Τεχνητή Νοημοσύνη
$1^{ο}$ Θέμα

Βελέγκας Γρηγόριος
Α.Μ. 03113011

Λωτίδης Κυριάκος
Α.Μ. 03113033

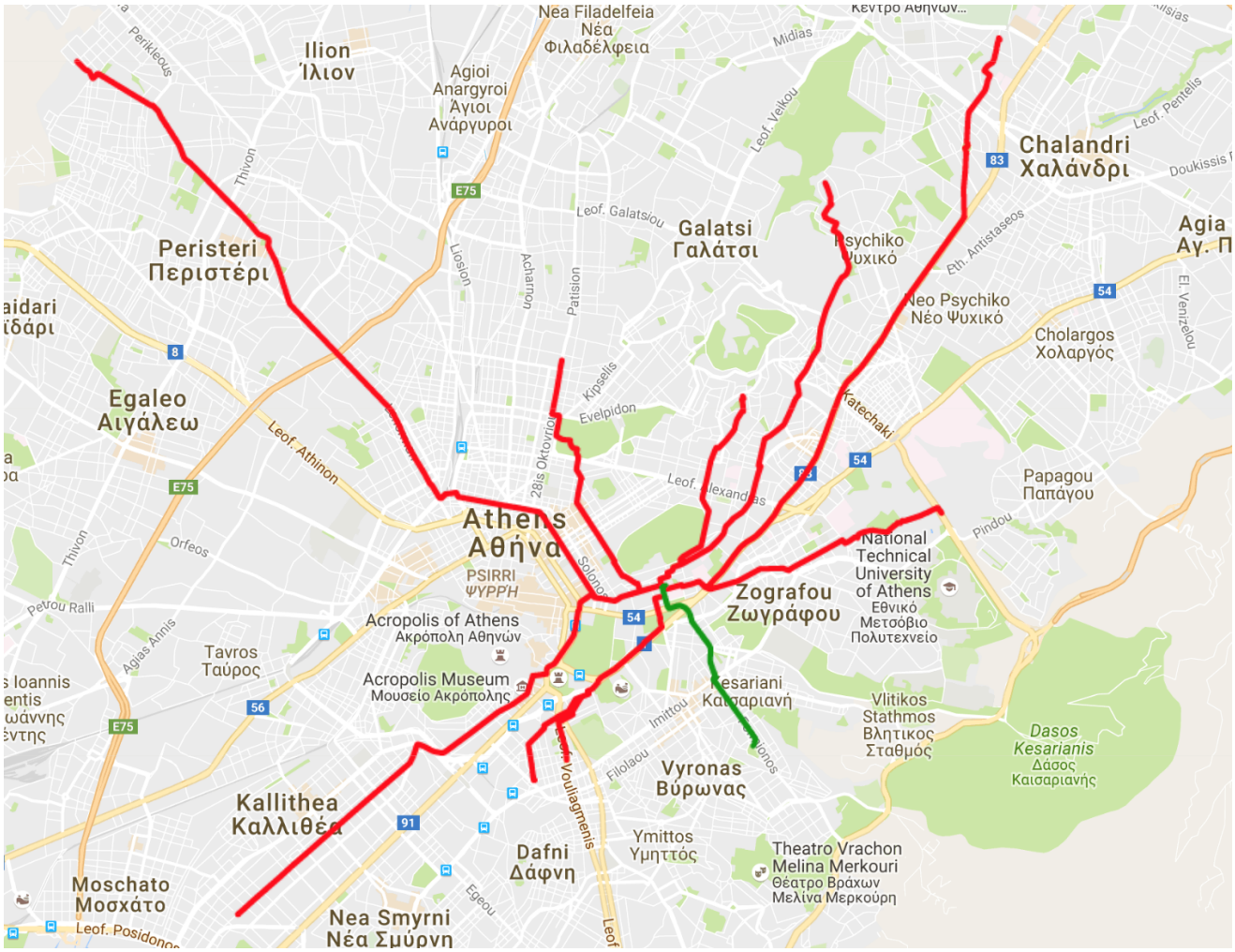
ΗΜΜΥ ΕΜΠ
5/12/2016

Για το διάβασμα των δεδομένων από τα αρχεία χρησιμοποιήθηκαν bufferedReaders και για κάθε γραμμή γίνεται ένα parsing ώστε να πάρουμε μεμονωμένα τα στοιχεία που μας ενδιαφέρουν. Δημιουργήθηκε μια κλάση Node για την αναπαράσταση των κόμβων του γράφου, που περιέχει τις συντεταγμένες του κάθε σημείου σε μορφή String και Double, αφού και οι δύο τύποι ήταν χρήσιμοι για διαφορετικές πράξεις, μια λίστα με τους γείτονές του και την εκάστοτε απόσταση (σε μορφή ArrayList), έναν δείκτη προς τον πατέρα του στο μονοπάτι που δημιουργείται από τον A*, καθώς και πεδία τύπου Double για τις τιμές real και heuristic των αποστάσεων. Ο γράφος υλοποιείται με ένα Hashtable που περιέχει τους κόμβους που αναφέρθηκαν πριν. Η επιλογή του Hashtable έγινε επειδή προσφέρει αναζήτηση για τυχαίο στοιχείο σε χρόνο $\Theta(1)$, κάτι που ήταν πολύ χρήσιμο αφού κατά το διάβασμα των συντεταγμένων των σημείων θέλαμε να κάνουμε γρήγορο έλεγχο για το κατά πόσο υπάρχει ήδη ένας κόμβος, αφού με αυτόν τον τρόπο δηλώνονται οι διασταυρώσεις. Για τον A* χρησιμοποιήθηκε ένα HashMap που περιέχει τους κόμβους που έχουμε ήδη επισκεφθεί έτσι ώστε όταν τους ξανασυναντήσουμε να μην τους επεκτείνουμε (η επιλογή της δομής έγινε και πάλι με κριτήριο τη γρήγορη αναζήτηση στοιχείου), μια PriorityQueue που μοντελοποιεί το μέτωπο αναζήτησης (με comparator το άθροισμα της ευριστικής και της πραγματικής απόστασης) καθώς και άλλο ένα HashMap που περιέχει επίσης τους κόμβους που βρίσκονται στο μέτωπο αναζήτησης, αφού όταν συναντήσουμε έναν κόμβο θέλουμε να εξετάσουμε κατά πόσο υπάρχει ήδη στο μέτωπο. Σε περίπτωση που υπάρχει και η νέα του real απόσταση είναι μικρότερη από την προηγούμενη πρέπει να του αλλάξουμε την προτεραιότητα και ο μόνος τρόπος για να το κάνουμε αυτό είναι με την εξαγωγή και έπειτα την εισαγωγή του. Η αναζήτηση στοιχείου σε μια PriorityQueue της Java χρειάζεται γραμμικό χρόνο, οπότε χρησιμοποιώντας