COMP1002 - Data Structures and Algorithms Final Assignment

Optimising Urban Parcel Logistics Using Data Structures and Algorithms

Description

CityDrop Logistics is expanding its operations in a metropolitan area and required me to develop a suite of inter-connected software modules to optimise there delivery network.

I ended up developing a system that is divided into four modules that are interconnected:

1. Graph Based Route Planning :- This allows CityDrop Logistics to determine the most effective and efficient paths from a hub to a destination
2. Hash Based Customer Lookup :- This allows CityDrop Logistics to retrieve customer information quickly
3. Heap Based Parcel Scheduling :- This allows CityDrop Logistics to prioritise deliveries based on urgency and customer priority
4. Sorting of Delivery Records :- This allows CityDrop Logistics to organise records at the end of the day usign Merge and Quick sorts

Getting Started

Dependencies

- Python 3.13
  - ☑ Numpy

Recommended

- Microsoft VSCode
  - ☑ Pylance
  - ☑ Python
  - ☑ Python Debugger
  - ☑ Rainbow CSV

Installing

1. Download the zipfile either available on BlackBoard
2. Unsip the contents of the file into a clean folder. The name can either be the same as the zipfile or something else 😄

Executing program

Please note that all instructions provided to run the program assume that you are utilising Visual Studio Code. If you do not have access to this please refer to Recommended Software

- How to run the program

1. Open *Visual Studio Code*
2. Ensure that all modules outlined in recommended are installed.
3. Open the working directory of the project by pressing ***"File"*** then ***"Open Folder"***.
4. Press ***"File"*** --> ***"Preferences"*** --> ***"Settings"***.
5. In the search bar of the ***"Settings"*** page type in ***"Terminal Scrollback"***
6. Find the option labeled as ***"Terminal > Integrated: Scrollback"*** and set the value to **10000**. Save this by pressing *CTRL+S or CMD+S* to save. This is done for the test cases for the sorting algorithms as the display outputs can get very large.
7. Open the DSAAssignment.py file and Run the file by pressing *F5* or the *Play Button* on the top right of the source-code window.
8. Now you will be presented with the ***Delivery System Main Menu***

- Interacting with the Software to Run Test Cases This section will cover details on how to ***run*** the pre-determined test cases.

1. Open the *Graph Based Route Planning System* by typing *1* into the input and pressing *ENTER*
2. Now you will be presented with the *Graph Based Route Planning System*, here select *OPTION* **10** which will load the test graph.
3. To do this, type *8* into the selection prompt and press *ENTER*. You will now get an output of all of the test graph information including Adjacency List/Matrix, BFS and DFS Traversals, Dijkstra's Shortest Path Results and Cycles.
4. In the terminal prompt, now type *12* to return back to the ***Delivery System Main Menu***
5. Open the *Hash Based Customer Look Up* by typing *2* into the input and pressing *ENTER*.
6. Now you will be presented with the *Hash Based Customer Lookup*, here select *OPTION* **7** this will allow you to load the preconfigured CSV file which contains customer data.
7. When prompted to enter the CSV filename, type in `customertest.csv` and press *ENTER*
8. To verify that data has been loaded, the output should mention *Hash table loaded from CSV*.
9. Further verfication can be done by entering *OPTION* **10** and pressing *ENTER* in the terminal, to display the data which has been entered into the hash table.
10. You can also see the collision handling by entering *OPTION* **9** and pressing *ENTER* which will show the hash table values of two customer ID's "*aa*" and "*ah*" which then shows that they are indexed to subsequent values due to linear probing.
11. We can add a new customer to the Hash Table by entering *OPTION* **1** and pressing *ENTER*.
12. Start by selecting entering a *starting node* which can be selected from the outputted *Adjacency List*. (P.S. You are welcome to try Node M to any other node, and the end of the prompts it will issue an error saying that XXX is unreachable from M)
13. Then enter the *Customer ID*. Ensure COMMAS are not included in the *Customer ID*.
14. Enter the *Customer Name*. Ensure COMMAS are not included in the *Customer Name*
15. Enter the *Address* which can be selected from the outputted *Adjacency List* above. (P.S. You are welcome to try Node M to any other node, and the end of the prompts it will issue an error saying that M is unreachable from XXX)
16. The system will now output the *Calculated Time* from Dijkstra's Shortest Path algorithm and the path it takes to get there.
17. Enter the *Prioirty* from 1 to 5 (1 = Lowest, 5 = Highest). (P.S. You are welcome to try other values, and the system will reset your customer input progress and issue an error)
18. You will now be returned back to the menu for *Hash Based Customer Lookup*. From here you can verify the addition of a customer by displaying the hash table *OPTION* **10**

