WGU

Nicolas SAM <nsam2@wgu.edu>

C950 Outreach...

1 message

Jack Lusby <jack.lusby@wgu.edu> Thu, Jun 17, 2021 at 5:48 PM
To: "nsam2@wgu.edu" <nsam2@wgu.edu>

WGU Course C950:
Data Structures and Algorithms II

Hello again!

Just wanted to make sure that you have everything you need from me and the Computer Science Team for C950.

In general, the task directions make the project sound more difficult than it is. Check out our C950 FAQ page and read the student comment in the course chatter. Many common questions are answered there, and other students provide some great advice.

Some helpful points for the Performance Assessment:

- By “optimal,” they mean less than 140 miles. The lowest we’ve seen is 76 miles.
- You can manually load the trucks if you find difficulty in creating a prioritization algorithm for the packages. Keep in mind that if you do this, you must have a dynamic delivery algorithm that will determine the truck route. You cannot deliver the packages in the same order that they were manually loaded.
- You can hold a truck at the hub until a given time. Keep in mind that you want to minimize miles, not time. This also helps with the delayed packages.
- You can treat the wrong address package like a delayed package.
- Do NOT follow the sample Core Algorithm Overview for your write-up. Instead, use each rubric item as section headers in your document and respond to each one underneath of it. This will make your submission easier to grade and the evaluation feedback easier to locate.

Most students also get overwhelmed with lack of specific directions / requirements for this performance assessment. Here is one way to break it down into 4 phases:

NOTE: If the below approach confuses you, please feel free to implement another approach. You can also make an appointment with me so that we can go over this.

A) Phase I – Load Input Data

a. Validate the data found in the Excel spreadsheets.

i. Ensure that all addresses listed in the Packages spreadsheet match EXACTLY with the addresses listed in the Distances spreadsheet.

1. Some addresses that have a direction in the street name may not match and you will need to update them. For example, one address might say “North” with the street in one file but “N” in the other file.

2. This is important because the address is the primary key that links the package data to the distance data.

b. Export the Excel spreadsheets into a CSV comma delimited file that you can use. (OPTION 1)

i. You can use Excel to do this automatically through the File->Save As options (and select CSV)

ii. Open the .csv files and clean it up so it only has the data you need

1. For the packages.csv file, you need 40 rows (each with 8 fields / columns) of package data only.

2. For the distances.csv file, you need 27 rows (each with 27 fields / columns) of numeric distance values only.

3. The addresses that correlate to the distance file should be stored in a separate addresses.csv file that are listed in the same order as they appear in the distance Excel spreadsheet. Then you can correlate a specific package address to a specific distance table index by finding the address location within the address data.

iii. Create a hash table from scratch with an insert() and lookup() function that is called out in the requirements.

1. Verify that the hash table insert() and lookup() function works properly using some test data.

2. The learning resource within the classroom has an entire chapter dedicated to hash tables along with full Python examples.

iv. Utilize the “csvreader” library that is built into python to extract the package data from the csv file from above.

1. Create a Package class that includes the 8 fields from the package csv file.

2. Read a row from the package csv file and create a Package object with it

3. Add the package object to the hash table using the insert() function.

v. Utilize the “csvreader” library that is built into python to extract the distance data from the csv file from above.

1. Create a 2D array (list of lists) that will store the distance information (OPTION 1.1)

a. Read a row from the distance csv file and insert it into the 2D array at the proper position.

2. OR Create Node objects with that are connected to other Node objects via edge weights. (OPTION 1.2)

a. Read a row from the distance csv file and populate the Node and edge weight information at the proper position.

vi. Utilize the “csvreader” library that is built into python to extract the address data from the csv file from above.

1. Utilize a list or dictionary to store the address information

2. Create an address_lookup(string address) function that will translate the specific string address into a numeric value that can be returned.

a. NOTE: The address_lookup() function is only needed if you utilize the 2D array (from OPTION 1.1 above).

c. OR Hard code the Excel spreadsheet data directly into your program (OPTION 2)

i. Create a hash table from scratch with an insert() and lookup() function that is called out in the requirements.

1. Verify that the hash table insert() and lookup() function works properly using some test data.

2. The learning resource within the classroom has an entire chapter dedicated to hash tables along with full Python examples.

ii. Create a Package class that includes the 8 fields needed for the Excel spreadsheet.

1. Create 40 package objects with the appropriate data for each one

2. Add each package object to the hash table using the insert() function.

iii. Create a 2D array (list of lists) that will store the distance information (OPTION 2.1)

1. Manually enter data from distance Excel file into the 2D array at the proper position.

2. Utilize a list or dictionary to store the address information from the distance Excel spreadsheet.

a. Create an address_lookup(string address) function that will translate the specific string address into a numeric value that represents the row / column location of the address within the 2D distance array.

iv. OR Create Node objects with that are connected to other Node objects via edge weights. (OPTION 2.2)

1. Manually enter data from distance Excel file into the proper Node objects.

B) Phase II – Implement Dynamic Core Algorithm

a. Optimal Industry Solution (OPTION 1)

i. Create an algorithm that will look at ALL package data (distance, time restrictions, special notes) and will determine which package goes on which truck and in which order.

