HW1

Aristofanis Chionis Koufakos 1115201500177

Σε αυτή την εργασία κατάφερα να κάνω όλα τα ερωτήματα όπως ζητήθηκαν στην εκφώνηση και στο piazza με όλες τις παραλλαγές.Το έλεγξα και με το αρχείο που δόθηκε με τις 5000 συναλλαγές και δούλεψε μια χαρά.Το μόνο που δεν έχω προλάβει ακόμα είναι η εντολή exit να αποδεσμεύει σωστά όλη την δεσμευμένη μνήμη.Θα το προσπαθήσω αφού γράψω το readme και κάνω μια υποβολή να είμαι σίγουρος ότι παρέδωσα εντός προθεσμίας.

Οι δομές που έχω χρησιμοποιήσει :
→ 2 Hashtables με παρόμοια δομή για τα wallets και τα bitcoins αντίστοιχα, τα οποία βρίσκω το μέγεθός τους διαβάζοντας μια φορά το αρχείο με τα bitcoinbalances που δίνεται. Οι αλλαγές γραμμών ειναι ο αριθμός των πορτοφολιών και ο αριθμός των κενών ειναι ο αριθμός των bitcoin.Αυτά τα πολλαπλασιάζω με ένα συντελεστή 1.42 ώστε στο τέλος τα hashtables αυτά να είναι γεμάτα με τον ιδανικό φόρτο 70%.

→ 1 generic linked list που χρησιμεύει σε πολλά σημεία όπως λίστα με τα userbitcoins, me transactions, trx ids , buckets και άλλα.

→ 1 hashtable ίδιο για sender & receiver όπως ζητείται από την εκφώνηση με buckets.

→ 1 δέντρο όπως ζητείται στην εκφώνηση.
άλλες δομές : wallet, userbitcoin, bitcoin, btcnode, trxobject, bucketnode, bucket όλα αυτά για να με βοηθήσουν να υλοποιήσω τα ζητούμενα της εκφώνησης.Αναλυτικά τι περιέχει η καθεμία μπορείτε να βρείτε στο structs.h .

Περιγραφή του flow:
Αρχικά, η Input Reader συνάρτηση μου διαβάζει τις παραμέτρους από γραμμή εντολής, μια μια με απαραίτητους ελέγχους φτιάχνει τις αρχικές δομές που θα δώσει σαν ορίσματα στην input manager, αυτή είναι η βάση όλης της εργασίας αφού είναι υπεύθυνη να διαβάσει τα δύο αρχεία που δόθηκαν στη γραμμή εντολής να αρχικοποιήσει τα bitcoin & wallet hashtables κάνοντας πάντα τους απαραίτητους ελέγχους μοναδικότητας, ημερομηνίες, ύπαρξης χρηστών στα hashtables και όλα τα συναφή.
Με το διάβασμα του αρχείου με τις συναλλαγές ερχόμαστε και πρώτοι φορά σε επαφή με την 2η βασική συνάρτηση του προγράμματος τη processTrx, η οποία είναι υπεύθυνη για την εκτέλεση των συναλλαγών, το πιο σημαντικό στην εργασία αυτή.
Μετά το πέρας του διαβάσματος των αρχείων αυτών μπαίνουμε σε ένα while loop που διαβάζει εντολές από τον χρήστη, και ανάλογα το μέγεθος της εντολής κάνει συγκρίσεις και αποφασίζει ποιά συνάρτηση πρέπει να δοθεί για να ικανοποιηθεί το αίτημα του χρήστη καλώντας μια από τις high level συναρτήσεις του αρχείου utils.c .
Σε αυτό το σημείο να πω για την requestTransactions που δίνουμε συναλλαγές τη μια μετά την άλλη ότι τερματίζει όταν διαβάσει τον χαρακτήρα “q” .
Κόβω τα ; στο τέλος όπου τα βρίσκω όπως και τις αλλαγές γραμμής.
Όλες οι συναρτήσεις που έχουν να κάνουν με ημερομηνίες τις δέχονται με όποια σειρά πχ date1 time1 date2 time2 or time1 date1 time2 date2 . Αυτά είναι αποδεκτά.

