ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ 2016-2017

ΘΕΜΑ 1

ΚΑΤΣΑΡΑΓΑΚΗΣ ΕΜΜΑΝΟΥΗΛ-03113059

ΚΟΥΒΑΤΖΗΣ ΚΩΣΤΑΝΤΙΝΟΣ-03113112

7^{Ο} ΕΞΑΜΗΝΟ ΣΗΜΜΥ

Στα πλαίσια αυτής της εργασίας κληθήκαμε να υλοποιήσουμε το βασικό κορμό μιας ευφυούς υπηρεσίας εξυπηρέτησης πελατών ταξί. Πιο συγκεκριμένα, γνωρίζοντας τις συντεταγμένες του πελάτη, τις συντεταγμένες των ελευθερών οδηγών ταξί και ένα id για καθέ έναν απ'αυτούς, καθώς επίσης και τις γεωγραφικές συντεταγμένες διαφόρων σημείων των οδών, μαζί με το αντίστοιχο id και το όνομα της οδού για κάθε σημείο, υλοποιήσαμε ένα πρόγραμμα, το οποίο υπολογίζει και εμφανίζει για κάθε έναν από τους διαθέσιμους οδηγούς των ταξί τη βέλτιστη διαδρομή προς τον πελάτη και επιλέγει το ταξί που βρίσκεται πιο κοντά στον πελάτη.

Σύντομη Επεξήγηση του Κώδικα και της Υλοποίησης

Το πρόγραμμα υλοποιήθηκε στο BlueJ version 3.1.7 .

Αρχικά, λοιπόν, διαβάζουμε τα δεδομένα μας από τα αντίστοιχα αρχεία εισόδου 'taxis.csv' για τα στοιχεία των οδηγών, 'nodes.csv' για τα στοιχεία των κόμβων και 'client.csv' για τα στοιχεία του πελάτη. Έχουμε υλοποιήσει επιπλέον τις κλάσεις Taxi_driver, Node και Client, που κάθε μία έχει τα αντίστοιχα πεδία και μεθόδους που χρειαζόμαστε για τους οδηγούς, τους κόμβους(σημεία στο χάρτη) και τους πελάτες. Για τον πελάτη δημιουργούμε ένα αντικείμενο τύπου Client και για τους οδηγούς των ταξί ένα ArrayList από αντικείμενα τύπου Taxi_driver, όπου κάθε ένα από αυτά περιέχει τα δεδομένα για τον αντίστοιχο οδηγό. Όσον αφορά την αποθήκευση των κόμβων στη μνήμη μας αυτό το πετύχαμε τοποθετόντας σε ένα HashMap αντικείμενα τύπου Node. Για κάθε

στοιχείο του HashMap θεωρούμε ως κλειδί το concatenation των συντεταγμένων των κόμβων ως String. Κάθε Node έχει ως πεδία τις συντεταγμένες του, το id του, το όνομά του, μία σημαία η οποία ενημερώνεται κάθε φορά ανάλογα με το αν ο τρέχων οδηγός ταξί έχει περάσει απ'αυτόν τον κόμβο και, τέλος, ένα ArrayList, το οποίο περιέχει τα κλειδιά όλων των κόμβων που συνδέονται με αυτόν. Με αυτόν τον τρόπο, λοιπόν, για κάθε κόμβο γνωρίζουμε όλα τα 'παιδιά'-γείτονες του, ενώ η πρόσβαση σε αυτόν γίνεται κάθε φορά σε σταθερό χρόνο O(1), αφού γνωρίζοντας το κλειδί του μπορούμε να αναφερθούμε κατευθείαν σ'αυτόν, κερδίζοντας έτσι σε ταχύτητα.

Αναφέρεται, επίσης, ότι για την ανάγνωση και αποθήκευση των δεδομένων έχουμε δημιουργήσει τις κλάσεις CSV_Node_Reader, CSV_Client_Reader και CSV_Taxi_Reades αντιστοιχα, όπου από κάθε μία καλούμε την αντίστοιχη μέθοδο για την σωστή ανάγνωση και αποθήκευση των δεδομένων του πελάτη, των οδηγών και των γεωγραφικών συντεταγμένων των κόμβων.

Τώρα, έχοντας αποθηκευμένα τα δεδομένα στις παραπάνω δομές αρκεί να εφαρμόσουμε τον αλγόριθμο A*. Για την υλοποίηση του A* πρέπει, σε 1^{ο} στάδιο να προσδιορίσουμε την κατάλληλη ευριστική συνάρτηση. Εμείς θεωρήσαμε ως καταλληλότερη ευρυστική την ευκλείδια απόσταση, προκειμένου να εξασφαλίσουμε το γεγονός ότι δεν θα υπερεκτιμάμε ποτέ την απόσταση του στόχου από τον κόμβο στον οποίο βρισκόμαστε και έτσι να αποφύγουμε πιθανή εύρεση μη βέλτιστης λύσης.

Η εφαρμογή του αλγορίθμου A* γίνεται με κλήση της μεθόδου find_route της κλάσης PathFinder, δίνοντας φυσικά τα κατάλληλα ορίσματα. Σημειώνεται εδώ ότι έχουμε δημιουργήσει και μία κλάση Way, η οποία αποτελείται από μία τιμή f, μία τιμή g και μία ArrayList, από κλειδιά κόμβων. Η γενική ιδέα του αλγορίθμου A* που έχουμε υλοποιήσει είναι ότι δημιουργούμε ένα ArrayList από αντικείμενα τύπου Way. Σε κάθε κόμβο στον οποίο βρισκόμαστε, ελέγχουμε το πλήθος των γειτονικών του κόμβων. Αν υποθέσουμε ότι η λίστα μέχρι και τον κόμβο που είμαστε είναι η A και ότι αυτοί οι κόμβοι-γείτονες είναι κ στο πλήθος, τότε για κάθε κόμβο j απ'αυτούς δημιουργούμε νέα αντικείμενα Way, όπου για το κάθε ένα υπολογίζουμε το αντίστοιχο f και g, ενώ στην ArrayList του έχουμε κάθε φορά την A και βάζουμε στο τέλος τον j-οστό κόμβο. Φυσικά, οι κόμβοι για οποίους γνωρίζουμε ότι έχουμε ήδη περάσει δεν επανατοποθετούνται στη λίστα.

