ECE241 PROJECT 1: Sorting and Searching

Due: Oct 8th, 2020, 11PM on Moodle

Introduction:

COVID-19 (Coronavirus) has affected our daily life and is slowing down the global economy. This pandemic has affected thousands of peoples, who are either seriously sick or die due to the spread of the virus. We need to know our enemies before we defeat them. In this project, we will analyze the public COVID-19 data collection from the Harvard Dataverse Library to have a better understanding of the spread of the virus.

In this data collection, we have the number of daily confirmed cases for different cities. These are Metropolitan level daily cases for 942 cities from May 1st to July 1st, 2020. For each city, we have information about the City_ID, City Name and its State, population, and the total number of confirmed cases on a specific day.

In this project, we will read the COVID-19 Dataset from an Excel file (cov19_city.csv). We manage the data with our own data structures so that we can quickly find the information we are interested in and the spreading patterns. Then we will perform search and sort operations on the data to demonstrate the efficiency of different sorting and searching algorithms.

Task Overview

In this project, you will perform the following specific tasks.

1. Manage the data in array-based objects with Python. Specifically, you can follow the instructions below to start. **Note, you should use the required name for the classes and functions**.
    a. Each city is managed with the **City class.** In the City class, you should have variables for the different attributes of the city (specify the names attributes as ***cid*** for City_ID, ***cname*** for the City name, ***cstate*** for the state name, ***pop*** for population and array ***cities*** for a list that contains the numbers of the confirmed cases for two months).
    b. Implement a **COV19Library class** as a library to store the whole dataset. In the COV19Library class, you should have basic information about the library which usually contains an array for all the City objects, an integer that indicates how many cities are in the library. You may create additional variables, for example, a Boolean variable that shows whether the library is sorted or not.
2. Implement a ***LoadData(self, filename: str)*** method in COV19Library class. The method takes the file name (string type) of the input dataset and stores the data of cities into the library. Make sure the order of the cities is the **same** as the one in the input file.
3. Implement a linear search algorithm to search the cities based on City_ID or city Name. The ***linearSearch(self, city: str, attribute: str)*** method takes two arguments as the key string for search and the attribute that we want to search ("id" or "name"). It prints out the details of the city, including the ID, name, state, population, and the latest number of confirmed cases. Print a "City not found" message when there is no match.
4. Implement a ***quickSort(self)*** algorithm to sort the city library database based on the city name. The sorted cities should be stored in the same city array. Make sure you change the status of the Boolean variable that shows whether the library is sorted or not.
5. Implement a function ***buildBST(self)*** to build a balanced BST of the cities based on the city cid. Record the time spent on building the BST. Ideally, the height of a balanced BST for 942 cities is 10. In this task, you need to build a tree with height **AT MOST 20**.

6. Implement a search function ***searchBST(self, cid: str)*** for city ID on BST.
7. Perform a linear search based on ***cid*** in the sorted database for the 100 cities. You can use a random number generator to arbitrarily select 100 city ***cid***s. Record the average time spent on the linear search.
8. Perform a search of the same 100 cities in the BST. Record the average time spent.
9. In order to perform a search on BST, we need to spend additional time to build the BST. If we have many cities to search, it is worthwhile for the additional time. Using the information calculated above (Task 5, 7, 8), determine how many linear searches $\boldsymbol{n}$ would be necessary before the overall run time of the linear searches would be more than the combined total of building a BST (only performed once) and $\boldsymbol{n}$ searches in the BST.
10. Plot a figure to shows the daily increasing cases for the city with the highest population. (Note that, the confirmed cases in the input data are the total number, not the daily increasing number). Can you tell from the figure what stage the pandemic is at? (An **early-stage** with the increasing number still increasing; A **peak-stage** if the increasing number is at a high level for several days; A **late-stage** shows the increasing number is decreasing)
11. Find the city with the highest rate of confirmed cases in June. The rate is computed from the number of increased cases divided by the population.

What to submit

For Task 1-6, you should submit your code to Gradescope for auto-grading. Remember to comment your code properly.

For Task 5, 7-11, put your recorded running time of building BST, the average time for linear search and search on BST into a Document (.doc or .pdf). Explain how you compute the number of searches $\boldsymbol{n}$ for task 10 in the document. Submit the document in Moodle.

Reminder: The course honesty policy requires you to write all code yourself, except for the code that we give to you. Your submitted code will be compared with all other submitted code for the course to identify similarities. Note that our checking program is not confused by changed variable or method names.

Grading

- Code works on Gradescope (70%)
- Assignment results (search number $\boldsymbol{n}$, plot, city and stage) (20%)
- Program structure and comments (10%)