ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΕΧΝΗΤΗ ΝΟΗΜΟΣΥΝΗ

ΘΕΜΑ 1ο

ΟΝΟΜΑΤΕΠΩΝΥΜΟ: Πέτρου Γεώργιος
Α.Μ: 03113145
Εξάμηνο: 7ο

Σχεδιαστικές Επιλόγες

- Αρχικά φτιάχνουμε μια κλάση node για να μοντελοποιήσουμε κάθε σημείο του αρχείου nodes.csv (σημεία οδών) με ένα κόμβο ενός γραφήματος. Κάθε κόμβος node του γραφήματος περιέχει συντεταγμένες X,Y κωδικό code και ένα πεδίο name που έχει το όνομα της οδού στην οποία βρίσκεται ή αν δεν δίνεται το όνομα τότε το πεδίο name είναι κενό. Επιπλέον κάθε κόμβος του γραφήματος περιέχει μια μεταβλητή h που είναι η τιμή της ευριστικής συνάρτησης απόστασης του κόμβου από τον τελικό κόμβο στόχο. Σαν τιμή στο πεδίο h αναθέτουμε την ευκλείδεια απόσταση του σημείου-κόμβου από το τελικό κόμβο-στόχο. Έτσι γνωρίζουμε ότι η ευκλείδεια απόσταση μεταξύ ενός κόμβου του γραφήματος και του τελικού κόμβου που είναι το μήκος του ευθυγράμμου τμήματος που τα ενώνει είναι η ελάχιστη δυνατή απόσταση (κάθε άλλη διαδρομή από τον κόμβο στον τελικό κόμβο είναι μεγαλύτερη ή ίση από το ευθύγραμμο τμήμα που ενώνει τον κόμβο με τον τελικό κόμβο-στόχο) οπότε ο ευριστικός μηχανισμός που χρησιμοποιούμε θα είναι admissive πράγμα που εγγυάται ότι ο αλγόριθμος A* θα βρίσκει τη βέλτιστη λύση.

- Στη συνέχεια δημιουργούμαι μία κλάση graph ώστε να μοντελοποιήσουμε το γράφο που θα χρησιμοποιήσουμε. Η κλάση αυτή έχει ένα vector (graph_nodes) που περιέχει όλους τους κόμβους του γραφήματος (μόλις τους διαβάζουμε από το αρχείο nodes.csv τους αποθηκεύουμε στο vector) καθώς επίσης περιέχει και ένα vector από vectors για να αναπαραστήσει τος ακμές (δηλαδή σαν ένα πίνακα γειτνίασης).

- Επειδή εκεί που τέμνονται δύο οι παραπάνω οδοί τα σημεία αυτά περιέχονται πολλές φορές στο αρχείο nodes.csv (ένα σημείο για κάθε διαφορετικό id οδού) κάνουμε την σχεδιαστική επιλογή να κρατάμε όλα τα σημεία αυτά (που έχουν ίδιες συντεταγμένες X,Y) και τα ενώνουμε με ακμές οι οποίες όμως θα έχουν μηδενικό κόστος (αφού τα σημεία αυτά στη πραγματικότητα ταυτίζονται),οπότε δεν επηρεάζουν σε τίποτα τον αλγόριθμο απλά μπορεί να υπάρχουν στο τελικό αρχείο kml (διαδοχικά) ορισμένοι ίδιοι κόμβοι ως προς τις συντεταγμένες (που ήταν τα σημεία διασταύρωσης) .Αυτό επίσης δεν επηρεάζει ούτε το τελικό γραφικό αποτέλεσμα (αφού τον να έχουμε μία ή παραπάνω φορές τον ίδιο κόμβο (διαδοχικά) στο kml αρχείο δεν επηρεάζει την γραφική απεικόνιση στο χάρτη). Επιπλέον ακμές τοποθετούμε για να ενώσουμε τους κόμβους που ανήκουν στη ίδια οδό ως εξής : ενώνουμε κάθε κόμβο με τον προηγούμενο του αν ανήκει στην ίδια οδό (ίδιο code) και με τον επόμενο του αν ανήκει στην οδό (δηλαδή έχει ίδιο code).

- Έχοντας λοιπόν δημιουργήσει τον γράφο διαβάζουμε τις συντεταγμένες του κόμβου – πελάτη από το αρχείο client.csv οι οποίες επειδή μπορεί να μην συμπίπτουν με τις συντεταγμένες κάποιου κόμβου-σημείοο οδού (που αποτελούν το γράφο), θέτουμε ως κόμβο στόχο το πλησιέστερο κόμβο δηλαδή των κόμβο που έχει την μικρότερη (ευκλείδια) απόσταση από τις συντεταγμένς του κόμβου πελάτη που δόθηκε. Οπότε κρατάμε τον κόμβο αυτό του γραφήματος που είναι πιο κοντά στο σημείο-πελάτη και το θέτουμε ώς κόμβο-στόχο (και με βάση αυτόν τον κόμβο υπολογοίζονται οι ευρεστικές τιμές των υπόλοιπων κόμβων όπως περιγράφηκε προηγουμένως).

