Κ24: Προγραμματισμός Συστήματος
1η Εργασία – Εαρινό Εξάμηνο '17
Προθεσμία Υποβολής: Κυριακή 19 Μαρτίου Ώρα 23:59

Εισαγωγή στην Εργασία:

Στην εργασία αυτή θα υλοποιήσετε σύνολο δομών που επιτρέπουν την εισαγωγή/τροποποίηση όπως και επερωτήσεις σε μεγάλο όγκο εγγραφών τύπου `CDR (Call Detail Record)`. `CDR`s είναι εγγραφές που δημιουργούνται από υποδομές τηλεφωνίας/Internet όταν πελάτες επιχειρούν ένα τηλεφώνημα, κάνουν μια κλήση δεδομένων, στέλνουν SMS κλπ. Αν και τα δεδομένα της Άσκησης θα προέρχονται από αρχεία, τελικά όλες οι εγγραφές θα αποθηκεύονται μόνο στην κύρια μνήμη και μαζί με τις μεθόδους προσπέλασης που θα δημιουργήσετε θα αποτελούν ένα απλό warehouse για `CDR`s.

Το κόστος εισαγωγής αλλά και εκείνο της ανάκτησης μιας εγγραφής με βάση το κλειδί της (το τηλέφωνο του καλούντα) στην «βασική δομή» της άσκησης θα πρέπει να είναι πρακτικά *O(1)* ανεξάρτητα από τον αριθμό των εγγραφών που υπάρχουν ήδη αποθηκευμένες (δηλ. θα πρέπει να υιοθετήσετε μια μορφή hashing). Θα πρέπει επίσης να δημιουργήσετε ταυτόχρονα άλλες «δευτερεύουσες δομές» που θα επιτρέπουν γρήγορες απαντήσεις σε επερωτήσεις που ένας χρήστης πιθανόν θα ήθελε να εξάγει από το σύνολο των εγγραφών. Πιο συγκεκριμένα θα πρέπει να δημιουργήσετε:

- Μία δομή κατακερματισμού που χρησιμοποιεί buckets και που αποθηκεύει όλα τα εισαγόμενα `CDR`s στην κύρια μνήμη. Το κλειδί προσπέλασης στην εν λόγω δομή θα είναι το τηλέφωνο του καλούντα και αλλαγές (προσθαφαιρέσεις) στην εν λόγω βασική δομή επιτρέπονται οποιαδήποτε χρονική στιγμή.
- Μια δεύτερη δομή (πιθανόν με παρόμοια διάταξη) που θα επιτρέπει γρήγορη προσπέλαση στις εγγραφές από την πλευρά όμως των καλουμένων αριθμών.
- Μια δομή που να δίνει την δυνατότητα να μπορεί η/ο χρήστης της εφαρμογής να δημιουργήσει το σύνολο των κορυφαίων-καλούντων που δημιουργούν το top k% της χρηματικής εισροής του τηλεπικοινωνιακού οργανισμού.
- Δυνατότητα προσθαφαίρεσης εγγραφών στη διάρκεια εκτέλεσης του προγράμματος σας. Τέτοιες λειτουργίες μπορεί να επηρεάζουν όλες τις δομές που χρησιμοποιούνται και που θα πρέπει να αναδιατάσσονται κατάλληλα.
- Το πρόγραμμα σας θα πρέπει να ελευθερώνει όλη την μνήμη που έχει δεσμεύσει όταν τερματίζει με εύλογο τρόπο.

Η Σύνθεση των Εγγραφών `CDR`s:

Κάθε `CDR` είναι ουσιαστικά μια γραμμή ASCII κειμένου που αποτελείται από τα εξής στοιχεία:

