



ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ
ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

Τεχνητή Νοημοσύνη:

1^ο Θέμα

Χειμερινό εξάμηνο 2018

Μασούρης Αθανάσιος
03115189 7^ο εξάμηνο

Πίνακας περιεχομένων

Γενικός σχεδιασμός συστήματος	1
Διάβασμα αρχείων εισόδου	1
➤ Κλάση Client	1
➤ Κλάση Taxi	1
➤ Κλάση Node	1
Κλάση Position	2
Κλάση Heuristic.....	2
Παραλλαγή αλγορίθμου αναζήτησης A*	2
➤ Κλάση SearchNode.....	2
➤ Κλάση SearchNodeComparator	2
➤ Κλάση SearchSpace	3
Κλάση Taxigation.....	4
Κλάση Route	5
Κλάση RouteComparator.....	5
Δομές δεδομένων	5
Σχεδιαστικές λεπτομέρειες και συμβάσεις για τη λειτουργία της εφαρμογής	5
Παραδείγματα χρήσης της εφαρμογής.....	6
Έξοδος εφαρμογής στα αρχεία εισόδου που δίνονται	6
Έξοδος εφαρμογής στα δικά μου αρχεία εισόδου (taxis2.csv και client2.csv).....	6
Given Inputs	7
My Inputs.....	8
Παρατηρήσεις	9

Γενικός σχεδιασμός συστήματος

Στα πλαίσια του πρώτου θέματος κληθήκαμε να κατασκευάσουμε το βασικό κορμό μιας ευφυούς υπηρεσίας εξυπηρέτησης πελατών ταξί. Η υπηρεσία αυτή, δεχόμενη ως εισόδους τις συντεταγμένες τοποθεσίας ενός πελάτη, θα πρέπει να υπολογίζει τις διαδρομές όλων των ταξί, που ανήκουν στην εταιρεία, από τη γεωγραφική θέση που βρίσκονται εκείνη τη χρονική στιγμή μέχρι και τον πελάτη και να ειδοποιεί το ταξί που βρίσκεται πλησιέστερα σε αυτόν. Επιπροσθέτως θα πρέπει να εμφανίζει όλες τις δυνατές (ισοδύναμες) εναλλακτικές διαδρομές που θα μπορεί να ακολουθήσει το ταξί αυτό προκειμένου να φτάσει στον πελάτη.

Η υπηρεσία αυτή υλοποιήθηκε με την κατασκευή ενός προγράμματος σε γλώσσα Java. Το πρόγραμμα αυτό λαμβάνει ως παραμέτρους εισόδου τρία αρχεία csv (nodes.csv, client.csv, taxis.csv) τα οποία περιέχουν τις απαραίτητες πληροφορίες για το οδικό δίκτυο της περιοχής, τον πελάτη και τα διαθέσιμα ταξί. Στη συνέχεια με την υλοποίηση και την εφαρμογή μια παραλλαγής του γνωστού αλγορίθμου A* η εφαρμογή εντοπίζει όλες τις δυνατές μικρότερες διαδρομές από κάθε ταξί προς τον πελάτη καθώς και τις εναλλακτικές τους. Στη συνέχεια παράγει ως έξοδο στην κονσόλα το ταξί (ή τα ταξί) που είναι πλησιέστερα του πελάτη και παράγει ένα αρχείο εξόδου kml (routes.kml). Το αρχείο αυτό μπορεί στη συνέχεια να εισαχθεί στην υπηρεσία My Maps της Google και να παρουσιαστεί μία οπτική απεικόνιση των διαδρομών στον χάρτη επισημαίνοντας με πράσινο χρώμα (λόγω κατασκευής του kml) τη συντομότερη (ή τις συντομότερες αν υπάρχουν εναλλακτικές διαδρομές).

Διάβασμα αρχείων εισόδου

Για το διάβασμα των τριών αρχείων εισόδου έχουν δημιουργηθεί οι εξής κλάσεις:

➤ Κλάση Client

Η κλάση Client, η οποία επεκτείνει την κλάση Position (αναφέρεται παρακάτω), αποτελεί την κλάση των πληροφοριών τοποθεσίας του πελάτη και περιλαμβάνει τη μέθοδο διαβάσματος του αρχείου client.csv .