- Στη συνέχεια διαβάζουμε από το αρχείο taxis.csv τις συντεταγμένες των taxi και τις αποθηκεύουμε σε ένα vector. Για κάθε ταξί που (είναι η αφετηρία από όπου θα ξεκινήσουμε τον αλγόριθμο A*) επειδή οι συντεταγμένες του δεν συμπίπτουν αναγκαστικά με τις συντεταγμένες κάποιου κόμβου του γραφήματος θέτουμε σαν σημείο αφετηρίας Start το κόμβο του γραφήματος που απέχει το λιγότερο από το σημείο που βρίσκεται το ταξί (όπως κάναμε και με το σημείο του πελάτη-στόχου (Goal)) . Έτσι για κάθε ταξί

βρίσκουμε το κοντινότερο κόμβο του γραφήματος , το θέτουμε ως αφετηρία και εκτελούμε τον αλγόριθμο A* (όπως περιγράφεται παρακάτω).

- Για τον αλγόριθμο **A*** αρχικά έχουμε ορίσει μια κλάση path ώστε να μοντελοποιήσουμε τις διαδρομές που ακολουθεί και εξετάζει ο αλγόριθμος. Ένα μονοπάτι και περιέχει ένα vector που κρατάει όλους τους κόμβους του μονοπατιού καθώς επίσης την συνολική απόσταση που έχουμε διανύσει για το συγκεκριμένο μονοπάτι , τον τελευταίο κόμβο του μονοπατιού και την ευριστική τιμή του κόμβου αυτού. Στη συνέχεια ορίζουμε μια ουρά προτεραιότητας που περιέχει τέτοια μονοπάτια (δηλαδή αντικείμενα της κλάσης path) που έχει εξετάσει ο αλγόριθμος μέχρι στιγμής. Η ουρά περιέχει τα στοιχεί ταξινομημένα με βάση το συνολικό κόστος που είναι η απόσταση που έχουμε διανύσει και η ευριστική τιμή της απόστασης του τελευταίου κόμβου του μονοπατιού από τον στόχο (δηλαδή μια προσέγγιση του πόσο κοστίζει (πόσο απέχουμε) να φτάσουμε στον στόχο). Σε κάθε βήμα ο αλγόριθμος αφαιρεί το πρώτο στοιχείο από την ουρά προτεραιότητας δηλαδή αυτό που έχει το μικρότερο συνολικό κόστος εκείνη τη στιγμή και βρίσκει τα παιδία του τελευταίου κόμβου του μονοπατιού ώστε να επεκτείνει το μονοπάτι (ενώ παράλληλα κρατάει σε ένα πίνακα ότι επισκέφτηκε τον κόμβο αυτό). Έτσι για κάθε ένα παιδί επεκτείνει το μέχρι στιγμής μονοπάτι προσθέτοντας της απόσταση (του κόμβου από το παιδί του) στη συνολική απόσταση και έχοντας σαν ευρεστική τιμή την ευρεστική τιμή του κόμβου-παιδιού και εισάγει το νέο στοιχείο-μονοπάτι στην ουρά. Μόλις φτάσει να βγάλει από την ουρά προτεραιότητας κάποιο στοιχείο-μονοπάτι που έχει σαν τελευταίο κόμβο τον κόμβο στόχο σταματάει (καθώς πάει α επεκτείνει κόμβο –στόχο) επιστρέφοντας αυτό το μονοπάτι και την συνολική απόσταση.

- Τέλος η παραπάνω διαδικασία επαναλαμβάνεται για κάθε ταξί και έχοντας κρατήσει το συνολικό μονοπάτι (βέλτιστο για κάθε ταξί) που πρέπει να διανύσει κάθε ταξί και την συνολική απόσταση παράγουμε σαν output το αρχείο final.kml που περιέχει τις συντεταγμένες για τη διαδρομή που ακολουθεί κάθε ταξί και επιλέγουμε σαν βέλτιστο αυτό που έχει να διανύσει τη μικρότερη συνολική απόσταση το οποίο χρωματίζεται με πράσινο χρώμα ενώ τα υπόλοιπα με κόκκινο.

Οι κλάσεις που χρησιμοποιήθηκαν και η χρησιμότητά τους είναι η εξής:

Class Node: μοντελοποιεί το κάθε κόμβο στο τελικό γράφο. Περιέχει τις συνταταγμένες X,Y τον κωδικό της οδού που ανήκει (code) ,το όνομα της οδού που βρίσκεται (name) , την ευριστική τιμή h και μια μέθοδο για τον υπολογοισμό της τιμής του πεδίου h.