19. You can also lookup a customer by entering *OPTION* **2** and entering there *Customer ID*. This will return the Customer Name. Alternatively using *OPTION* **4** which will just simply return exists/does not exist (P.S. You are welcome to try something not in the table)
20. To remove a customer enter *OPTION* **3** and enter the *Customer ID*. For the purposes of demonstration try *002* and then display the hash table using *OPTION* **10**. You will notice that *Index 3* now returns *Formley Used* as this was a mapped index.
21. Enter **12** to return back to the ***Delivery System Main Menu***
22. Now enter **3** to enter the *Heap Based Parcel Scheduling*
23. Enter **6** to load the hash table data from module two (*Hash Based Customer Look Up*). This will mention that '*Data Loaded*'
24. Enter **3** to the terminal to show the heap. Note that this heap is showing the RAW HEAP STRUCTURE AND IT IS NOT SORTED AS A LIST FROM HIGHEST PRIORITY TO LOWEST
25. We can manually add a parcel to the scheduler by entering **1** and following the same procedure for the *Hash Based Customer Look Up* (P.S. The same mis-inputs will be detected here too!)
26. After entering all the required data, the *Heap* will be displayed showing the newly updated *Heap*. Also take note of the delivery status which has updated from "Not-Delivered" to "In-Transit"
27. We can remove a customer by entering **2**. This will remove the highest priority delivery.
28. These removed items can be exported to a CSV file by entering *OPTION* **7** and typing in a `name.csv`. Note the exported CSV file has the delivery status changed to "*Delivered*". (P.S. You can remove as many of the items as you want until the heap is empty before an error is thrown.)
29. Continue to remove deliveries from the *heap* until the heap is empty and the error shows up and export the CSV to the SAME CSV file you named earlier. This will result in a dynamically updating CSV file. Note that the output of this CSV file will be in a sorted order from highest to lowest priority
30. Now enter *OPTION* **4** to load the CSV file you just exported as we need this for the final module and its interoperability.
31. By entering *OPTION* **3** the loaded CSV file can be verified as you can see the loaded heap.
32. Enter *OPTION* **8** to return to ***Delivery System Main Menu***.
33. Enter *OPTION* **4** to enter the *Sort Delivery Records Menu*.
34. By entering *OPTION*(s) **1** to **4** you can perform the available sorting methods and recieve an output of the *time* taken to complete the sort and the sorted output. This will sort records based on *time* (lowest to highest).
35. For larger test cases, enter *OPTION* **5** to return back to ***Delivery System Main Menu***.
36. Now enter *OPTION* **5** to enter the *Test Sorts* module.
37. Enter `test_data.csv` to the input and press *ENTER* as this file contains the test cases.
38. By entering *OPTION*(s) **1** to **3** we can perform sorts on 100, 500 and 1000 items respectivley. Each *OPTION* will perform a ***random***, ***nearly-sorted*** and ***reversed*** sort with *time* taken to perform the sort and its *output*.
39. By entering *OPTION* **4** you can return back to the main menu

- Additional Menu Tinkering

1. In the *Graph Based Route Planning* module you can add hubs/intersections by using *OPTION* **1** and entering the location. To confirm this addition enter *OPTION* **5** to show the *Adjacency List* to show the graph structure.
2. A route can be added by entering *OPTION* **2** and entering the *start* and *end* location as well as the *time/weight* of the route.

3. A Location/Hub/Intersection can be deleted by entering *OPTION* **3**. This will delete all of the associated routes to that Location/Hub/Intersection. (You are welcome to try a location that does not exist).
4. A route can be deleted by using *OPTION* **4** and entering the two respective locations. (You are welcome to try a location that does not exist).
5. The graph can be displayed using *OPTION*s **5** and **6** which will give you the Adjacency List and Matrix respectivley. The Matrix will show the edge weights of each path.
6. A BFS and DFS search can be completed using *OPTION*s **7** and **8** and entering a start location. (You are welcome to try locations that do not exist)
7. The Dijsktra's shortest path can be outputted using *OPTION* **9** and entering the start node. (You are welcome to try locations that do not exist)
8. Finally, in the *Hash Based Customer Lookup* you can save the Hash Table as a CSV in the same way as other modules. Similarly in *Heap Based Parcel Scheduling* you can load data in a CSV and export the heap as a CSV (no deliveries completed) in the same manner as other modules.
9. You are also able to pull data from Module 2 back into Module 3 using the function built into module 2. (Note that this might be a bit buggy, so your milage may vary. I recommend using the CSV imports)

Author

Muhammad Annas Atif | 22224125 Contact Me

Version History

- 1.0
    - Finalised all modules: Graph Based Route Planning, Hash Based Customer Lookup, Heap Based Parcel Scheduling, and Sorting of Delivery Records.
    - Completed CSV import/export functionality for interoperability between modules.
    - Optimised and debugged all modules for final submission.
    - Added annotations and comments for better code readability.
- 0.9
    - Linked modules together for seamless integration.
    - Completed Module 3 (Heap Based Parcel Scheduling) and Module 4 (Sorting of Delivery Records).
    - Fixed bugs and improved error handling across modules.
- 0.8
    - Completed Module 2 (Hash Based Customer Lookup) and started Module 3.
    - Implemented BFS and DFS functionality for Graph Based Route Planning.
    - Added Dijkstra's Shortest Path algorithm.
- 0.7
    - Completed Module 1 (Graph Based Route Planning).
    - Added adjacency list and matrix representations for the graph.
    - Fixed issues with graph definitions and edge cases.

- 0.6
    - Initial implementation of core modules.
    - Added basic CSV handling and data structures.
    - Created README with setup and usage instructions.
- 0.1
    - Project initialisation.
    - Basic structure and initial commit.

Acknowledgments

- Thank you to the tutors of this unit for coping with me all semester 😄
- Me, Myself and I.