➤ Κλάση Taxi

Η κλάση Taxi, η οποία επεκτείνει την κλάση Position, αποτελεί την κλάση πληροφοριών για τα ταξί. Περιέχεται η τοποθεσία του εκάστοτε ταξί καθώς και το μοναδικό id του. Περιλαμβάνει επίσης τη μέθοδο για το διάβασμα του αρχείου taxis.csv, από την οποία επιστρέφεται μία δομή δεδομένων κλάσης ArrayList που περιλαμβάνει objects της κλάσης Taxi.

➤ Κλάση Node

Η κλάση Node αποτελεί την κλάση πληροφοριών για τον κάθε κόμβο του οδικού χάρτη που δίνεται μέσα από το αρχείο nodes.csv και επεκτείνει και αυτή την κλάση Position. Οι πληροφορίες που έχει κάθε object αυτής της κλάσης είναι οι γεωγραφικές συντεταγμένες του κόμβου, το id του δρόμου στον οποίο ανήκει καθώς και το όνομα του δρόμου αυτού (πιθανόν και να μη δίνεται). Η κλάση Node περιλαμβάνει και αυτή τη μέθοδο read για το διάβασμα του αρχείου nodes.csv και η οποία επιστρέφει μια δομή δεδομένων κλάσης ArrayList που

περιλαμβάνει objects της κλάσης Node. Τέλος περιέχει τη μέθοδο findClosestNode με την οποία βρίσκουμε τον πλησιέστερο κόμβο ως προς αυτόν που καλεί τη μέθοδο.

Κλάση Position

Η κλάση Position αποτελεί την κλάση που περιέχει πληροφορίες σχετικά με τις συντεταγμένες ενός σημείου. Αποτελεί την υπερκλάση των Client, Taxi και Node. Περιλαμβάνει επίσης την ιδιαίτερα σημαντική μέθοδο distanceTo η οποία υπολογίζει την απόσταση μεταξύ του σημείου που την καλεί και ενός άλλου σημείου που δίνεται ως παράμετρος, χρησιμοποιώντας και τη μέθοδο distance της κλάσης Heuristic που περιγράφεται παρακάτω.

Κλάση Heuristic

Η κλάση Heuristic αποτελεί την κλάση που υπολογίζει την τιμή της ευριστικής συνάρτησης για κάθε έναν από τους κόμβους του οδικού δικτύου που δίνεται. Περιλαμβάνει τη μέθοδο distance η οποία χρησιμοποιεί τη συνάρτηση haversine προκειμένου να υπολογίσει την απόσταση μεταξύ δύο σημείων δίνοντας ως παραμέτρους εισόδου τις συντεταγμένες τους. Η υλοποίηση της συνάρτησης haversine στη γλώσσα προγραμματισμού Java βρέθηκε [εδώ](#) (Rosetta code).

Παραλλαγή αλγορίθμου αναζήτησης A*

Για την υλοποίηση της παραλλαγής του αλγορίθμου αναζήτησης A* έχουν δημιουργηθεί οι εξής κλάσεις:

➤ Κλάση SearchNode

Η κλάση SearchNode αποτελεί την κλάση που περιλαμβάνει πληροφορίες για ένα object που βρίσκεται στο μέτωπο αναζήτησης του αλγορίθμου A*. Οι πληροφορίες αυτές είναι ο κόμβος στο οποίο βρίσκεται ο αλγόριθμος, το μονοπάτι που έχει διανύσει ο αλγόριθμος μέχρι να φτάσει σε αυτό τον κόμβο, η συνολική απόσταση του μονοπατιού καθώς και η τιμή της απόστασης από τον στόχο που υπολογίζει η ευριστική συνάρτηση. Περιέχει επίσης και τη μέθοδο getRealPlusHeuristic η οποία υπολογίζει και επιστρέφει το άθροισμα της απόστασης του μονοπατιού προς αυτό τον κόμβο και της απόστασης από τον στόχο όπως αυτή υπολογίζεται από την ευριστική συνάρτηση. Με βάση την τιμή που επιστρέφει η παραπάνω μέθοδος, ταξινομείται το μέτωπο αναζήτησης του αλγορίθμου.

