CMSC 27200 - Spring 2022 — Lorenzo Orecchia
Last updated - 5/2/22

Course Project

1 The Best Possible Vacation

1.1 Problem Statement

As everyone knows, there's no better way to spend a day off than by visiting the theme park TheoryWorld. After all, TheoryWorld has customizable churros, graph games, recursive roller-coasters, and Fermat figurines; what more could you ask for?

Clover, Erasmo, Jonathan, and Konstantinos need a vacation, so of course they decide to go to TheoryWorld. There's so much to do, but unfortunately they can only spend 24 hours in Theoryworld. To make things worse, the rides open and close at weird times (so disorganized - typical theorists). Unfortunately, that means there probably isn't enough time to do everything. Help the TAs find the best way to spend their day off!

1.2 Formalization

1.2.1 Input

As input, your program will be given the following:

- N: The number of attractions in TheoryWorld.
- N lines, one for each attraction A_i in $A_1, \ldots, A_n$:
  - x_i, y_i: the coordinate of A_i in TheoryWorld ($0 \leq x_i, y_i \leq 400$).
  - o_i, c_i: the lower and upper bound on the number of minutes after TheoryWorld opens that we may enter the attraction A_i ($0 \leq o_i \leq c_i \leq 1440$). Note, then, that we may *enter* the ride at any time between o_i and c_i.
  - u_i: the utility that the TAs get from attending A_i ($0 < u_i \leq 27200$).
  - t_i: the amount of time in minutes that the TAs spend at A_i.

Every number given in the input is an integer.

1.2.2 Output

Your program will need to output the following lines:

- M: the number of attractions visited.
- A list of unique attractions (by number) in the order visited.

1.2.3 Aim

Your program should output a *feasible* sequence of attractions that earns them the most utility. To determine if a sequence is *feasible*, we begin by describing how the TAs behave:

- The TAs start at $(200, 200)$ at minute 0 of the day.

- The TAs walk at a rate of 1 unit per minute, and walk from one destination to the next in a straight line (which means they may walk right through attractions). To prevent any potential precision errors, the TAs wait until the next whole minute to reach their destination.
- If A_i is closed, they wait until it opens. When it is open, they spend t_i time at A_i. They may experience A_i as long as they get to the location by the time it closes.
  - If A_i closes at minute 17 and they arrive at exactly minute 17, they can still spend the whole t_i time there. If they would arrive at minute 17.01, they are unable to do so.
- Once they are done with an attraction, they head to the next one and continue the process.
- After they finish their list of attractions, they head back to $(200, 200)$, which they must arrive at by minute 1440.

If the TAs can experience every attraction on the sequence and return to the start by minute 1440, then the sequence is *feasible*. If they go to an attraction after it closes, repeat any attractions, or can not return to $(200, 200)$ before minute 1440, then the sequence is *not feasible*.

1.3 Example Input and Output

Example Input:

```
10
50 350 0 1440 100 20
200 150 0 1440 200 10
200 100 0 1440 20 100
350 50 0 1440 15 40
200 400 0 1440 1000 10
300 150 0 1440 10 1000
400 20 0 1440 100 100
200 275 0 1440 300 30
100 50 0 1440 200 200
225 175 0 1440 100 100
```

Example Output:

```
5
5 8 10 2 9
```

2 Your Tasks

2.1 Phase 0: Team Formation (Due: 4/27/22, 11:59pm)

In this phase, you need to form a team, and let us know about your team on Gradescope. Teams should be no more than 3 people, but 1 or 2 is fine. That said, the amount of work is the same per group, so having more people in your group means less work per person. There is a group finder megathread on Ed.

Don't stress too much about team name - if you'd like to change it during the project, just let us know!

2.2 Phase 1: Inputs (Due: 5/6/22, 7:00pm)

In this phase, **your team will create a set of problem instances of various sizes for this problem**. More specifically, you will be submitting 9 different inputs, with the following names:

- `small1.in, small2.in, small3.in`

  Three inputs with $15 \leq N \leq 25$.

- `medium1.in, medium2.in, medium3.in`

  Three inputs with $40 \leq N \leq 60$.

- `large1.in, large2.in, large3.in`

  Three inputs with $150 \leq N \leq 200$.

2.2.1 Objective

You will receive points from this phase for submitting valid inputs **on time**, but you will also be graded on how well your solutions to your inputs in phase 2 compare to others' solutions. For that reason, **it will benefit you to provide tricky inputs that will be difficult for others to solve**. You may use code to help you generate inputs (and probably should for larger inputs).