1. Greedy algorithms are good for this because they can optimize solutions based on more than one parameter.

2. Dijkstra's is good to produce a minimum spanning tree that usually contains branches. In our case, you need the best round trip solution without branches. So be careful choosing Dijkstra's as your solution unless you really understand the algorithm well and know how you are going to convert the MST to a round trip solution.

ii. Load the trucks with the appropriate packages.

1. This typically means creating a list that stores the package ids associated with each truck

a. Later, you can use the hash table.lookup(package id) method to access that package associated with each package id.

b. The hash table should be a primary table for the latest information about the packages.

c. Do not duplicate the package objects outside of the hash table. This will lead to synchronization issues later on.

iii. Deliver the packages

1. Loop through the packages assigned to the truck in order (since the order was determined by the algorithm above)

2. Deliver the package by timestamping it in the hash table.

a. NOTE: Don't be afraid to add an additional timestamp field within the package class.

3. Keep track of the distance traveled starting from the hub

4. Display total round-trip distance once all packages have been delivered.

a. This needs to include the distance back to the hub when the truck is finished

b. OR Separate Prioritization and Delivery Algorithms (OPTION 2)

i. Prioritization Algorithm

1. Create an algorithm that will look at the package data ONLY (no distances) and will figure out which package goes on which truck. (OPTION 2.1)

a. This typically means creating a list that stores the package ids associated with each truck

i. Later, you can use the hash table.lookup(package id) method to access that package associated with each package id.

ii. The hash table should be a primary table for the latest information about the packages.

iii. Do not duplicate the package objects outside of the hash table. This will lead to synchronization issues later on.

2. OR Manually load the trucks (OPTION 2.2)

a. If an algorithmic approach becomes too difficult, evaluation is fine with "manual loading of trucks".

b. This means that you can hard code into your program which package goes on which truck.

c. If you do this, you cannot deliver the packages in the same order in which they were loaded.

ii. Delivery Algorithm

1. Loop through the packages assigned to the truck (since the order was determined by the algorithm above)

a. For each package, have the truck figure out what the next best address is to go to (shortest distance)

i. Nearest neighbor algorithm works well for this and will come in under 140 miles.

2. Deliver the package by timestamping it in the hash table.

    a. NOTE: Don't be afraid to add an additional timestamp field within the package class.

3. Keep track of the distance traveled starting from the hub

4. Display total round-trip distance once all packages have been delivered.

    a. This needs to include the distance back to the hub when the truck is finished

C) Phase III – Implement Command Line Interface

a. At a minimum, you need a command line interface that will ask the user for a specific time.

b. Once you have the time, display the status of all packages at that specific time as Delivered or Not Delivered.

i. You can utilize the timestamp of the packages in the hash table to determine their delivered state at the particular time.

1. If the timestamp of the package is before the user time, the package is Delivered and vice versa.

c. Ensure that the package status output is a neatly formatted table listed in ascending order by package ID.

i. You can loop through all package ids and use the hash table lookup() function to retrieve each package.

D) Phase IV – Documentation

a. Python Code Comments

i. Ensure that each class and function has a brief description above it as well as it's Big O runtime and Big O space complexity within the Python code itself.

b. Write Up

i. Utilize each rubric item as a sub heading in your write-up.

ii. The write-up should act as a table of contents to your solution. Do not be afraid to call out specific code files and line numbers to be specific.

iii. Evaluation wants to see that you can connect the dots between the learning resource material and your solution.

iv. DO NOT USE THE CORE ALGORITHM OVERVIEW DOCUMENT AS YOUR TEMPLATE AS STATED IN THE DIRECTIONS. This leads to evaluation performing an Easter egg hunt to find each rubric item.

v. Ensure that your write up is a separate document and not in the code itself.

Here is a recommended study guide:

C950: Data Structures and Algorithms II (Python)
Recommended Pacing Guide

- This table describes my recommended eight-week duration C950 pacing guide.
- If desired, this standard pace can be either accelerated or decelerated.
- Please discuss non-standard pacing with your course instructor and/or program mentor.

Week	Online zyBook:	Other
1	* Ch. 1 Getting Started * Ch. 2 Introduction * Ch. 3 Algorithms	* Explore the C950 home page, including the *Announcements* and *Course Tips*. * Setup Development Environment - Install a recent Python 3. - Install community edition of the PyCharm IDE.
2	* Ch. 4 Trees * Ch. 5 Balanced Trees	
3	* Ch. 6 Graphs	
4	* Ch. 7 Hashing	
5	* Ch. 8 Sets * Ch. 9 Dictionaries	
6	--	* Review/Prepare for Performance Assessment. C950 FAQ page
7	--	* Review/Start the Performance Assessment.
8	--	* Complete the Performance Assessment.

SCHEDULE LINK: Here is the link to schedule an appointment with any currently available C950 instructors.

You can contact me via jack.lusby@wgu.edu, 1-385-428-2680, or my scheduling link.

Thanks for your attention and happy studies! Please let me know how I can help.

Prof. Lusby

Jack C Lusby

Course Instructor, College of Information Technology

Western Governors University

Toll Free 1.877.435.7948 Ext. 2680

jack.lusby@wgu.edu

WebEx Meeting Info
Link: https://wgu.webex.com/meet/jack.lusby
Call In Phone: 1-855-282-6330 #
Access Code: 928 330 069 #
Attendee ID: #