➤ Κλάση SearchNodeComparator

Η κλάση αυτή κάνει implement τον Comparator για τα objects της κλάσης SearchNode και παραγράφει τη μέθοδο compare προκειμένου να ορίσουμε η σύγκριση μεταξύ των αντικειμένων της κλάσης SearchNode να γίνεται ως προς το αποτέλεσμα που επιστρέφει η μέθοδος getRealPlusHeuristic. Ο Comparator αυτός θα χρησιμοποιηθεί στην υλοποίηση του μετώπου (περιγράφεται παρακάτω) με τη χρήση TreeSet. Επίσης ιδιαίτερα σημαντικό είναι πως η μέθοδος compare που ορίζουμε δεν επιστρέφει μηδέν καθώς για την υλοποίηση της παραλλαγής του αλγορίθμου A είναι απαραίτητο να διατηρούμε στο μέτωπο αναζήτησης και τους ίσους μεταξύ τους κόμβους (ως προς το παραπάνω άθροισμα αποστάσεων) και σε περίπτωση που επιστραφεί μηδέν το TreeSet δεν προσθέτει το καινούριο object.

➤ Κλάση SearchSpace

Η κλάση αυτή ουσιαστικά αποτελεί το μέτωπο της αναζήτησης του αλγορίθμου και περιέχει τις μεθόδους διαχείρισής του. Αποτελείται από ένα TreeSet που περιλαμβάνει αντικείμενα της κλάσης SearchNode και αντιστοιχεί στο μέτωπο αναζήτησης, ένα HashMap το οποίο χρησιμοποιείται για να βρίσκουμε τους γειτονικούς κόμβους σε ένα σημείο (περιγράφεται παρακάτω), καθώς και τον κόμβο που αποτελεί το στόχο του αλγορίθμου αναζήτησης A*.

Περιλαμβάνει τις εξής μεθόδους:

- **initSearchSpace**

Η μέθοδος αυτή αρχικοποιεί το μέτωπο της αναζήτησης. Θέτει τον στόχο και προσθέτει στο μέτωπο τον αρχικό κόμβο της αναζήτησης με κενό αρχικό μονοπάτι και μηδενική απόσταση μονοπατιού.

- **addChildren**

Η μέθοδος αυτή είναι ιδιαίτερα σημαντική. Είναι υπεύθυνη για την επέκταση του μετώπου αναζήτησης. Μέσω του HashMap που αναφέρθηκε παραπάνω και περιγράφεται στη συνέχεια, εντοπίζει τους γειτονικούς κόμβους στον κόμβο της τρέχουσας κατάστασης. Κάθε ένας από αυτούς τους κόμβους, αν δεν έχει μπει ήδη στο κλειστό σύνολο (σύνολο από κόμβους που έχουμε ήδη επισκεφθεί), αποτελεί παιδί του κόμβου της τρέχουσας κατάστασης. Ωστόσο για να μπουν στο μέτωπο αναζήτησης πραγματοποιείται έλεγχος για το αν υπάρχουν ήδη σε αυτό. Από τον έλεγχο αυτό καταλήγουμε στις εξής τρεις περιπτώσεις:

1. Υπάρχει ήδη στο μέτωπο με μικρότερο άθροισμα ευριστικής και πραγματικής απόστασης. Σε αυτή την περίπτωση ο κόμβος που εξετάζουμε δε θα μπει στο μέτωπο.
2. Υπάρχει ήδη στο μέτωπο με μεγαλύτερο άθροισμα ευριστικής και πραγματικής απόστασης. Σε αυτή την περίπτωση ο κόμβος που εξετάζουμε θα μπει στο μέτωπο και ο κόμβος (ή οι κόμβοι) που έχουν μεγαλύτερο άθροισμα θα αφαιρεθούν.
3. Υπάρχει ήδη στο μέτωπο με ίσο άθροισμα ευριστικής και πραγματικής απόστασης. Σε αυτή την περίπτωση θέλουμε να κρατήσουμε και τους δύο (ή περισσότερους) κόμβους στο μέτωπο διότι έχουν ίσες αποστάσεις με διαφορετικά μονοπάτια και είναι απαραίτητο προκειμένου να βρούμε τις εναλλακτικές διαδρομές.

