ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
1η Εργασία - Τμήμα: Περιττών Αριθμών Μητρώου
K22: Λειτουργικά Συστήματα – Χειμερινό Εξάμηνο '24
Ημερομηνία Ανακοίνωσης: Τετάρτη 2 Οκτωβρίου 2024
Ημερομηνία Υποβολής: Παρασκευή 25 Οκτωβρίου 2024 Ώρα 23:59

Εισαγωγή στην Εργασία:

Ο στόχος αυτής της εργασίας είναι να εξοικειωθείτε με το περιβάλλον Linux και τα σχετικά εργαλεία προγραμματισμού και ανάπτυξης λογισμικού. Σε C/C++, θα αναπτύξετε μια εφαρμογή με όνομα `miris`, η οποία επιτρέπει την διαχείριση και παρακολούθηση μικρών χρηματικών μεταφορών μεταξύ χρηστών.

Η εφαρμογή θα υλοποιεί μια δομή γράφου που απεικονίζει μεταφορές χρηματικών ποσών μεταξύ τραπεζικών λογαριασμών/χρηστών. Ο γράφος παρέχει ένα μηχανισμό που μπορεί να βοηθήσει στην παρακολούθηση διακίνησης χρηματικών ποσών, στην διαλεύκανση διάχυσης μεταφορών, αλλά και την κατανόηση όσον αφορά στην διαμόρφωση ομάδων πελατών. Θα πρέπει να υλοποιήσετε μια δομή γράφου στην κυρίως μνήμη στην οποία να μπορείτε να εισάγετε λογαριασμούς και δοσοληψίες, καθώς επίσης και να διαπιστώνετε καταστάσεις που πιθανόν δημιουργούνται από την δυναμική συμπεριφορά χρηστών με την βοήθεια επερωτήσεων. Εκτός της βασικής δομής του γράφου, η εφαρμογή σας θα πρέπει να μπορεί να βρίσκει τα στοιχεία ενός πελάτη με την βοήθεια μιας *επιπλέον* δομής που έχει κόστος προσπέλασης *O(1)*. Κάτι τέτοιο μπορείτε να το επιτύχετε με την χρήση κάποιας μορφής κατακερματισμού[1] που θα πρέπει να επιλέξετε και να δικαιολογήσετε.

Μερικές βασικές προϋποθέσεις για την εφαρμογή `miris`, είναι οι εξής:

1. Δημιουργία δομής κατευθυνόμενου (πολυ-)γράφου (directed multi-graph) που μπορεί να αλλάζει δυναμικά στην κυρίως μνήμη. 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 και την ημερομηνία που έγινε η δοσοληψία μεταξύ δύο κόμβων.
2. Έλεγχο γράφου για την εύρεση κύκλων, εύρεση της διάχυσης χρηματικών ποσών στο δίκτυο, και εκτύπωση της δομής με ένα κατανοητό στους χρήστες τρόπο.
3. Δυνατότητα πρόσθεσης κόμβων και ακμών στη διάρκεια εκτέλεσης του προγράμματος σας και αναπροσαρμογής των τιμών των (υπαρχουσών) ακμών.
4. Έλεγχο συνδεσιμότητας γράφου και ανάδειξη ομάδων χρηστών.
5. Το πρόγραμμα σας θα πρέπει στο τέλος να ελευθερώνει σταδιακά όλη την μνήμη που έχει δεσμεύσει.

