Προγραμματιστική Εργασία σε Μεγάλα Σύνολα Δεδομένων (2.5 μονάδες)

Καταληκτική Ημερομηνία Παράδοσης: 31/5/2015

Η παρακάτω προγραμματιστική εργασία είναι 1-2 ατόμων. Τα παραδοτέα της εργασίας θα είναι μία αναφορά με σύντομη ανάλυση της μεθοδολογίας που ακολουθήσατε, με τα πειραματικά αποτελέσματα, καθώς και ο κώδικας σε ηλεκτρονική μορφή. Για τον κώδικα προτείνεται να χρησιμοποιήσετε C(++). Ο κύριος στόχος της εργασίας αυτής είναι να υλοποιήσετε βασικούς αλγορίθμους του μαθήματος, να τους εφαρμόσετε σε πραγματικά μεγάλα σύνολα δεδομένων και να πειραματιστείτε με τις παραμέτρους τους.

1. Κατασκευή Συνόλων Δεδομένων

Από την βιβλιοθήκη **Stanford Large Network Dataset Collection (SNAP)** στο http://snap.stanford.edu/data/index.html θα επιλέξετε τα εξής μεγάλα σύνολα δεδομένων για τα πειράματά σας:
(α) **Road network of California** [αρχείο: **roadNet-CA.txt.gz**]
(β) **LiveJournal online social network** [αρχείο: **soc-LiveJournal1.txt.gz**]
(γ) **Friendster online social network** [αρχείο: **com-friendster.ungraph.txt.gz**]
Τα σύνολα αυτά είναι δίκτυα/γράφοι (κατευθυνόμενοι ή μη) και το βασικό στοιχείο των δεδομένων τους είναι οι συνδέσεις/ακμές που έχουν. Κάθε ακμή ορίζεται από τα **id** των κόμβων που αποτελείται, το **id** του κόμβου της αρχής της, έστω **id1**, και το **id** του κόμβου του τέλους της, έστω **id2**, (ακμή: **id1→id2**). Αν **V** είναι το πλήθος των κόμβων, τα **id** είναι ακέραιοι αριθμοί από **0** έως **V-1**. Στην παρούσα εργασία μας ενδιαφέρει να κωδικοποιήσουμε και να δεικτοδοτήσουμε/ταξινομήσουμε όλες τις ακμές του γράφου μετατρέποντας με έναν απλό τρόπο το ζεύγος των **id** τους σε έναν μεγάλο ακέραιο αριθμό. Θα κατασκευάσετε λοιπόν έναν μικρό κώδικα με τον οποίο θα αποσπάσετε από τα παραπάνω σύνολα τα ζεύγη **(id1,id2)** των ακμών και θα τα καταγράψετε σε νέα αρχεία ως ακεραίους της μορφής: **id1*(R+1)+id2**, όπου **R=max(id2)** το μεγαλύτερο από τα **id2.** Τα τελικά αρχεία των μεγάλων ακεραίων θα είναι τα σύνολα δεδομένων για τα πειράματα που θα ακολουθήσουν.

2. Ταξινόμηση

Υλοποιήστε τον αλγόριθμο συγχώνευσης εξωτερικής μνήμης (**Mergesort**) για την ταξινόμηση των συνόλων δεδομένων που κατασκευάσατε. Ο αλγόριθμος θα χρησιμοποιεί έναν συγχωνευτή **d**-δρόμων, στον οποίο θα δίνονται **d** ταξινομημένες ροές εισόδου ακεραίων και θα φτιάχνει μία ροή εξόδου που θα περιλαμβάνει όλα τα στοιχεία από τις ροές εισόδου ταξινομημένα. Αν **N** το πλήθος των στοιχείων (ακεραίων) του συνόλου δεδομένων και **M** το μέγεθος της διαθέσιμης μνήμης, τότε το πρόγραμμά σας θα πρέπει να παίρνει τις παραμέτρους **N, M** και **d** και θα ταξινομεί ως εξής:

- Θα διαβάζει το αρχείο εισόδου και θα το σπάει σε $\lceil N/M \rceil$ ροές, όπου η κάθε μία θα έχει μέγεθος $\leq M$. Κάθε παραγόμενη ροή θα ταξινομείται στην κύρια μνήμη με χρήση της **QuickSort** πριν γραφεί πάλι στο δίσκο.
- Θα αποθηκεύει τις αναφορές στις $\lceil N/M \rceil$ ροές σε μία ουρά (αν χρειάζεται να την αποθηκεύσετε και αυτή στον δίσκο).

