<u>Σικαλιάς Φώτιος - 1115201000155</u>

Προγραμματισμός Συστήματος – Project 1

Το πρόγραμμα οργανώνεται ως εξής. Για κάθε ζητούμενη λειτουργία υπάρχει και ένα ξεχωριστό αρχείο .c μαζί με ένα αντίστοιχης ονομασίας αρχείο .h. Το μόνο αρχείο .h που δεν έχει ένα αντίστοιχης ονομασίας .c είναι το definition.h που περιέχει όλες τις αρχικές δομές για να στηθεί το πρόγραμμα.

Όσον αφορά τα ζητούμενα τις άσκησης, υλοποιήθηκαν όλα εκτός του dump και της bonus υλοποίησης.

Σχεδιαστικά, ακολούθησα την εξής λογική.

- Το hashtable1(HT1) είναι ένα hashtable που η κάθε του θέση περιέχει έναν δείκτη σε ένα struct-κόμβο λίστας bucketNode1_caller και έναν ακεραιο μετρητή που μας δείχνει για λόγους ευκολίας πόσοι τέτοιοι κόμβοι έχουν δημιουργηθεί για τη συγκεκριμένη θέση του HT1.
- Ο κάθε τέτοιος κόμβος περιέχει μεταξύ άλλων και έναν δείκτη στην αρχή ενός πίνακα από structs bucket1_caller, που είναι το blue bucket της εκφώνησης. Το bucket1_caller περιέχει τον caller, έναν δείκτη σε ένα struct-κόμβο λίστας bucketNode2_caller, έναν ακέραιο μετρητή και έναν δείκτη στον αντίστοιχο “κόμβο” του max-heap μας.
- Ο κάθε κόμβος bucketNode2_caller περιέχει μεταξύ άλλων και έναν δείκτη στη αρχή ενός πίνακα από structs bucket2_caller, που είναι το lightpink bucket της εκφώνησης. Το bucket2_caller περιέχει τον callee και γενικά όλα τα υπόλοιπα CDR της κάθε κλήσης.
- Οσον αφορά το heap, δημιουργείται παράλληλα με το insert. Άμα πρόκειται για καινούριο κόμβο τότε καλώ την insertNode, εάν όχι, τότε καλώ την updateNode. Όποια συνάρτηση από τις 2 και να κληθεί, μετά καλείται μια heapify για τον κόμβο αυτόν για να σιγουρευτούμε ότι ο max-heap μας διατηρεί την ιδιότητά του ως προς τη συνολική χρηματική προσφορά του κάθε συνδρομητή στην εταιρεία. Οι κόμβοι του max-heap είναι διπλά συνδεδεμένοι μεταξύ τους και περιέχουν και έναν δείκτη προς το αντίστοιχο bucketNode1_caller.
- Το αρχείο κοστολόγησης το ανοίγω και βάζω τους πιθανούς συνδυασμούς σε μια λίστα με κεφαλή το headNodeCharge.

(Τα ίδια ισχύουν και για το hashtable2 με τη διαφορά ότι όπου είχα caller γίνεται callee, origNum γίνεται destNum και ότι το bucket2_callee δεν περιέχει κάποια πληροφορία για το max-heap)

Ειδικότερα, λίγα λόγια για το κάθε ερώτημα.

- **Insert:** Ο αριθμός caller μπαίνει σε μια hashFunction και μας γυρνάει τη θέση του HT που πρέπει να μπεί. Στη θέση αυτή, παίρνουμε τον δείκτη στην αρχή της λίστας με τα bucketNode1_caller και πάμε να δούμε αν χωράει σε κάποια

θέση του bucket1_caller. Αν δε χωράει, απλά προχωράμε στη λίστα μέχρι να βρούμε κενή θέση και αν δε βρούμε πουθενά τότε δημιουργούμε bucketNode1_caller, bucket1_caller, bucketNode2_caller, bucket2_caller και βάζουμε μέσα την εγγραφή μαζί με τα CDR της. Αν βρούμε κενή θέση τότε βάζουμε το ένα κομμάτι της εγγραφής και βλέπουμε αν χωράει το υπόλοιπο διατρέχοντας τη λίστα με κόμβους bucketNode2_caller. Ακολουθούμε παρόμοια διαδικασία με το bucketNode1_caller για το αν κάτι χωράει-δε χωράει. (αντίστοιχα και για τον callee)

