Κ23γ: Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
Χειμερινό εξάμηνο 2017-18

1η Προγραμματιστική Εργασία
Κατακερματισμός και αναζήτηση για πολυγωνικές καμπύλες στη C/C++

Η άσκηση πρέπει να υλοποιηθεί σε σύστημα Linux και να υποβληθεί στις Εργασίες του e-class το αργότερο την Παρασκευή 27/10 στις 23.59.

Περιγραφή της εργασίας

Θα υλοποιήσετε τον Locality Sensitive Hashing (LSH) για πολυγωνικές καμπύλες όπως παρουσιάστηκε στο μάθημα για την εύρεση, μέσα σε σύνολο πολυγωνικών καμπύλων, α) του πλησιέστερου γείτονα καμπύλης q και β) [υποχρεωτικό μόνο για τις ομάδες 2 φοιτητών] των γειτόνων εντός ακτίνας R καμπύλης *q.*
Κάθε πολυγωνική καμπύλη ορίζεται ως μια ακολουθία σημείων στον ευκλείδειο χώρο $\Re^d$, d ϵ {2,3,4}, με διαφορετικό πλήθος σημείων.

Το πρόγραμμα θα υποστηρίζει τις εξής μετρικές / συναρτήσεις απόστασης μεταξύ καμπυλών α) Discrete Frechet Distance και [υποχρεωτικά μόνο για τις ομάδες δύο φοιτητών] β) Dynamic Time Warping.

Το LSH καμπυλών στις διαλέξεις απεικονίζει καμπύλες σε «καμπύλες πλέγματος» (grid curves) οι οποίες μπορούν να αναπαριστώνται ως διανύσματα. Υλοποιήστε την αποθήκευση και αναζήτηση στις καμπύλες πλέγματος σε πίνακα κατακερματισμού α) με κλασική ντετερμινιστική συνάρτηση κατακερματισμού διανυσμάτων (classic hash function) και β) [υποχρεωτικό μόνο για τις ομάδες δύο φοιτητών] πιθανοκρατική LSH συνάρτηση κατακερματισμού ευκλείδειων διανυσμάτων.

Ο σχεδιασμός του κώδικα θα πρέπει να επιτρέπει την εύκολη επέκτασή του για διαφορετικές συναρτήσεις απόστασης καθώς και τη μελλοντική χρήση μεμονωμένων συναρτήσεων σε μελλοντικές εργασίες

ΕΙΣΟΔΟΣ

1) Ένα αρχείο κειμένου για την είσοδο του συνόλου δεδομένων (αρχείο dataset) διαχωρισμένο με στηλοθέτες (tab-separated), το οποίο θα έχει την ακόλουθη γραμμογράφηση:

```
@dimension {2,3,4} //default: 2 else define
curve_id1      m1      (x11,y11)  (x12,y12)  ...   (x1m1,y1m1)
    .                      .          .       ...        .
curve_idN      mN      (xN1,yN1)  (xN2,yN2)  ...   (xNmN,y1mN)
```

όπου `(xij,yij)` οι συντεταγμένες double του j σημείου της καμπύλης i, όπου j ≤ mi και mi το πλήθος των σημείων της καμπύλης i, όταν ο αριθμός των διαστάσεων (dimension) = 2.

2) Αρχείο κειμένου που περιλαμβάνει το σύνολο των καμπυλών για τις οποίες αναζητούμε τον πλησιέστερο γείτονα και [υποχρεωτικά για ομάδες 2 φοιτητών] τους τους γείτονες εντός ακτίνας *R*: πρόκειται για το σύνολο αναζήτησης, που περιέχει τουλάχιστον μία καμπύλη (αρχείο αναζήτησης). Ο θετικός double *R* δίνεται στην πρώτη γραμμή. Όταν δίνεται R=0 το πρόγραμμα βρίσκει μόνο τον πλησιέστερο γείτονα των καμπυλών στο σύνολο αναζήτησης.

Το πρόγραμμα αρχικά ζητά από τον χρήστη το μονοπάτι του αρχείου dataset, το είδος της συνάρτησης απόστασης που θα χρησιμοποιηθεί και το είδος της συνάρτησης κατακερματισμού για την αποθήκευση των καμπυλών πλέγματος. Μετά τη δημιουργία της δομής αναζήτησης, το πρόγραμμα ζητά από τον χρήστη το μονοπάτι του αρχείου αναζήτησης και του αρχείου εξόδου των αποτελεσμάτων. Μετά την εκτέλεση του αλγορίθμου και την παραγωγή των αποτελεσμάτων, το πρόγραμμα ζητά από τον χρήστη αν θέλει να τερματίσει το πρόγραμμα ή αν θέλει να επαναλάβει την αναζήτηση για ένα διαφορετικό σύνολο / αρχείο αναζήτησης. Το αρχείο έχει την ακόλουθη μορφή για προβλήματα σε διανυσματικό χώρο, αντιστοίχως με τη μορφή του αρχείου εισόδου:

```
R: <double>
curve_idS1      mS1      (xS11,yS11)  (xS12,yS12)  ...   (xS1mS1,yS1mS1)

    .                        .            .        ...          .

curve_idSN      mSN      (xSN1,ySN1)  (xSN2,ySN2)  ...   (xSNmSN,ySNmSN)
```

Τα ονόματα των καμπύλων στο σύνολο αναζήτησης curve_idSj (1<=*j*<=*Q*) μπορούν να είναι μοναδικοί ακέραιοι ή συμβολοσειρές.