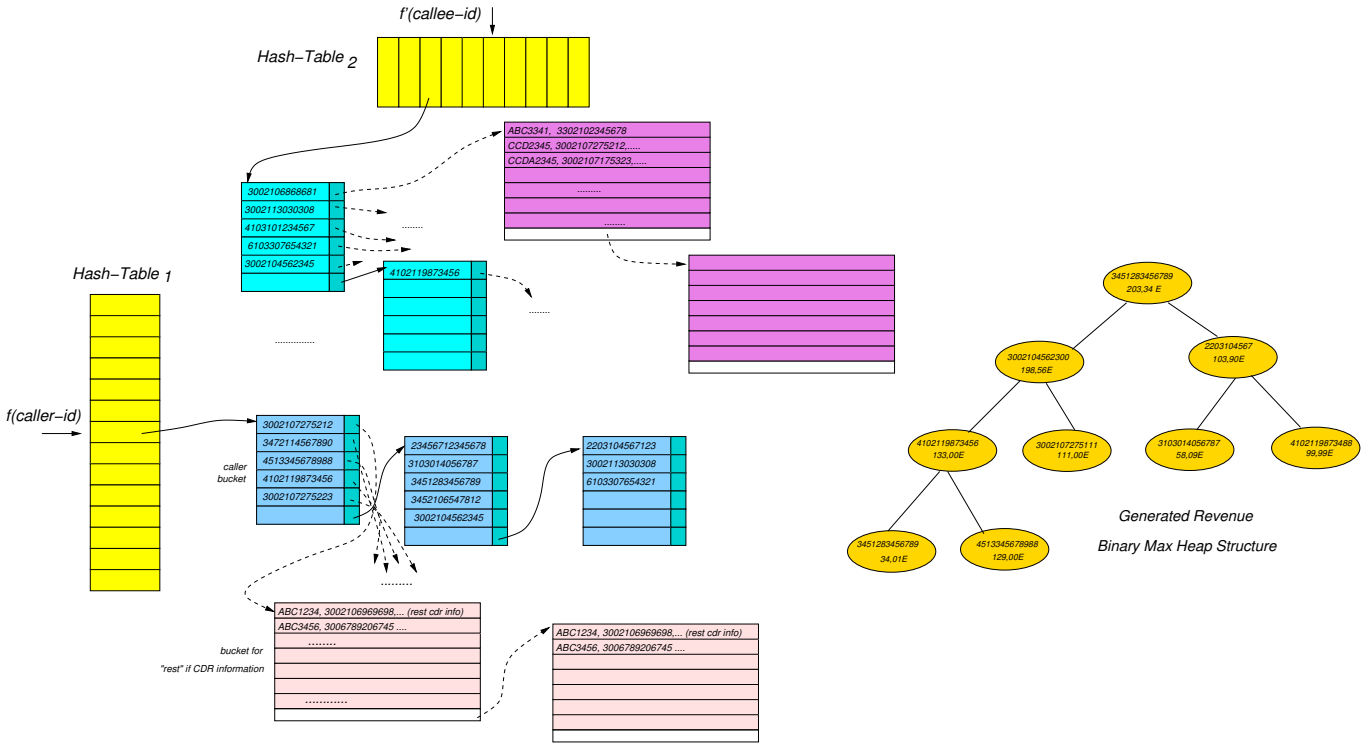
1. `cdr-uniq-id:` μια συμβολοσειρά (μπορεί να έχει και μόνο ψηφία) που με μοναδικό τρόπο καθορίζει την κάθε τέτοια εγγραφή.
2. `originator-number:` είναι ο αριθμός που καλεί και αποτελείται από ένα 3–ψήφιο κωδικό χώρας και ένα 10–ψήφιο τηλεφωνικό νούμερο (π.χ. 357-2127272314 είναι ένα νούμερο για Κύπρο).
3. `destination-number:` ο καλούμενος αριθμός που έχει την ίδια δομή όπως ο παραπάνω.
4. `date:` ημερομηνία κλήσης με την μορφή DDMMYYYY όπου το DD εκφράζει την ημέρα, το MM το μήνα και το YYYY το χρόνο κλήσης (π.χ. 12012017 να δηλώνει την 12η Ιανουαρίου του 2017).
5. `init-time:` ώρα στο 24ωρο με την μορφή HH:MM στην οποία ξεκίνησε η κλήση (π.χ. 21:25 να υποδηλώνει ότι η κλήση έγινε στις 9 και 25 μμ.).

6. `duration:` διάρκεια της κλήσης σε λεπτά.
7. `type:` είδος κλήσης που μπορεί να είναι ομιλία, διεκπεραίωση δεδομένων και SMS.
8. `tarrif:` τιμολόγιο που ισχύει για την εν λόγω κλήση και μπορεί να είναι ενδοχώρια, διεθνής για κλήσεις ομιλίας και δεδομένων, και κόστος μικρού μηνύματος. Για απλοποίηση μπορεί να έχουμε μόνο τριών ειδών τιμολογήσεις.
9. `fault-condition:` περιγράφει με ένα ακέραιο το πρόβλημα (αν υπήρξε τέτοιο) το οποίο και εμπόδισε την ομαλή εξέλιξη της κλήσης (π.χ., διακοπή κλήσης λόγω προβλήματος στο router, λόγω hand-off, κλπ.).

Τα διάφορα πεδία του `CDR` μπορούν να χωρίζονται μεταξύ τους στο αρχείο εισόδου με ';'.