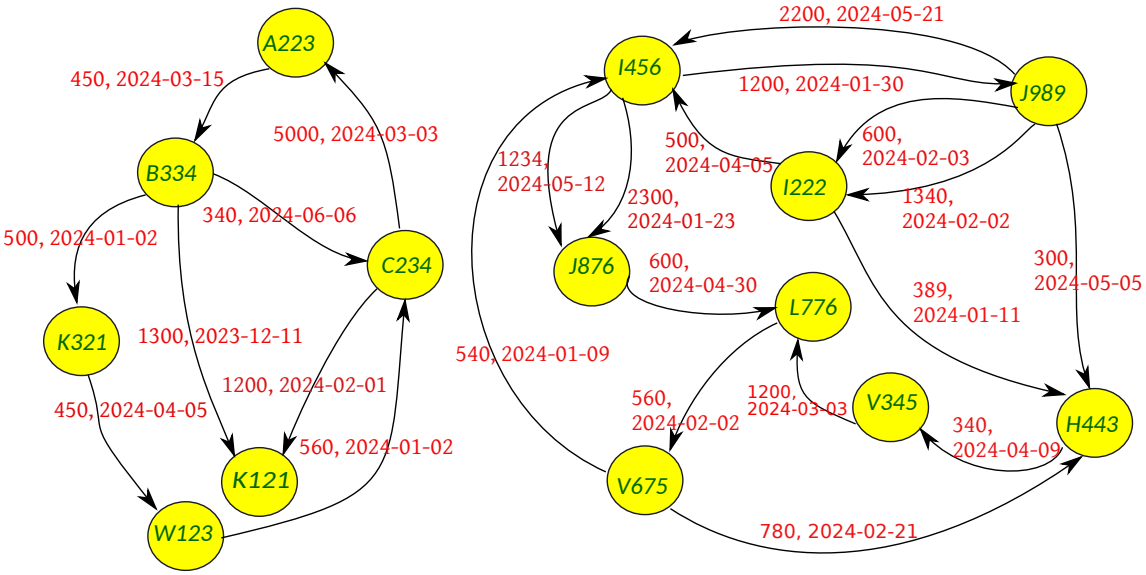
Διαδικαστικά:

Το πρόγραμμα σας θα πρέπει να γραφτεί σε `C` (ή `C++` αν θέλετε αλλά χωρίς την χρήση `STL/Templates`) και να τρέχει στις μηχανές Linux *workstations* του τμήματος.

Ο πηγαίος κώδικας σας (source code) πρέπει να αποτελείται από ***τουλάχιστον δυο*** (και κατά προτίμηση πιο πολλά) διαφορετικά αρχεία και θα πρέπει *απαραιτήτως να γίνεται χρήση* separate compilation .

Παρακολουθείτε την ιστοσελίδα του μαθήματος `https://www.alexdelis.eu/k22/` για επιπρόσθετες ανακοινώσεις.

- Υπεύθυνοι για την άσκηση αυτή (ερωτήσεις, αξιολόγηση, βαθμολόγηση κλπ.) είναι ο Δρ. Σαράντης Πασκαλής (`paskalis+AT-di`), και ο κ. Δημήτρης Ροντογιάννης (`dronto+AT-di`).
- Στη διάρκεια της 1ης εβδομάδας του εξαμήνου θα πρέπει να 'εγγραφείτε' στο μάθημα στην πλατφόρμα `https://eclass.uoa.gr/courses/DI651/`. Με αυτό τον τρόπο μπορούμε να γνωρίζουμε ποιος/-α προτίθεται να υποβάλλει την πρώτη άσκηση *ώστε να προβούμε στις κατάλληλες ενέργειες για την υποβολή της εργασίας σας.*



Σχήμα 1: Παράδειγμα γράφου που χρησιμοποιείται από το `miris`.

- Με βάση την παραπάνω λίστα, θα σας γράψουμε επίσης στο σύστημα `piazza.com`. Στη σελίδα που βρίσκεται στο `https://piazza.com/uoa.gr/fall2024/197eb` μπορείτε να κάνετε ερωτήσεις και να δείτε απαντήσεις ή/και διευκρινήσεις που δίνονται σχετικά με τις προγραμματιστικές ασκήσεις του μαθήματος.