Ως optional παράμετροι μπορεί να δίνονται στη γραμμή εντολών: ο ακέραιος *k* των locality-sensitive συναρτήσεων *hi* που χρησιμοποιούνται για τον ορισμό των συναρτήσεων *g*, καθώς και ο ακέραιος αριθμός *L* των πινάκων κατακερματισμού. Αν τα *k, L* δεν δίνονται, το πρόγραμμα χρησιμοποιεί default τιμές: *k*=4, *L*=5.

Τα αρχεία εισόδου και αναζήτησης θα μπορούν να δίνονται και μέσω παραμέτρων στη γραμμή εντολών, οπότε η εκτέλεση θα γίνεται μέσω της εντολής:

```
$./lsh –d <input file> –q <query file> –k <int> -L <int> -o <output file>
-stats [optional] –function {DFT, DTW} –hash {classic, probabilistic}
```

ΕΞΟΔΟΣ

Αρχείο κειμένου που περιλαμβάνει για κάθε καμπύλη του συνόλου αναζήτησης με τη χρήση των κατάλληλων ετικετών: α) το όνομα του πλησιέστερου γείτονα που βρέθηκε από τον LSH και την απόστασή του από το *q*, β) [υποχρεωτικά μόνο για ομάδες δύο φοιτητών] τα ονόματα των γειτόνων ακτίνας *R.* Αν δοθεί η παράμετρος γραμμής εντολών –stats το πρόγραμμα θα εκτελείται 100 φορές *q*) κατά τις οποίες θα υπολογίζεται ο προσεγγιστικός πλησιέστερος γείτονας του *q* με ανακατασκευή της δομής αναζήτησης και θα υπολογίζονται γ) η απόλυτη τιμή της διαφοράς της ελάχιστης, μέσης και μέγιστης απόστασης του *q* από τον προσεγγιστικό πλησιέστερο γείτονα από την απόσταση του *q* από τον πραγματικό πλησιέστερο γείτονα (όπως αυτός προσδιορίζεται μέσω εξαντλητικής αναζήτησης), δ) ο ελάχιστος, μέσος και μέγιστος χρόνος εύρεσης του πλησιέστερου γείτονα μέσω

LSH και ε) ο χρόνος εύρεσης του αληθινού πλησιέστερου γείτονα με εξαντλητική αναζήτηση. Το αρχείο εξόδου ακολουθεί υποχρεωτικά το παρακάτω πρότυπο στην περίπτωση που δεν ζητείται ο υπολογισμός στατιστικών.

```
Query: curveJ
DistanceFunction: {DFT, DTW}
HashFunction: {Classic, Probabilistic}
FoundGridCurve: {True, False}
LSH Nearest neighbor: curveY
True Nearest neighbor: curveX
distanceLSH: <double>
distanceTrue: <double>
R-near neighbors:
curveA
curveB
. . .
curveW
κ.ο.κ.
```

Στις περιπτώσεις που ζητείται ο υπολογισμός στατιστικών:

```
Query: curveJ
DistanceFunction: {DFT, DTW}
HashFunction: {Classic, Probabilistic}
|minDistanceLSH – distanceTrue|: <double>
|maxDistanceLSH – distanceTrue|: <double>
|avgDistanceLSH – distanceTrue|: <double>
tLSHmin: <double>
tLSHmax: <double>
tLSHavg: <double>
tTrue: <double>
```

Οι απαντήσεις στις αναζητήσεις των R-κοντινών γειτόνων πρέπει να είναι ταξινομημένες σύμφωνα με τη σειρά στο σύνολο αναζήτησης. Οι R-κοντινοί γείτονες σε κάθε απάντηση πρέπει να είναι ταξινομημένοι λεξικογραφικά.

Επιπρόσθετες απαιτήσεις

1. Το πρόγραμμα πρέπει να είναι καλά οργανωμένο με χωρισμό των δηλώσεων / ορισμών των συναρτήσεων, των δομών και των τύπων δεδομένων σε λογικές ομάδες που αντιστοιχούν σε ξεχωριστά αρχεία επικεφαλίδων και πηγαίου κώδικα. Βαθμολογείται και η ποιότητα του κώδικα (π.χ. αποφυγή memory leaks). Η μεταγλώττιση του προγράμματος πρέπει να γίνεται με τη χρήση του εργαλείου make και την ύπαρξη κατάλληλου Makefile.

2. Στην υλοποίηση μπορείτε να χρησιμοποιείτε συναρτήσεις από την πρότυπη βιβλιοθήκη της C/C++ εξαιρουμένων των βιβλιοθηκών για τις απαραίτητες δομές δεδομένων (συνδεδεμένες λίστες, vectors, hashtables), για τις γεννήτριες τυχαίων αριθμών (εκτός από τις συναρτήσεις που δηλώνονται στο `stdlib.h / cstdlib` της πρότυπης βιβλιοθήκης της C/C++), για τις συναρτήσεις κατακερματισμού, για τις συναρτήσεις υπολογισμού αποστάσεων ούτε για άλλη λειτουργία του προγράμματος.

3. Το παραδοτέο πρέπει να είναι επαρκώς τεκμηριωμένο με πλήρη σχολιασμό του κώδικα και την ύπαρξη αρχείου readme το οποίο περιλαμβάνει κατ' ελάχιστο: α) τίτλο και περιγραφή του προγράμματος, β) κατάλογο των αρχείων κώδικα / επικεφαλίδων και περιγραφή τους, γ) οδηγίες μεταγλώττισης του προγράμματος, δ) οδηγίες χρήσης του προγράμματος και ε) πλήρη στοιχεία των φοιτητών που το ανέπτυξαν.

4. Η υλοποίηση του προγράμματος θα πρέπει να γίνει με τη χρήση συστήματος διαχείρισης εκδόσεων λογισμικού και συνεργασίας (Git ή SVN) .