ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ

ΜΥΕ041 - ΠΛΕ081: Διαχείριση Σύνθετων Δεδομένων

(ΕΑΡΙΝΟ ΕΞΑΜΗΝΟ 2020-21)

ΕΡΓΑΣΙΑ 3 – Χωρικά δίκτυα και ερωτήσεις κορυφαίων κ

Προθεσμία: 28 Μαΐου 2021, 9μμ

Για να κατεβάσετε τα δεδομένα χωρικού δικτύου που απαιτούνται για την εργασία, πλοηγηθείτε στο http://www.cs.utah.edu/~lifeifei/SpatialDataset.htm και πατήστε τους συνδέσμους

California Road Network's Nodes (Node ID, Longitude, Latitude)

California Road Network's Edges (Edge ID, Start Node ID, End Node ID, L2 Distance)

Στο αρχείο του 1ου συνδέσμου, κάθε γραμμή είναι ένας κόμβος του δικτύου, ο οποίος χαρακτηρίζεται από έναν identifier (ένα μοναδικό ακέραιο Node ID) και τις χωρικές συντεταγμένες του. Στο αρχείο του 2ου συνδέσμου, κάθε γραμμή είναι μία ακμή του δικτύου, που χαρακτηρίζεται από ένα ζευγάρι κόμβων (μαζί με την Ευκλείδεια απόσταση της ακμής). Θεωρούμε ότι το δίκτυο είναι μη κατευθυνόμενο (δηλ. ότι οι ακμές δεν έχουν κατεύθυνση: αν υπάρχει ακμή από τον κόμβο 0 στον 1 τότε υπάρχει ακμή από τον 1 στον 0).

Μέρος 1: Αποθήκευση και δεικτοδότηση στη μνήμη

Γράψτε ένα πρόγραμμα που διαβάζει τα δεδομένα από τα δοθέντα αρχεία και να κατασκευάζει μια δομή στη μνήμη για αυτά. Συγκεκριμένα, για κάθε κόμβο, η δομή αποθηκεύει (1) τις συντεταγμένες του και (2) σε μια λίστα (λίστα γειτνίασης - adjacency list) τους γείτονές του στο δίκτυο (οι οποίοι υπονοούν την ύπαρξη των αντιστοίχων ακμών). Η δομή σας μπορεί να είναι τύπου λίστας (ή array) στην οποία η θέση ενός κόμβου υπονοεί το ID του. Π.χ. ο κόμβος με Node-ID=0 βρίσκεται στη θέση 0 της δομής. Για κάθε γείτονα αποθηκεύστε στη δομή σας και την Ευκλείδεια απόσταση (L2 Distance) από τον κόμβο στο γείτονα που δίνεται στο αρχείο. Κατόπιν γράψτε σε ένα αρχείο εξόδου με όνομα out.txt τα δεδομένα της δομής. Για παράδειγμα, η πρώτη γραμμή του αρχείου δείχνει ότι ο κόμβος 0 έχει συντεταγμένες (-121.904167,41.974556) και γείτονες τους κόμβους 1 και 6 σε απόσταση 0.002025 και 0.005952 αντίστοιχα:

```
0 -121.904167 41.974556 1 0.002025 6 0.005952
1 -121.902153 41.974766 2 0.01435
...
```

Μπορείτε να χρησιμοποιήστε έτοιμες δομές array, vector ή λιστών που προσφέρονται από τη γλώσσα προγραμματισμού που θα χρησιμοποιήσετε (Python, C, C++ ή Java).

Μέρος 2: Dijkstra και A*

Ζητείται να υλοποιήσετε τους αλγορίθμους κοντύτερου μονοπατιού Dijkstra και A*. Οι συναρτήσεις θα δέχονται σαν είσοδο (α) τον identifier ενός αρχικού κόμβου (source node) και (β) τον identifier ενός τερματικού κόμβου (target node) και θα πρέπει να υπολογίζουν σωστά και να επιστρέφουν (σε μορφή λίστας) το κοντύτερο μονοπάτι από τον αρχικό στον τερματικό κόμβο και την αντίστοιχη δικτυακή απόσταση (δηλ. την απόσταση κοντύτερου μονοπατιού). Επίσης, θα πρέπει να επιστρέφεται και ο αριθμός των επαναλήψεων (δηλαδή των επισκέψεων κόμβων) που λαμβάνουν χώρα κατά την κλήση του αλγορίθμου. Οι αλγόριθμοι θα χρησιμοποιούν τη δομή που φτιάξατε στο πρώτο μέρος της άσκησης. Για τον υπολογισμό του κάτω ορίου της δικτυακής απόστασης μεταξύ 2 μη συνδεδεμένων κόμβων που χρησιμοποιεί ο A* υπολογίστε την Ευκλείδεια απόσταση με βάση τις συντεταγμένες των κόμβων, ενώ για την απόσταση μιας ακμής χρησιμοποιήστε τις έτοιμες τιμές που υπάρχουν στη δομή σας. Το κύριο πρόγραμμα θα δέχεται σαν είσοδο στη γραμμή διαταγών τον αρχικό και τον τερματικό κόμβο, θα καλεί τις 2 συναρτήσεις, και θα τυπώνει τα αποτελέσματά τους (κοντύτερο μονοπάτι και αριθμό επαναλήψεων). Επιτρέπεται η χρήση έτοιμων δομών (π.χ. ουρά προτεραιότητας) από βιβλιοθήκες της γλώσσας προγραμματισμού που θα χρησιμοποιήσετε.