Η Σύνθεση της Δομής που θα Δημιουργήσετε:

Οι εγγραφές `CDRs` θα διαβάζονται από ένα η πολλαπλά αρχεία εισόδου. Η εισαγωγή των εγγραφών οδηγεί στην δημιουργία και διαχείριση ενός αριθμού από δομές που είναι δύο πίνακες κατακερματισμού (hash-tables) και ένας διαδικό-μαξιμαλιστικός σωρός (binary max heap [;]). Το Σχήμα 1 δείχνει μια μερική αλλά αντιπροσωπευτική κατάσταση της σύνθετης δομής που θα δημιουργήσετε για να υλοποιήσετε την εφαρμογή werhauz. Η σύνθετη δομή αποτελείται από δύο hash-tables (που χρησιμοποιούν κουβάδες συγκεκριμένου μεγέθους για να αποθηκεύουν τα στοιχεία των εγγραφών που διαβάζονται) και ένα binary max heap το οποίο για κάθε τηλέφωνο που υπάρχει διαθέτει το συνολικό ποσό που έχει δημιουργήσει σαν εισόδημα για την εταιρεία τηλεφωνίας.



Σχήμα 1: Παράδειγμα Δομών για την εφαρμογή werhauz

Τα βασικά χαρακτηριστικά των δομών του Σχήματος1 είναι τα εξής:

1. Υπάρχουν δύο πίνακες κατακερματισμού (hash-table1 και hash-table2) που προσφέρουν γρήγορες προσπελάσεις στα στοιχεία των εγγραφών από τους «καλούντες αριθμούς» αλλά και από τους «καλούμενους

τηλ. αριθμούς».

2. Οι πίνακες κατακερματισμού θα χρησιμοποιούν κουβάδες για να εξυπηρετήσουν εγγραφές που που παρουσιάζουν «σύγκρουση»/collision (δηλ. το αποτέλεσμα της συνάρτησης κατακερματισμού οδηγεί στο ίδιο στοιχείο του hash table. Αν χρειάζονται πιο πολλοί από ένα κουβάδες για να αποθηκευτούν δεδομένα, δημιουργούνται δυναμικά και διατάσσονται σε μια λίστα. Για παράδειγμα τα τηλεφωνικά νούμερα 3002107275212 και 4513345678988 καταλήγουν στο ίδιο στοιχείο του πρώτου πίνακα κατακερματισμού και συνεπώς στο ίδιο κουβά.

   Για κάθε τέτοιο αριθμό υπάρχει ένα σύνολο από τηλέφωνα (CDRs) στα οποία έχει γίνει κλήση. Η δευτερεύουσα λίστα (που φαίνεται στο κάτω τμήμα του Σχήματος 1 (σε χρωματισμό light pink) ουσιαστικά παρέχει «όλες» τις πληροφορίες για τις κλήσεις που έχει πραγματοποιήσει το νούμερο 3002107275212.

3. Ο πρώτος πίνακας κατακερματισμού ουσιαστικά παρέχει προσπέλαση για τους ποιους έχει καλέσει ένας τηλ. αριθμός. Ο δεύτερος πίνακας κάνει το *αντίστροφο*: για κάθε αριθμό που έχει δεχτεί κλήση προσφέρει πληροφορίες για τους αριθμούς που τον/την έχουν καλέσει (δηλ. inverted access method).

4. Και οι δύο πίνακες μπορούν να οργανωθούν με τον ίδιο τρόπο. Αυτό είναι και το κατ΄ ελάχιστον αναμενόμενο στην άσκηση αυτή αφού έτσι τα δεδομένα θα αποθηκευτούν μαξιμαλιστικά «δύο» φορές στους 2 πίνακες κατακερματισμού. Ο *στόχος* σε αυτή την άσκηση είναι να υπάρξει *όσο το δυνατόν μικρότερη επικάλυψη στοιχείων (data duplication)*.

5. Καθώς οι εγγραφές εισάγονται, επίσης δημιουργούμε ένα binary max-heap που ο στόχος του είναι να προσφέρει γρήγορη προσπέλαση στους «καλύτερους πελάτες» (δηλ. αυτούς που δαπανούν πιο πολλά χρήματα για τις τηλεπικοινωνιακές τους ανάγκες). Στο δεξί τμήμα του Σχήματος 1 παραθέτουμε ένα δείγμα τέτοιας δομής. Το τηλ. νούμερο 3451283456707 έχει δαπανήσει μέχρι στιγμής 203,34 €και είναι η κάλλιστη επίδοση από πλευράς πελατών. Το binary max-heap είναι ουσιαστικά ένα priority queue.

