K24: Προγραμματισμός Συστήματος
1η Εργασία – Εαρινό Εξάμηνο '16
Προθεσμία Υποβολής: Κυριακή 20 Μαρτίου Ώρα 23:59

Εισαγωγή στην Εργασία:

Στην εργασία αυτή θα υλοποιήσετε μια δομή που να μπορεί να εισάγει/προσπελάσει/τροποποιεί εγγραφές για δοσοληψίες σε ένα τραπεζικό σύστημα και να παρέχει λειτουργίες παρακολούθησης των λογαριασμών (και του ξεπλύματος μαύρου χρήματος). Θεωρούμε ότι τα δεδομένα εγγραφών βρίσκονται μόνο στην κυρίως μνήμη και στην βάση τους αποτελούν ένα κατευθυνόμενο γράφο. Το κόστος εισαγωγής αλλά και εκείνο την ανάκτησης μιας εγγραφής στο γράφο με βάση το κλειδί της πρέπει να είναι (πρακτικά) *O(1)* ανεξάρτητα από τον αριθμό των εγγραφών που υπάρχουν ήδη αποθηκευμένες στην δομή σας. Με βάση την παραπάνω οργάνωση δεδομένων θα πρέπει να να δημιουργήσετε έναν αριθμό από λειτουργίες. Πιο συγκεκριμένα:

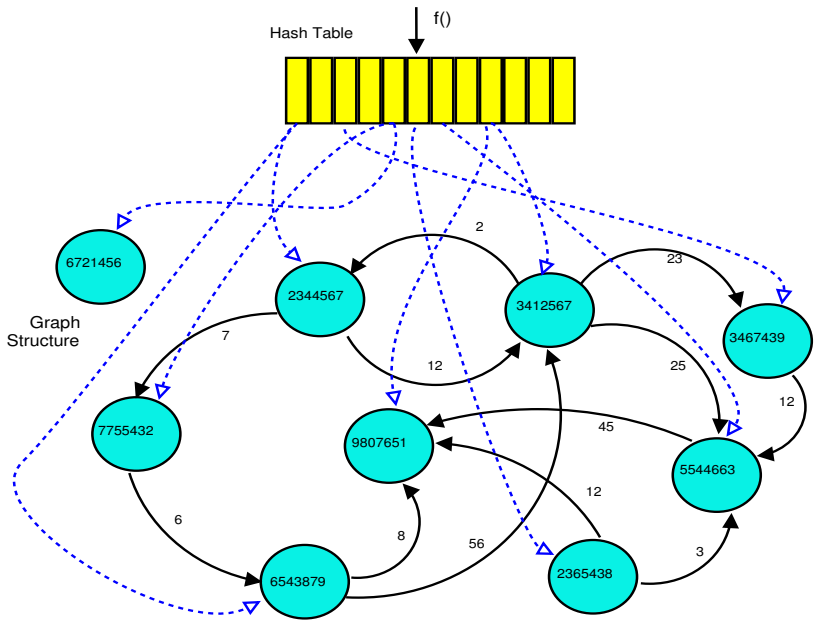
1. Δημιουργία δομής κατευθυνόμενου γράφου (directed graph) που μπορεί να αλλάζει δυναμικά στην κυρίως μνήμη. Οι κόμβοι αναπαριστούν τραπεζικούς λογαριασμούς και οι ακμές εκροές/εισροές ποσών. Δεν υπάρχουν περιορισμοί όσον αφορά στο μέγεθος του γράφου (δηλ. αριθμό κόμβων και ακμών που διαθέτει). Η κάθε (κατευθυνόμενη) ακμή έχει βάρος που ουσιαστικά υποδηλώνει το *συνολικό πόσο* μεταφοράς χρημάτων που έχει λάβει χώρα μεταξύ δύο κόμβων (λογαριασμών).
2. Έλεγχο γράφου για την εύρεση κύκλων, εύρεση της διάχυσης χρημάτων στο δίκτυο, ανεύρεση κυκλικών μεταφορών και διάχυσης ποσών καθώς και εκτύπωση τής δομής με ένα κατανοητό για τους χρήστες τρόπο.
3. Δυνατότητα προσθαφαίρεσης κόμβων και ακμών στη διάρκεια εκτέλεσης του προγράμματος σας και αναπροσαρμογής των τιμών των (υπαρχουσών) ακμών.
4. Ύπαρξη ενός δευτερεύοντος μηχανισμού που να προσφέρει την δυνατότητα εισαγωγής/ανεύρεσης στοιχείων με πρακτικό κόστος *O(1)*. Προφανώς κάτι τέτοιο μπορεί να πραγματοποιηθεί με κατακερματισμό (hashing).
5. Το πρόγραμμα σας θα πρέπει να ελευθερώνει όλη την μνήμη που έχει δεσμεύσει όταν τερματίζει.