2.2.2 Submission

We will be providing code to help you test the validity of your inputs. You should submit all nine inputs in a zip file, with the names listed above. Other names or submission formats will not be accepted. You should only submit once per team, and add all your teammates to your submission. **Due to the time-sensitivity of this project, late submissions will not be accepted.**

2.3 Phase 2: Outputs

In this phase, we will be providing you with the combined set of everyone's inputs. You will be competing with other teams to design and implement an algorithm that best identifies solutions to the inputs provided. **We strongly recommend that you start thinking about this during Phase 1.**

2.4 Phase 2a: Design Document (Due: 5/13/22, 7:00pm)

Two weeks before the end of the project, you and your team should have spent a decent amount of time thinking about the algorithm you want to implement, and should hopefully already have a first version of it implemented. To ensure that this is the case, you will submit a design document answering the following questions:

- What have you thought about so far? Which ideas seemed promising, and which ideas didn't seem so promising?
- What is your approach for the algorithm? How will it process the input, and how will it find a good solution? What heuristics or ideas do you plan to leverage, and why? What tools or services do you plan to use?

Your design document only needs to be around 500 words long, but can be more if you have more to say.

2.5 Phase 2b: Solutions (Due: 5/28/22, 7:00pm)

2.5.1 Objective

You will be submitting solutions to the problem instances provided. After Phase 2a, we will put up a leaderboard for you to see how well your submissions are performing against other teams in the class, as well as against our solutions. You will be graded on the following:

- Solution validity
- Performance against a baseline staff solution
- Performance against other teams
- Performance against a strong staff solution

The leaderboard will include the performance of a baseline staff solution intended as a metric to gauge that each time is putting some amount of thought into their algorithm. It will also include a strong staff solution - teams outperforming our solution will be granted extra credit.

2.5.2 Submission

You will also be allowed to start submitting solutions soon after Phase 2 starts, but will also be provided with starter code to help you determine whether or not your outputs are valid. Submission format will be updated soon.

You will also need to submit all of your algorithm's code in a separate zip file.

2.6 Phase 2c: Reflection (Due: 5/28/22, 7:00pm)

By the end of the project, your team will also submit a written reflection. Your reflection should provide a more detailed picture of your work in the project. It should be written in paragraph form, should aim to be around 2-3 pages, and should be divided into the following sections:

1. The Algorithm
    - How does your final algorithm work? (You may take what you wrote from your design doc, but there should be more detail here).
    - Why did you settle on it, and how well do you think it performed?
    - What ideas did you have after the design doc that you ended up abandoning, and why?
2. Running your code
    - What steps would we have to take to run your code on the inputs and get the outputs you've submitted? If different inputs were run on different algorithms, please describe how and why.
    - Did you do anything outside of running the code (like manually solving inputs or tweaking outputs)? If so, what did you do, which outputs are affected, and why did you choose to do so?
3. External Resource Usage
    - What outside computational tools/services/resources did you use, and to what extent did you use them?
4. Team Reflection
    - How was work distributed among the team? [Note: if you have concerns about your team's work distribution, please send us an email]

- What do you think you did well as a team?
- What surprised you as you worked on the project?
- What would you have done differently if you could do the same project again?
- What would you change about the project itself, if you could?

2.7 Rules

2.7.1 Computational Resources

You may use any online resources that **everyone in the class has access to**. This means you are welcome to use freely available software or packages you find online, as well as paid tools that have easily available academic licenses for UChicago students. Furthermore, your reflection document must indicate every tool/service used, how much it was used, and why. You are also welcome to use instructional machines, but only one at a time per person.

2.7.2 Academic Honesty

You should treat your work on this project the same way you'd treat a homework question - clarifying questions are welcome, but the sharing of specific strategies/ideas/answers outside of your team is discouraged. Of course, the sharing of code or outputs between teams is super discouraged. If you find a particularly helpful tool/service/resource online, you are welcome to share it, but you should do so publicly on Ed so that everyone can see it. If you don't know whether you are allowed something, you should ask us.

2.8 Grading

The project is out of 100 points total. The points are distributed as follows:

- Phase 1
  - Submitting 9 valid inputs: 20 points
  - Quality (difficulty) of inputs: 10 points
- Phase 2
  - Design Document: 5 points
  - Submitting all feasible solutions: 20 points
  - Performance of solutions over baseline: 10 points
  - Performance of solutions over classmates: 20 points
  - Reflection: 15 points