Tom Nguyen

12/6/24

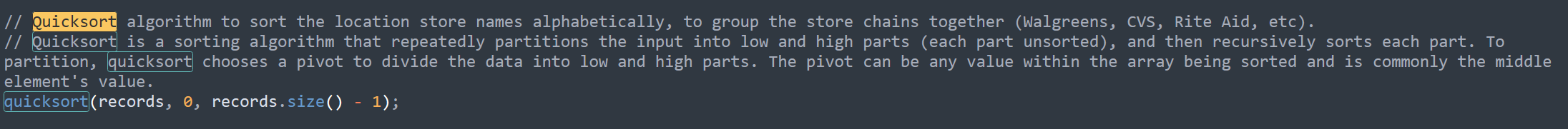
CPSC 39

1. What your game or app does and how it is useful or entertaining.
   1. The app lets the user input their city or zip code, and then it finds the COVID vaccination locations within the user’s city or zip, per what they selected. This is helpful because it encourages people to get vaccinated by showing them a close vaccination location for them. It also counts how many locations there are in the city/zip code, as well as in total in America. The datasets were gathered from the CDC.gov and vaccines.gov website. Furthermore, it displays the store’s name, city, state, zip code, phone number, operating hours, vaccine brand, minimum age, whether insurance and walk-ins are accepted, and any provider notes listed (which is useful, since some specify that they want users to call ahead for an appointment), and the location’s website.
2. Includes at least 3 algorithms as steps or a flowchart, and a snapshot of the algorithms code.
   1. Reading CSV File and Storing Data into an ArrayList
      1. Function readCSVFile (filepath: “Vaccines.gov\_\_COVID-19\_vaccinating\_provider\_locations.csv”):
         1. Initialize the empty list ‘records’
         2. Open the CSV file at the file name that I specify to start reading
         3. For each line in the CSV file,
            1. Split each line by commas using .split(“,”). (each column of the CSV file)
            2. Create a new VaccineProviderInfo object with the split value.
            3. Add the object to the ‘records’ list.
         4. Close the CSV file
      2. 
      3. A screen shot of a computer program

         Description automatically generated
      4. A screen shot of a computer

         Description automatically generated
   2. Quicksort algorithm
      1. First make a partition. Pick middle element as the pivot.
         1. Initialize lowIndex and highIndex. And set done = false.
         2. lowIndex is incremented until a value greater than or equal to the pivot is found.
            1. We can use compareToIgnoreCase to compare the strings lexicographically to see which string is “greater” than the other, in order to sort alphabetically
         3. highIndex is decreased until a value less than or equal to the pivot is found.
         4. Elements at the index of lowIndex and highIndex are swapped, moving those elements to the correct partitions.
         5. This partitioning process repeats until the indices of lowIndex and highIndex reach or pass each other. This indicates all elements have been partitioned. Then we can set done = true.
         6. Once partitioned, the algorithm returns highIndex, which is the highest index of the low partition. These partitions are not yet sorted.
      2. Quicksort is called recursively to sort low and high partitions.
         1. If highIndex is less than or equal to low index, then return because the subarray is sorted (contains one or no elements). This will only sorts if at least two elements exist. Base cases are important for recursion otherwise it will go on forever.
         2. Quicksort then partitions the array
         3. Quicksort/recursively sort the left partition
         4. Quicksort/recursively sort the right partition
      3. A screenshot of a computer program

         Description automatically generated
      4. A black background with many small colored lines

         Description automatically generated with medium confidence
      5. 
   3. Have the User type in city or zip code, and print out COVID vaccination location details in that specified area.
      1. Use the CSV arraylist data we obtained from an earlier algorithm.
      2. Ask the user if they want to type in either their city, or zip.
      3. Ask the user if they want to see all COVID vaccine brands offered. This way we can remove the location duplicates in the CSV File, if they do not wish to see all the brands offered.
      4. Have the user type in their city or zip.
      5. Either use Function printLocationsByCity or printLocationsByZip to filter and print locations by their city/ zip code.
         1. For each record in list ‘records’:
            1. If the record ‘locAdminCity’ or ‘locAdminZip’ (if the city or zip) matches the user input

Then print the record’s information, including store name, address, phone number, notes such as if they prefer appointments, etc.

For zip code, I had to use substring to grab the first 5 digits, because some of the zip codes in the CSV had 9 digits, and that’d be harder to match with the user input.

I also remembered to keep in mind some of the data in the CSV was not consistently formatted, as some cities had lower case, and some were all upper case, so I had to account for that.