- **getHead**

Επιστρέφει το πρώτο στοιχείο του μετώπου αναζήτησης αφαιρώντας το ταυτόχρονα από το μέτωπο αναζήτησης.

- **exists**

Εξετάζει αν υπάρχει άλλο ίδιο ως προς την τοποθεσία αντικείμενο της κλάσης SearchNode στο μέτωπο αναζήτησης.

Κλάση Taxigation

Η κλάση αυτή αποτελεί την κλάση που συνδέει ουσιαστικά τις υπόλοιπες μεταξύ τους και υλοποιεί σε συνδυασμό με τις παραπάνω τον αλγόριθμο A*. Στην κλάση αυτή περιέχονται ένα ArrayList το οποίο αποτελεί το κλειστό σύνολο της αναζήτησης και ένα TreeSet στο οποίο περιέχονται αντικείμενα της κλάσης Route η οποία θα περιγραφεί στη συνέχεια. Η κλάση αυτή περιλαμβάνει τη main μέθοδο της εφαρμογής μας.

Η main λαμβάνει ως παραμέτρους τις τοποθεσίες ή τα ονόματα των τριών αρχείων εισόδου. Στη συνέχεια δημιουργεί αντικείμενα αντίστοιχα της κλάσης που επιστρέφει η μέθοδος read της εκάστοτε εκ των τριών κλάσεων που χρησιμοποιούνται για το διάβασμα των αρχείων αυτών (Client, Node, Taxi).

Επειδή είναι πιθανό οι τοποθεσίες του πελάτη και των ταξί να μην αποτελούν και κόμβους του οδικού δικτύου που μας δίνεται, χρησιμοποιείται η μέθοδος findClosestNode της κλάσης Node προκειμένου να βρούμε τους αρχικούς κόμβους των taxi και τον τελικό στόχο.

Με τη χρήση της μεθόδου initMap, που υλοποιείται στην κλάση Taxigation, δημιουργείται ένα HashMap το οποίο έχει για κλειδιά αντικείμενα της κλάσης Position και για τιμές ArrayList από αντικείμενα της κλάσης Node που αποτελούν τους γειτονικούς κόμβους στο Position key.

Έπειτα για κάθε ένα από τα taxi (loop) αρχικοποιείται το μέτωπο της αναζήτησης και μέσα σε ένα while loop επαναλαμβάνουμε διαρκώς την εξής ακολουθία βημάτων μέχρις ότου είτε να αδειάσει το μέτωπο αναζήτησης, είτε να βρεθούμε σε κατάσταση που αποτελεί στόχο και η συνολική απόσταση να είναι μεγαλύτερη από την προηγούμενη που είχαμε βρει, είτε τέλος έχοντας βρει ήδη μια διαδρομή προς το στόχο να είμαστε σε ένα κόμβο με μεγαλύτερη συνολική απόσταση από αυτή που έχουμε βρει:

1. Πάρε το head από το μέτωπο αναζήτησης
2. Έλεγχος για ικανοποίηση μίας εκ των παραπάνω συνθηκών τερματισμού του βρόχου
3. Αν βρισκόμαστε σε κατάσταση στόχος δημιουργήσε αντικείμενο της κλάσης Route και πρόσθεσε το στο TreeSet. Έπειτα συνέχισε από το βήμα 1.
4. Διαφορετικά έλεγξε αν το head είναι στο κλειστό σύνολο.
 - Αν είναι πήγαινε στο βήμα 1
 - Αν όχι, τότε έλεγξε αν υπάρχει άλλος αντικείμενο της κλάσης SearchNode με τοποθεσία ίδια με το head στο μέτωπο αναζήτησης. Λόγω της υλοποίησης της μεθόδου addChildren που περιγράψαμε παραπάνω, αν υπάρχει τότε θα έχει ίση συνολική απόσταση με το head και διαφορετικό μονοπάτι. Οπότε θέλουμε να το ελέγξουμε και αυτό. Άρα σε αυτή την περίπτωση δε βάζουμε το head στο κλειστό σύνολο. Αν δεν υπάρχει τότε τοποθετούμε το head στο κλειστό σύνολο.
5. Πρόσθεσε τα παιδιά του head μέσω της μεθόδου addChildren στο μέτωπο αναζήτησης.
6. Πήγαινε στο βήμα 1.