Χαρακτηριστικά Γράφων σε Συντομία:

Τα βασικά συστατικά της δομής είναι τα εξής:

1. Κόμβοι (nodes) έχουν συγκεκριμένα ονόματα (πχ. alphanumeric strings) που αντιπροσωπεύουν όνομα χρηστών ή αριθμούς λογαριασμών.
2. Ακμές/σύνδεσμοι (edges) έχουν συγκεκριμένη φορά και 'βάρος' που είναι γνωστά. Το βάρος αποτελείται από ένα ποσό μεταφοράς και την ημερομηνία που η εν λόγω δοσοληψία συνέβη (Σχήμα 1).
3. Μπορούν να υπάρχουν πολλαπλές ακμές με ίδια φορά μεταξύ δύο κόμβων οι οποίες μπορούν να έχουν και το ίδιο αριθμητικό βάρος. Κάθε μία από αυτές τις ακμές αντιπροσωπεύει μια δοσοληψία.
4. Κάθε δοσοληψία γίνεται σε μια συγκεκριμένη ημερομηνία.
5. Ένας κόμβος μπορεί να συνδέεται με κανέναν, έναν ή πολλούς άλλους. Το πόσο μεγάλο μπορεί να είναι αυτό το πλήθος δεν είναι γνωστό και γενικά μπορεί να αυξάνεται/μειώνεται δυναμικά.
6. Νέες ακμές μπορούν να εισαχθούν οποιαδήποτε χρονική στιγμή.
7. Γενικά δεν υπάρχει περιορισμός στον αριθμό των κόμβων και των ακμών. Έτσι οποιαδήποτε μορφή υλοποίησης που αναλίσκεται σε *στατικούς* πίνακες για την απεικόνιση του γράφου *δεν είναι αποδεκτή*.
8. Το Σχήμα 1 δείχνει μια αντιπροσωπευτική κατάσταση της δομής που θα δημιουργήσετε. Στο σχήμα αυτό υπάρχει μία μεταφορά 560 χρηματικών μονάδων από τον λογαριασμό W123 στον C234.
9. Ο αριθμός των αναμενόμενων εισαγωγών (ή και διαγραφών) κόμβων και ακμών στην δομή *δεν είναι γνωστός εξ' αρχής*.

10. Ο γράφος που θα δημιουργείτε δεν είναι απαραίτητο να έχει όλα τα στοιχεία του συνδεδεμένα (strongly connected). Για παράδειγμα στο Σχήμα 1, οι κόμβοι A223, B334, C234, K321, K121 και W123 αποτελούν ένα συνδεδεμένο κομμάτι του γράφου, ενώ οι υπόλοιποι κόμβοι αποτελούν ένα άλλο.

11. Ανεξάρτητα από την δομή του γράφου που φαίνεται στο Σχήμα 1, θα πρέπει να υπάρχει μια επιπλέον δομή που επιτρέπει «γρήγορη προσπέλαση» σε ένα κόμβο με συγκεκριμένο `id`.

Η Κλήση της Εφαρμογής `miris`

Η εφαρμογή θα πρέπει αυστηρά να μπορεί να κληθεί με βάση την παρακάτω γραμμή εντολής (`tty`):
mymachine-promt >> `./miris -i inputfile -o outfile`

- `./miris` είναι μονοπάτι για το πρόγραμμα σας,
- σημαία `-i` ονοματίζει το αρχείο `input` που περιέχει ένα αρχικό σύνολο κόμβων και μεταφορών,
- σημαία `-o` είναι το αρχείο εξόδου που την στιγμή της εξόδου της εφαρμογής σας καταγράφει την κατάσταση του γράφου όπως αυτή έχει εξελιχτεί με την περάτωση της εφαρμογής.

Αν το κρίνετε απαραίτητο, μπορείτε να χρησιμοποιήσετε περαιτέρω σημαίες/παραμέτρους γραμμής εντολών. Η σειρά με την οποία δίνονται οι διάφορες σημαίες/παράμετροι ως είσοδο στη γραμμή εντολών μπορεί να είναι τυχαία.