το HashMap έχουμε σημαντική βελτίωση στο χρόνο αναζήτησης, έχοντας όμως κάποιο επιπλέον κόστος στη μνήμη. Η real απόσταση κάθε κόμβου είναι το άθροισμα της μέχρι τώρα διαδρομής από το αρχικό σημείο, δηλαδή το εκάστοτε ταξί, και η heuristic είναι η ευκλείδια απόσταση μέχρι τον πελάτη, η οποία αποτελεί υποεκτίμηση της πραγματικής απόστασης οπότε είμαστε βέβαιοι ότι ο αλγόριθμος θα βρει τη βέλτιστη λύση.
Οι συντεταγμένες του πελάτη καθώς και τον ταξί αποθηκεύονται επίσης σε ένα αντικείμενο τύπου Node. Το σύνολο των ταξί αποθηκεύεται σε ένα HashMap με κλειδί το προαναφερθέν Node και τιμή τον κωδικό του κάθε ταξί. Τα σημεία από το αρχείο nodes αποθηκεύονται στην ίδια δομή και η εύρεση των γειτόνων τους γίνεται με την εξής διαδικασία: Διαβάζουμε τις συντεταγμένες κάθε γραμμής και κρατάμε τον κωδικό της οδού. Ελέγχουμε κατά πόσο υπάρχει ήδη το σημείο στο γράφο, αφού στην περίπτωση αυτή έχουμε διασταύρωση και δε χρειάζεται να το τοποθετήσουμε εκ νέου. Αν ο κωδικός οδού του τρέχοντος σημείου είναι ίδιος με του προηγούμενου προσθέτουμε μια ακμή, δηλαδή έναν κόμβο στη λιστα γειτόνων του κάθε σημείου, διαφορετικά δεν προσθέτουμε κάποια σύνδεση. Από τα δεδομένα που μας δόθηκαν δε χρειάστηκε να κρατήσουμε αποθηκευμένα ούτε τον κωδικό της οδού ούτε και το όνομά της. Επειδή οι συντεταγμένες των ταξί και του πελάτη δεν ήταν σημεία του χάρτη επιλέξαμε να προσθέσουμε μια ακμή μεταξύ του καθενός απ΄ αυτά με τον κοντινότερό του κόμβο ώστε να μπορέσουμε να κάνουμε τον γράφο συνεκτικό.
Για την εκτέλεση του προγράμματος, τα ονόματα των αρχείων δίνονται στον κώδικα και τα αποτελέσματα γράφονται στο standard output. Για να τα δούμε στο αρχείο kml τα αντιγράφουμε στις κατάλληλες θέσεις.
Στα αρχεία my_client_athens.csv, my_taxis_athens.csv φαίνονται κάποιες επιπλέον δοκιμές για τον δοθέντα χάρτη. Έγιναν επιπλέον πειραματισμοί με τον χάρτη της Θεσσαλονικής, ο οποίος βρίσκεται στο αρχείο thessaloniki.csv, και τα αρχεία των ταξί και του πελάτη είναι τα thess_taxis.csv και thess_client.csv αντίστοιχα. Το αρχείο athina_default.kml είναι για τα

