CS 228 Fall 2019
Homework 4 (Part 2) - Dijkstra's Algorithm

(There is NO LATE DUE DATE - LATE HWs WILL NOT BE ACCEPTED)

Introduction

For this part of the assignment, you will be submitting a single class that implements the interface Dijkstra<V> that is provided along with this assignment statement. Your implementation of this interface will be used to find shortest paths and the total costs of those paths through a graph that will be given to you as a reference to an instance of DiGraph<V>, an interface that is also provided to you.

Instructions

Write a generic class CS228Dijkstra<V> that implements Dijkstra<V>. Please read the comments in both interfaces very carefully, as they detail the specific behavior of each method and the appropriate use of the interfaces. Place this file alongside the two interfaces we give you in the cs228hw4.graph package. The ". . . \graph" directory should be a sibling directory of the ". . . \game" directory where you will have your Agent implementation for Part 1 of this assignment. A single .zip file can be submitted with a single "hw4\cs228hw4\. . . " directory structure. You will want to implement DiGraph<V> in order to test your CS228Dijkstra<V> class, but you will not be required to turn this in. Note that proper implementation of these two interfaces could be helpful in your Agent implementation for Part 1, so feel free to use this package in your solution for Part 1 if you like.

We will test your CS228Dijkstra<V> class with our own implementation of DiGraph<V> that behaves as specified in the interface comments. Even though you are not submitting your implementation of DiGraph<V>, do not share this code since it can be used for Part 1. You may, however, share JUnit tests for these classes in the usual manner on Canvas.

In order to receive full credit for this part, you must:

- Place all your files in the cs228hw4.graph package.
- Implement all methods as specified in the Dijkstra<V> interface. Pay attention to runtime requirements.
- You are free to use anything you can find in the Java Collections Framework.
- Your class is required to have the one-parameter constructor described in the documentation.
- You must use Dijkstra's algorithm for computing the shortest path. Using the algorithm described in the book or lecture is acceptable.