Όταν η εφαρμογή αρχικοποιηθεί, παρουσιάζει ένα *prompt*. Κάθε φορά που η χρήστης της εφαρμογής δίνει μια εντολή, το `miris` 'αντιδρά' και φέρνει σε πέρας την σχετική ενέργεια. Στο *prompt*, το πρόγραμμα σας πρέπει να μπορεί να διαχειρίζεται τις παρακάτω εντολές:

1. `i(nsert) Ni [Nj Nk ...]`
   εισήγαγε ένα νέο `Ni` ή πολλαπλούς νέους κόμβους όπως π.χ. `Ni, Nj, Nk`.
2. `(i)n(sert2) Ni Nj sum date`
   εισήγαγε μια ακμή με φορά από `Ni` προς `Nj` με ημερομηνία `date` και ποσό `sum`. Οι κόμβοι `Ni` και `Nj` μπορεί να εμφανίζονται για πρώτη φορά στην δομή.
3. `d(elete) Ni [Nj Nk ...]`
   διάγραψε από το γράφο ένα `Ni` ή πολλαπλούς κόμβους όπως π.χ. `Ni, Nj, Nk, ....` Επίσης όλες τις εισερχόμενες/εξερχόμενες ακμές (και πληροφορίες για ημερομηνίες και ποσά) διαγράφονται.
4. `(de)l(ete2) Ni Nj`
   διέγραψε την ακμή μεταξύ `Ni` και `Nj`. Αν υπάρχουν πολλαπλές ακμές, διέγραψε μια από τις υπάρχουσες.
5. `m(odify) Ni Nj sum sum1 date date1`
   άλλαξε το ποσό `sum` και την ημερομηνία `date` σε νέες τιμές `sum1` και `date1`. Αν υπάρχουν πολλαπλές ακμές με τις ίδιες τιμές `sum` μετάτρεψε την πρώτη που θα βρεις.
6. `f(ind) Ni`
   παρουσίασε όλες τις εξερχόμενες δοσοληψίες που ο εν λόγω κόμβος παρουσιάζει.
7. `r(eceiving) Ni`
   παρουσίασε όλα τις δοσοληψίες που ο κόμβος `Ni` δέχεται σαν είσοδο.
8. `c(irclefind) Ni`
   βρίσκει αν ο κόμβος `Ni` εμπλέκεται σε απλούς κύκλους (simple cycles) και παρουσιάζει τους εν λόγω κύκλους.
9. `(fi)n(dcircles) Ni k`
   βρίσκει αν ο κόμβος `Ni` εμπλέκεται σε κυκλικές δοσοληψίες με άλλους. Ωστόσο το ελάχιστο ποσό που κάθε ακμή πρέπει να έχει, είναι `k` μονάδες (τιμή για minimum `sum`).

10. `t(raceflow) Ni m`
    βρίσκει όλες τις ροές ποσών που ξεκινούν από τον κόμβο `Ni` και διατρέχουν μονοπάτια μήκους (βάθους) `m`. Το `m` δίνει το μέγιστο μήκος της διαδρομής σε αριθμό ακμών που θα πρέπει να διατρεχτούν.
11. `(c)o(nnected) Ni Nj`
    εντοπίζει αν υπάρχει μονοπάτι από το `Ni` στο `Nj` και το τυπώνει.
12. `e(xit)`
    το πρόγραμμα τερματίζει αφού τυπώσει στο αρχείο εξόδου την κατάσταση του γράφου όπως έχει εξελιχτεί, ελευθερώνει όλο τον χώρο που δυναμικά έχει πάρει το `miris` στην διάρκεια της εκτέλεσή του, και τυπώνει τον αριθμό των εν λόγω Bytes.

Η ακριβής αναμενόμενη *μορφή εξόδου* της κάθε μία από τις παραπάνω εντολές δίνεται με *λεπτομέρεια* στην σελίδα του μαθήματος.