Και επαναλαμβάνεται διαρκώς η παραπάνω ακολουθία για κάθε ένα από τα taxi. Καθαρίζοντας και κάθε φορά το κλειστό σύνολο.

Τέλος η μέθοδος main καλεί τη μέθοδο createKML της κλάσης Taxigation. Η μέθοδος αυτή δημιουργεί ένα αρχείο KML σύμφωνα με το πρότυπο που έχει δοθεί στην εκφώνηση του 1^{ου} θέματος, ενώ ταυτόχρονα τυπώνει και στην κονσόλα το (ή τα) ταξί που βρίσκονται πλησιέστερα στον πελάτη, χρησιμοποιώντας το TreeSet με τα αντικείμενα της κλάσης Route.

Κλάση Route

Τα αντικείμενα της κλάσης αυτής περιέχουν τις εξής πληροφορίες: λίστα από αντικείμενα της κλάσης Node που αποτελούν το μονοπάτι κόμβων από τον αρχικό κόμβο προς το στόχο, συνολική απόσταση του μονοπατιού και το αναγνωριστικό του ταξί για το οποίο δημιουργήθηκε η διαδρομή.

Κλάση RouteComparator

Η κλάση RouteComparator υλοποιεί την κλάση Comparator για αντικείμενα κλάσης Route και χρησιμοποιείται ως comparator για το TreeSet που αναφέρεται παραπάνω. Παραγράφει τη μέθοδο compare ώστε να συγκρίνει μόνο τις συνολικές αποστάσεις μεταξύ των δύο αντικειμένων Route. Επίσης είναι ιδιαίτερα σημαντικό η compare μας να μην επιστρέφει μηδέν ώστε να διατηρηθούν στο TreeSet και διαδρομές ίσης απόστασης, δηλαδή οι εναλλακτικές.

Δομές δεδομένων

Οι δομές δεδομένων που χρησιμοποιήθηκαν στην εφαρμογή μας είναι οι εξής:

- ArrayList
- TreeSet, για το μέτωπο και τις διαδρομές προκειμένου να διατηρούνται ταξινομημένες. (Το καθένα με δικό του comparator)
- HashMap, προκειμένου να μπορούμε να εντοπίζουμε άμεσα τα παιδιά ενός κόμβου χρησιμοποιώντας την τοποθεσία του ως κλειδί.

Σχεδιαστικές λεπτομέρειες και συμβάσεις για τη λειτουργία της εφαρμογής

Αρχικά οι κόμβοι στο αρχείο nodes.csv δίνονται διαδοχικά όπως σχηματίζεται ο δρόμος. Συνεπώς γνωρίζουμε ότι παιδιά ενός κόμβου είναι τα κελιά ± 1 του κελιού που βρίσκεται ο κόμβος αν έχουν το ίδιο id. Επίσης, αν πρόκειται για διασταύρωση, παιδιά είναι και τα κελιά ± 1 του αντίστοιχου κελιού της διασταύρωσης.

Όπως αναφέρθηκε παραπάνω, είναι πιθανό οι τοποθεσίες του πελάτη καθώς και των ταξί να μην αποτελούν κόμβους του αρχείου nodes.csv. Συνεπώς θεωρούμε ότι η διαδρομή θα ξεκινήσει από τον πλησιέστερο κόμβο στον πελάτη και θα τερματίσει στον πλησιέστερο κόμβο στο εκάστοτε ταξί και στη συνέχεια προσθέτουμε τις δύο αποστάσεις από τους πλησιέστερους κόμβους. Ωστόσο είναι πιθανό η σύμβαση αυτή που κάνουμε, ότι ο πελάτης θα βρίσκεται στον πλησιέστερό του κόμβο, να μην είναι σωστή επειδή δε θα υπάρχει στην πραγματικότητα η δυνατότητα μετάβασης από την τοποθεσία του πελάτη στον πλησιέστερο κόμβο. Αντίστοιχα και για τα ταξί.

