Homework 4 Self-Evaluation

4/5 Points

2/20/2024

Attempt 1

Review Feedback
2/19/2024

Attempt 1 Score:
4/5

Add comment

Anonymous grading: **no**

Unlimited Attempts Allowed

2/22/2024

Details

Please answer the following questions about the code you submitted to the **first** submission deadline (i.e., not the resubmission, and not your work in progress for the resubmission). If you overwrote it locally, you can redownload your submission from Canvas.

For each question, answer either with the line number (or a range of line numbers) that is relevant to the question, or with "no" if your code does not do what the question is asking about. *Your answer to a question will get 1 point if your answer accurately answers the question and you did not answer "no"; your answer will get a 0 in all other cases.*

Make sure to **double-check your answer and line numbers** to make sure they are correct and also that they are referencing the correct version of the file (see above). To ensure consistency in grading across all students, **your line numbers MUST correspond to the relevant lines of code, otherwise the question will get a 0** (even if the relevant code lies elsewhere).

---

1. Are you using an adjacency list or adjacency matrix representation for your graph? Specify which representation you chose, and tag the line in your code where you define the field that stores the "spine" vector (the vector containing the "rows"). If your implementation is neither an adjacency list nor an adjacency matrix, answer "no".
2. Does your `get_adjacent` method have the correct complexity? (O(V) for adjacency matrices, O(d) for adjacency lists.) Tag the line in your code where you loop over the matrix row (adjacency matrix) or the list of adjacent vertices (adjacency list). If your implementation is neither an adjacency list nor an adjacency matrix, answer "no".
3. Does your `set_edge` method maintain the undirectedness invariant by recording the edge in both directions? If using an adjacency list, tag the lines where you add both `u` to `v`'s list and `v` to `u`'s list. If using an adjacency matrix, tag the lines showing that you add both `u` to `v`'s row and `v` to `u`'s row. If your implementation is neither an adjacency list nor an adjacency matrix, answer "no". If you're maintaining the invariant some other way, point to the lines where you're doing so, and explain what your invariant is.

4. Do you have a unit test for `dfs` on a graph that is not fully connected, showing that it does not ever reach unreachable vertices?

5. *For this last question, answer with a short paragraph. We will not be looking for a specific correct answer, but rather for thoughtfulness and reflection.*

   The way we represent our data shapes the way we think about problems. For example, your `my_neck_of_the_woods` graph represents road connections specifically, omitting other modes of transportation: e.g., train lines, bike paths, etc. As a result, someone using it for city planning would naturally gravitate towards solving problems using roads, as opposed to other modes of transportation, neglecting the needs of people who rely on these other modes. How would you adapt `my_neck_of_the_woods` (e.g., changing the meaning of nodes or edges, adding another ADT, etc.) to allow representing different modes of transportation, while still being able to distinguish roads from, e.g., rail?

View Rubric

Select Grader

Bennett Lindberg (TA)

Self-Eval

Criteria	Ratings		Points
Q1	**1 pts** **Got it**	**0 pts** **Missing/Incorrect**	1 / 1 pts
Q2	**1 pts** **Got it**	**0 pts** **Missing/Incorrect**	1 / 1 pts
Q3	**1 pts** **Got it**	**0 pts** **Missing/Incorrect**	1 / 1 pts
Q4	**1 pts** **Got it**	**0 pts** **Missing/Incorrect**	0 / 1 pts
Q5	**1 pts** **Got it**	**0 pts** **Missing/Incorrect**	1 / 1 pts
			Total points: 4

1. adjacency matrix; 80 (definition) and 85 (initialization)
2. 104
3. 94 & 95
4. no

5. Instead of one `OptWeight`, store a `struct` containing as many fields as there are modes of transport. Each field contains an `OptWeight` indicating the distance/travel time if the mode of transport exists, or `None` if it doesn't. So, for example, if cities 1 and 2 are connected only by train and the train journey is 3 miles, then `graph[1][2]` and `graph[2][1]` in an adjacency matrix representation will contain identical `struct`s where `road` is `None`, `train` is `3`, and `air` is `None`.