Class CSVreader: η κλάση αυτή χρησιμοποιείται για το διάβασμα τον αρχείων CSV και έχει ένα vector που κρατάει τα δεδομένα που διάβασε.

Class Graph: χρησιμοποιείται για να μοντελοποιήσει το γράφο. Χρησιμοποειέι την προηγούμενη κλάση για να διαβάσει και να αποθηκεύσει τους κόμβους από το αρχείο nodes.csv και υπολογίζει και κρατάει τις ακμές του γράφου.

Class Project1: χρησιμοποιεί την κλάσση CSVreader για το διάβασμα του αρχείου που περιέχει τις συντεταγμένες του πελάτη (client.csv) και τις συντεταγένες των ταξί (taxis.CSV) και στη συνέχει α υπολογίζει τον κόμβο στόχο (Goal) και τον κόμβο αφετηρία (Start) για τον αλγόριθμο A*. Δημιουργεί ένα γράφο δηλαδή ένα αντικείμενο της κλάσης graph , και περιέχει την υλοποίηση για τον αλγόριθμο A* . Ο αλγόριθμος A* κρατάει μια ουρά προτεραιότητατς για την οποία για να έχει κατάλληλα ταξινομηένα τα δεδομένα χρησιμοποιεί την κλάσση DistComparator. Τέλος χρησιμοποιεί την κλάση KMLcreator για την δημιουργία του kml αρχείου που είναι η έξοδος του προγράμματος.

Class Path: χρησιμοποιείται από την κλάση Project1 για να αποθηκέυει τα μονοπάτια που εξετάζει ο αλγόριθμος A* ώστε στο τέλος όπου έχει υπολογιστεί το βέλτιστο μονοπάτι να το έχουμε κρατήσει. Η κλάση path έχει σαν πεδία ένα vector όπου κρατούνται οι κόμβοι του συγκεκριμένου μονοπατιού, ένα πεδίο που κρατάει την συνολική απόσταση ,ένα πεδίο που κρατάει τον τελευταίο-πιο πρόσφατο κόμβο και ένα πεδίο που κρατάει την ευριστική τιμή το τελευταίου κόμβου(πιο πρόσφατου- δηλαδή που προστέθηκε τελευταία).

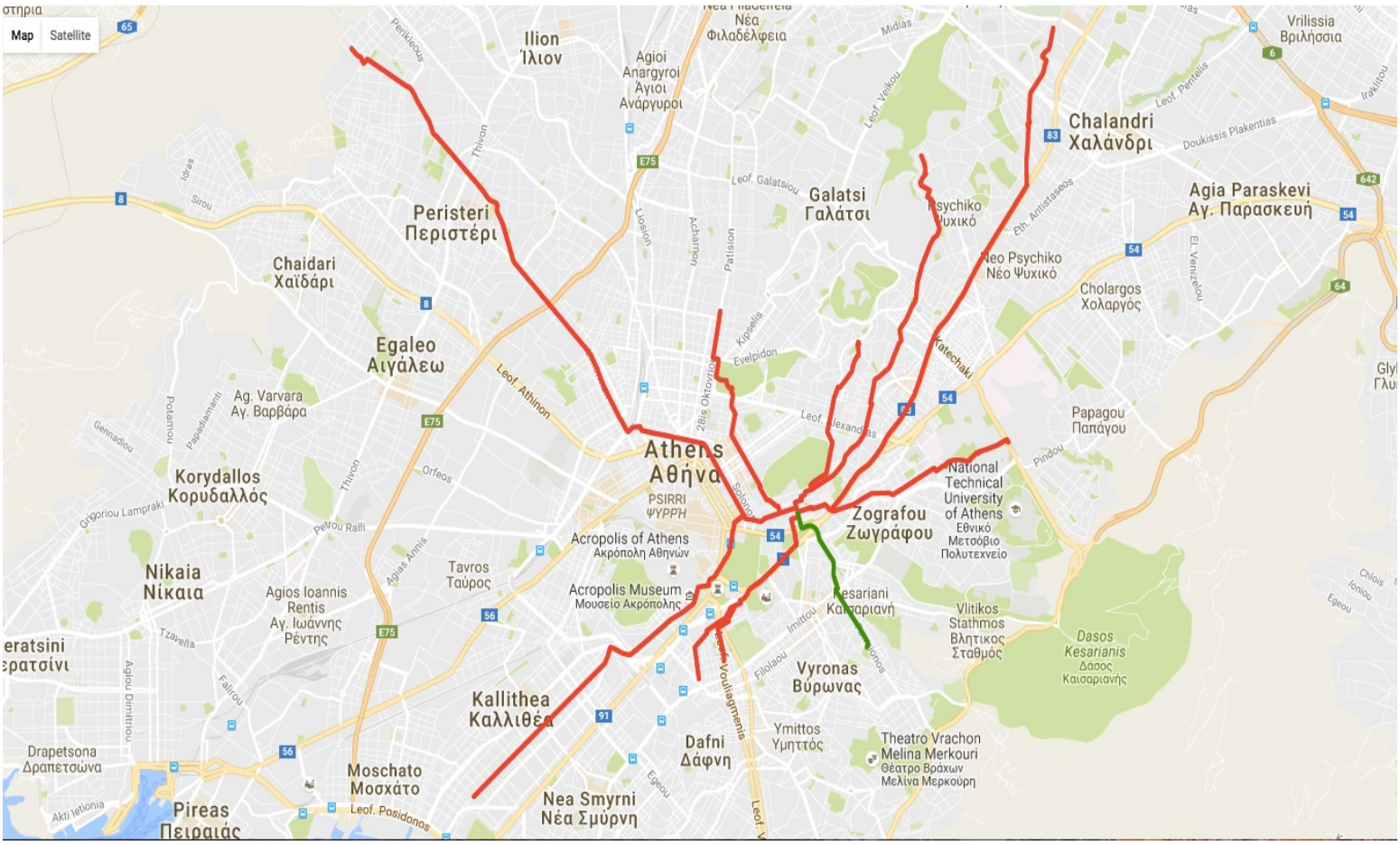
Class DistComparator: χρησιμοποιείται από την class Project1 για την ταξινόμηση των στοιχείων στην ουρά προτεραιότητας που χρησιμοποιείται για τον αλγόριθμο A*.