Ακόμα, λαμβάνουμε υπόψη και τις παραδοχές που περιγράφονται στην εκφώνηση. Όλοι οι δρόμοι είναι διπλής κατεύθυνσης, επιτρέπεται να κινηθεί σε αυτούς το ταξί, επιτρέπουν ίδια ταχύτητα κίνησης και όλα τα σημεία τομής είναι σημεία διασταύρωσης.

Επίσης η ευριστική συνάρτηση εκτίμησης απόστασης haversine που χρησιμοποιήθηκε έχει ακρίβεια αρκετών δεκαδικών ψηφίων. Στη εφαρμογή μας, προκειμένου να εμφανιστούν και εναλλακτικές διαδρομές, στρογγυλοποιούμε στα τρία δεκαδικά ψηφία που αντιστοιχούν στα μέτρα.

Τέλος, θεωρούμε δεδομένο ότι ισχύει η συνθήκη σωστής λειτουργίας του αλγορίθμου A^* και ότι ο ευριστικός μηχανισμός είναι αποδοκός και ικανοποιεί το κριτήριο αποδοχής. (Πρέπει για κάθε κατάσταση η τιμή της ευριστικής να είναι μικρότερη ή το πολύ ίση με την πραγματική απόσταση της S από την τελική κατάσταση, προκειμένου ο A^* να βρίσκει πάντα τη βέλτιστη λύση.)

Παραδείγματα χρήσης της εφαρμογής

Έξοδος εφαρμογής στα αρχεία εισόδου που δίνονται

Το αποτέλεσμα που εμφανίζει στην κονσόλα η εφαρμογή για τα αρχεία εισόδου που δίνονται είναι το εξής:

```
Finding nearest taxi(s) to the client.....  
-----  
Taxi 100 is near to the client.  
Estimated distance is 1.402 km  
-----  
Use the routes.kml file to see all available routes on a map.  
(Shortest routes' lines will appear green colored on the map)
```

Το αρχείο routes.kml που παράγεται μπορείτε να το δείτε απεικονισμένο στον χάρτη είτε [εδώ](#), είτε στην επόμενη σελίδα.

Έξοδος εφαρμογής στα δικά μου αρχεία εισόδου (taxis2.csv και client2.csv)















Το αποτέλεσμα που εμφανίζει στην κονσόλα η εφαρμογή για τα αρχεία εισόδου που δημιουργήσα είναι το εξής:

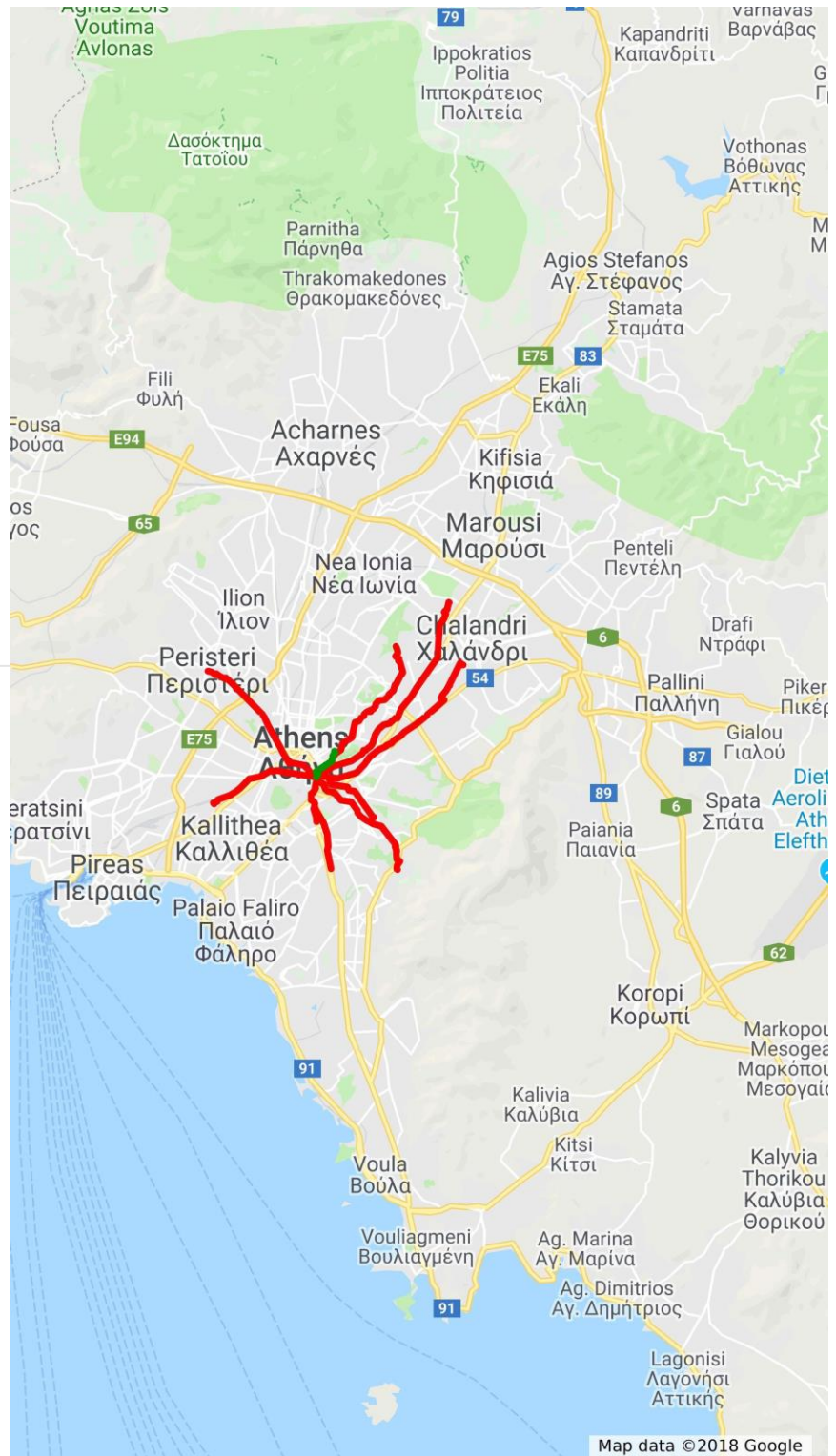
```
Finding nearest taxi(s) to the client.....  
-----  
Taxi 100 is near to the client.  
Estimated distance is 1.921 km  
-----  
Use the routes.kml file to see all available routes on a map.  
(Shortest routes' lines will appear green colored on the map)
```

Το αρχείο myroutes.kml που παράγεται μπορείτε να το δείτε απεικονισμένο στον χάρτη είτε [εδώ](#), είτε στη [σελίδα 8](#).