Γραμμή Κλήσης της Εφαρμογής:

Η εφαρμογή μπορεί να κληθεί με τον παρακάτω τρόπο:
./werhauz -o Operations -h1 Hashtable1NumOfEntries -h2 Hashtable2NumOfEntries -s BucketSize -c config-file
όπου

- ./werhauz 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). Εάν υπάρχει και αρχείο (με -o) και εντολές του prompt, οι λειτουργίες του αρχείου εφαρμόζονται πρώτες.
- Hashtable1NumOfEntries είναι ο αριθμός θέσεων του πίνακα κατακερματισμού 1.
- Hashtable2NumOfEntries είναι ο αριθμός θέσεων του πίνακα κατακερματισμού 2.
- BucketSize είναι ο αριθμός των Bytes που δίνει το μέγεθος του κάθε bucket στους πίνακες κατακερματισμού.
- config-file είναι ένα προαιρετικό configuration αρχείο που μπορεί να χρησιμοποιηθεί για την παραμετροποίηση της εφαρμογής σας. Η χρήση της εν λόγω σημαίας είναι προαιρετική.

Οι σημαίες μπορούν να χρησιμοποιηθούν με οποιαδήποτε σειρά στην γραμμή εκτέλεσης του προγράμματος και δεν μπορείτε να κάνετε αλλαγές στη ονομασία τους.

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. insert cdr-record
   εισαγωγή ενός CDR στις δομές.
2. delete caller cdr-id
   διαγραφή μιας εγγραφής που έχει συγκεκριμένο cdr-id και ο καλών είναι ο caller.
3. find caller [time1][year1] [time2][year2]
   παρουσίασε όλα τα CDRs ενός συνδρομητή (με επιλογή στο εύρος χρόνου ή/και ημερομηνίας). Αν υπάρχει ορισμος για [time1] θα πρέπει να υφίσταται και ορισμός για [time2]. Επίσης το ίδιο ισχύει και για την χρήση των μη υποχρεωτικών παραμέτρων [year1] και [year2].
4. lookup callee [time1][year1] [time2][year2]
   για τον συνδρομητή βρείτε όλες τις κλήσεις που έχει δεχτεί (επίσης με επιλογή στο εύρος χρόνου ή/και ημερομηνίας).
5. indist1 caller1 caller2
   παρουσίασε όλους τους συνδρομητές που έχουν επικοινωνήσει και με την caller1 και την caller2 αλλά οι δύο τελευταίοι δεν έχουν καμία άμεση επικοινωνία μεταξύ τους.
6. topdest caller
   βρες τον κωδικό χώρας που έχει καλέσει ο caller τις πιο πολλές φορές. Ποιος είναι ο κωδικός αλλά και ο αριθμός των κλήσεων;
7. top k
   βρες τους αριθμούς των συνδρομητών που έχουν δημιουργήσει το top-k% του εισοδήματος της εταιρείας.
8. bye
   η λειτουργία του werhauz έχει ολοκληρωθεί. Ελευθερώνετε όλο το χώρο που έχουν καταλάβει οι δομές που έχετε χρησιμοποιήσει. Η εντολή αυτή μπορεί να ακολουθείται από δημιουργία νέων δομών.
9. print hashttableX
   τυπώνει το περιεχόμενο του hastable X στην οθόνη εργασίας με ένα κατανοητό τρόπο.
10. dump hashttableX filename(προαιρετικό)
    αν επιλέξετε να υλοποιήσετε την λειτουργία αυτή ουσιαστικά σε οποιαδήποτε φάση εκτέλεσης κάνετε εξαγωγή σε στο filename όλη την δομή για το hashtable X που έχει κατασκευαστεί μέχρι στιγμής.

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 *δεν είναι αποδεκτές επιλογές*. Γενικά, θα πρέπει να *αποφύγετε την χρήση στατικών πινάκων*.
2. Αν πιθανώς κάποια κομμάτια του κωδικά σας προέλθουν από κάποια δημόσια πηγή, θα πρέπει *οπωσδήποτε να δώσετε αναφορά στην εν λόγω πηγή* είτε αυτή είναι βιβλίο, σημειώσεις, Internet URL κλπ. και να εξηγήσετε πως ακριβώς χρησιμοποιήσατε την εν λόγω αναφορά.
3. Έχετε πλήρη ελευθερία να επιλέξετε το τρόπο με τον οποίο τελικά θα υλοποιήσετε βοηθητικές δομές.

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/spring2017/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.

Αναφορές

[1] J. Williams, Binary Heap: Description of A Priority Queue Structure, `https://en.wikipedia.org/wiki/Binary_heap`, February 2017.