- **Delete:** Ψάχνουμε τον αριθμό που μας δώθηκε στις δομές μας. Αν τον βρούμε στο struct bucket1_caller τότε κοιτάμε στο bucket2_caller αν υπάρχει το cdr_uniq_id. Αν δεν βρούμε κάτι σε οποιαδήποτε περίπτωση γυρνάμε μήνυμα λάθος. Αν το βρούμε, το σβήνουμε και κοιτάμε αν χρειάζεται να σβήσουμε το bucket2_caller και τον κόμβο λίστας (τύπου bucketNode2_caller) που το περιέχει.
- **Find/lookup**: Δεν έχει γίνει κάτι εξεζητημένο σε αυτά τα ερωτήματα. Δόθηκε ιδιαίτερη προσοχή στους ελέγχους και τη σειρά τους ανάλογα με το τι έχει δώσει ο χρήστης/αρχείο. Γενικά παίρνουμε σειριακά το HT1/HT2 και ψάχνουμε εγγραφές που να ικανοποιούν τη συνθήκη μας.
- **Indist1:** Σε αυτό το ερώτημα παίρνουμε σειριακά τη δομή του caller και για κάθε εγγραφή κοιτάμε αν έχει επικοινωνήσει και με τους 2 αριθμούς που μας δίνονται, με τη βοήθεια και της δομής του callee. Αν περάσει αυτό το τέστ τότε κοιτάζουμε αν μπορεί να μπει σε μια βοηθητική λίστα, η οποία περιέχει αριθμούς που έχουν επικοινωνήσει με τους 2 δοθέντες αλλά όχι μεταξύ τους. Αν βρω ότι έχει επικοινωνήσει με κάποιον τότε όχι μόνο δεν βάζω μέσα στη λίστα τον αριθμό αυτό, αλλά σβήνω και όλους αυτούς από τη λίστα με τους οποίους έχει επικοινωνήσει.
- **Topdest:** Εδώ, για τον συγκεκριμένο caller κοιτάζω όλους τους αριθμούς με τους οποίους έχει επικοινωνήσει και κρατάω τους κωδικούς σε μια λίστα. Στο τέλος διατρέχω τη λίστα και βρίσκω τον/τους max και τους επιστρέφω.
- **Top:** Για την εύρεση του top δημιουργώ μια βοηθητική λίστα και ο αλγόριθμος είναι ο εξής. Καταρχάς, κοιτάζω αν ο πρώτος κόμβος πληρεί τα κριτήρια του ποσοστού και τον βάζω σίγουρα μέσα στη λίστα. Ύστερα από αυτο η λογική είναι ότι παίρνουμε με τη σειρά κάθε κόμβο που βρίσκεται στη λίστα και (απο τα παιδιά που δεν είναι ήδη στη λίστα) βρίσκουμε το μεγαλύτερο παιδί μεταξύ τους. Αν πληρεί κάποια κριτήρια τότε το βάζουμε και αυτό στη λίστα και επαναλαμβάνουμε τη διαδικασία μέχρι να εκπληρωθεί το ποσοστό K που μας δόθηκε.
- **Bye:** Αδειάζει όλες τις δομές και καθιστά το πρόγραμμα έτοιμο να δεχτεί εκ νέου εντολές. Χωρίζεται σε 2 συναρτήσεις ragnarok1, ragnarok2 μία για κάθε hashtable.
- **Print:** Χωρίζεται σε 2 συναρτήσεις print1, print2 μία για κάθε hashtable.

Τέλος, ακολουθήθηκε το πρότυπο εκτύπωσης που ανέβηκε στο piazza με τη μόνη “αλλαγή” ότι εκτυπώνεται ποια εντολή εκτελείται για ευκολία ανάγνωσης της εξόδου.

Μέσα στον φάκελο υπάρχει αρχείο Makefile. Η εκτέλεση γίνεται με την εντολή της εκφώνησης με τη διαφορά ότι έχει προστεθεί το flag “-b” που πρέπει να ακολουθείται από το όνομα του αρχείου κοστολόγησης.

Παράδειγμα:

```
>make
>./werhauz -h1 10 -h2 10 -s 200 -o input.txt -b charge-config.txt
```