Η Σύνθεση της Δομής που θα Δημιουργήσετε:

Το Σχήμα 1 δείχνει μια μερική αλλά αντιπροσωπευτική κατάσταση της σύνθετης δομής που θα δημιουργήσετε για να υλοποιήσετε την εφαρμογή elegxos. Η σύνθετη δομή αποτελείται από ένα κατευθυνόμενο γράφο όπως επίσης και από ένα πίνακα κατακερματισμού που βοηθά στην γρήγορη προσπέλαση στοιχείων του γράφου.

Τα βασικά συστατικά της δομής γράφου είναι τα εξής:

1. Κόμβοι (nodes/vertices) αναπαριστούν τραπεζικούς λογαριασμούς και περιγράφονται από 7 ψηφία.
2. Ακμές/σύνδεσμοι (edges/links) έχουν συγκεκριμένη φορά και 'βάρος' που είναι αρχικά γνωστό και αργότερα μπορεί να αλλάξει σύμφωνα με τις συναλλαγές που έχουν συμβεί μεταξύ δυο οποιοδήποτε κόμβων.
3. Ένας κόμβος μπορεί να συνδέεται με πολλούς άλλους. Το πλήθος γειτόνων δεν είναι γνωστό εξαρχής και φυσικά μπορεί να αυξάνεται δυναμικά.
4. Νέες ακμές μπορούν να εισαχθούν οποιαδήποτε χρονική στιγμή (εφόσον δεν υπάρχουν ήδη στο γράφο).
5. Γενικά δεν υπάρχει περιορισμός στον αριθμό των κόμβων και των ακμών. Υλοποιήσεις με πίνακες για την απεικόνιση του γράφου δεν είναι επαρκείς.



Σχήμα 1: Παράδειγμα Δομής(-ων) για την εφαρμογή elegxos

6. Το Σχήμα 1 δείχνει μια αντιπροσωπευτική κατάσταση της δομής που θα δημιουργήσετε. Στο σχήμα αυτό ο λογαριασμός 2344567 φαίνεται να έχει στείλει 12K Ευρώ στον 3412567 και 7K στον 7755432, ενώ έχει λάβει 2K από τον λογαριασμό 3412567, κοκ.

Μπορούμε να επιτύχουμε γρήγορη προσπέλαση στα δεδομένα που αποθηκεύονται στο γράφο με την βοήθεια ενός (απλού) πίνακα κατακερματισμού ο οποίος φαίνεται στο πάνω κομμάτι του Σχήματος 1.

Γραμμή Κλήσης της Εφαρμογής:

Η εφαρμογή μπορεί να κληθεί με τον παρακάτω τρόπο:
./elegxos -o Operations -b HashtableEntries
όπου

- elegxos είναι το εκτελέσιμο,
- Operations είναι ένα αρχείο σειριακής εισόδου με λειτουργίες που θα πρέπει να εφαρμοστούν στις δομές (δηλ. ερωτήσεις, εισαγωγές, προσαυξήσεις ποσών σε συγκεκριμένες εγγραφές, κλπ.). Η λίστα αυτών των λειτουργιών δίνεται παρακάτω. Τέτοιες λειτουργίες μπορούν να εισαχθούν και χειρωνακτικά και από το prompt της εφαρμογής (δηλαδή το standard input). Εαν υπάρχει και αρχείο (με -o) και εντολές του prompt, οι λειτουργίες του αρχείου εφαρμόζονται πρώτες.
- HashtableEntries είναι ο αριθμός θέσεων του πίνακα γραμμικού κατακερματισμού.

Οι σημαίες -o/-b μπορούν να χρησιμοποιηθούν με οποιαδήποτε σειρά στην γραμμή εκτέλεσης του προγράμματος και δεν μπορείτε να κάνετε αλλαγές στη ονομασία τους. Από αυτές τις in-line παραμέτρους μόνο η -b είναι υποχρεωτική για την επιτυχή εκτέλεση του προγράμματος.