Given Inputs










Taxi Routes

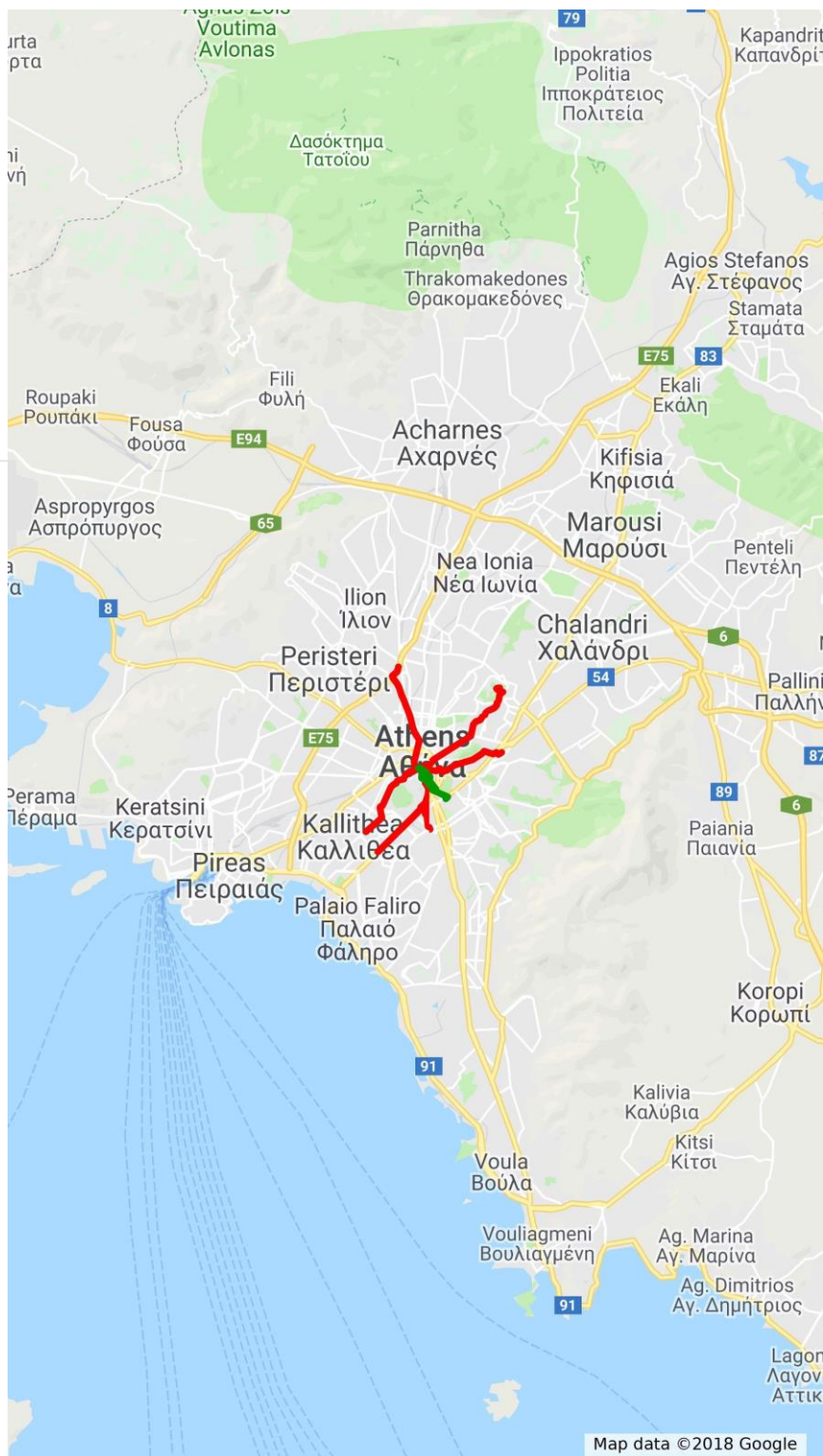
-  Taxi 100
-  Taxi 100
-  Taxi150
-  Taxi120
-  Taxi120
-  Taxi170
-  Taxi160
-  Taxi110
-  Taxi180
-  Taxi190
-  Taxi140
-  Taxi140
-  Taxi200
-  Taxi130



My Inputs

Taxi Routes

-  Taxi 100
-  Taxi 100
-  Taxi300
-  Taxi200
-  Taxi600
-  Taxi800
-  Taxi800
-  Taxi400
-  Taxi700



Παρατηρήσεις

Τόσο στα δοσμένα αρχεία εισόδου, όσο και στα δικά μου παραδείγματα παρατηρούμε την ικανότητα του αλγορίθμου να βρίσκει το ταχί που αντιστοιχεί στη συντομότερη διαδρομή καθώς και τις εναλλακτικές αυτής της διαδρομής.

Ωστόσο παρατηρήθηκε ότι για κάποια σημεία εισόδου, ο αλγόριθμος δεν ήταν σε θέση να υπολογίσει διαδρομή, κάτι που πιθανόν οφείλεται στην έλλειψη απαραίτητων κόμβων από το οδικό δίκτυο.