Lab 6: Graphs and Search

1 Objectives

- **Learn about graph structures in NetworkX**
- **Understanding search algorithms**

2 NetworkX

NetworkX is a Python package for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. We will be using it as a way to create and display graphs for our search algorithms. Begin by installing networkx `https://pypi.org/project/networkx/` **and going through the tutorial** `https://networkx.github.io/documentation/stable/tutorial.html`**.**

3 Homework Questions

3.1 Importing Nodes

While you can go through and manually input nodes and edges, oftentimes it may be necessary to import information from another file. Download 'contiguous-usa.dat' which contains the connectivity of US states. In Lab7.py, create a function in nodes() that reads the dat file, creates a graph, and plots it.

3.2 Shortest Paths

NetworkX has built-in functions to find the shortest path between nodes within a graph. In the function plot_paths(), implement a function that takes in a graph, a starting node, and an end node. The function will find the shortest path using the shortest_path function, A*, and Dijkstra. Then, the function will plot the entire graph, with Dijkstra's path highlighted.

3.3 Adding Weights

The graph created in the previous questions did not contain any weights, so shortest paths were calculated purely in terms of degrees (i.e., nodes) away from each other. Now consider that each edge has a number of different weights associated with it. Create a new dat file that contains one additional columns to the contiguous states one. The third column contains a distance (e.g., geographic distance) between those nodes. You can input in realistic numbers, or completely random ones. Implement this in function weights(). The function should create a new dat file, and then return a weighted graph.

3.4 Oregon Trail Redux

The Oregon Trail was an amazing computer game first developed in the 1970s designed to teach children about the realities of 19th century pioneer life. The player assumes the role of a wagon leader who is guiding a party of settlers from Independence, Missouri to Oregon's Willamette Valley.

A number of incarnations have popped up since the original. If you're interested in playing, some devoted fan wrote a version in Python a few years ago. Feel free to download and try it out: `https://github.com/philjonas/oregon-trail-1978-python`**.**

We're going to play a slightly pared down version of Oregon Trail using the contiguous states dat file that you created in Question 3. For this question, you're going to write your Oregon Trail. The game will do the following:

- **Begin by asking the user what state they are starting from and what state they would like to get to.**
- **Calculate the shortest path (in distance) between these two states. This value is kept hidden from the user.**
- **On the screen, present all the states that the user can travel to, and ask which one they would like to go to. Keep asking the user until the goal state is reached.**
- **Once the player reaches the goal state, if the entire path matches Dijkstra's path in weights (distance), then the user has won. Otherwise, the player has lost. Plot Dijkstra's path and the user's path. Note that it's possible to get different paths but the same weighted value. In this case, consider that a win for the user.**
- **Present the player the option to quit at any time in the game.**

Implement this game in the function oregon() in Lab7.py

4 Submitting Homework to ELMS

List all assumptions, and use comments to notate your code whenever possible. Clarity and style will matter. Your TF should be able to read your code and understand everything. You will be submitting one python file containing your code/functions.

1. **DirectoryID_Lab6.py**