* + 1. A screen shot of a computer code

       Description automatically generated
    2. A screenshot of a computer code

       Description automatically generated

1. Explain in a paragraph or 3 the algorithms that you created and how they are used in the game or app. How you created them is important, and if you used ChatGPT here you can explain how you used it - ChatGPT should not be able to write your algorithms in their totality.
   1. I created a CSV file parsing algorithm like we learned from our Starbucks assignment. I used this to read the CSV datasets I found from the CDC on cdc.gov/vaccines.gov for COVID/Flu vaccination location datasets. I stored this COVID Vaccination data in an arraylist originally to show the user all the vaccine brands, and then converted it into a hashmap to remove duplicates (since a lot of the data had duplicate stores to account for multiple vaccine brands).
   2. I created an algorithm to filter vaccination locations based on a city name or zip code that a user types in. I used what we learned in class of having a scanner and having the user type what they wanted. I then used that user input to compare it with an if loop, to see if it matched if they typed “city” or “zip”. VS code suggested using enhanced for loops for cleaner code. I also remembered to convert the city names to lowercase since not all the data in the CSV was consistent (some were uppercase). I then went through the arraylist with this for loop and then used an if loop to see if the city in the data matched the user’s input. Then I proceeded to print out specific details that I adjusted under the toString method override in the VaccineProviderInfo.java file with the setters, getters, and constructors.
   3. Using what I learned in Zybooks, I created a Quicksort algorithm to sort the location store names alphabetically, to group the store chains together (Walgreens, CVS, Rite Aid, etc). This would help the user choose which store they want to go to.
      1. Quicksort is a sorting algorithm that repeatedly partitions the input into low and high parts (each part unsorted), and then recursively sorts each part. To partition, quicksort chooses a pivot to divide the data into low and high parts. The pivot can be any value within the array being sorted and is commonly the middle element's value.
   4. I created an algorithm to count the number of vaccination locations in total in America, as well as in the specific city/zip code that the user typed in.
   5. I learned how to use a HashMap data structure from ChatGPT to count how many Flu vaccination locations per California city. I also added code to print out all the details/information of the flu vaccination providers in a city, but it would be too much output and bombard the terminal with too much information, so I commented it out for now. It would also be somewhat redundant with the previous COVID vaccination code/dataset.
   6. I used ChatGPT for how to convert the array list into a hash set (to remove duplicates from all the vaccine brand options). I also used it for the equals and hashNode overrides in VaccineProvider info, to finish the hash set. We did not learn hashSets in class, so the AI was helpful for me to learn how to use it to remove duplicates.
      1. Set<VaccineProviderInfo> uniqueRecords = new HashSet<>(records);
      2. // Convert the HashSet back to an ArrayList
      3. records = new ArrayList<>(uniqueRecords);
      4. A screen shot of a computer

