Come and take it!

Optimization Challenge, MSBA 6311, Summer 2023

Description

"Come and take it!" was the battle cry during the American revolution in 1778 at Fort Morris, Georgia, and again in 1835 at the Battle of Gonzales during the Texas Revolution. So it seems we have a history of fighting the status quo.

When data was gathered for the `usstates.csv` file, the US population was approximately 275 million people (it is currently believed to be over 300 million). Continuing in the tradition of early Americans, from time to time various US states (notably Texas and California) threaten to secede from the Union and become their own country. In this programming challenge, you are a data scientist for the US government and you have been asked to explore some rudimentary secession scenarios and what those scenarios might mean for the US Federal government.

Objectives

After participating in this activity, you will be able to:

- Review and solve a "real life" data analysis scenario that has no one right answer.
- Explore and implement advanced analysis techniques.
- Apply the knowledge and skills you've gained during the first 5 labs in a holistic manner.

Instructions

1. Create a new GitHub repository to hold your work for this challenge. This should be a private repository. Only the course staff should have access to this repository. Name your repository whatever you like, but be sure to *setup the webhook as you did for your labs repository.*
2. Register your team name to initiate your participation in the programming challenge. You will not appear on the leaderboard until you register successfully.
3. Write the 2 functions described below and commit each function to your repository in a .py file of the same name. Each time you commit a new version of a file, it will be evaluated / timed and the leader board will be updated.

Timeline

Unlike a traditional assignment with a singular due date, the programming challenge lasts approximately two weeks each, concluding on the due date posted on the course web site. Pay attention to the course website and announcements for detailed timing information.

Getting Started

You can find a sample (but not very good) implementation of each function in the challenge repository. The file `border_data.csv` contains information on which US states share borders and the length of each border. Use this file in combination with the `usstates.csv` file to write two functions:

1. `new_nation_n_states`: This function accepts 1 argument (n) and returns the most populous "new nation" carved out from n US states. The states must be contiguous; that is they must each share a border with another state in the new nation.

2. `new_nation_with_pop`: This function accepts 1 argument (a population, in millions) and returns a list of all possible new nations, consisting of a minimum number of states, with at least that many people. Once again the states in each new nation must be contiguous!

Place each function in its own .py file of the same name. For example, the `new_nation_n_states` function should go in the `new_nation_n_states.py` file. Here are some example function calls:

```
In [1]: new_nation_n_states(1, 'usstates.csv', 'border_data.csv')
Out[1]: (('CA',), 34888000)

In [2]: new_nation_n_states(2, 'usstates.csv', 'border_data.csv')
Out[2]: (('CA', 'AZ'), 39325000)

In [3]: new_nation_n_states(3, 'usstates.csv', 'border_data.csv')
Out[3]: (('WA', 'CA', 'OR'), 44362000)

In [4]: new_nation_n_states(4, 'usstates.csv', 'border_data.csv')
Out[4]: (('NM', 'TX', 'CA', 'AZ'), 61187000)
```

The most populous US state is California (CA), so it makes sense that the largest nation carved from 1 existing state would simply be CA. The largest nation consisting of 2 states would be California and Arizona, which share a border, and so on.

Here are some additional example function calls:

```
In [1]: new_nation_with_pop(34, 'usstates.csv', 'border_data.csv')
Out[1]: [('CA',)]

In [2]: new_nation_with_pop(40, 'usstates.csv', 'border_data.csv')
Out[2]:
[('WA', 'CA', 'OR'),
 ('NM', 'CA', 'AZ'),
 ('NY', 'PA', 'OH'),
 ('NV', 'CA', 'AZ'),
 ('CO', 'CA', 'AZ'),
 ('CA', 'AZ', 'UT'),
 ('CA', 'AZ', 'OR')]

In [3]: new_nation_with_pop(50, 'usstates.csv', 'border_data.csv')
Out[3]: [('NJ', 'NY', 'PA', 'OH'), ('NM', 'TX', 'CA', 'AZ'), ('NY', 'PA', 'OH', 'MI')]
```

California has over 30 million people, so the minimum number of states required for a nation of at least 30 million is just 1. To get at least 40 million people you need a combination of 3 contiguous states. There are 7 options that have at least 40 million. To get at least 50 million people you need a combination of 4 contiguous states, and there are 3 options that achieve this result.

Evaluation

Because this is a computationally intense problem, your solution will be given a maximum time window (about 5 seconds) for evaluation. Your place on the leader board will be calculated using the following metrics (in this order):

1. **Highest n (or pop)**: A function that calculates for $n = 6$ in 5 seconds or less will place higher than a function that can only calculate for $n = 4$.
2. **Time**: A function that calculates $n = 6$ in 4.536 seconds will place higher than a function that calculates the same n in 4.846 seconds.

In addition, a couple of scenarios will result in not appearing on the leader board or disappearing from a previously held position:

1. **Invalid Solution**: For example, the states you return do not border each other or the population of those states does not add up correctly.
2. **Suboptimal Solution**: You hold the top spot on the leaderboard with $n = 4$ and a valid solution with a population of 70 million. Another team then calculates a solution for $n = 4$ with a population of only 61,187,000.

You will earn up to 100 points for this programming challenge. To allow you to self-select your personal level of effort, you can expect to earn points according to the following definitions:

Approx. %	Expectation	Approx. Letter Grade	Description
Up to 75%	Meets	C	- Participate only a few times - Basic solution with obvious limitations - Final points based on participation level
80 – 90%	Exceeds	B	- Participate many times over the course of the week - Brute force implementations accepted - Final points based on participation level
90 – 100%	Outstanding	A	- Participate many times over the course of the week - Consistently on top of the leader board - Final points based on participation + performance

A Few Notes

1. As with the labs, please have reasonable expectations of the infrastructure. This is a computationally intense problem, and performing the evaluations can be time consuming even for the fastest computer. Per the syllabus we will aim to resolve technical issues within 24 hours.

2. Please do not attempt to defeat, subvert, or hack the system on your way to the top. Obviously you could precompute some of these scenarios on a powerful computer and then hard-code them into your function, but that is not the objective. If you have any question as to what is an acceptable strategy, please ask! Attempting to subvert the system could result in disqualification (0 points for the assignment) and in extreme cases could be considered academic misconduct.