CS 170 Homework 12 [Coding Portion]

Due **Monday 4/29/2024, at 10:00 pm (grace period until 11:59pm)**

6 [Coding] Approximation Algorithms

For this week's coding questions, we'll implement some approximation algorithms for the **Traveling Salesperson** problem you saw earlier. There are two ways that you can access the notebook and complete the problems:

1. **On Datahub**: click here and navigate to the `hw12` folder.
2. **On Local Machine**: `git clone` (or if you already cloned it, `git pull`) from the coding homework repo,

   https://github.com/Berkeley-CS170/cs170-sp24-coding

   and navigate to the `hw12` folder. Refer to the `README.md` for local setup instructions.

Extra Credit Opportunity

Additionally, we are offering up to **3 points of post-curve extra credit** if you'd like to keep on exploring ways to cope with NP-complete problems and try to do better than the 2-approximation for the Metric TSP problem.

Since the extra credit will be applied *post-curve*, attempting this question can only help your grade, and skipping this part *cannot hurt your grade.*

To earn extra credit points, the total cost of your algorithm's outputs on our test set will need to beat certain *fixed* thresholds that we will release early next week. So, *you will be evaluated purely on how well you do, and will not be compared against your classmates.* However, we will enable the Gradescope leaderboard so you can see how your classmates are doing in a friendly, low-stakes competition! *You do not need to use your real name on the leaderboard, but please keep aliases professional and appropriate.*

For this extra credit portion only, we ask you to implement your code inside of a `.py` file that we provide. There are two ways that you can access the `.py` file:

1. **On Datahub**: click here and navigate to the `extra_credit` folder.
2. **On Local Machine**: `git clone` (or if you already cloned it, `git pull`) from the coding homework repo,

   https://github.com/Berkeley-CS170/cs170-sp24-coding

   and navigate to the `extra_credit` folder. Refer to the `README.md` for local setup instructions.

While we ask that you only submit your final `.py` file to Gradescope, feel free to create your own Jupyter notebooks or supporting `.py` files to help you test out your algorithm.

The test set is designed so that the DP algorithm will take forever to run. Instead, you should try to implement some other approximation strategy – your creativity is the limit here! Course staff have some ideas on how to get started (in no particular order):

(None of these algorithms or techniques are in scope for the final, but they are still useful and cool!)

- Implement the 3/2-approximation for Metric TSP (Christofides Algorithm)
- Use branch-and-bound to explore subproblems in a more efficient way. See DPV Section 9.1.2 for more information.
- Improve an existing approximate solution using local search. See DPV Section 9.3.1 for more information.
- Try to use randomness or *simulated annealing* to get lucky and find good approximate solutions. See DPV Section 9.3.3 for more information.
- Reduce the problem to another problem (such as integer LP) and use a pre-made solver library.

Your algorithm will be run on the entire test set on Gradescope, so it must be efficient enough to run on all inputs within the 40 minute timeout limit.

For the extra credit only, feel free to use any third-party library imports that you like, as long as these library imports do not *directly solve the TSP problem.* So importing a Python TSP solver is not allowed, but importing modules to help you compute an MST or a linear program is OK. *If you're in doubt, ask!*

Notes:

- *Submission Instructions:* For the normal assignment, please download your completed submission `.zip` file and submit it to the Gradescope assignment titled "Homework 12 Coding Portion".

  For the extra credit, please upload your completed `.py` file and submit it to the Gradescope assignment titled "(Extra Credit) Coding Portion".

- *Getting Help:* Conceptual questions are always welcome on Edstem and office hours; *note that support for debugging help during OH will be limited.* If you need debugging help first try asking on the public Edstem threads. To ensure others can help you, make sure to:

  1. Describe the steps you've taken to debug the issue prior to posting on Ed.
  2. Describe the specific error you're running into.
  3. Include a few small but nontrivial test cases, alongside both the output you expected to receive and your function's actual output.

  If staff tells you to make a private Ed post, make sure to include *all of the above items* plus your full function implementation. If you don't provide them, we will ask you to provide them.

- *Academic Honesty Guideline:* We realize that code for some of the algorithms we ask you to implement may be readily available online, but we strongly encourage you to not directly copy code from these sources. Instead, try to refer to the resources mentioned in the notebook and come up with code yourself. That being said, we **do acknowledge** that there may not be many different ways to code up particular algorithms and that your solution may be similar to other solutions available online.