vertices
- Medium: up to 50 vertices
- Large: up to 100 vertices

We will collect everyone's input files and release them after Phase 1 is due. You will receive full points for your inputs as long as they follow the input format and submission guidelines. In Phase 2, you will be graded based on how well your solver performs compared your classmates' solvers on these inputs, so it is in your favor to construct difficult/tricky inputs.

Submission Details

The three input files you submit should be named 25.in, 50.in, and 100.in. If your files do not satisfy these requirements, or if your input is invalid, you will not receive any credit for this portion of the project. **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**. We believe two weeks to be more than enough time to come up with 3 inputs for the project, and would like to give as much time as possible with the final list of inputs for writing solvers. 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.**

Phase 2 (Due May 3rd, 11:59 pm)

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**. 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 (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 must be able to replicate your results by running your code. Furthermore, we cannot guarantee that staff will be able to help you with usage of languages and libraries outside of Python and NetworkX.

Services

You may only use free services (academic licenses included). This includes things like free AWS credits for students. If you use AWS or any other cloud computing service, you must cite exactly how many hours you used it for in your project report. We should be able to replicate the results that you cite. If you use any non-standard libraries, you need to explicitly cite which you used, and explain why you chose to use those libraries. 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**.

[1] *Note on accesing 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.

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.

Submission

Place your output files in a zip folder and submit on Gradescope. For example, the output file for the input named "50.in" should be named "50.out". Please make sure you **include your teammates in your Gradescope submission**. The auto-grader will score your output files immediately (although it may take a while to run).

We will maintain a leaderboard on Gradescope showing how well your solution performs compared to those of the rest of the class. You will enter a team name when you submit to Gradescope, and this name will be used to display your team's results. **Please keep team names civil and PG-13 and no more than 30 characters long**. We will not tolerate any inappropriate team names. **Also note that the leaderboard is only an approximation of your final grade.**

You will also submit your code as well as a project reflection in Phase 2. **Your reflection should address the following questions:**

- Describe the algorithm you used to generate your outputs. 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.)

Your reflection should be **no more than 1000 words** long, although it may be shorter as long as you respond to the questions above. You will also submit the code for your solver, along with a README containing precise instructions on how to run it. If we cannot parse your instructions then you run the risk of receiving a penalty to your overall grade. We should be able to replicate all your results using the code you submit as well as your project report. More details on how to submit your code and project report will be released closer to the Phase 2 deadline.

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.

Grading

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

- 1% will come from your inputs.
- 1% will come from your reflection.
- 3% will come from the the quality of your outputs.

We will release specific 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.

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!