Χαρακτηριστικά του Προγράμματος που Πρέπει να Γράψετε:

1. Δεν μπορείτε να κάνετε pre-allocate με στατικό τρόπο οποιοδήποτε χώρο αφού 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 *δεν είναι αποδεκτές επιλογές*.
2. Για *δυναμικές* δομές που θα υιοθετήσετε, θα πρέπει να μπορείτε να εξηγήσετε (και να δικαιολογήσετε στην αναφορά σας) την πολυπλοκότητα που παρουσιάζουν.
3. Πριν να τερματίσει η εκτέλεση της εφαρμογής, το περιεχόμενο των δομών απελευθερώνεται με σταδιακό και ελεγχόμενο τρόπο.
4. Έχετε πλήρη ελευθερία να επιλέξετε το τρόπο με τον οποίο τελικά θα υλοποιήσετε βοηθητικές δομές.
5. Τέλος, μπορείτε να εισαγάγετε και επιπλέον εντολές της επιλογής σας στις οποίες το `miris` μπορεί να ανταποκρίνεται.

Τι πρέπει να Παραδοθεί:

1. Μια σύντομη και περιεκτική εξήγηση για τις επιλογές που έχετε κάνει στο σχεδιασμό του προγράμματος σας (2-3 σελίδες σε ASCII κείμενο είναι αρκετές).
2. Οπωσδήποτε ένα `Makefile` (που να μπορεί να χρησιμοποιηθεί για να γίνει αυτόματα το compile του προγραμματός σας). Πιο πολλές λεπτομέρειες για το (`Makefile`) και πως αυτό δημιουργείται δίνονται στην ιστοσελίδα του μαθήματος.
3. Ένα `zip/7z` αρχείο με όλη σας τη δουλειά σε έναν κατάλογο που πιθανώς να φέρει το ονομά σας και θα περιέχει όλη σας την δουλειά δηλ. `source files, header files, output files` (αν υπάρχουν) και οτιδήποτε άλλο χρειάζεται.

Βαθμολόγηση Προγραμματιστικής Άσκησης:

Σημεία Αξιολόγησης Άσκησης	*Ποσοστό Βαθμού (0–100)*
Ποιότητα στην Οργάνωση Κώδικα & Modularity	20%
Οργάνωση Δομής Γράφου	10%
Βοηθητική Δομή για Προσπέλαση Κόμβων	10%
Σωστή Εκτέλεση Εντολών `miris`	40%
Χρήση Makefile & Separate Compilation	08%
Αποφίλωση Λαθών με *Valgrind*	06%
Επαρκής/Κατανοητός Σχολιασμός Κώδικα	06%

Άλλες Σημαντικές Παρατηρήσεις:

1. Η εργασία είναι **ατομική**.
2. Το πρόγραμμα σας θα πρέπει να τρέχει στα `Linux` συστήματα του τμήματος αλλιώς **δεν μπορεί να βαθμολογηθεί**.
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** και δεν πρέπει να γίνει. Οποιοσδήποτε βρεθεί αναμεμειγμένος σε αντιγραφή κώδικα **απλά παίρνει μηδέν** στο μάθημα. Αυτό ισχύει για **όσους εμπλέκονται ανεξάρτητα** από το ποιος έδωσε/πήρε κλπ.
4. Το παραπάνω επίσης ισχύει αν διαπιστωθεί ***έστω και μερική άγνοια*** του κώδικα που έχετε υποβάλει ή ***άπλα υπάρχει υποψία*** ότι ο κώδικας είναι προϊόν συναλλαγής με τρίτο/-α άτομο/α ή με συστήματα αυτόματης παραγωγής λογισμικού ή με αποθηκευτήρια (repositories) οποιασδήποτε μορφής.
5. Αναμένουμε ότι όποια υποβάλει την εν λόγω άσκηση θα πρέπει να έχει πλήρη γνώση και δυνατότητα εξήγησης του κώδικα. Αδυναμία σε αυτό το σημείο οδηγεί σε μηδενισμό στην άσκηση.
6. Προγράμματα που **δεν χρησιμοποιούν** separate compilation χάνουν αυτόματα 8% του βαθμού.
7. Σε καμιά περίπτωση τα Windows *δεν είναι επιλέξιμη* πλατφόρμα για την παρουσίαση αυτής της άσκησης.

Αναφορές

[1] R. Sedgewick, *Algorithms in C: Parts 1-4*, Addison-Wesley, Boston, MA 2002.