Problem Statement

The "Cannonball Run" challenge is a speed record inspired by the 1970s Cannonball "Sea-to-Shining-Sea" Dash, where competitors attempted to make it from New York City to Los Angeles as quickly as possible. Recent decrease in road traffic has lead to an uptick of interest in the challenge; the record was beaten 3 times in 2020 alone, with the current record standing at 25 hours and 39 minutes.[1][2]

The Federal Highway Administration (FHA) has grown tired of so many people attempting the reckless challenge and has decided to do something about it. However, as it would be impractical to enforce a broad ban of the challenge, the FHA has determined that the best way is to *frustrate* attempts at the challenge.

You are now the head of the FHA, and you have been given the following information:

1. a weighted, undirected, connected graph (i.e. a map of cities and roads): $G(V,E,\ell)$
   - the weights are given by $\ell : E \to \mathbb{R}_{>0}$
2. a source and a destination $s,t \in V$
3. two integer budgets: $0 \leq k \leq |E|$ and $0 \leq c \leq |V|$

You are tasked with *maximizing* the distances between the specified source and destination, where the distance between two nodes is measured as the shortest distance between them. Formally, you wish to remove at most k edges and at most c nodes *without disconnecting the graph* to form G' which maximizes

$$\Delta_{G'}(s,t) = d_{G'}(s,t) - d_G(s,t)$$

where $d_G(s,t)$ is the distance of the shortest path from s to t in G. Again, the solution where graph G' is disconnected from s to t (so $d_{G'}(s,t) = \infty$ and $\Delta_{G'}(s,t) = \infty$) is ***invalid***. Note that removing a node removes the surrounding edges, which does not subtract from your budget k (however, you are allowed to remove an edge adjacent to a node you are removing against your budget k, but this is not particularly useful).

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

Overview

You will submit three input graphs of different sizes. That is, the graphs you are submitting are instances of the problem that are to be solved by your peers in Phase 2.

You must submit exactly 1 input for each of the following size categories with *fixed* k and c parameters:

- **Small:** $|V| \leq 30$, $k = 15$, $c = 1$
- **Medium:** $|V| \leq 50$, $k = 50$, $c = 3$

[1]Note: CS 170 staff *strongly discourages* anyone from attempting this challenge.
[2]To reiterate: *do not try this challenge*.

- **Large:** $|V| \leq 100$, $k = 100$, $c = 5$

To clarify, you will be submitting 3 graphs with sizes of up to 30, 50, and 100, and each one will have the fixed k and c values above. For example, when we receive a small input, we are tasked with solving the problem in the setting where $k = 15$ and $c = 1$ (s and t are defined implicitly, see below).

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 30.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.**

Input Format

You will be submitting three graphs of sizes 30, 50, and 100 respectively (see below for problem description in each setting). 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 $\ell(u,v)$" defining an edge of the graph. u and v are integers between 0 and $|V|-1$ corresponding to vertices and $\ell(u,v)$ is a floating point number corresponding to the weight of the edge. $\ell(u,v)$ **must be greater than** 0, **less than** 100**, and have at most 3 decimal places**.
- Every vertex in the input graph should have a degree of at least 2.
- *Note:* you **will not** specify $s,t \in V$, instead, $s = 0$ and $t = |V|-1$, implicitly, based on your input.

Sample input:

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

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 and *publicly available* services (academic licenses included). This includes things like free AWS credits for students and excludes things like servers available only to those in your research lab. 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[3], **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.

Submission

Please follow submission instructions here.

We will maintain a leaderboard here (we will make a Piazza post when it is available) showing how well your solution performs compared to those of the rest of the class. **Please keep team names civil and PG-13 and no more than 30 characters long**. We will not tolerate any inappropriate team names.

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 is formatted as follows:

- The first line denotes the number of nodes being deleted ($\leq c$); each subsequent line lists the node being deleted.
- Thereafter, a line lists the number of edges being deleted ($\leq k$); each subsequent line lists the edge being deleted.

[3] *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 Output:

```
2        ← number of cities being deleted
1
2
5        ← number of edges being deleted
2 3
3 4
5 4
3 4
2 9
```

Reflection

You will also submit a project reflection in addition to your code 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.[4]
- 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.

[4]Inputs will be graded for completion.

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!