CS 170 Efficient Algorithms and Intractable Problems
Fall 2021 Jelani Nelson Project

Problem Statement

It is the year 2022, and technology has progressed far enough for humans to teleport anywhere in the world! Because technology has progressed, so has capitalism. Businesses are taking advantage of such technological strides, and we are no exception. CS 170 has decided that solving NP-Hard problems is not as lucrative as running an igloo polishing service. With teleportation, we can instantly move from igloo to igloo, but because we pride ourselves in offering great service, we cannot leave an igloo partially polished or polish two igloos at once. Additionally, not all igloo polishing tasks are equally profitable. Our goal is to strategically polish igloos in an order that would maximize our total profit. From our customer list (which consists almost entirely of penguins), we know what deadline they expect us to finish polishing their igloo by, how long polishing their igloo would take, and the profit we would gain from polishing that igloo by the given deadline. Additionally, if we finish polishing an igloo late (probably because we were polishing another igloo), our profit decays exponentially according to the following function:

$$p_i e^{-0.0170 s_i}$$

where p_i is the total profit we would have earned from polishing igloo i if we finished the task by the deadline and s_i is the number of minutes we were late in completing the task after the deadline.

Unfortunately, since CS 170 started focusing on igloo polishing instead of algorithms, we are having a hard time coming up with an order in which we should polish igloos to maximize our profit. So, we need your help figuring this out!

More formally, given a list of n igloos to polish with id i, deadline t_i in minutes, duration d_i in minutes, and profit p_i, provide a sequence of igloos to polish in one day (1440 minutes) that would maximize the total profit.

For the sake of the problem assume the following:

- We start scheduling polishing tasks at the 0th minute of the day.
- We start the next polishing task at the same minute we finish the previous polishing task. For example, if the very first task we schedule has duration 7, it starts at minute 0 and completes at minute 7. Now, the second task starts at minute 7, and assuming it has duration 10, it ends at minute 17.
- We receive whole profit for tasks that complete any time before or on their deadline. For example, if a task has duration 5 and deadline 10, we get the whole profit p for the task if we schedule it at any time before minute 5 (including at minute 5) because it would complete by minute 10. If we schedule this task at minute 6, we would complete it by minute 11, and gain profit corresponding to completing the task 1 minute late.
- We can schedule tasks such that the last task **completes by minute 1440**, For example, we can schedule a task with duration 1 at minute 1439 since it would complete by minute 1440. But, we cannot schedule a task with duration 1 at minute 1440 since it would only complete by minute 1441.

1 Your Tasks

1.1 Phase 1: Input Phase (Due 11/23, 11:59pm)

Overview

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

We will collect everyone's input files and release them after Phase 1 is due. Inputs are graded on completion and 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 100.in, 150.in, and 200.in (referring to the maximum size of each input — see "Input Format" for details). 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 they should 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 homework, **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.**

1.2 Phase 2: Solver Phase (Due 12/6, 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. Your grade 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 (more information is in the Grading section)

We have released starter code (see Piazza) to parse the provided input files and check outputs (you do not have to use the starter code). Teams are free to choose any programming language they wish, as long as the input and output files follow the specified format, since the Phase 2 autograder just requires the output files. Given the variety of possible programming languages and packages, we cannot guarantee that course staff will be able to provide support in office hours.

Note 1: Please choose a unique team name for the leaderboard (to keep the leaderboard anonymous) **Please keep team names civil and PG-13 and no more than 30 characters long**.

Note 2: Any further details regarding the Phase 2 submission, autograder and leaderboard shall be released via Piazza when Phase 2 begins.

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 we have this rule is that in the past CS170 has overloaded instructional machines and we do not want to make these resources unavailable to other students.

Input Format

Each input should be formatted as follows:

- The first line should contain a single positive integer equal to n, the events to be scheduled.
- Each of the following 'n' lines should be formatted as a space-separated list of four numbers "$i\ t\ d\ p$" defining the parameters of an event.
  - i is the task id, 1-indexed (integer)
  - t is the deadline of an task (integer)
  - d is the duration of an task (integer)
  - p is the profit for completing the task by the deadline (floating point number with at most 3 decimal places)
- The range of values are specified as follows, for each input size:
  - **Small:**
    $75 < n \leq 100, \quad 1 \leq i \leq n, \quad 0 < t \leq 1440, \quad 0 < d \leq 60, \quad 0 < p < 100$
  - **Medium:**
    $100 < n \leq 150, \quad 1 \leq i \leq n, \quad 0 < t \leq 1440, \quad 0 < d \leq 60, \quad 0 < p < 100$
  - **Large:**
    $150 < n \leq 200, \quad 1 \leq i \leq n, \quad 0 < t \leq 1440, \quad 0 < d \leq 60, \quad 0 < p < 100$

Sample input:

```
5
1 187 30 99.5
2 30 12 7.751
3 15 55 12.987
4 200 15 55.2
5 1420 20 1.25
```

Note 1: The i^{th} line in the input file corresponds to the parameters of the i^{th} igloo polishing task (1-indexed).

Note 2: The input file must contain all the n tasks with 1 event corresponding to each task number to be considered a valid input for the input autograder.

[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.

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 "100.in" must be named "100.out". The output is formatted as follows:

- There is one number per line.
- Each number represents the index of the igloo to be polished. The index of the igloo is its position as listed on the inputs list (starting from 1).

Sample Output:

```
4
5
1
2
```

This shows that we will polish igloo 4, igloo 5, igloo 1, and then igloo 2. Note that we decided not to polish igloo 3.

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.[2]
- 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!

[2] Inputs will be graded for validity and completion.