CS 170 Efficient Algorithms and Intractable Problems
Fall 2019 P. Raghavendra and S. Rao Version 1

Problem Statement

Professor Rao and his army of TAs are working in Soda Hall late at night, writing the final exam for CS 170. Rao offers to drive and drop TAs off closer to their homes so that they can all get back safe despite the late hours. However, the roads are long, and Rao would also like to get back to Soda as soon as he can. Can you plan transportation so that everyone can get home as efficiently as possible?

Formally, you are given an undirected graph $G = (L, E)$ where each vertex in L is a location. You are also given a starting location s, and a list H of unique locations that correspond to homes. The weight of each edge (u, v) is the length of the road between locations u and v, and each home in H denotes a location that is inhabited by a TA. Traveling along a road takes energy, and the amount of energy expended is proportional to the length of the road. For every unit of distance traveled, the driver of the car expends $\frac{2}{3}$ units of energy, and a walking TA expends 1 unit of energy. The car must start and end at s, and every TA must return to their home in H.

You must return a list of vertices v_i that is the tour taken by the car (cycle with repetitions allowed), as well as a list of drop-off locations at which the TAs get off. You may only drop TAs off at vertices visited by the car, and multiple TAs can be dropped off at the same location.

We'd like you to produce a route and sequence of drop-offs that **minimizes total energy expenditure**, which is the sum of Rao's energy spent driving and the total energy that all of the TAs spend walking. **Note TAs do not expend any energy while sitting in the car.** You may assume that the TAs will take the shortest path home from whichever location they are dropped off at.

Input Format

The first line of the input should contain a single integer, which equals the number of locations, $|L|$. The second line should also be an integer, which equals the number of homes, $|H|$. The third line should be a list of distinct names, separated by spaces. These are the names of your locations, L. The names must be alphanumeric, can contain up to 20 characters, and can contain only characters A-Z, a-z, and 0-9. The fourth line must be a list of names, separated by spaces, that are the names of your homes, H. All the homes need to be contained in the location list on the previous line. In other words H should be a subset of L. The fifth line must be the name of the location that is your starting point. Your output does not need to specify the walking paths of the TAs; we assume the TAs know how to find the shortest path home.

The next $|L|$ lines should contain an adjacency matrix representation of your graph. Location i refers to the location name at index i of L. For example, in the sample input shown below, location 2 refers to Wheeler. If there is no road between two locations i and j on the map, then the corresponding entry in the adjacency matrix, M_{ij}, should be the lowercase 'x'. For $i = j$, the entry M_{ij} should also be 'x', as there should not be a road from a location to itself. If there is a road between i and j, the entry M_{ij} should be the length of that road, ℓ_{ij}. The ℓ_{ij} must be strictly positive integers or floating-point numbers. They must be less than 2 billion, and floats must have at most 5 decimal places. It must be the case that $\ell_{ij} = \ell_{ji}$ for all values of i and j, since the graph is undirected. In other words, your adjacency matrix must be symmetric.

Your graph must be connected and the edge weights must obey the triangle inequality. This means that for any two vertices u and v such that the edge (u,v) is in your graph, the shortest path to get from u to v needs to be the edge (u,v), and not a path through any of the other vertices. Formally,

$$\ell_{uv} \leq \ell_{uw} + \ell_{wv}$$

where ℓ_{ij} is defined as the length of the edge (i,j). Additionally, it must be that $\ell_{ii} = 0$ (in the case of our input, we will have an 'x' present) and $\ell_{ij} \neq 0$ when $i \neq j$.

Sample input:

```
7
4
Soda Dwinelle Wheeler Campanile Cory RSF Barrows
Wheeler Campanile Cory RSF
Soda
x 1 x 1 x x 1
1 x x 1 x x x
x x x 1 x x x
1 1 1 x 1 1 1
x x x 1 x x x
x x x 1 x x x
1 x x 1 x x x
```

Output Format

The output file corresponding to an input must have the same name, except with the extension replaced by ".out". For example, the output file for "1.in" must be named "1.out".

Your output should contain the cycle the car took, the number of distinct locations at which TAs were dropped off, and the homes of the TAs who got off at each drop-off location. The first line should be a space-separated list of location names which represents the route taken by the car in your solution. These locations should be in the order in which they are visited and the list must start and end with the starting location defined in the corresponding input file. Locations may be repeated. The second line should contain the number of drop-off locations.

Each line in the rest of the output file should be a list of locations, the first of which corresponds to the drop-off location and the rest of which correspond to the homes of the TAs who were dropped off at that location. For example, if Julia lives in Cory and was dropped off at Soda, and Neha lives in the Campanile and was dropped off at the Campanile, we should have the two lines 'Soda Cory' and 'Campanile Campanile'. The order in which these lines appear in the output file does not matter. If no TAs get off at a location, it should not be included. If multiple TAs get off at the same location, they should all be included on one line. For example, if Dee and Emaan were both dropped off at Dwinelle, and Dee lives in Wheeler and Emaan lives in the RSF, one of the lines in the output file would be 'Dwinelle Wheeler RSF'.