Class KMLcreator: χρησιμοποιείται για να εξάγει το κατάλληλο αρχείο KML.

Αποτελέσματα Εκτέλεσης

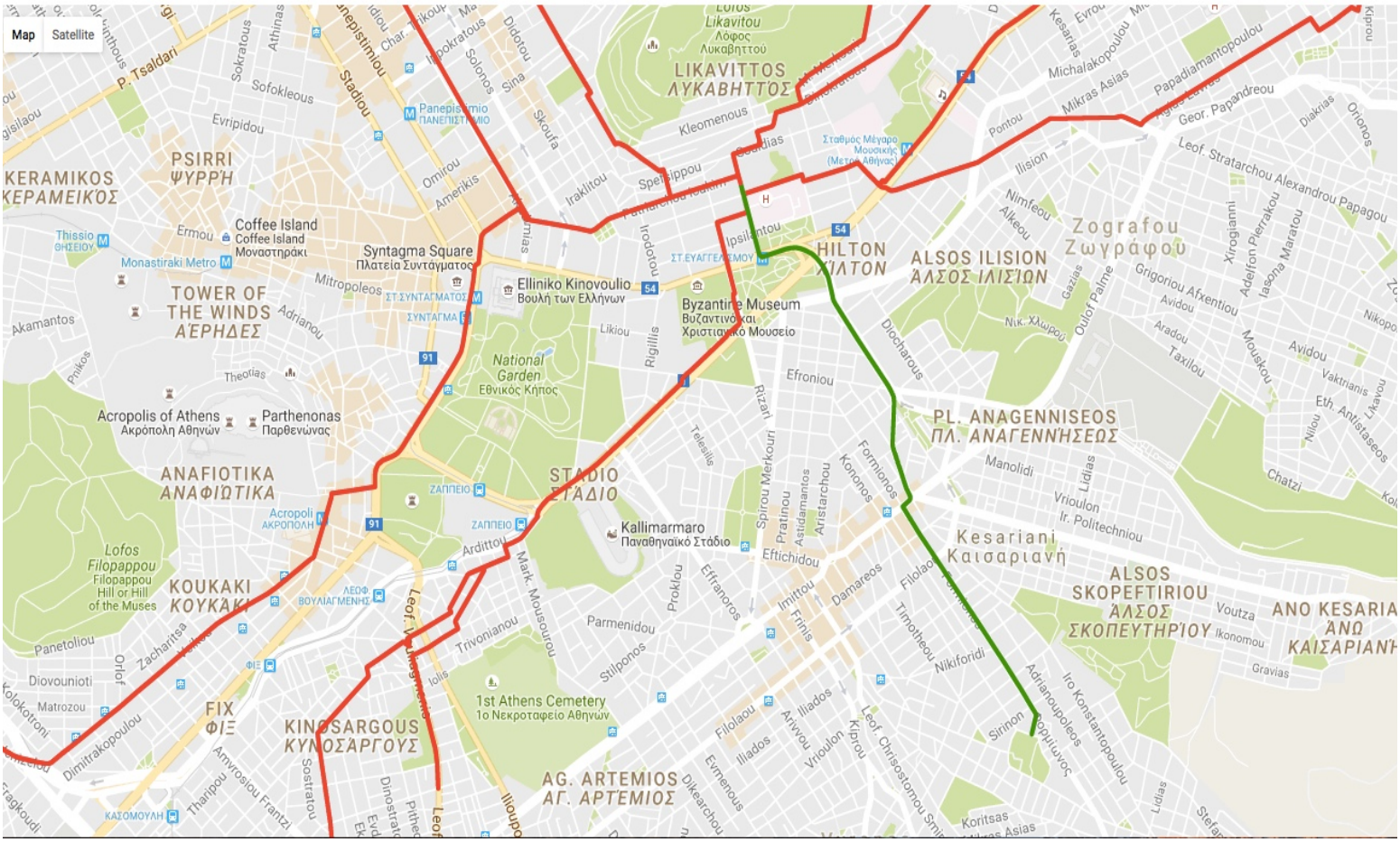
Για τα δοσμένα αρχεία nodes.CSV, taxis.CSV και client.SCV η έξοδος που φαίνεται στο χάρτη είναι η εξής:

