The R4dices

**A Program Specification Document**

**for the course on**

**Introduction to Data Structures and Algorithms**

**(DASALGO)**

Submitted by:

**Ang, Jerald Edric**

**Lao, Hans Dylan**

**Lumba, Jon Jitsugen**

**Morales, Jomari Alex A.**

**Rivera, Joanna Pauline**

Teacher

**July 24, 2018**

1. Introduction

* What program did you develop?

Our group made a program that enables the user to simulate a mail delivery service. Using a premade map, the program aims to show the user the whole process of mail delivery which includes the arrangement of each mail utilizing methods such as the Prim’s algorithm, route planning towards the mailman’s destinations, and a proper illustration of the whole mail delivery simulation. Moreover, with the arrangement of the mail, thus the delivery path, is processed and arranged by Radix Sort in order to have a much better and proficient way in the handling and sorting of each mail a postman delivers.

* What can the user do with your program? (e.g. The user will be able to select the starting post office. The user will also be able to input the name and address of the recipient of the mails.)

With this program, the user is able to input delivery locations, determine the best path based on these locations, and display the result on a map. The program is advantageous to the public, more specifically modern jobs relating to business and entrepreneurship, as it finds a faster and more efficient route especially in terms of delivery services. Some real life examples of this would include DLSU bids-to-pick, whereas if the delivery involves the shipment of various products, the carrier would be able to utilize this simulation in order to determine the best route, saving transportation money as well as time.

II. Data Structures

1. ArrayLists - used for convenience of auto adjusting once an item is added or deleted.

|  |  |
| --- | --- |
| **Name** | **Purpose** |
| mail | To store all the mails inputted by the user. |
| maplocations | To store all the addresses with its respective post office and distances. |
| postloc | To store the location index needed for prioritizing mails |
| sub | Acts as place holders to do an auto-generated method stub |
| sentmails | Temporary placeholders for the batch of mails sent within a post |
| mapdetails | Temporary placeholders to copy the mail attributes from the .csv file to the ArrayList of mails |

2. Arrays - used for splicing a string of characters into parts, having a fixed set of pattern with its indices.

|  |  |
| --- | --- |
| **Name** | **Purpose** |
| spost | Converts a string into a string with the first letter in uppercase |
| country | Acts as demarcation for the information included in the CSV file. It separates the columns into different elements |

III. Algorithms

In this chapter the different searching and/or sorting algorithms that you used in your program are presented.

1. Searching Algorithms

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Purpose** |
| Linear | Checking process that involves a one-by-one comparison. | To sort the post by index and by distance |

2. Sorting Algorithms

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Purpose** |
| Insertion | A sorting process that involves inserting values in the list and adjusting the preceding/following data stored | To sort the arrangement of the postman’s mails from most prioritized to least prioritized |
| Radix | A sorting process that involves checking the radix positions of positive whole numbers | To sort the distances of the inputted destinations to their respective post offices |
| Swap | Exchanging of one’s attributes to the other | To implement the swapping of mail components |

3. Greedy Algorithms

|  |  |  |
| --- | --- | --- |
| **Name** | **Description** | **Purpose** |
| Prim’s Algorithm | To sort the arrangement of the postman’s mails from the most prioritized to the least prioritized | In the case of sorting starting from one location, the use of Prim’s algorithm is most effective as it traverses routes of the least amount of distance. |

IV. Program Flowchart

Figure 1. Program Flowchart

Ask for Distances (Map)

Choose Starting Location

Input Mail Addresses

Sort Mail

Deliver within the Post Office area

Empty?

Play Again?

Program Exit

Check Mail Priority

Go to Location

Yes

No

No

Yes

Start New Delivery

Simulation

Pre-Simulation

Program Start

Describe in detail the things that your program can do. For each function/feature, state the assumptions, scope and limitations. Include the screenshots of your program as well.

1. Read Map File

The user can choose the Map file in CSV (Comma-Separated Values) format. The first line must be the category names, while the rest of the lines must be the data. The program would not be able to read the file if it contains commas that are not separators, if there are extra separators, and if there are missing separators. If the file selected is not a CSV file, it will simply display an error. The program will not begin unless a valid map is chosen.



Figure 2. Read Map File Screenshot

Figure 3. No Map Selected Error Screenshot



Figure 4. Invalid File Selected Error Screenshot

2. Select Post Office

In line with the format of the map provided, the user can select the initial post office from the 4 post offices: Manila City Post Office, Quezon City Post Office, Pasay City Post Office, Makati City Post Office. This feature was made to be non case sensitive, but must contain the words “post office” after the location. Furthermore, any post offices outside the data given by the map will not be considered.