In the sample output below, the TA who lives in Cory is dropped off at Soda, the TAs who live in Wheeler and the RSF are dropped off at Dwinelle, and the TA who lives in the Campanile is dropped off at the Campanile.

Sample output:

```
Soda Dwinelle Campanile Barrows Soda
```

```
3
Soda Cory
Dwinelle Wheeler RSF
Campanile Campanile
```

Submission Evaluation

Your score on a particular output will be determined by the sum of the energy it takes the car to drive the route and the total energy it takes for all the TAs to walk home. You may assume that the TAs will take the shortest path home from whichever location they are dropped off at. Formally, let the path of the car be $u_0 \dots u_{n-1}$ where $u_0 = u_{n-1} = s$, let b_j be the location you drop off the j-th TA, and let h_j be that TA's home. Let ℓ_{ij} be the length of the road between locations i and j (which must be adjacent), and d_{ij} be the shortest path distance between locations i and j (which need not be adjacent). Your score is given by:

$$\frac{2}{3} \cdot \sum_{i=1}^{n} \ell_{u_{i-1}u_i} + \sum_{j=0}^{|H|-1} d_{b_j h_j}$$

Note that the $\frac{2}{3}$ factor in the first term comes from the fact that the car driver expends $\frac{2}{3}$ the amount of energy as a TA.

Phase 1 (Due November 22nd, 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. While you are allowed to work alone or in a group of 2, we highly encourage you to find a group of 3. Beyond making the project more manageable, working collaboratively is a skill in and of itself, and working alone only deprives you of an opportunity to develop this skill.

Each group must submit three input files of different sizes, along with a valid solution to each of those instances. You will also submit a design doc detailing a few approaches you plan to take and why you think they will work well.

You must submit exactly 1 input and 1 corresponding valid output for each of the following size categories:

- Small: up to 50 locations and at most 25 TAs
- Medium: up to 100 locations and at most 50 TAs
- Large: up to 200 locations and at most 100 TAs

Your design doc should be **at least 200 words**, and **no more than 500 words long**. It should be a high-level description of algorithms/approaches that you have tried or plan to try. Additionally, you must include your reasoning for what led you to these approaches, and why you think they will work.

We will collect everyone's input files and release them to you all after Phase 1 is due. **The difficulty of your input files will determine part of your grade for this Phase** (details of this under the Grading section).

Some starting points for solver ideas include:

- Figuring out what problems this problem is similar to and reducing this problem to those problems. We recommend doing some Googling in addition to looking at recent discussion and homework questions.
- Reading about approximation algorithms for those other problems
- Finding solvers to those other problems (you have access to all freely available libraries!)

Feel free to use some of these ideas in your preliminary report, but make sure you explain concretely what you intend to do. For instance, if you intend to use some greedy heuristics, please specify those heuristics and state why you believe them to be good.

Submission Details

The three input files you submit should be named 50.in, 100.in, and 200.in. The respective output files should be named 50.out, 100.out, and 200.out. If your files do not satisfy these requirements, your input is invalid, and you will not receive any credit for this portion of the project. You will also submit the design doc on Gradescope.

Only one member of your team needs to submit the design doc and inputs, 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** for this part of the project. 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 December 6th, 11:59 pm)

Overview

You will be provided the pool of inputs generated by the class. These inputs will be divided into folders based on their size category. Design 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 final report** on the approaches you took and how they performed. You will also **give a brief presentation** to a TA about the ideas in your report, on either **December 12 or December 13**.

We have released starter code (see Piazza) to iterate over a folder and 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 ouput 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, **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

Put 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 solutions perform 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 final project report in Phase 2. The report should be a summary of how your algorithm works, what other methods you tried along the way, and what computing resources you used (e.g. AWS, instructional machines, etc.). Your final report should be **at least 400 words**, and **no more than 1000 words** long. 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.

In addition to this report, you will sign up for a presentation slot where you will be asked to be give a brief **5 minute presentation** to a TA. In this presentation, your entire group will explain a few of the approaches you took in attempting to solve the project problem. More information will be released closer to the final 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 initial report.
- 1% will come from your final report and project presentation.
- 1% will come from the quality of the inputs you provided us.
- 2% will come from the the quality of your outputs.

Your inputs will be scored based on your performance compared to that of other teams. We consider this a good metric of quality as we want inputs which have good solutions that are not easy to find. The score per input will range from 1 to 5 based on the following:

- 5 points (full credit): your solution performs better than 80% of student submissions.
- 4 points: your solution performs better than 60-80% of student submissions.

- 3 points: your solution performs better than 40-60% of student submissions.
- 2 points: your solution performs better than 20-40% of student submissions.
- 1 points: your solution performs better than up to 20% of student submissions.

Given the above grading scheme, we believe that any team that puts in a reasonable amount of effort into this project will receive at least **3% out of the overall 5%**. Of course, 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!