Κάρτης ταξί



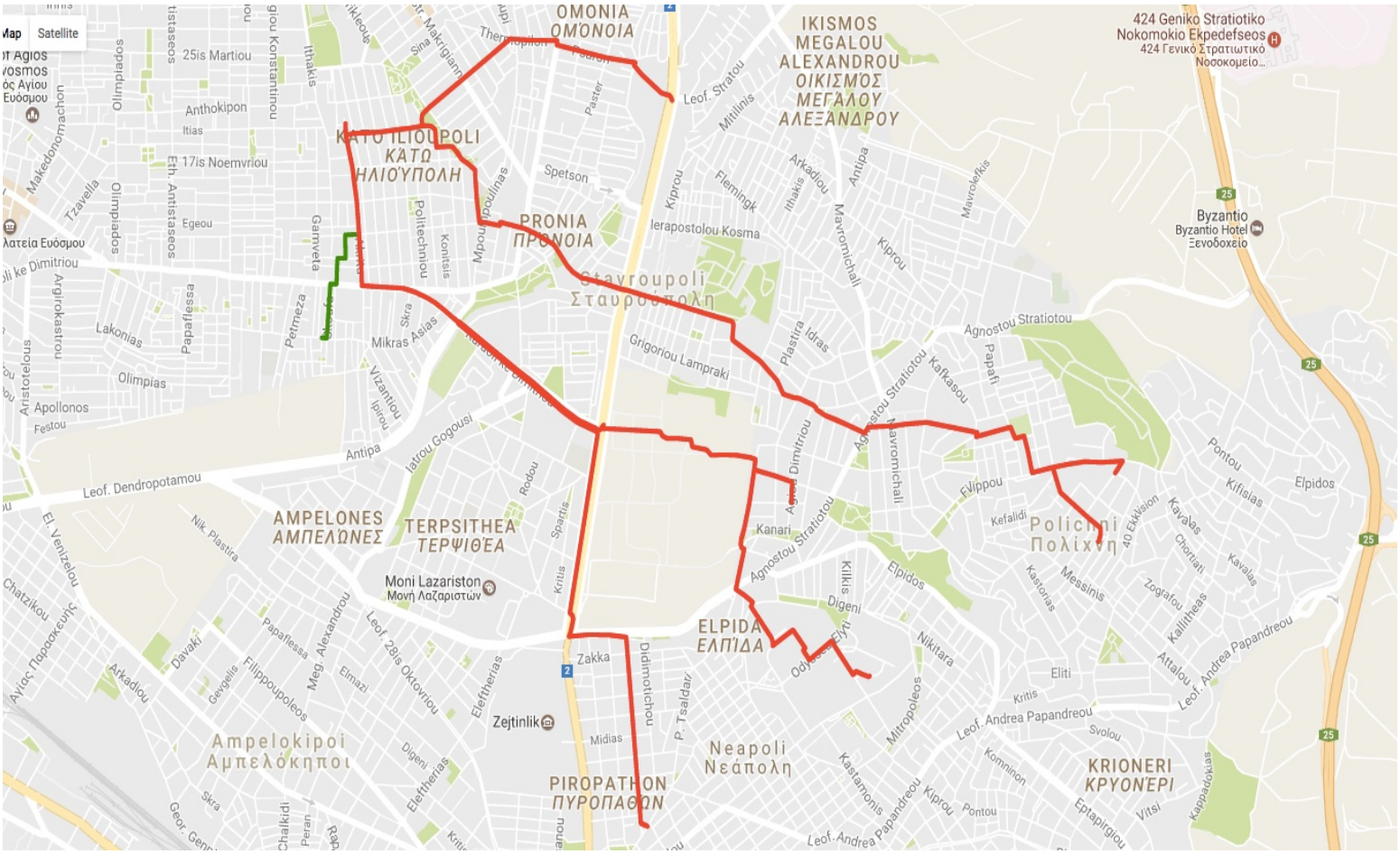
Εστιάζοντας στο ταξί που βρίσκεται πιο κοντά στο στόχο (διαδρομή σχεδιασμένη πράσινο χρώμα) έχουμε και την παρακάτω εικόνα:

Χάρτης ταξί

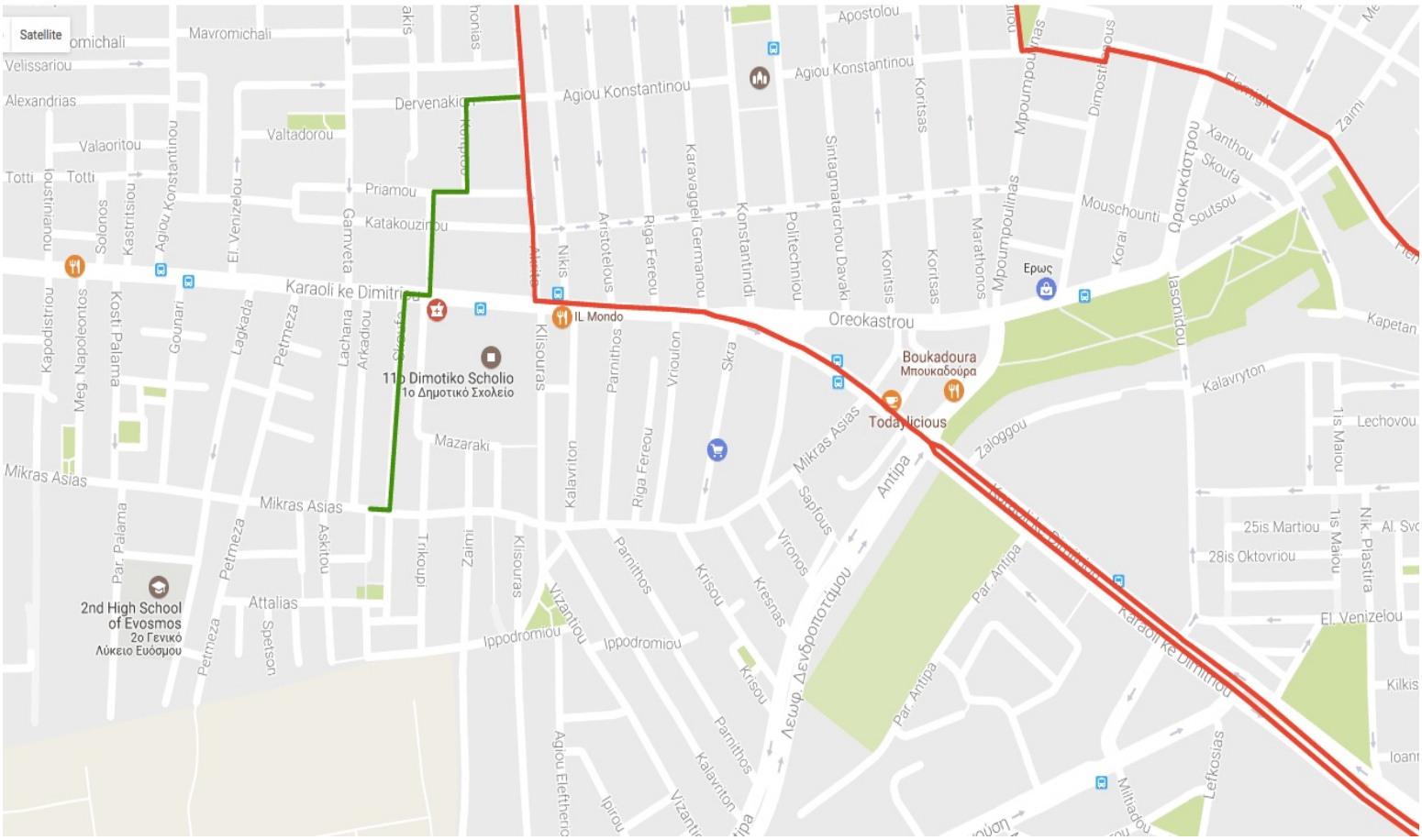


Τα αρχεία εισόδου (που είχαν δοθεί για την εργασία) το αρχείο εξόδου final2.kml ,το αρχείο map.html καθώς και οι παραπάνω εικόνες (screenshot1.jpeg ,screenshot2.jpeg) βρίσκονται στο φάκελο outpu1.

