2020 CAREER FAIR PROGRAMMING COMPETITION
Alhassan Issifu

Data Structures Used: Hash tables, ArrayLists and Arrays

TASK 1 APPROACH:

In my approach, I read each input file once. While reading the *covid_data* file, my algorithm sums up the deaths to get the total number of deaths and also inserts data into hash tables as follows.

Hash table name	Key	Value
`mytable`	country	total number of infections
`weekly`	country	array of infections per day for the most recent week
`entire`	country	array of all infections for that country

I used hash tables because they have a runtime of O(1) for data insertion and retrieval.

To answer (a) and (b), I realized I had to loop through the hash table which is not time efficient. Therefore, while updating the hash table `mytable`, I created variables to keep track of the highest and second highest total number of infections and their countries. If we ignore going through the input file, this is O(1) and O(n) otherwise.

Once this was set, I had all the information needed to answer the rest of the questions and since most of them were independent, I used multithreading. One thread determined which country had the highest infection rate (c). While reading through the *population_data* file, it takes a country, gets its total infections from the `mytable` hash table (runtime O(1)) and divides by the country's population to get the rate. For each country, it compares its rate to the current highest rate and makes changes when necessary. The next thread was responsible for overall death rate (d). It divides total number of deaths by the total number of infections since these values were already calculated. The next thread determined the country with the highest death rate(e).

Tasks (f) and (g) are related, so are (h), (i) and (j). Both groups made use of the `weekly` hash table. To find the slope of the best fit line, I used the least square method $m = \frac{n\times\sum(xy)-\sum(x)\times\sum(y)}{n\times\sum(x^2)-(\sum(x))^2}$. I then used one thread for (f) and (g) since after finding those with positive slope (f), I can find the highest slope (g). I used another thread for (h), (i) and (j). Thus, after finding those with negative slope (g), I can find the least slope (h). To solve (i), for each of the countries in (j), I got their values in the hash table called `entire` which I then looped through to find the peak value and the date it occurred. I created a method called `earlier(date1, date2)`. It returns true if `date1` is earlier than `date2` and false if otherwise. I used it to compare the dates and took the country with the earliest.

Since multithreading might get my answers randomly (maybe c, g, e, i, d…), I created a hash table with integer keys and string values mapped as 1→ (a), 2→ (b), 3→ (c) and so on. I then created a last thread that prints the content of this hash table into an output text file. This thread prints them in order and if a value is not yet inserted into the hash table, it waits. I have a runtime of O(n) for each question in task 1 except (d) which has O(1). The others will remain O(n) if we consider reading through the input file (no nested loops) but (d) will become O(n).

TASK 2 APPROACH

For task 2, I read through *partial_time_series* and *covid_data* and write them to separate *ArrayLists*. I then take the *partial_time_series ArrayList*, compare its first data to the number of infections in the *covid_data ArrayList* until there is a match. If there is a match, I compare subsequent data. Once there is a mismatch, I go back and continue from where I stopped until a match is found. This process is demonstrated in ***figure 1***.

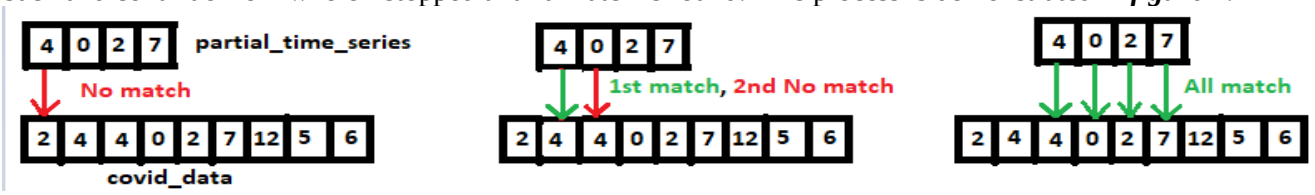

Figure 1: approach for task 2

If they all match, I check if they are from the same country. If they are, I stop looping, print the country and obtain the start date. If they are not from the same country or if some data do not match, I go back and continue my comparison from where I stopped. This approach has a time complexity of $O(n^2)$ worst case scenario and best case, O(n) since I will still have to compare all the data in *partial_time_series*.

Formula for m is from: https://mathbitsnotebook.com/Algebra1/StatisticsReg/ST2lCalculatorBest.html