Οι findPayments findEarnings δουλεύουν με ορίσματα χρόνου και ημερομηνίας όπως έχει ζητηθεί αλλά και χωρίς.Γενικά όλες δουλεύουν μια χαρά δεν έχω κάτι ιδιαίτερο να τονίσω όσων αφορά τις high level συναρτήσεις.

----------Σημαντικές backend συναρτήσεις----------

Συναρτήσεις για sender/receiver hashtable insert, ουσιαστικά αυτό το κάνω ως εξής.
Έχω το hashtable και έχει ένα πίνακα από δείκτες σε λίστες από bucket*.
Αυτά ουσιαστικά έχουν μέσα από ένα array bucketnode* και δυο ακέραιους μετρητές.
Τα οποία bucketnodes έχουν ένα δείκτη σε ένα πορτοφόλι και μια λίστα από trxobject* .
Τώρα αν θέλω να προσθέσω μια συναλλαγή, καλώ την search, η οποία θα μου επιστρέψει ένα bucketnode* αν βρει αυτο το χρήστη μέσα στο hashtable, αλλιώς null.
Αν τον βρει είμαστε τυχεροί απλά πουσάρουμε τη συναλλαγή στην αρχή της λίστας συναλλαγών του bucketnode.
Αν όχι καλώ φτιάχνω καινούριο bucketnode, και βάζω στην αρχή της λίστας τη συναλλαγή, και τώρα αυτό πρέπει να το βάλω μέσα στο hashtable , καλώ την insertSRHT για αυτό.
Αυτή με τη σειρά της θα ψάξει στην κατάλληλη θέση αν υπάρχει bucket που να το χωράει το bucketnode, αν ναι το βάζει αν όχι φτιαχνει καινούργιο και το πουσάρει στη λιστα και βαζει εκεί μεσα το bucketnode.

processTrx:
μετά τους απαραίτητους ελέγχους, αν η συναλλαγή είναι έγκυρη,
φτιάχνω το trxobject* μου και καλώ την πολύ σημαντική findbitcoins για να βρει τα κατάλληλα bitcoins που θα συμμετέχουν στη συναλλαγή και θα σπάσει τα δέντρα και θα μειωσει/αυξησει τα ποσά και όλα τα απαραίτητα για το κάθε μπιτκοιν.
Γενικά διαλέγω πάντα από την αρχή μπιτκοιν και αν δεν υπάρχουν στην λίστα του receiver τα βάζω στο τέλος για να μη διαλέγονται συνέχεια τα ίδια και τα δέντρα τους γίνονται πολύ μεγάλα και όλα τα άλλα δέντρα μου είναι πολύ μικρά ή και ανέγγιχτα και μεγαλώνει ο χρόνος εκτέλεσης των συναλλαγών και ο χώρος που πιάνουν.
Γενικά έχω κόψει τον κώδικα σε πολλές μικρές συναρτήσεις που έχουν νόημα, και έχω βάλει παντού σχόλια και καλά ονόματα που δίνουν την περιγραφή της κάθε συνάρτησης οπότε πιστεύω θα είναι εύκολο να καταλάβετε ότι δεν έχω εξηγήσει εδώ.

updateTree:
πολύ προσεκτική δουλειά, παίρνει ένα δέντρο και αναδρομικά το τσεκάρει κάθε κόμβο παιδί με όνομα walletid και μαζεύει όσα λεφτά χρειάζεται από αυτό για να φτάσει το ποσό που έχω ήδη υπολογίσει από πριν καλέσω την συνάρτηση αυτή.Δηλαδή την καλώ και της λέω θέλω 120 ευρώ από αυτό το δέντρο ψάχνει τα φύλλα και μαζεύει μαζεύει μέχρι να φτάσει την τιμή αυτή και όταν καλεί τα παιδιά της περνάει σαν όρισμα πόσο ακόμα μένει να μαζευτεί.
΄Υστερα φτιάχνει αριστερό και δεξί παιδί αριστερά βάζω receiver και δεξιά sender.

Τα λέμε στην προφορική για παραπάνω λεπτομέρειες και ότι απορίες έχετε.
Ευχαριστώ για το χρόνο σας,
Αριστοφάνης.