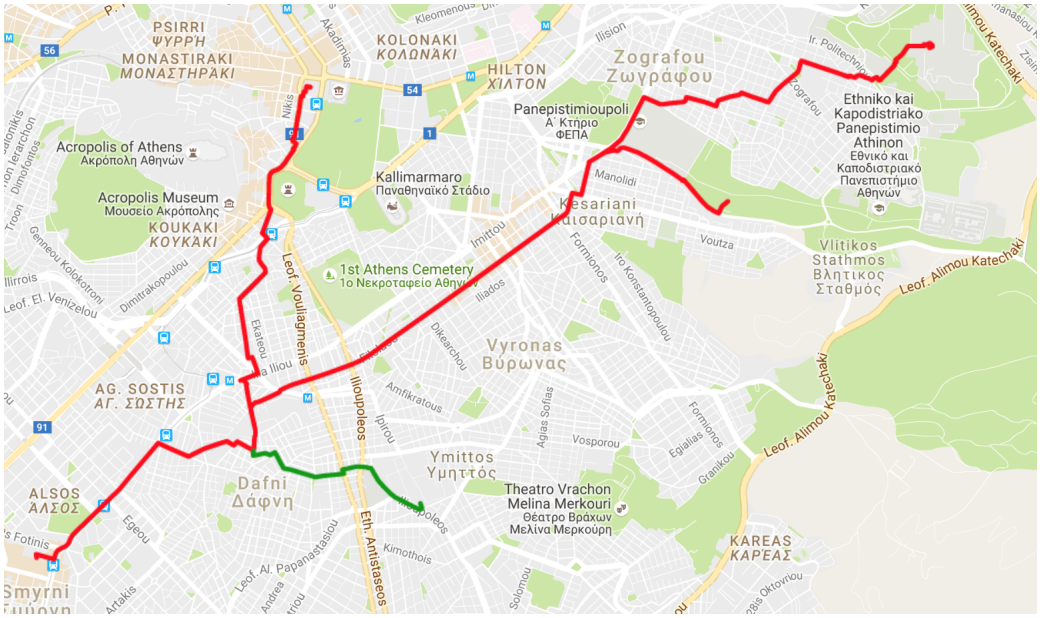
δεδομένα που μας δόθηκαν, το athina_sample.kml είναι το αποτέλεσμα του δικού μας πειραματισμού στο χάρτη της Αθήνας ενώ το αρχείο thessaloniki.kml είναι το αποτέλεσμα της εκτέλεσης στο χάρτη της Θεσσαλονίκης. Στο φάκελο src βρίσκεται ο πηγαίος κώδικας καθώς και τα διάφορα αρχεία που χρειάζονται. Η main βρίσκεται στην κλάση Make.

Στη συνέχεια παρατίθενται εικόνες από τα αποτελέσματα του προγράμματός μας για τις εκάστοτε εκτελέσεις.

• Αθήνα-Δοθέντα δεδομένα:



• Αθήνα-Επιπλέον δεδομένα (όπως βλέπουμε οι διαδρομές κάποιων ταξί ταυτίζονται σε κάποιο σημείο γι΄ αυτό δεν απεικονίζονται όλα):



• Θεσσαλονίκη: