# Κ23γ: Ανάπτυξη Λογισμικού για Αλγοριθμικά Προβλήματα
# Χειμερινό εξάμηνο 2018-19

## 1η Προγραμματιστική Εργασία
### Υλοποίηση δομής για την εύρεση κοντινών γειτόνων στη γλώσσα C/C++

Η άσκηση πρέπει να υλοποιηθεί σε σύστημα Linux και να υποβληθεί στις Εργασίες του e-class το αργότερο την Κυριακή 28/10 στις 23.59.

**Περιγραφή της άσκησης**

(1) Υλοποιήσετε τον αλγόριθμο LSH για διανύσματα στον *d*-διάστατο χώρο βάσει της Ευκλείδειας μετρικής και της ομοιότητας συνημιτόνου (cosine similarity).
(2) Υλοποιήσετε τη μέθοδο τυχαίας προβολής στον υπερκύβο Ο αλγόριθμος θα στηρίζεται στις συναρτήσεις LSH για Ευκλείδεια απόσταση και για cosine similarity.
Αμφότερες μέθοδοι (1-2) θα υλοποιηθούν ώστε, με είσοδο διάνυσμα *q*, να επιστρέφουν (α) τα διανύσματα εντός ακτίνας *R* από το *q* και (β) του κοντινότερου γείτονα στο *q*.
Ο σχεδιασμός του κώδικα θα πρέπει να επιτρέπει την εύκολη επέκτασή του σε διανυσματικούς χώρους με άλλη μετρική, π.χ., *p*-norm, ή διαφορετικούς χώρους καθώς και σε διανύσματα με double συντεταγμένες

**ΕΙΣΟΔΟΣ**

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

```
@metric {euclidean, cosine} // default: euclidean else define
item_id1     X11      X12      ...      X1d

    .         .        .       ...       .

item_idN     XN1      XN2      ...      XNd
```

όπου **Xij** οι συντεταγμένες **int** του διανύσματος *i* στον *d*-διάστατο Ευκλείδειο χώρο. Τα ονόματα (item_idK) μπορούν να είναι μοναδικοί ακέραιοι ή συμβολοσειρές.

2) Αρχείο κειμένου που περιλαμβάνει το σύνολο αναζήτησης δηλ. των διανυσμάτων *q*, και περιέχει τουλάχιστον ένα διάνυσμα. Ο θετικός double *R* δίνεται στην πρώτη γραμμή. Όταν δίνεται R=0 το πρόγραμμα βρίσκει μόνο τον κοντινότερο γείτονα των διανυσμάτων στο σύνολο δεδομένων.

Το πρόγραμμα αρχικά ζητά από τον χρήστη το μονοπάτι του dataset. Μετά τη δημιουργία της δομής αναζήτησης, το πρόγραμμα ζητά από τον χρήστη το μονοπάτι του αρχείου αναζήτησης και του αρχείου εξόδου. Μετά την εκτέλεση του αλγορίθμου και την παραγωγή των αποτελεσμάτων, το πρόγραμμα ζητά από τον χρήστη αν θέλει να τερματίσει το πρόγραμμα ή να επαναλάβει την αναζήτηση για διαφορετικό σύνολο / αρχείο αναζήτησης. Το αρχείο αναζήτησης έχει την ακόλουθη μορφή:

```
Radius: <double>
item_idS1    X11     X12      ...      X1d

     .        .       .       ...       .

item_idSQ    XQ1     XQ2      ...      XQd
```

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

Για το LSH, μπορούν να δίνονται οι εξής παράμετροι προαιρετικά στη γραμμή εντολών: ακέραιο πλήθος *k* των locality-sensitive συναρτήσεων *hi* που χρησιμοποιούνται για τον ορισμό των *g*, ο ακέραιος αριθμός *L* των πινάκων κατακερματισμού. Αν τα *k, L* δεν δίνονται, το πρόγραμμα χρησιμοποιεί default τιμές *k*=4, *L*=5.

Για την τυχαία προβολή στον υπερκύβο, μπορούν να δίνονται οι εξής προαιρετικές παράμετροι στη γραμμή εντολών: η διάσταση στην οποία προβάλλονται τα σημεία *k (=d')*, το μέγιστο επιτρεπόμενο πλήθος υποψήφιων σημείων που θα ελεγχθούν *M*, το μέγιστο επιτρεπόμενο πλήθος κορυφών του υπερκύβου που θα ελεγχθούν probes. Οι default τιμές ειναι: *k*=3, *M*=10, probes=2.

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>
$./cube -d <input file> -q <query file> -k <int> -M <int> -probes <int> -
o <output file>
```

**ΕΞΟΔΟΣ**

Για κάθε μέθοδο: αρχείο κειμένου που περιλαμβάνει για κάθε αντικείμενο του συνόλου αναζήτησης με την χρήση των κατάλληλων ετικετών: α) τα ονόματα των γειτόνων ακτίνας *R*, β) το όνομα του προσεγγιστικά κοντινότερου γείτονα που βρέθηκε και την απόστασή του από το *q*, γ) την απόσταση του *q* από τον αληθινά πλησιέστερο γείτονα (μέσω εξαντλητικής αναζήτησης), δ) τον χρόνο εύρεσης του (β), και ε) τον χρόνο εύρεσης του (γ). Το αρχείο εξόδου ακολουθεί υποχρεωτικά το εξής πρότυπο:

```
Query: itemJ
R-near neighbors:
itemJ
itemK
. . .
itemW
Nearest neighbor: itemY
distanceLSH: <double>
distanceTrue: <double>
tLSH: <double>
tTrue: <double>
```
και ούτω καθεξής.

Επίσης να τυπώνεται στο τέλος της αναζήτησης α) το Μέγιστο (από όλα τα αντικείμενα του συνόλου αναζήτησης) κλάσμα προσέγγισης = Απόσταση προσεγγιστικά κοντινότερου γείτονα / Απόσταση αληθινά κοντινότερου γείτονα, β) ο μέσος χρόνος εύρεσης του προσεγγιστικά κοντινότερου γείτονα.

Συγκρίνετε το LSH και την προβολή σε υπερκύβο: για ένα dataset, αρχείο αναζήτησης και συνάρτηση LSH της επιλογής σας (από τις 2 που υλοποιήσατε), παραμετροποιήστε τις δύο μεθόδους ώστε να πετυχαίνουν σχεδόν ίσο Μέγιστο κλάσμα προσέγγισης. Ποιος είναι ο μέσος χρόνος εύρεσης κοντινότερου γείτονα από κάθε δομή? Ποιος είναι ο χώρος που καταναλώνει κάθε δομή? Σχολιάσετε τα αποτελέσματα στο readme.

## 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

2. 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 του φοιτητή που το ανέπτυξε.

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