Αυτά τα αντικείμενα Way που βάζουμε στη λίστα(με τα αντικείμενα Way) τα ταξινομούμε πάντα βάσει του f σε αύξουσα σειρά. Στη συνέχεια επιλέγουμε το 1^{ο} στοιχείο της λίστας, αφού έχει την μικρότερη τιμή για το f, μέχρι κάποια στιγμή να φτάσουμε στον κόμβο-στόχο, δηλαδή στον πελάτη. Αυτή η διαδικασία επαναλαμβάνεται για όλους τους οδηγούς των ταξί και, κάθε φορά, επιστρέφεται στο τέλος η ακολουθία με τους κόμβους που πρέπει να ακολουθήσει ο εκάστοτε οδηγός ώστε να φτάσει στον πελάτη.

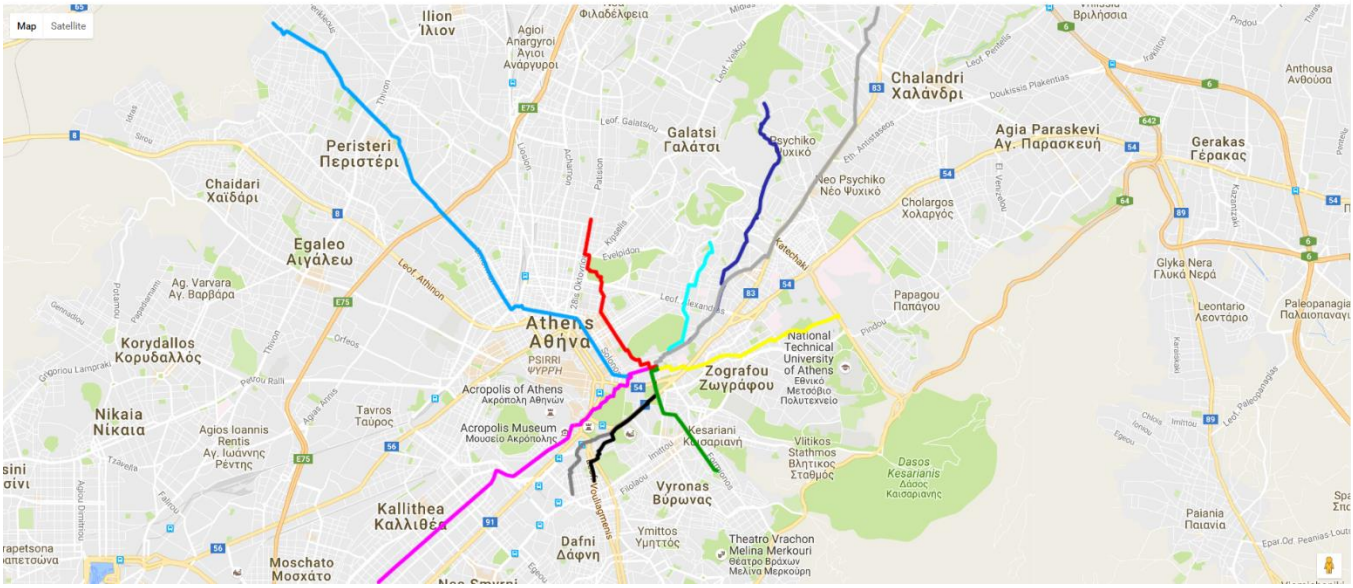
Τέλος, αφού γίνει η διαδικασία αυτή για όλους τους οδηγούς επιλέγεται ο οδηγός, ο οποίος βρίσκεται πιο κοντά στον πελάτη.

Αποτελέσματα εκτέλεσης του προγράμματος

Για τα αρχεία εισόδου που μας δόθηκαν παρατίθεται μαζί με την αναφορά η έξοδος που προέκυψε στο αρχείο RESULTS_FINAL.txt

Επιπλέον, όσον αφορά την οπτική απεικόνιση των αποτελεσμάτων, πάλι για τα δεδομένα εισόδου που μας δόθηκαν,παραθέτουμε το αρχείο KML_file.txt, το οποίο είναι το αρχείο KML που δημιουργήσαμε, καθώς επίσης και το html αρχείο map, το οποίο εμφανίζει τα αποτελέσματα του παραπάνω KML. Για τα δοθέντα αρχεία εισόδου προέκυψε η εξής οπτική απεικόνιση, η οποία είναι ίδια με αυτή του html αρχείου:

Χάρτης ταξί



Με την πράσινη γραμμή απεικονίζεται η βέλτιστη διαδρομή για το κοντινότερο ταξί, ενώ οι υπόλοιπες απαικονίζονται με άλλα χρώματα, εκτός του πρασίνου.

(Επισήμανση:Λόγω περιορισμένου πλήθους χρωμάτων υπάρχουν δύο διαφορετικά γκρι στην εικόνα, ένα πιο σκούρο και ένα πιο ανοιχτο τα οποία αναφέρονται σε δύο διαφορετικούς οδηγούς)