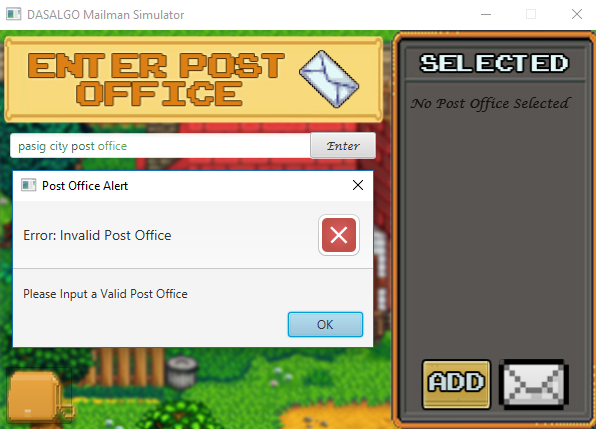
Figure 5. Invalid Post Office Screenshot

Figure 6. Enter Post Office Screenshot

Figure 7. Post Office Successfully Selected

3. Input Mail Addresses

The user can now input a particular school location or address where the postman will deliver the mail. For simplicity, all of the addresses within the selected map are already provided in a choicebox, allowing the user to select one and press the add button easily. Similarly, the addresses are only valid according to the map. This feature repeats for every post visited, and will only stop when there are no mails left.

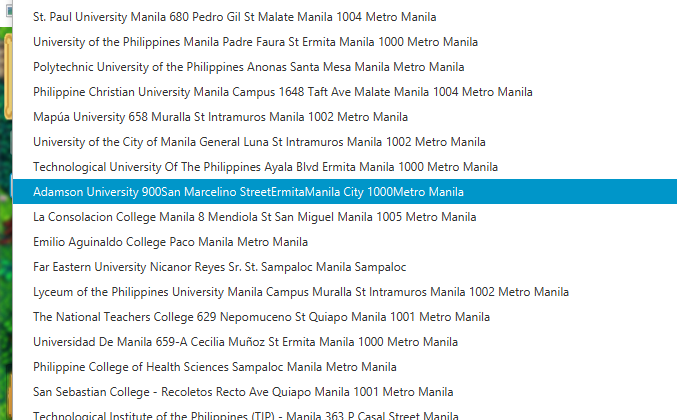
Figure 8. Input Mail Address Screenshot

Figure 9. Input Mail Address Screenshot 2

Figure 10. Mail Address Successfully Added

4. Sorting by Location and Distance

After input, the simulation will then sort out the mails in two categories: by Location and by Distance. Since the input of mails allows different post office locations, the program will sort out which mails are to be delivered within the post. For case 2 (having two different posts), this allows the program to set aside mails from another post. For case 3 (multiple posts), this prioritizes which post location will go next, with the first input of a different post being served first, much like a queue. Furthermore, for mails of similar posts, the program also sorts their respective distances from the map through mapping the shortest route possible, as of real life planning. The program makes use of a greedy algorithm called prim’s method.

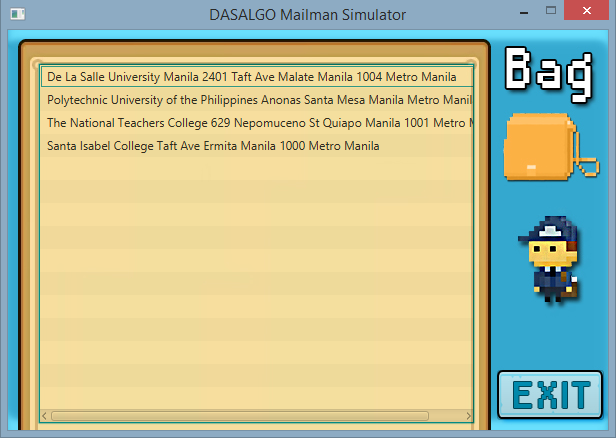


Figure 11. Mailman’s Bag Screen Showing Unsorted Mails Screenshot



Figure 12. Delivery of Mails in a Strategized Route Screenshot

5. Choice to Play Again

When the simulation exits due to having no other mails in the mailman’s bag, it asks the user if they want to run another test. This enables them to start the entire program fresh, having no correlation to past attempts. To do this, the program presents the user a yes or no question. If the user chooses to click yes, The program will loop back to the opening scene so that the player may choose to input a different map file. If the user chooses to click no, the program simply terminates.



Figure 13. Asking to Mail Again Screenshot

VI. References

Agrawal, D. (n.d.). Radix sort. Geeksforgeeks. Retrieved from [https://www.geeksforgeeks.org/ radix-sort/](https://www.geeksforgeeks.org/radix-sort/)

Eker, S. (2014, December 11). A way to pause a method for 1 second. Retrieved from https:// stackoverflow.com/questions/27432472/a-way-to-pause-a-method-for-1-second/

27432483

Envelope - ASCII Art. (n.d.). ASCII. Retrieved from <http://ascii.co.uk/art/envelope>

Imms, D. (2016, December 28). Radix sort (LSD). Growing with the web. Retrieved from <http://www.growingwiththeweb.com/sorting/radix-sort-lsd/>

Ransom, M. (2010, November 10). When should we use radix sort. Stack overflow. Retrieved from <https://stackoverflow.com/questions/4146843/when-should-we-use-radix-sort>

Sefidgar, R. (2014). Prim’s Algorithm. Retrieved from

<https://www-m9.ma.tum.de/graph-algorithms/mst-prim/index_en.html>

Tierman, P. (2000, April 1). Radix sort revisited. Codercorner. Retrieved fromhttp:// codercorner.com/RadixSortRevisited.htm

Walters, A. (2014, July 9). Radix sort in C. Austin G. Walters. Retrieved from https:// austingwalters.com/radix-sort-in-c/