Περιγραφή της Διεπαφής (Λειτουργίες) του Προγράμματος:

Η εφαρμογή επιτρέπει στον χρήστη να αλληλεπιδρά με την δομή και να ανασύρει, αποθηκεύει, ή υπολογίζει διάφορες πληροφορίες με τις παρακάτω εντολές (η μορφή των εντολών είναι αυστηρή):

1. createnodes N1 N2 N3 N4 ...
   η εντολή δημιουργεί έναν η περισσοτέρους κόμβους που ορίζονται ώς: N1, N2, N3, N4, ... Ο κόμβος/-οι παραμένουν ωστόσο ασύνδετοι (όχι μόνο μεταξύ τους αλλά και σε σχέση με προϋπάρχοντες κόμβους).
2. delnodes N1 N2 N3 N4 ....
   διάγραψε από τον γράφο έναν κόμβο N1 (ή και πιο πολλούς N2, N3, N4...) εφόσον δεν υπάρχουν ακμές που να φεύγουν από ή να εισέρχονται στον εν λόγω κόμβο(ή κόμβους).
3. addtran N1 N2 amount
   η εντολή αυτή προσθέτει μια ακόμα ακμή μεταξύ N1 και N2 με βάρος amount αν μια τέτοια ακμή δεν υπάρχει. Αν υπάρχει ήδη η ακμή, απλά το amount προστίθεται στην τιμή που ήδη υπάρχει.
4. deltran N1 N2
   η εντολή διαγράφει την ακμή μεταξύ N1 και N2 ανεξάρτητα από το amount που υπάρχει.
5. lookup [in | out | sum] N
   βρίσκει το συνολικό ποσό που εισέρχεται/εξέρχεται/εισέρχεται-και-εξέρχεται από το λογαριασμό N.
6. triangle N k
   βρίσκει αν ο λογαριασμός N εμπλέκεται σε *τριγωνικές κυκλικές* δοσοληψίες με άλλους (δηλ. υπάρχουν κυκλικές σχέσεις μεταξύ τριών κόμβων εκ των οποίων ο ένας είναι ο N. Ωστόσο το ελάχιστο ποσό που κάθε ακμή πρέπει να έχει, είναι k Ευρώ (εκφρασμένο σε χιλιάδες).
7. conn N1 N2
   βρίσκει αν υπάρχει μονοπάτι απο τον λογαριασμό N1 στον N2 και, εφόσον υπάρχει, το παραθέτει σε αναγνώσιμη μορφή.
8. allcircles N
   βρίσκει και τυπώνει όλους του κύκλους στους οποίους εμπλέκεται ο κόμβος N ανεξάρτητα από τα ποσά στις ακμές. Οι κύκλοι περιλαμβάνουν τουλάχιστον 3 κόμβους και οι κόμβοι εμφανίζονται στους κύκλους μία μόνο φορά. Σημειώστε ότι οι παρακάτω είναι όλοι φυσικά *ένας* (ο ίδιος!) κύκλος: ΑΒΓΔΕ, ΒΓΔΕΑ, ΓΔΕΑΒ, ΔΕΑΒΓ και ΕΑΒΓΔ.
9. traceflow N l
   βρίσκει και παραθέτει όλες τις εκροές χρηματικών ποσών ξεκινώντας από τον κόμβο N και προχωρώντας σε μονοπάτια μήκους/βάθους l. Στα μονοπάτια αυτά κάθε κόμβος εμφανίζεται ακριβώς μία φορά. Κάθε μονοπάτι τυπώνεται ξεχωριστά μαζί με το σύνολο των εκροών που του αντιστοιχούν.
10. bye
    η λειτουργία του γράφου έχει ολοκληρωθεί. Ελευθερώνετε όλο το χώρο που έχουν καταλάβει οι δομές που έχετε χρησιμοποιήσει. Η εντολή αυτή μπορεί να ακολουθείται από δημιουργία νέου γράφου.
11. print
    τυπώνει το γράφο στην οθόνη εργασίας με ένα κατανοητό τρόπο.
12. dump filename(προαιρετικό)
    αν επιλέξετε να υλοποιήσετε την λειτουργία αυτή ουσιαστικά σε οποιαδήποτε φάση εκτέλεσης κάνετε εξαγωγή σε στο filename όλη την δομή που έχει κατασκευαστεί μέχρι στιγμής. Αυτό μπορεί να επιτευχθεί σαν μια ακολουθία από createnodes και addtran για όλους του κόμβους και τις ακμές του γράφου σε αυτή την χρονική στιγμή.