Στη συνέχεια δοκιμάσαμε να τρέξουμε το πρόγραμμα για ένα δικό μας χάρτη (που κατεβάσαμε και επεξεργαστήκαμε με βάση τις οδηγίες της άσκησης). Ο χάρτης απεικονίζει την πόλη Θεσσαλονίκη και έχουμε 7 διαθέσιμα ταξί .Η έξοδος φαίνεται στις παρακάτω εικόνες:

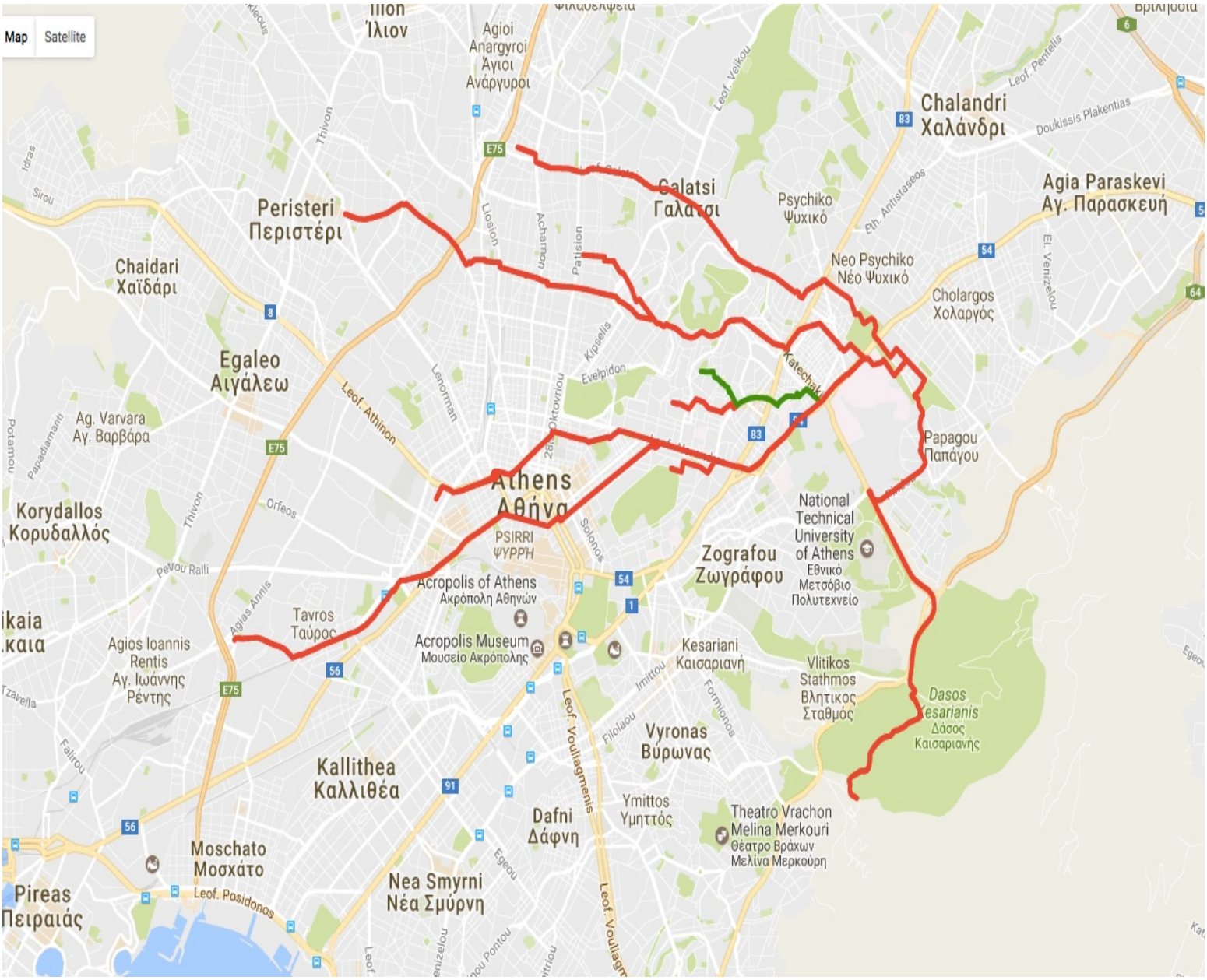


Εστιάζοντας στο ταξί που βρίσκεται πιο κοντά στο στόχο (διαδρομή σχεδιασμένη πράσινο χρώμα) έχουμε και την παρακάτω εικόνα:



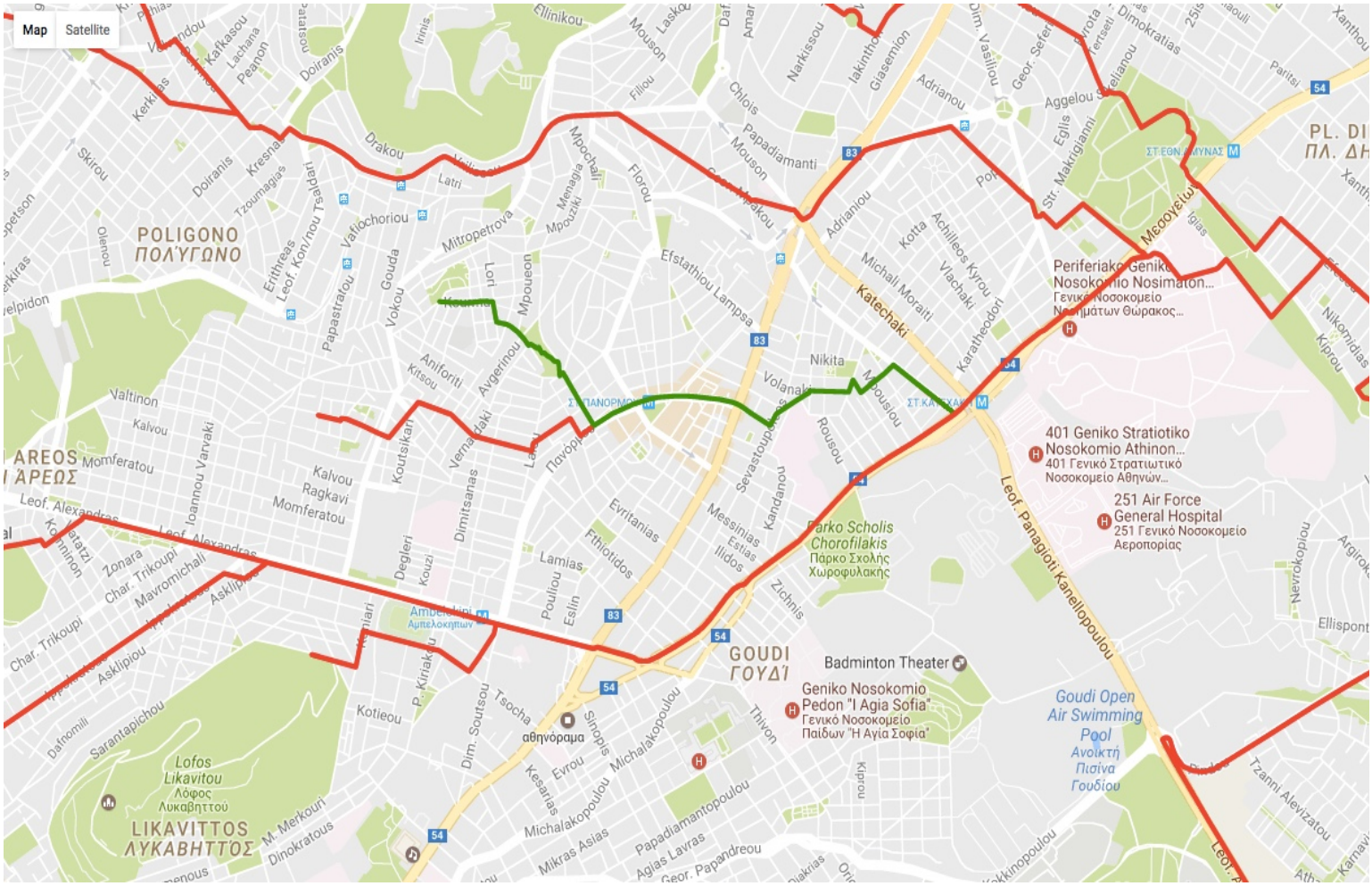
Τα αρχεία εισόδου (που δημιουργήσαμε εμείς) ,το αρχείο εξόδου final.kml , το αρχείο map2.html καθώς και οι παραπάνω εικόνες (screenshot3.jpeg ,screenshot4.jpeg) βρίσκονται στο φάκελο output2.

Στη συνέχεια δοκιμάσαμε να τρέξουμε το πρόγραμμα με το αρχικό αρχείο nodes.CSV (που δίνεται μαζί με την εκφώνηση της άσκησης) αλλάζοντας τώρα τα αρχεία taxis.CSV και client.CSV. Τα αποτελέσματα φαίνονται στις παρακάτω εικόνες:



Στην παραπάνω εικόνα φαίνονται εννέα διαδρομές ταξί ωστόσο στο kml αρχείο είναι δέκα πράγμα που οφείλεται στο ότι για δύο ταξί η διαδρομή τους επικαλύπτεται.

Εστιάζοντας στο ταξί που βρίσκεται πιο κοντά στο στόχο (διαδρομή σχεδιασμένη πράσινο χρώμα) έχουμε και την παρακάτω εικόνα:



Τα αρχεία εισόδου (nodes.CSV (ίδιο με αυτό που δόθηκε), client.SCV ,taxis.SCV) ,το αρχείο εξόδου final3.kml , το αρχείο map3.html καθώς και οι παραπάνω εικόνες (screenshot5.jpeg ,screenshot6.jpeg) βρίσκονται στο φάκελο output3.

Τέλος στο φάκελο source περιέχονται οι κώδικες των κλάσσεων (όλα τα αρχεία .java).