- Θα επαναλαμβάνει την συγχώνευση των **d** πρώτων ροών που βρίσκονται στην ουρά (ή λιγότερων όταν τελειώνουν) με τον συγχωνευτή **d**-δρόμων και θα αποθηκεύει την αναφορά της παραγόμενης ροής στο τέλος της ουράς. Αυτή η διαδικασία θα τερματίσει όταν στην ουρά παραμείνει αναφορά για μία μοναδική ροή (θα είναι όλο το αρχείο ταξινομημένο).

Πειραματιστείτε με διαφορετικές τιμές των **M** και **d** (τουλάχιστον 3) και δείξτε ποιος είναι ο κατάλληλος συνδυασμός για το κάθε ένα πειραματικό σύνολο δεδομένων. Ως βασική μετρική της απόδοσης θα είναι ο συνολικός χρόνος ταξινόμησης. Στην αναφορά σας να καταγράψετε σε γραφήματα ή πίνακες τα αποτελέσματα που βρήκατε και προσπαθήστε να τα ερμηνεύσετε.

3. Κατακερματισμός

Το σύνολο δεδομένων **Road network of California** είναι το πιο μικρό σε μέγεθος και έτσι αποφασίζουμε ότι μπορούμε να το δεικτοδοτήσουμε στην κεντρική μνήμη με κάποια κατάλληλη μέθοδο **hash**. Θα πρέπει να υλοποιήσετε τις εξής μεθόδους:

1. **Κατακερματισμός με ανοικτή διευθυνσιοδότηση:**
    a. **Με γραμμικό έλεγχο**
    b. **Με τετραγωνικό έλεγχο**
2. **Κατακερματισμός με αλυσίδες.**

Θα πρέπει να αποφασίσετε το κατάλληλο μέγεθος του πίνακα hash και τις κατάλληλες συναρτήσεις hash με τις παραμέτρους τους για την κάθε μέθοδο (σύμφωνα με τις στρατηγικές που διατυπώθηκαν στο μάθημα). Μπορείτε αν θέλετε να δοκιμάσετε και διαφορετικά μεγέθη ή συναρτήσεις πριν αποφασίσετε. Οι τελικές επιλογές που θα κάνετε πρέπει να είναι τεκμηριωμένες στην αναφορά σας. Για την αξιολόγηση των σχεδιαστικών σας επιλογών αλλά και για την σύγκριση μεταξύ των μεθόδων θα πρέπει να μετρήσετε και να αναφέρετε:

(α) Τον συνολικό χρόνο κατασκευής της δομής κάνοντας εισαγωγή όλων των ακεραίων του συνόλου δεδομένων έναν προς έναν.

(β) Τον συνολικό αριθμό συγκρούσεων που έγιναν κατά τη διάρκεια της εισαγωγής.

(γ) Τον μέσο χρόνο αναζήτησης. Επειδή μπορεί να είναι εξαιρετικά μικρός, για να τον μετρήσετε θα εργαστείτε ως εξής:

- Θα παράγετε μία τυχαία ακμή **id1→id2** με **id1,id2** τυχαίους ακεραίους στο σύνολο **{0,1,…,V-1}.**
- Θα βρείτε τον αντίστοιχο ακέραιο x=i**d1*(R+1)+id2** της κωδικοποίησής της.
- Θα εξετάσετε αν υπάρχει ο **x** ή όχι στον πίνακα hash (αναζήτηση ακμής).
- Θα επαναλάβετε τόσες φορές ώστε ο χρόνος να είναι μετρήσιμος.

<u>Κάποια βοήθεια:</u>

1. Στη C μπορείτε να χρησιμοποιήσετε την time.h για να μετρήσετε χρόνους.
2. Για να έχετε καλύτερη απόδοση στους χρόνους να θέσετε των compiler να κάνει τη μέγιστη δυνατή βελτιστοποίηση.
3. Για την ανάγνωση και την εγγραφή των στοιχείων σε αρχεία να προτιμήσετε τη χρήση των fread και fwrite συναρτήσεων από τη βιβλιοθήκη stdio καθώς αυτές οι συναρτήσεις υλοποιούν αποδοτικούς μηχανισμούς ενδιάμεσης μνήμης (buffering).
4. Κατά τη μετατροπή των ακμών σε μεγάλους ακεραίους (κωδικοποίηση) θα πρέπει να προσέξετε τα όριά τους (int, long, long long,…) για τις δηλώσεις μεταβλητών και δομών.