         Description automatically generated
2. Discusses the Big O time of these algorithms.
   1. For flu vaccination location counting for California cities, the efficiency advantage is that Hashmap lets me access elements in constant time, O(1), using a key of the city name, to directly retrieve the value (list of vaccine providers). Arraylist would be slower and O(n) linear time complexity, because it needs to iterate through the entire list to find the specific city. As the dataset grows, Arraylist search time increases and becomes less efficient.
   2. Reading the CSV file and storing the data into an array list is O(n) linear time complexity because it takes as long as the number of lines in the CSV file. When I try to run the program, I can feel a slight pause due to reading all the lines and data in the CSV file.
   3. O(n) for converting the Array List into a Hash Set since it goes through the list.
   4. O(n) for counting all the COVID vaccination locations in total in America, since it goes through the whole list.
   5. O(n) for going through all the records and counting locations in a specific city or zip code, since it still must go through all the data.
   6. O(n) for printing locations by city/zip code, since it still has to go over each piece of data to see if it matches the city or zip code that the user typed in.
   7. Per Zybooks, quicksort’s average and best case runtimes are both O(N log N). Quicksort has several partitioning levels, the first level dividing the input into 2 parts, the second into 4 parts, the third into 8 parts, etc. At each level, the algorithm does at most N comparisons moving the lowIndex and highIndex indices. If the pivot yields two equal-sized parts, then log N levels will exist, requiring the N \* log N comparisons.
      1. Worst case run time for quicksort is O(N2), but this rarely occurs fortunately.
      2. Quicksort was a better choice than selection sort because selection sort is O(N2), which is slower. It always performs the same number of comparisons and swaps them, regardless of input (even if the list is already sorted or nearly sorted).
3. An explanation of the data structures that you used, why you chose them, and how they were used.
   1. I used an array list, and a hash set data structures. The hash set was very helpful in removing duplicates because the duplicates of the same store repeated for each vaccine brand. This is because a hash set only allows unique elements. Without removing these duplicates, the output would look very cluttered and messy. I gave the user a choice to choose whether they wanted to see all brand options available or not, since some independent stores only carry one COVID brand, and some users may be picky about which COVID vaccine brand they want. If the user did not want to see all the vaccine brand options available, then I converted the array list into a hash set to help remove duplicates.
   2. The array list was helpful for storing lists of objects (the COVID vaccination locations from the CSV dataset). Unlike arrays, an array list can dynamically resize, which is useful when reading data from the CSV file that I used for COVID vaccination locations. The size of an array must be specified when initialized and they are fixed-size data structures, meaning their size cannot be changed after creation. The array list also maintains the order as they are inserted. The array list also has methods like the get() method, which is used to get the element of a specified index within the list. This get() method was useful for counting how many locations there were, or being able to match the user’s input with the locations at the specified city or zip code.
   3. HashMap was used to store flu vaccine provider locations where the key is the city name and the value is a list of VaccineProviderInfoFlu objects for that city. It was useful for quickly looking up vaccine providers for a specific city. HashMap has the benefit of fast retrieval of all providers of a specific city using the city name as the key. A city can have multiple providers, so the HashMap is storing a list of the VaccineProviderInfoFlu objects.
   4. String[] arrays are used to split each line of the CSV into their own fields for CSV parsing (each line is divided by commas).
4. Explain a step in the design or development process where you encountered an opportunity and how you used this.
   1. I saw an opportunity to learn how to use hash set data structures to remove duplicate code. This was very helpful because the duplicates of the same store repeated for each vaccine brand. Without removing these duplicates, the output would look very cluttered and messy. I gave the user a choice to choose whether they wanted to see all brand options available or not, since some independent stores only carry one COVID brand, and some users may be picky about which COVID vaccine brand they want.
5. Explain a step in the design or development process where you encountered an error and how you resolved this.
   1. One error was trying to read in the data file, and we had an error where the program was struggling to read in the latitude/longitude sections. It was probably because I was trying to set them as double values, but there was some value that was not letting me read it in. I'm guessing it was the latitude/longitude string in the first row, but I'm not sure. To resolve it, I set the latitude/longitude equal to strings so the code would run, and I could tackle other parts of the projects. Per the Data Elements Specification pdf file, it says the latitude and longitude data types were set to be varchar(255), so that could be another reason why it did not accept the double type.
   2. The Data Elements Specification pdf file was useful, since I was trying to run the zip code as an int, but it was meant to be run as a string. I got a lot of errors trying to run it as an int, so I had to go back and change it all back to string.
   3. Another error was that in the CSV dataset, some of the the zipcodes had 9 digits and some had 5 digit zipcodes. To solve this, I used the substring() method, to take the first 5 digits of the zip code column and ignore the extra 4 digits in the 9 digit zipcodes.
   4. Some cities were in all capital letters, while some weren’t. This made it difficult to compare user input to the city they typed in, as it would not capture all the results. To solve this, I was able to use toLowerCase or .equalsIgnoreCase.
   5. Another rare error was that my city of Stockton also showed up a few times in other states, like Illinois. To resolve this, I had to specify the state of California.
   6. One error I came across was why my Quicksort was not working. It was because I converted the Array list to a Hashset later in the code, and back to an array list. To solve this, I put the quicksort after the new array list. I was confused because the quicksort was working when I typed in ‘y’, but this was because I had a separate if statement where it returned early and didn’t convert to a hash set and back to an array list.
6. Explain what you would change or add in the next version of your game or app.
   1. It would be cool to add graphics or pictures of the pharmacy chain (Walmart, CVS, Walgreens).
   2. I found a zip code and coordinate CSV file that I could have added. It would be cool to somehow find the accurate distance between the user’s location and the nearest closest vaccination location.
   3. I would have liked to add more COVID statistics from data sets to help encourage the user to get vaccinated and to emphasize the seriousness of it. It would be nice to also add visuals and compare which city or population is most affected by COVID.
   4. I could maybe take a question out of the user input. Instead of asking the user to type in “city” or “zip”, I could try to make the program automatically detect if the user is entering a string or an integer, because string = city, and a number = zip code.

A blue and white list with white text

Description automatically generated