Homework 9

Due Friday, April 19 at 5:00 PM

Familiar with the personality traits “exceptionally intelligent”, “sadistic” and “vengeful”? Like most Geminis, with his unparalleled sociopathic visions, Syndrome leads the retired super villians with the mantra: “You can’t count on anyone, especially your heroes!” - Syndrome

Handing In

To hand in a homework, go to the directory where your work is saved and run `cs0160_handin hwX` where X is the number of the homework. Make sure that your written work is saved as a .pdf file, and any Python problems are completed in the same directory or a subdirectory. You can re-handin any work by running the handin script again. We’ll only grade your most recent submission. To install stencil Python files for a homework, run `cs0160_install hwX`. **You will lose points if you do not hand in your written work as a .pdf file.**

Written Problems

Problem 9.1

Negative weights

The proof of Dijkstra’s algorithm relies on all the weights being non-negative. Consider the following algorithm for computing shortest paths in the presence of negative weights. Let m be the weight of the most negative (i.e., smallest) edge. Add $|m|$ to the weight of each edge, thus making all weights non-negative. Now use Dijkstra’s algorithm to find the shortest path. Does this work? Explain why, or give a counterexample.

Problem 9.2

Connecting People

Silly premise: Sulley is trying to raise his scare total to beat out Randall. He has to scare as many children as possible, but can only order one door at a time. Sulley realized the best way to scare the most amount of children is if he can order the doors closest to the scare floor to minimize the amount of time wasted (he only has an hour to scare children before the scare floor closes). Help Sulley create a map so that he can see which doors are closest to the scare floor to order doors as quickly as possible.

There is always a door that starts on the scare floor, which is referred to as location Q. All of the locations of the doors need to be reachable by the map from this starting point, Q. The locations of the doors are at a set of points P_i ($i = 1 \ldots\ldots n$). *Note: Q is not included in the set of points P_i.* A door's location can be connected to the starting point directly or by a chain: location 5 could be connected by path to location 2, which could then be connected by path to the starting point Q; now locations 2 and 5 are both connected to the starting point as required. The chain can be arbitrarily long, not just two locations as in this example.

Assume that you're given the locations of the points P_i and the starting point, Q, and that $d(A, B)$ is a constant-time function that gives the cost of connecting A to B to the ideal path (e.g., perhaps it's the straight-line distance). Your goal is to find the lowest-cost path network to connect all the doors' locations to the starting point.

1. Describe how to model this problem as a graph problem, and then give a brief paragraph description of your solution. What is the running time of your algorithm in terms of n (the number of doors)?

2. Now suppose that while Sulley is inside one of the rooms, Randall calls one of the doors on Sulley's map. Sulley's original plan will be ruined, so he needs to make sure the door chains are not too long. So he can minimize the risk of a door being called before he can order it. Fortunately, Sanderson gets a "Code 2319" and needs to be quarantined, so Sulley gets an additional dock, K, on the scare floor from which he can call doors. Sulley can enter rooms from either dock and call doors while he's scaring children. This gives Sulley two starting points to take into account, Q and K, at different locations. As stated, Q and K are already connected by a path. Each item P_i must be connected to one of these two starting points (perhaps by a chain, as before). Once again describe how to model this as a graph problem, and explain how to solve it. Specify the running time in terms of n. Your solution should be just a few sentences.

Python Problems

As always you're expected to test for valid input, please do so using the exception **InvalidInputException**.

Problem 9.3

Multiple shortest paths

Silly premise: Mike Wazowski has just scared a girl and harvested her scream when he hears footsteps from down the hallway. Her parents! Mike needs to escape the room before the girls parents arrive, but her floor is littered with toys and Mike needs your help to find the shortest path for his escape.

Let $G = (V, E)$ be an undirected graph with *unit* edge lengths (i.e., the length of each edge is 1). Note that there might be multiple shortest paths between a pair of nodes in G. Write python code for a linear-time algorithm that finds the number of distinct shortest paths between nodes u and v. *Hint: O(VE) is not linear, but O(V + E) is.* You may assume that there are no parallel edges or self loops in G, and you may assume that there is at least one path between u and v, if u and v are both in the graph (which is not guaranteed). You may also assume that u and v are distinct nodes. Keep in mind that your code need only return the **number** of distinct shortest paths, not the paths themselves. However, in order to find the shortest path(s) you will need to calculate the distance (in this case equal to number of edges) between the given pair of nodes in G.

For this assignment, the only data structure you are allowed to use is python's list. If you plan to use the list as a queue, you'll need to figure out how to pop off only its first element. You are allowed to manipulate and decorate the graph. A stencil for this code, named **numShortestPaths**, has been provided for you.

Problem 9.4

Topological Sort

Implement a toplogical sort on a directed acyclic graph (DAG) in Python.

Requirements

Your function will take in a DAG and return a list of vertices (instances of GraphVertex) in topologically sorted order. If there are multiple valid orderings, you should only return one.

You should **not** assume that the input graph is truly acyclic. In the event that the input graph has a cycle (making it impossible to generate a topological sort), you should raise a `GraphCycleException`. You also should **not** assume that the input graph is fully connected. Its possible that the graph could have two unconnected parts.

You will need to use some auxiliary data structure to help generate your topological sort. We suggest you use a Python list as a stack (e.g., `s = []`).

`topological_sort` **must** run in linear $O(|V| + |E|)$ time. **Assume that the `incidentEdges` and `eminentEdges` methods run in O(1) time instead of O(n) time.**

Lastly, keep in mind that your algorithm must **not** manipulate, destroy, or decorate the input graph. Instead, think about what data structure you could use to map the number of incident edges to a given vertex in O(1) time. In this case, the number of incident edges would act like a decoration. Your data structure must not change the runtime of the original algorithm.

In this case, directly adding decorations is considered manipulating the graph. Do not do this.

Support Code

You will need to use the `mydigraph` module to complete this assignment. It may be helpful to look through the file to see the methods available to you.

Testing

Test your code in the provided separate Python file as usual.

Problem 9.5

Dijkstra's Algorithm

Implement Dijkstra's algorithm on a graph.

Requirements

Your function will take in a **connected** graph and a vertex. It should return a **tree** that contains the shortest paths from the given vertex to all other vertices. Your tree will be an instance of **MyGraph** but, since it should have the properties of a tree, it must not contain any cycles. You may not manipulate or destroy the graph. Decorations are okay, but do not add or remove vertices or edges. We have provided a `HeapPriorityQueue`.

How To Test Your Code

Test your code in a separate Python file as usual. See the sample test files.