### DASALGO Machine Project; Tag Sorting Algorithm

### A Program Specification Document

### for the course on

### Introduction to Data Structures and Algorithms

### (DASALGO)

### Submitted by

### Aquino, Raphael

### Batinga, Jewn

### Chua, Gian

### Wong, Michael Edmund

### 

### Rivera, Joanna Pauline

### Teacher

### August 24, 2018

### Introduction

### This program serves as a visual representation for tag sorting algorithm in the form of ‘mail delivery’. Users may input multiple entries, thereafter called as ‘mails’ and will be sorted either according to distance or time it was inputted, while being on the same area of response, or ‘post offices.’ While the program is being used, the user may change how fast the application would do so, even after it was executed. It is also possible to completely halt the system and later be resumed to its labor. If the user wishes to choose a new set of ‘mails’, they may do so by clicking the ‘reset’ button.

### The program provides a comprehensible representation of each criteria, whereas the locations will be represented as red circles, the ‘post offices’ as slightly bigger blue circles, white lines that serve as paths, and a light blue arrow that represents the ‘mailman.’

### To begin, the user must select at least 1 mail. These mails must have a recipient and will beforehand be provided with random names stored in the application. The user may choose a different name if they wish to do so. Clicking the ‘confirm’ button will show a ‘map-like’ structure of which locations will be involved with the mails. In order to begin, the user must select where the mailman must begin. The blue circles, or the ‘post offices’ are the only locations where the mailman can go to – clicking on them will set the starting location to it. Pressing the ‘play’ button, or the arrow pointing towards the right direction located at the bottom-left corner, will execute the application’s procedure until it is done.

### The procedure of which is as followed: message sorting, where the mailman will arrange its mails before delivering; mail delivery, where the mailman will attempt to locate the designated location and deliver the mail.

### Data Structures

### In this chapter, the different data structures that you applied in your program are presented, along with the description of the purpose of each data structure.

### Arrays

### Works as a database, where all data must be fixed and not be modified. These serve as means to demonstrate ‘tag sorting’.

|  |  |
| --- | --- |
| Name | Purpose |
| Created Mails | Mails that the user has created. These serve as reference and will not be tampered during the program’s execution. This includes all the previous mails that were already sent (not visible in the screen anymore). |

### Linked Lists

### Used in actual sorting procedures. These only contain the indices that is desired to point to ‘Created Mails’ array, fulfilling the criteria of a ‘tag sort’ algorithm.

|  |  |
| --- | --- |
| Name | Purpose |
| Unsorted Mails | Mails that have yet to be sent and sorted. These are assorted and may be designated to different cities. |
| Sorted Mails | Mails that are sorted according to a path that will intercept all the designated addresses with the shortest length. The designated addresses will be in the same city. |

### Tree

### The output after converting the ‘.csv’ file.

|  |  |
| --- | --- |
| Name | Purpose |
| Addresses | Converted version of the ‘.csv’ file. These helps ‘visualize’ the ‘map’ and provide an easier control when creating ‘routes’ for the ‘mailman’ to deliver mails. |

### Algorithms

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

### Searching Algorithms

|  |  |  |
| --- | --- | --- |
| Name | Description | Purpose |
| Linear | Looks for mails with the same city as the currently selected post office’s city. | Used to separate mails that are within the currently selected post office’s domain. |

### Greedy Algorithms

|  |  |  |
| --- | --- | --- |
| Name | Description | Purpose |
| Kruskal’s | Looks for the shortest path that intercepts all addresses that has pending mails. Each address does not require to have a pending mail and only acts as a ‘shortcut’ for faster travel. This will attempt ALL possible combinations, but not completely traverse if there is already a known route that is shorter. | Used for path computation to find the shortest route from the post office. The resulting route is ‘cached’ to prevent repeating the computation. |

### Program Flowchart

### Insert the flowchart of your program here. Using Microsoft Word, you will find various flowcharting shapes under Insert > Shapes.

### Figure 1. Flowchart

### Program Features

### Read Map File

### The user can input 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. The user may drop a .csv file directly on the screen or paste the .csv file’s contents into the screen via clipboard (CTRL + C and CTRL + V).

### 

### Figure 2. Read Map File Screenshot

### Create Mail

### The user can create mails to a designated location with the recipient’s name. This also accounts the date and time it was created.

### 

### Figure 3. Read Map File Screenshot

### Select Post Office

### 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. These are done by clicking the blue circles.

### 

### Figure 4. Select Post Office (Before selecting) Screenshot

### 

### Figure 5. Select Post Office (After selecting) Screenshot

### References

### GeeksforGeeks (N.D.). *Tag Sort (To get both sorted and original)*. Retrieved July 7, 2018 from https://www.geeksforgeeks.org/tag-sort/