Μέρος 3: Βέλτιστο σημείο συνάντησης

Ζητείται να υλοποιήσετε μια παραλλαγή του top-k αλγορίθμου NRA, η οποία θα λύνει το εξής πρόβλημα. Δίνονται ν κόμβοι του δικτύου, οι οποίοι αναπαριστούν τις αρχικές θέσεις ν χρηστών οι οποίοι θέλουν να συναντηθούν σε κάποιον κόμβο του δικτύου (σημείο συνάντησης). Το βέλτιστο σημείο συνάντησης είναι αυτό το οποίο ελαχιστοποιεί τις αποστάσεις που θα διανύσουν οι χρήστες. Το «κόστος» λοιπόν ενός σημείου συνάντησης είναι συνάθροιση (με χρήση συνάρτησης γ=max) των αποστάσεων που διανύουν οι ν χρήστες για να συναντηθούν. Η απόσταση για έναν χρήστη μετριέται με το μήκος του κοντύτερου μονοπατιού από την αρχική του θέση μέχρι το σημείο συνάντησης. Για παράδειγμα, έστω $n_1, n_2, \ldots, n_v$ οι αρχικές θέσεις (κόμβοι δικτύου) των ν χρηστών και έστω n_σ το σημείο συνάντησης των χρηστών. Τότε, το κόστος του n_σ ορίζεται σαν $\gamma(dist(n_1, n_\sigma), dist(n_2, n_\sigma),..., dist(n_v, n_\sigma))$, όπου $dist(n_i, n_\sigma)$ είναι η απόσταση του κοντύτερου μονοπατιού από το n_i στο n_σ για i=1,…,ν. Το βέλτιστο σημείο συνάντησης είναι το n_σ το οποίο ελαχιστοποιεί το κόστος $\gamma(dist(n_1, n_\sigma), dist(n_2, n_\sigma),..., dist(n_v, n_\sigma))$.

Η λογική του NRA αλγορίθμου είναι:

Από κάθε αρχική θέση χρήστη n_i πάρε τον πρώτο πιο κοντινό κόμβο (ο οποίος θα είναι ο n_i). Για κάθε κόμβο που έχουμε επισκεφτεί από οπουδήποτε κρατάμε τις γνωστές αποστάσεις και τα μονοπάτια από όλες τις αρχικές θέσεις. Συνεχίζουμε την αναζήτηση Dijkstra από τον κόμβο ο οποίος θα μεγαλώσει την ελάχιστη δυνατή απόσταση του σημείου συνάντησης. Συγκεκριμένα, έστω ότι οι αρχικές θέσεις είναι $n_1, n_2, \ldots, n_v$ και ότι οι τελευταίοι κόμβοι που έχουμε επισκεφτεί από αυτές είναι οι $m_1, m_2, \ldots, m_v$ αντίστοιχα. Αν $\min(dist(n_1, m_1), dist(n_2, m_2),..., dist(n_v, m_v)) = dist(n_i, m_i)$ τότε θα συνεχίσουμε το Dijkstra από τον κόμβο n_i. Αν ένας κόμβος n_x έχει επισκεφθεί από όλους τους χρήστες, η ακριβής απόστασή του είναι πλέον

γνωστή και συγκρίνεται με το καλύτερο σημείο συνάντησης μέχρι τώρα, το οποίο και ενημερώνεται ανάλογα. Ο αλγόριθμος σταματάει αν οι υποψήφιοι κόμβοι των οποίων η απόσταση από όλους τους χρήστες δεν είναι γνωστοί, δεν μπορούν να είναι καλύτεροι από το καλύτερο σημείο συνάντησης μέχρι τώρα, με βάση το κάτω όριό τους.

Υλοποιήστε και δοκιμάστε τον αλγόριθμο.

Το πρόγραμμά σας θα πρέπει να παίρνει σαν παράμετρο στη γραμμή διαταγών ένα σύνολο από κόμβους του δικτύου (τα node IDs τους) που είναι οι αρχικές θέσεις των χρηστών. Επιλέξτε σύνολα χρηστών που είναι κοντά η μακριά μεταξύ τους και συγκρίνετε τις επιδόσεις του αλγορίθμου για αυτά. Μπορείτε να χρησιμοποιήσετε έτοιμες δομές και συναρτήσεις της γλώσσας προγραμματισμού που χρησιμοποιείτε.

Οδηγίες ως προς τις υποβολές:

1) Μπορείτε να χρησιμοποιήσετε δομές όπως priority queue ή heap από τις βιβλιοθήκες της γλώσσας προγραμματισμού (π.χ. το module heapq της Python) εάν αυτό απαιτείται.
2) Αν χρησιμοποιήσετε Java, το πρόγραμμά σας θα πρέπει να γίνεται compile και να τρέχει και εκτός Eclipse στους υπολογιστές του εργαστηρίου. Μην χρησιμοποιείτε packages.
3) Αν χρησιμοποιήσετε Python, μην χρησιμοποιήστε τη βιβλιοθήκη pandas και μην υποβάλετε κώδικα για interactive programming (π.χ. ipython)
4) Υποβάλετε τις εργασίες σας σε ένα zip αρχείο (όχι rar) το οποίο πρέπει να περιλαμβάνει όλους τους κώδικες καθώς και ένα PDF αρχείο τεκμηρίωσης το οποίο να περιγράφει τη μεθοδολογία σας. Μην υποβάλετε αρχεία δεδομένων.
5) Μην ξεχνάτε να βάζετε το όνομά σας (σε greeklish) και το ΑΜ σε κάθε αρχείο που υποβάλετε.
6) Ο έλεγχος των προγραμμάτων σας μπορεί να γίνει σε άλλα αρχεία εισόδου από αυτά που σας δίνονται, άρα θα πρέπει ο κώδικάς σας να μην εξαρτάται από τα συγκεκριμένα αρχεία εισόδου που σας δίνονται.
7) Κάντε turnin την εργασία στο assignment3@mye041