Χαρακτηριστικά του Προγράμματος που Πρέπει να Γράψετε:

1. Δεν μπορείτε να κάνετε pre-allocate οποιοσδήποτε χώρο αφού η δομή(-ές) θα πρέπει να μπορεί(-ουν) να μεγαλώσει(-ουν) χωρίς ουσιαστικά κανέναν περιορισμό όσον αφορά στον αριθμό των εγγραφών που μπορούν να αποθηκεύσουν. Η χρήση στατικών πινάκων/δομών που δεσμεύονται στην διάρκεια της συμβολομετάφρασης του προγράμματος σας γενικά δεν είναι αποδεκτές επιλογές. Γενικά θα πρέπει να *αποφύγετε*

την χρήση στατικών ή δυναμικών πινάκων.

2. Αν πιθανώς κάποια κομμάτια του κωδικά σας προέλθουν από κάποια δημόσια πηγή, θα πρέπει *να δώσετε αναφορά στη εν λόγω πηγή* είτε αυτή είναι βιβλίο, σημειώσεις, Internet URL κλπ. και να εξηγήσετε πως ακριβώς χρησιμοποιήσατε την εν λόγω αναφορά.

Διαδικαστικά:

- Το πρόγραμμά σας θα πρέπει να γραφτεί σε *C* (ή *C++* αλλά χωρίς την χρήση έτοιμων δομών της standard βιβλιοθήκης/STL) και να τρέχει στα `Linux workstations` του τμήματος.
- Το πρόγραμμα σας (source code) πρέπει να αποτελείται από **τουλάχιστον** δυο (και κατά προτίμηση πιο πολλά) διαφορετικά αρχεία. Η χρήση του separate compilation είναι *επιτακτική!*
- Η οργάνωση και η αναγνωσιμότητα (μαζί με την ύπαρξη σχολίων) του κώδικα περιλαμβάνονται στα κριτήρια βαθμολόγησης.
- Για επιπρόσθετες ανακοινώσεις, παρακολουθείτε το forum του μαθήματος στο piazza.com. Η πλήρης διεύθυνση είναι `https://piazza.com/uoa.gr/spring16/k24/home`. Η παρακολούθηση του φόρουμ στο Piazza είναι *υποχρεωτική*.
- Στην διάρκεια των μαθημάτων της πρώτης εβδομάδας κυκλοφορούμε hard-copy λίστα στην τάξη στην οποία θα πρέπει να δώσετε το όνομά σας και το `Unix` user-id σας.

  Με αυτό το τρόπο μπορούμε να γνωρίζουμε ποιος/-α *προτίθεται να υποβάλλει* την 1η αυτή άσκηση και έτσι, μπορούμε να προβούμε στις κατάλληλες ενέργειες για την τελική υποβολή της άσκησης.

Τι πρέπει να Παραδοθεί:

1. Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματος σας (1-2 σελίδες ASCII κειμένου είναι αρκετές) σε αρχείο README.
2. Οποιαδήποτε πηγή πληροφορίας, συμπεριλαμβανομένου και κώδικα που μπορεί να βρήκατε στο Διαδίκτυο ή αλλού θα πρέπει να αναφερθεί και στον πηγαίο κώδικά σας αλλά και στο παραπάνω README.
3. Όλη η δουλειά σας σε ένα tar-file (*OnomaEponymoProject1.tar*) που να περιέχει όλα τα source files, header files, Makefile.

Άλλες Σημαντικές Παρατηρήσεις:

1. Οι εργασίες είναι **ατομικές**.
2. Όποιος υποβάλλει / δείχνει κώδικα που δεν έχει γραφτεί από την ίδια/τον ίδιο **μηδενίζεται** στο μάθημα.
3. Αν και αναμένεται να συζητήσετε με φίλους και συνεργάτες το πως θα επιχειρήσετε να δώσετε λύση στο πρόβλημα, αντιγραφή κώδικα (οποιαδήποτε μορφής) είναι κάτι που **δεν επιτρέπεται**. Οποιοσδήποτε βρεθεί αναμεμιγμένος σε αντιγραφή κώδικά **απλά παίρνει μηδέν** στο μάθημα. Αυτό ισχύει για **όσους εμπλέκονται ανεξάρτητα** από το ποιος έδωσε/πήρε κλπ.